Elements of Algorithmic Graph Theory An Exercise in Combining Precision with Concision

$\boxed{\text{Working Document}}$

Roland Backhouse[1]
with contributions by Henk Doornbos[2],
Roland Glück[3] and Jaap van der Woude[4]

April 8, 2022

[1]School of Computer Science, University of Nottingham, UK
[2]Questance, Groningen, The Netherlands
[3]Deutsches Zentrum für Luft- und Raumfahrt, Augsburg, Germany
[4]Vakgroep Informatica, Technische Universiteit Eindhoven, The Netherlands

Contents

List of Figures

Abstract

Algorithmic graph theory —as taught in many university courses— focuses on the notions of acyclicity and strongly connected components of a graph, and the related search algorithms. This document is about combining mathematical precision and concision in the context of algorithmic graph theory. Specifically, we use point-free reasoning about paths in graphs (as opposed to pointwise reasoning about paths between nodes in graphs), resorting to pointwise reasoning only where this is unavoidable. Our aim is to use the calculations as the basis of a machine-supported formal verification of graph algorithms in order to assess the current state of automated verification systems.

This document extends joint work with Henk Doornbos, Roland Glück and Jaap van der Woude published in [BDGv22].

Chapter 1

Introduction

This document is about formal calculations in an axiom system representing properties of graphs. Algorithmic graph theory is a subject that is extremely well known and there is little novelty in the content of the theorems that are presented. (That is, when appropriately interpreted, almost all the theorems can be found in undergraduate-level textbooks.) We use the calculations to illustrate the combination of concision and precision that is effected by the use of point-free reasoning. Our thesis is that this is a vital step towards making machine-verified proofs a practical reality.

The presentation is divided into four parts. The first part, comprising chapters 2 to 5 presents the axiomatic framework that we use in later chapters, and the final part, comprising chapters 10 to chapter 13, presents a detailed analysis of graph-searching algorithms, including topological search of an acyclic graph and depth-first search.

Formal analysis of the algorithms we present is based on *point-free* relation algebra rather than the commonly used *pointwise* reasoning about relations. In other words, we reason directly about relations rather than about whether or not a relation holds of a pair of points. Of course, pointwise reasoning is sometimes necessary. Chapter 6 gives a relational semantics to a simple imperative language as well as formulating an interface between pointwise and point-free calculations. Chapters 7 to 9 present well-known properties of relations almost exclusively in point-free form.

Theorems and lemmas are typically stated without proof in the initial chapters (in particular, chapters 2 to 5). Proofs are given, however, of properties that do not already appear in extant literature. (The reason for not including proofs is to maintain a balance between the lengths of the initial chapters and the chapters on graph algorithms. To make the paper self-contained we may include all proofs at a later date.)

This work is part of an ongoing endeavour to make the mathematics of program construction much more *calculational* than is customary in traditional mathematical documents. In order to achieve our goals, we often deviate from traditional mathematical practice, in particular with respect to notational conventions. For example, we use a uniform syntax for denoting all quantifications rather than the many different notations frequently seen in mathematics texts. The notational conventions we do adopt are strongly influenced by the work of Edsger W. Dijkstra. We refer the reader to [Bac03, Bac11] for specific details and raison d'être.

An Apology It is common to include up-to-date citations in scientific publications. With a small number of exceptions, we do not do so here for a number of reasons. First, the graph algorithms and properties of graphs discussed in the paper are now common knowledge having found their way into undergraduate curricula at least forty years ago — so long ago that we have forgotten where we ourselves learned about them. (We make no claim to novelty on this score.) Second, the foundations for the point-free calculations presented in the paper were first laid more than forty years ago [Bac75, BC75] and completed more than twenty years ago (eg. [ABH+92, Mat95, Doo96, DBvdW97]). That writing the paper would make a worthwhile contribution to current research, in particular our conviction that point-free calculations are vital to overcoming some of the challenges faced by modern theorem-proving systems, was inspired by Glück's work [Glü17] to which we refer the reader for more recent literature.

End of Apology

Part I

Mathematical Foundations

Chapter 2

Elements of Lattice Theory

This is the first of several chapters in which we provide an introduction to *relation algebra*, the axiomatic calculus of relations due to (among others) De Morgan, Schröder and Tarski. Full accounts appear in several monographs (see, for example, [SS93, TG87]); we will make do with just a summary of precisely those properties we need in our calculations.

Relation algebra is very rich, so much so that, for the novice, it can be daunting. Our approach is to separate out different substructures and the interfaces between these substructures. Briefly, we present relation algebra as a hierarchy of three substructures: a complete lattice, a regular algebra and finally relation algebra. This chapter is about complete lattices.

2.1 Partial Orderings

A (heterogeneous) binary relation between two sets $\mathcal{A}$ and $\mathcal{B}$ is a subset of the cartesian product $\mathcal{A}{\times}\mathcal{B}$. In other words, a relation is an element of the powerset $2^{\mathcal{A}\times\mathcal{B}}$.

In general, a powerset (the set of subsets of a set) is partially ordered by the subset relation; it is also "complete" and "completely distributive", it has "complements" and its elements (sets) themselves have elements. This section is about axiomatising such properties of partial orderings. Section 2.6 is about axiomatising properties of the element-of relation.

A *complete lattice* is a partially ordered set equipped with unrestricted supremum and infimum operators. Let us assume the set is denoted by $\mathcal{A}$ and the ordering is denoted by $\sqsubseteq$. (Later, when we specialise the discussion to power sets, we switch to using the conventional subset symbol $\subseteq$ but, for the moment, we don't do so in order to emphasise the greater generality of the discussion.) Of course, we assume that the ordering is reflexive, transitive and anti-symmetric.

That the ordering is *complete* means that every function f with target $\mathcal{A}$ has a supremum, denoted by $\sqcup f$, satisfying the property

(2.1) $\langle\forall x :: \sqcup f \sqsubseteq x \equiv \langle\forall u :: f.u \sqsubseteq x\rangle\rangle$

and an infimum, denoted by $\sqcap f$, satisfying the property

(2.2) $\langle\forall x :: x \sqsubseteq \sqcap f \equiv \langle\forall u :: x \sqsubseteq f.u\rangle\rangle$.

Properties (2.1) and (2.2) specialise to binary suprema and infima, which we denote in the usual way by infix operators. That is,

(2.3) $\langle\forall x,y,z :: y\sqcup z \sqsubseteq x \equiv y \sqsubseteq x \wedge z \sqsubseteq x\rangle$

and

(2.4) $\langle\forall x,y,z :: x \sqsubseteq y\sqcap z \equiv x \sqsubseteq y \wedge x \sqsubseteq z\rangle$.

We often use the definitions of supremum and infimum in our calculations without explicity citing the rules.
Aside In many cases, we want to use a function without giving it a specific name. In such cases, we use the notation $\langle x :: E\rangle$ rather than the more conventional $x \mapsto E$ or $\lambda x.E$. We also write $\langle\sqcup x :: E\rangle$ rather than the strictly correct $\sqcup\langle x :: E\rangle$. (The motivation for this is to avoid additional parentheses.) The expression $\langle\forall u :: f.u \sqsubseteq x\rangle$ used in (2.1) is an example: the universal quantifier, denoted by $\forall$, is the infimum operator in the complete lattice of booleans ordered by implication. The "x" in $\langle x :: E\rangle$ is a bound variable, and the scope of the binding is delimited by the angle brackets. The "E" is any well-defined expression of appropriate type. An expression of the form $\langle\oplus x :: E\rangle$ is called a *quantified expression*, the function denoted by $\oplus$ being called the *quantifier*. Typically, we omit type information in quantified expressions relying on the context to make the types clear. (For example, the dummy u in $\langle\forall u :: f.u \sqsubseteq x\rangle$ is assumed to range over the source type of the function f; the information is not provided because it is not relevant.) Occasionally we do include type information in expressions of the form $\langle\sqcup x : R : E\rangle$, where R is some expression. The expression R is called the *range* and the expression E is called the *term* of the quantification.

The advantage of using a consistent notation for quantification is that it is possible to formulate calculational rules based on assumed properties of the quantifier. We only use the quantifier notation when the binary form of $\oplus$ is associative and symmetric. We assume that the reader is familiar with the calculational rules.

As the reader may already have surmised, we use an infix dot to denote function application — as in "$f.u$". The dot is omitted when the argument is parenthesised; function application is then denoted by juxtaposition.) **End of Aside**

A complete lattice has a *top* (a greatest element, the infimum of the unique function with source the empty set), which we denote by $\top\!\!\top$, and a *bottom* (a least element, the supremum of the unique function with source the empty set), which we denote by $\perp\!\!\perp$. That is,

(2.5) $\langle\forall x :: \perp\!\!\perp \sqsubseteq x \sqsubseteq \top\!\!\top\rangle$.

(We use the notation $\top\!\!\top$ and $\perp\!\!\perp$ rather than the more common $\top$ and $\perp$ because $\top$ is easily confused with T.) More generally, we say that a partially ordered set is *bounded* if it has both a top, $\top\!\!\top$, and a bottom, $\perp\!\!\perp$, satisfying (2.5).

A complete lattice is said to be *completely distributive* iff for all sets $\mathcal{J}$ and $\mathcal{K}$ and all functions f of type $\mathcal{A}\leftarrow\mathcal{J}\times\mathcal{K}$, the following equality and its dual hold:

$$\langle\sqcap j : j\in\mathcal{J} : \langle\sqcup k : k\in\mathcal{K} : f(j,k)\rangle\rangle \;=\; \langle\sqcup g : g\in\mathcal{K}\leftarrow\mathcal{J} : \langle\sqcap j : j\in\mathcal{J} : f(j\,,g.j)\rangle\rangle \;\; .$$

(The dual equality is obtained by swapping the infimum and supremum operators.)

The reader may want to instantiate the above formula with $\forall$ as the infimum operator and $\exists$ as the supremum operator; the resulting formula is a statement of the axiom of choice in predicate calculus.

A powerset ordered by set inclusion is a complete, completely distributive lattice but the full power of the distributivity property is rarely used; so-called “universal distributivity” most often suffices. Formally, a complete lattice is said to be *universally distributive* if

$$\langle\forall x,f :: x\sqcup(\sqcap f) \;=\; \langle\sqcap j :: x\sqcup f.j\rangle \;\;\wedge\;\; x\sqcap(\sqcup f) \;=\; \langle\sqcup j :: x\sqcap f.j\rangle\rangle \;\; .$$

We frequently apply universal distributivity without specific reference to the rule. Particular examples that we use frequently are $x\sqcap\perp\!\!\perp=\perp\!\!\perp$ and $x\sqcup\top\!\!\top=\top\!\!\top$.

2.2 Pseudo-Complements

Suppose x is an element of a partially ordered set with top element $\top\!\!\top$ and bottom element $\perp\!\!\perp$. A *complement* of x is an element y such that $x\sqcup y=\top\!\!\top$ and $x\sqcap y=\perp\!\!\perp$. A partially ordered set is said to be *complemented* if it is bounded and every element of the set has a complement.

In our earlier work (see, for example, [ABH$^+$92, DBvdW97]) we explicitly avoided the use of complementation. This was because our goal was to develop a (point-free) relational theory of datatypes in which complementation has no role. In the current application —a theory of finite graphs— complementation does play a significant role. An example is acyclicity of graphs: defined as *not* having cycles.

A powerset ordered by set inclusion is complemented but, as for complete distributivity, the existence of complements is sometimes unnecessary. The weaker notion of "pseudo-complementation" is a consequence of universal distributivity. This section explores its properties. Throughout the section, we assume that $(\mathcal{A},\sqsubseteq)$ is a partially ordered set. We assume the existence of a bottom element $\perp\!\!\!\perp$ and top element $\top\!\!\!\top$, and binary suprema and infima. We also assume finite distributivity of infima over suprema and suprema over infima.

Definition 2.6 (Pseudo-Complement) Suppose $(\mathcal{A},\sqsubseteq)$ is a partially ordered set with bottom element $\perp\!\!\!\perp$ and finite infima. A *pseudo-complement* of an element p of $\mathcal{A}$ is a solution of the equation

(2.7) $x :: \langle\forall q :: q\sqsubseteq x \equiv q\sqcap p = \perp\!\!\!\perp\rangle$.

□

A simple calculation shows that an element p has at most one pseudo-complement: Suppose x and y both satisfy (2.7). Then[1]

$x\sqsubseteq y$

$=$ { assumption: y is a pseudo-complement of p,
(2.7) with $q,x:=x,y$ }

$x\sqcap p = \perp\!\!\!\perp$

$\Leftarrow$ { assumption: x is a pseudo-complement of p,
(2.7) with $q,x:=x,x$ }

$x\sqsubseteq x$

$=$ { reflexivity of $\sqsubseteq$ }

true .

Interchanging x and y, we get $y\sqsubseteq x$; combining the two inequalities, we get $x=y$.

Pseudo-complements may not exist — even when the poset is bounded and complete. However, in the case that the poset is bounded, the pseudo-complements of the top and bottom elements are guaranteed to exist. Indeed,

(2.8) $\sim\perp\!\!\!\perp = \top\!\!\!\top \;\wedge\; \sim\top\!\!\!\top = \perp\!\!\!\perp$

[1] See, for example, [DS90, Bac03, Bac11] for explanation of our notational choices and style of calculation. Briefly, the equality of booleans is denoted both by the symbol "$\equiv$" and the symbol "$=$". The use of two symbols is helpful to disambiguate the overloading of the "$=$" symbol, whilst also emphasising its most fundamental property. We do not use the symbol "$\Leftrightarrow$" for boolean equality because it emphasises its anti-symmetry instead. So-called "continued" relations —three or more expressions connected by relations, as in this calculation— should always be read conjunctionally.

since (as is easily verified)

$$\langle\forall q :: q\sqsubseteq \top\!\!\top \equiv q\sqcap\bot\!\!\bot = \bot\!\!\bot\rangle \;\wedge\; \langle\forall q :: q\sqsubseteq \bot\!\!\bot \equiv q\sqcap\top\!\!\top = \bot\!\!\bot\rangle \;\;.$$

(The first conjunct is (2.7) instantiated with $p := \bot\!\!\bot$ and the second is (2.7) instantiated with $p := \top\!\!\top$.)

If p has a pseudo-complement, we denote it by $\sim\! p$. Instantiating (2.7), the axiom defining $\sim\! p$ is thus

(2.9) $\langle\forall q \;::\; q\sqsubseteq {\sim} p \;\equiv\; q\sqcap p = \bot\!\!\bot\rangle$.

Several properties are immediate from (2.9). By instantiating (2.9) with $q := {\sim} p$, we get:

(2.10) ${\sim} p\sqcap p \;=\; \bot\!\!\bot$.

(This instantiation was used in the calculation above.) An immediate consequence is the "anti-monotonicity" property[2]

(2.11) ${\sim} p\sqsubseteq {\sim} q \;\Leftarrow\; q\sqsubseteq p$

since

$$\begin{array}{ll}
 & {\sim} p \sqsubseteq {\sim} q \\
= & \{\quad (2.9) \text{ with } p,q := q\,,{\sim} p \quad\} \\
 & {\sim} p\sqcap q \;=\; \bot\!\!\bot \\
= & \{\quad (2.10),\ \bot\!\!\bot \text{ is the least element of the ordering} \quad\} \\
 & {\sim} p\sqcap q \sqsubseteq {\sim} p\sqcap p \\
\Leftarrow & \{\quad \text{monotonicity of } (\,({\sim} p)\sqcap\,) \quad\} \\
 & q\sqsubseteq p \;\;.
\end{array}$$

Instantiating (2.9) with $p,q := {\sim} p\,,p$ and applying (2.10), we also get:

(2.12) $p\sqsubseteq {\sim}{\sim} p$.

The combination of (2.11) and (2.12) then gives ${\sim}{\sim}{\sim} p\sqsubseteq {\sim} p$. But, for the converse, we have:

[2]An endofunction f on the partially ordered set $(\mathcal{A},\sqsubseteq)$ is "anti-monotonic" if it is a monotonic function from $(\mathcal{A},\sqsupseteq)$ to $(\mathcal{A},\sqsubseteq)$.

$\sim p \sqsubseteq \sim\sim\sim p$

$=$ { (2.9) with $p,q := \sim\sim p \, , \sim p$ }

$\sim p \sqcap \sim\sim p \; = \; \perp\!\!\perp$

$=$ { symmetry of $\sqcap$ }

$\sim\sim p \sqcap \sim p \; = \; \perp\!\!\perp$

$=$ { (2.9) with $p,q := \sim p \, , \sim\sim p$ }

$\sim\sim p \sqsubseteq \sim\sim p$

$=$ { reflexivity }

true .

By anti-symmetry of the $\sqsubseteq$ relation, we conclude that

(2.13) $\sim\sim\sim p \; = \; \sim p$.

The existence of pseudo-complements is guaranteed by the assumption that the partially ordered set $(\mathcal{A}, \sqsubseteq)$ is a complete, universally distributive lattice. (This is an application of the theory of Galois connections discussed later in section 2.4: see corollary 2.32.)

Assuming distributivity of finite infimum over finite supremum, we can show that

(2.14) $\sim(p \sqcup q) \; = \; \sim p \sqcap \sim q$.

Specifically, for all r,

$r \sqsubseteq \sim(p \sqcup q)$

$=$ { (2.9) with $p,q := p \sqcup q \, , r$ }

$r \sqcap (p \sqcup q) \; = \; \perp\!\!\perp$

$=$ { assumption: finite distributivity }

$r \sqcap p \; = \; \perp\!\!\perp \;\; \wedge \;\; r \sqcap q \; = \; \perp\!\!\perp$

$=$ { (2.9) with $p,q := p,r$ and $p,q := q,r$ }

$r \sqsubseteq \sim p \;\; \wedge \;\; r \sqsubseteq \sim q$

$=$ { definition of binary infimum }

$r \sqsubseteq \sim p \sqcap \sim q$.

Property (2.14) follows by the rule of indirect equality. A similar calculation establishes that:

(2.15) $\sim(p \sqcup \sim p) \; = \; \perp\!\!\perp$.

Specifically, for all r,

$r \sqsubseteq {\sim}(p \sqcup {\sim}p)$

$=$ { (2.9) with $p,q := p \sqcup {\sim}p\ ,\ r$ }

$r \sqcap (p \sqcup {\sim}p) = \perp\!\!\perp$

$=$ { distributivity and (2.9) with $p,q := p,r$ and $p,q := {\sim}p\ ,\ r$ }

$r \sqsubseteq {\sim}p \ \wedge\ r \sqsubseteq {\sim}{\sim}p$

$=$ { definition of infimum and (2.10) with $p := {\sim}p$ }

$r \sqsubseteq \perp\!\!\perp$.

Thus (2.16) follows by the rule of indirect equality.

Combining (2.15) and (2.8) we get:

(2.16) $\ {\sim}{\sim}(p \sqcup {\sim}p) = \top\!\!\top$.

Dual to the notion of pseudo-complement is the notion of pseudo-cocomplement — the *pseudo-cocomplement* of element p in the partially ordered set $(\mathcal{A}, \sqsubseteq)$ is the pseudo-complement of p in the partially ordered set $(\mathcal{A}, \sqsupseteq)$. Formally, the pseudo-cocomplement of p, denoted by ${\backsim}p$, has the property

(2.17) $\ \langle \forall q :: q \sqsupseteq {\backsim}p \ \equiv\ q \sqcup p = \top\!\!\top \rangle$.

We leave it to the reader to dualise the above properties of pseudo-complement. For our purposes, it suffices to note that (assuming universal distributivity), for all p,

(2.18) $\ {\sim}p \sqsubseteq {\backsim}p$

since

${\sim}p$

$=$ { $\top\!\!\top$ is greatest element }

${\sim}p \sqcap \top\!\!\top$

$=$ { dual of (2.10) }

${\sim}p \sqcap (p \sqcup {\backsim}p)$

$\sqsubseteq$ { assumption: distributivity,
and $q \sqcap r \sqsubseteq r$ with $q,r := {\sim}p\ ,\ {\backsim}p$ }

$({\sim}p \sqcap p) \sqcup {\backsim}p$

$=$ { (2.10) }

$\perp\!\!\perp \sqcup {\backsim}p$

$= \quad \{ \quad \bot\!\!\bot$ is the least element $\}$

$\quad \backsim\!\neg p$.

In general, the pseudo-complement and pseudo-cocomplement may be different (even when both exist). A simple example is the 3-element set $\{\bot\!\!\bot, 0, \top\!\!\top\}$ ordered by $\bot\!\!\bot \sqsubseteq 0 \sqsubseteq \top\!\!\top$. The pseudo-complement of 0 is $\bot\!\!\bot$ and its pseudo-cocomplement is $\top\!\!\top$.

2.3 Complements

In this section we define complements in terms of pseudo-complements and pseudo-cocomplements and then show that this is equivalent to a simpler (and possibly more familiar) direct definition. (The reason for beginning with the more complicated definition is that we want to isolate properties that rely only on the weaker notion of pseudo-complement.) The section is concluded by a list of properties that are exploited frequently later.

Definition 2.19 (Complement) Suppose $(\mathcal{A}, \sqsubseteq)$ is a partially ordered set. A *complement* of an element p of $\mathcal{A}$ is an element of $\mathcal{A}$ that is simultaneously the pseudo-complement and pseudo-cocomplement of p . The poset is *complemented* if all of its elements have a complement. Formally, the poset is complemented iff it is pseudo-complemented and pseudo-cocomplemented and

$$\langle \forall p :: {\sim}p = \backsim\!\neg p \rangle \; .$$

(where ${\sim}p$ and $\backsim\!\neg p$ denote, respectively, the pseudo-complement and pseudo-cocomplement of the element p).
□

Lemma 2.20 Suppose $(\mathcal{A}, \sqsubseteq)$ is both pseudo-complemented and pseudo-cocomplemented. Then that it is complemented equivales

(2.21) $\langle \forall p :: p \sqcup {\sim}p = \top\!\!\top \rangle$

(where ${\sim}p$ denotes the pseudo-complement of p).

Proof

$\quad (\mathcal{A}, \sqsubseteq)$ is complemented

$= \quad \{ \quad$ definition 2.19 $\}$

$\quad \langle \forall p :: {\sim}p = \backsim\!\neg p \rangle$

$=\quad\{\quad (2.18) \text{ and anti-symmetry} \quad\}$

$\langle\forall p :: {\sim}p \sqsupseteq \backsim p\rangle$

$=\quad\{\quad (2.17) \text{ with } p,q := p\,,{\sim}p \quad\}$

$\langle\forall p :: {\sim}p \sqcup p = \top\!\!\top\rangle$.

□

Aside It is perhaps worth briefly mentioning that the difference between so-called "classical" and "constructive logic" is that negation in classical logic is a complement whereas in constructive logic it is a pseudo-complement operator. (In both logics, the ordering relation is everywhere-implication, $\perp\!\!\perp$ is the Boolean predicate false and $\top\!\!\top$ is the Boolean predicate true.) In the context of classical versus constructive logic, property (2.21) is called the law of the *excluded middle*. So, in words, a complemented lattice is a lattice that is both pseudo-complemented and pseudo-cocomplemented and in which the law of the excluded middle is universally valid.

Those familiar with constructive logic will recognise (2.16) as a weak form of the law of the excluded middle: property (2.16) (with supremum replaced by disjunction and the top element replaced by true) is valid in both constructive and classical logic whereas the law of the excluded middle is not generally valid in constructive logic.

Property (2.16) is an example of a meta-law relating classical and constructive logic: the double negation of any valid property in classical logic is a valid property of constructive logic. In constructive logic, the basic assumption is the so-called "Curry-Howard isomorphism" which is stronger than the assumption of the existence of pseudo-complements. In our formalism, the Curry-Howard isomorphism is the assumption that, for all p, the endofunction $(\sqcap p)$ has an upper adjoint. The assumption is thus that there is a (binary) function $\Rrightarrow$ such that, for all p, q and r,

$$q\sqcap p \sqsubseteq r \quad\equiv\quad q \sqsubseteq (p\Rrightarrow r) \ .$$

The definition of pseudo-complement is the instance of this property when $r = \perp\!\!\perp$. In general, it is not possible to express $p\Rrightarrow r$ as ${\sim}p \oslash r$ for some (binary) function $\oslash$. For example, take the 4-element set $\{\perp\!\!\perp,0,1,\top\!\!\top\}$ ordered by $\perp\!\!\perp \sqsubseteq 0 \sqsubseteq 1 \sqsubseteq \top\!\!\top$. Then ${\sim}0 = {\sim}1 = \perp\!\!\perp$, but it is required that $1\Rrightarrow 0 = 0$ and $0\Rrightarrow 0 = \top\!\!\top$. That is, if, for all p and r, $p\Rrightarrow r = {\sim}p \oslash r$, we must have ${\sim}0 \oslash 0 = \perp\!\!\perp\oslash 0 = 0$ and ${\sim}1 \oslash 0 = \perp\!\!\perp\oslash 0 = \top\!\!\top$, which is impossible.

End of Aside

An alternative, more direct (and possibly more familiar) definition of complements is given by the following lemma.

Lemma 2.22 Assuming finite distributivity, a complement of p is a solution of the equation

(2.23) $x \ :: \ x\sqcap p = \perp\!\!\perp \ \wedge \ x\sqcup p = \top\!\!\top$.

Proof By definition, a complement of p is a pseudo-complement of p and a pseudo-cocomplement of p; so a complement of p satisfies the equation (2.23).

Conversely, suppose x satisfies (2.23). Then

$\langle \forall q :: q \sqsubseteq x \equiv q \sqcap p = \perp\!\!\perp \rangle$

$=$ { mutual implication }

$\langle \forall q :: q \sqsubseteq x \Rightarrow q \sqcap p = \perp\!\!\perp \rangle \wedge \langle \forall q :: q \sqsubseteq x \Leftarrow q \sqcap p = \perp\!\!\perp \rangle$

$=$ { x satisfies (2.23) (in particular $x \sqcap p = \perp\!\!\perp$) and monotonicity }

$\langle \forall q :: q \sqsubseteq x \Leftarrow q \sqcap p = \perp\!\!\perp \rangle$

$=$ { q

  $=$ { $q \sqsubseteq \top\!\!\top$ for all q }

  $q \sqcap \top\!\!\top$

  $=$ { x satisfies (2.23) (in particular $x \sqcup p = \top\!\!\top$) }

  $q \sqcap (x \sqcup p)$

  $=$ { distributivity }

  $(q \sqcap x) \sqcup (q \sqcap p)$

Leibniz and $\perp\!\!\perp$ is zero of supremum }

$\langle \forall q :: q \sqcap x \sqsubseteq x \Leftarrow q \sqcap p = \perp\!\!\perp \rangle$

$=$ { definition of infimum }

true .

(The penultimate step in the above calculation uses Leibniz's rule: the rule sometimes called "substitution of equals for equals" identified by Gottfried Wilhelm Leibniz as the first rule of logic. Often —in common with conventional mathematical practice— we use Leibniz's rule without specific mention; often however, we do mention the rule explicitly, giving "Leibniz" as hint. Here it is mentioned because the antecedent of the implication is the equality between $q \sqcap p$ and $\perp\!\!\perp$, and this equality has been used in combination with the subcalculation to simplify the consequent.)

That is, x satisfies definition 2.6 of the pseudo-complement of p. Dualising the calculation, x also satisfies the definition of the pseudo-cocomplement of p. Since pseudo-complements and pseudo-cocomplements are the unique solutions of their defining equations, it follows that every element p has a pseudo-complement and a pseudo-cocomplement and both are equal.

$\Box$

We assume various properties of complements in a complete, universally distributive, complemented lattice. First, complements are unique. We denote the unique complement of element x by $-x$. (This notation is temporary: we want to retain the distinction between pseudo-complement and complement until the end of this chapter. After then, we blur the distinction.) Second, complementation is an order isomorphism of $(\mathcal{A},\sqsubseteq)$ and $(\mathcal{A},\sqsupseteq)$. Specifically, for all x and y in $\mathcal{A}$,

(2.24) $-(-x)=x$ and

(2.25) $-x\sqsubseteq y \;\equiv\; x\sqsupseteq -y$.

(Property (2.24) is a consequence of the fact that $-x=\sim x=\backsim x$ and (2.12) and the dual property of $\backsim x$. Property (2.25) then follows from (2.24) and the anti-monotonicity of pseudo-complements: property (2.11).) It follows that complementation distributes through infima and suprema: for all f,

(2.26) $-\langle\sqcap x::f.x\rangle \;=\; \langle\sqcup x::-(f.x)\rangle \;\;\wedge\;\; -\langle\sqcup x::f.x\rangle \;=\; \langle\sqcap x::-(f.x)\rangle$.

Finally, we have the *shunting rule*: for all x, y and z,

(2.27) $x\sqcap y\sqsubseteq z \;\equiv\; x\sqsubseteq -y\sqcup z$.

We leave the verification of the shunting rule to the reader: use excluded middle —see lemma 2.20— and its dual (2.10) (and, of course, that $-y$ is both the pseudo-complement and pseudo-cocomplement of y).

No doubt the rules we have mentioned in this section are familiar to the reader (even more so were we to replace "$\sqsubseteq$" by "$\subseteq$", "$\sqcup$" by "$\cup$" and "$\sqcap$" by "$\cap$"). It would take too much space to enumerate all the properties we assume. Where a property is assumed that we have not explicitly stated, the reader should be able to derive it from this short summary.

2.4 Galois Connections and Fixed-Point Calculus

We assume familiarity with Galois connections and fixed-point calculus. See [Bac02, DB02] for an introduction and [Bac00] for a detailed account of their properties. For ease of reference we summarise the most fundamental properties below.

A Galois connection involves two partially ordered sets $(\mathcal{A},\leq)$ and $(\mathcal{B},\preceq)$ and two functions, $F\in\mathcal{A}\leftarrow\mathcal{B}$ and $G\in\mathcal{B}\leftarrow\mathcal{A}$. These four components together form a *Galois connection* iff for all $x\in\mathcal{B}$ and $y\in\mathcal{A}$ the following holds

(2.28) $F.x\leq y \equiv x\preceq G.y$.

We refer to F as the *lower adjoint* and to G as the *upper adjoint*.

Examples of Galois connections are the definitions of supremum and infimum (2.1) and (2.2), the special cases (2.3) and (2.4), the order isomorphism (2.25) and the shunting rule (2.27). It is straightforward to see that (2.25) and (2.27) are Galois connections (in the case of (2.27), the lower and upper adjoints are $(\sqcap y)$ and $((\sim y)\sqcup)$, respectively) but, as is commonly the case, it is not immediately obvious that (2.1) and (2.2) fit the definition of a Galois connection. Some practice is needed to be able to readily spot that a function is an adjoint in a Galois connection. The skill is, however, well worthwhile acquiring. See, for example, the discussion of irreducibility in section 2.6.

Perhaps the most frequently used property is that a (left or right) adjoint is monotonic. That is, the lower adjoint F in (2.28) has the property that

(2.29) $F.x \leq F.z \Leftarrow x \preceq z$,

and similarly for the upper adjoint G. Of course, it is not the case that all monotonic functions are left or right adjoints.

Perhaps the most significant property (of which monotonicity is a corollary) is that lower adjoints preserve suprema and upper adjoints preserve infima. The theorem in the form that we use it here (thus not in its most general form) is the following.

Theorem 2.30 Suppose $F \in \mathcal{A} \leftarrow \mathcal{B}$ and $G \in \mathcal{B} \leftarrow \mathcal{A}$ are the lower and upper adjoints in a Galois connection of complete lattices $(\mathcal{A}, \leq)$ and $(\mathcal{B}, \preceq)$. Then, for all functions h and k of appropriate type,

$$F.(\sqcup_{\mathcal{B}} h) \;=\; \sqcup_{\mathcal{A}}(F \circ h) \;\wedge\; G.(\sqcap_{\mathcal{A}} k) \;=\; \sqcap_{\mathcal{B}}(G \circ k) \enspace .$$

□

The theorem predicts, for example, that the distributivity law (2.26) follows from the order isomorphism (2.25). Subscripts have been added to the supremum and infimum operators because the types of F and G may be significant. (See, for example, lemma 2.64.)

In fact, theorem 2.30 can be strengthened to an equivalence: the converse is that universal distributivity implies the existence of an upper adjoint.

Theorem 2.31 (Fundamental Existence Theorem) Suppose that $\mathcal{A}$ is a poset and $\mathcal{B}$ is a complete poset. A function $F \in \mathcal{A} \leftarrow \mathcal{B}$ is a lower adjoint in a Galois connection equivales F is supremum-preserving (i.e. for all functions h of appropriate type, $F.(\sqcup_{\mathcal{B}} h)$ satisfies the definition of the supremum in $\mathcal{A}$ of the function $F \circ h$).

Dually, suppose that $\mathcal{B}$ is a poset and $\mathcal{A}$ is a complete poset. A function $G \in \mathcal{B} \leftarrow \mathcal{A}$ is an upper adjoint in a Galois connection equivales G is infimum-preserving (i.e. for all functions k of appropriate type, $G.(\sqcap_{\mathcal{A}} k)$ satisfies the definition of the infimum in $\mathcal{B}$ of the function $G \circ k$).

□

We mentioned earlier that the existence of pseudo-complements is guaranteed by universal distributivity. This is a corollary of the above fundamental existence theorem:

Corollary 2.32 A complete, universally distributive lattice is pseudo-complemented and pseudo-cocomplemented (but not necessarily complemented).

Proof Suppose $(\mathcal{A}, \sqsubseteq)$ is complete and universally distributive. Then by definition of universal distributivity, for each element p of $\mathcal{A}$, the endofunction $(p\sqcap)$ is supremum preserving. By the fundamental existence theorem, theorem 2.31, it has an upper adjoint. Denoting the upper adjoint by $(p\sqcap)^\sharp$, define $\sim p$ to be $(p\sqcap)^\sharp \perp\!\!\perp$. Then, for all q,

$$\begin{array}{ll}
 & q \sqsubseteq \sim p \\
= & \{ \quad \text{definition of } \sim p \quad \} \\
 & q \sqsubseteq (p\sqcap)^\sharp \perp\!\!\perp \\
= & \{ \quad \text{definition of upper adjoint} \quad \} \\
 & p\sqcap q \sqsubseteq \perp\!\!\perp \\
= & \{ \quad \perp\!\!\perp \text{ is the least element} \quad \} \\
 & p\sqcap q = \perp\!\!\perp \quad .
\end{array}$$

That is, $\sim p$ satisfies definition 2.6 of the pseudo-complement of p. Similarly, the lower adjoint $(p\sqcup)^\flat \top\!\!\top$ of the endofunction $(p\sqcup)$ satisfies the definition of the pseudo-cocomplement of p.

The 3-element set $\{\perp\!\!\perp, 0, \top\!\!\top\}$ ordered by $\perp\!\!\perp \sqsubseteq 0 \sqsubseteq \top\!\!\top$ is an example of a complete, universally distributive lattice that is not complemented.

□

Finally, the theorem that is sometimes described as the most interesting property is the theorem that we call the "unity of opposites". The theorem in the form that we use it here is as follows.

Theorem 2.33 (Unity of Opposites) Suppose $F \in \mathcal{A} \leftarrow \mathcal{B}$ and $G \in \mathcal{B} \leftarrow \mathcal{A}$ are the lower and upper adjoints in a Galois connection of posets $(\mathcal{A}, \leq)$ and $(\mathcal{B}, \preceq)$. Then $F.\mathcal{B}$ and $G.\mathcal{A}$ are isomorphic posets. Moreover, if one of $\mathcal{A}$ or $\mathcal{B}$ is complete, $F.\mathcal{B}$ and $G.\mathcal{A}$ are also complete. Assuming $\mathcal{B}$ is complete, the infimum and supremum operators are given by

$$\begin{array}{rcl}
\sqcap_{G.\mathcal{A}} h & = & \sqcap_{\mathcal{B}} h \\
\sqcup_{G.\mathcal{A}} h & = & G.(F.\sqcup_{\mathcal{B}} h) \\
\sqcap_{F.\mathcal{B}} k & = & F.\sqcap_{\mathcal{B}}(G\circ k) \\
\sqcup_{F.\mathcal{B}} k & = & F.\sqcup_{\mathcal{B}}(G\circ k) \quad .
\end{array}$$

□

We now turn to fixed points. Suppose $\mathcal{A} = (A, \sqsubseteq)$ is a partially ordered set and suppose f is a monotonic endofunction on $\mathcal{A}$. Then a *prefix point* of f is an element x of the carrier set A such that $f.x \sqsubseteq x$. A *least prefix point* of f is a solution of the equation

$$x{::}\ f.x \sqsubseteq x \wedge \langle \forall y : f.y \sqsubseteq y : x \sqsubseteq y \rangle \ .$$

A least prefix point of f is thus a prefix point of f that is smaller than all other prefix points of f. A *least fixed point* of f is a solution of the equation

(2.34) $x{::}\ f.x = x \wedge \langle \forall y : f.y = y : x \sqsubseteq y \rangle \ .$

We use the notation $\mathsf{Pre}.f$ to denote the set of prefix points of f and $\mathsf{Fix}.f$ to denote the set of fixed points of f.

Theorem 2.35 (Least Prefix Point) Suppose $(A, \sqsubseteq)$ is an ordered set and the function f of type $(A, \sqsubseteq) \leftarrow (A, \sqsubseteq)$ is monotonic. Then f has at most one least prefix point, μf, characterised by the two properties:

(2.36) $f.\mu f \sqsubseteq \mu f$

and, for all $x \in A$,

(2.37) $\mu f \sqsubseteq x \ \Leftarrow\ f.x \sqsubseteq x \ .$

Moreover, a least prefix point of f is a fixed point of f:

(2.38) $f.\mu f = \mu f \ .$

□

Theorem 2.39 (Greatest Postfix Point) Suppose $(A, \sqsubseteq)$ is an ordered set. Suppose, also, that the function f of type $(A, \sqsubseteq) \leftarrow (A, \sqsubseteq)$ is monotonic. Then f has at most one greatest postfix point, νf, characterised by the two properties:

(2.40) $\nu f \sqsubseteq f.\nu f$

and, for all $x \in A$,

(2.41) $x \sqsubseteq \nu f \ \Leftarrow\ x \sqsubseteq f.x \ .$

Moreover, the greatest postfix point of f is a fixed point of f:

(2.42) $f.\nu f = \nu f \ .$

□

Theorems 2.35 and 2.39 do not assert the existence of least or greatest fixed points. Indeed, a simple example suffices to show that fixed points need not exist: Suppose A is the set $\{0,1\}$ and suppose the ordering $\sqsubseteq$ is the equality relation. Define the endofunction f by $f.0=1$ and $f.1=0$. Then f is monotonic but does not have any fixed points.

Theorems that do guarantee the existence of least and greatest functions are well known — and are applicable to the algebras discussed later in this document. For brevity, we omit the details and generally assume their existence.

A least fixed point of a monotonic function is, as we have seen in theorem 2.35, characterised by two properties. It is a fixed point, and it is least among all prefix points of the functions. This gives us two calculational rules for reasoning about a least fixed point μf of monotonic function f: the *computation rule*

$$\mu f = f.\mu f$$

and the *induction rule*: for all x,

$$\mu f \sqsubseteq x \Leftarrow f.x \sqsubseteq x \quad .$$

Theorem 2.43 (μ-fusion) Suppose $f \in A \leftarrow B$ is the lower adjoint in a Galois connection between the posets $(A, \sqsubseteq)$ and $(B, \preceq)$. Suppose also that $g \in (B, \preceq) \leftarrow (B, \preceq)$ and $h \in (A, \sqsubseteq) \leftarrow (A, \sqsubseteq)$ are monotonic functions. Suppose g and h both have least fixed points, μg and μh, respectively. Then

(a) $\quad f.\mu g \sqsubseteq \mu h \Leftarrow \langle \forall x :: f.(g.x) \sqsubseteq h.(f.x) \rangle$, and

(b) $\quad f.\mu g = \mu h \Leftarrow \langle \forall x :: f.(g.x) = h.(f.x) \rangle$.

Indeed, if the condition $f \circ g = h \circ f$, i.e.

$$\langle \forall x :: f.(g.x) = h.(f.x) \rangle \quad ,$$

holds, f is the lower adjoint in a Galois connection between the posets $(\mathsf{Pre}.h, \sqsubseteq)$ and $(\mathsf{Pre}.g, \preceq)$.

□

2.5 Closure Operators

Definition 2.44 An endofunction f on a partially ordered set $\mathcal{A}$ is a *closure operator* if

$$\langle \forall x,y :: x \sqsubseteq f.y \;\equiv\; f.x \sqsubseteq f.y \rangle \quad .$$

In words, f is a closure operator if, for all y, the set of elements at most $f.y$ is "closed" under application of the function f.
□

Closure operators frequently arise from Galois connections. If F and G are lower and upper adjoints in a Galois connection then $G{\circ}F$ is a closure operator.

It is easy to show that a closure operator is *extensive*

$$\langle\forall x :: x \sqsubseteq f.x\rangle$$

monotonic

$$\langle\forall x,y :: f.x \sqsubseteq f.y \Leftarrow x \sqsubseteq y\rangle$$

and *idempotent*

$$\langle\forall x :: f.x = f.(f.x)\rangle \ .$$

Examples of closure operators that will be discussed later are the reflexive closure, symmetric closure and the transitive closure of a relation.

Definition 2.45 Suppose f and g are both endofunctions of the same type. Then we say that *the fixed points of* f *are closed under* g iff

$$\langle\forall x :: f.(g.x) = g.x \ \Leftarrow \ f.x = x\rangle \ .$$

The function f is said to be g*-idempotent* iff $f{\circ}g{\circ}f = g{\circ}f$.
□

Lemma 2.46 Suppose f and g are both endofunctions of the same type and f is a closure operator. Then the fixed points of f are closed under g if and only if[3] f is g-idempotent.

Proof First, assume f is g-idempotent. Then, for all x,

$$
\begin{array}{cl}
 & f.x = x \\
\Rightarrow & \{ \quad \text{Leibniz} \quad \} \\
 & g.(f.x) = g.x \\
= & \{ \quad \text{assumption: } f \text{ is } g\text{-idempotent} \quad \} \\
 & g.(f.x) = g.x \ \wedge \ f.(g.(f.x)) = g.(f.x) \\
\Rightarrow & \{ \quad \text{Leibniz (apply leftmost equality to rightmost term)} \quad \} \\
 & f.(g.x) = g.x \ .
\end{array}
$$

[3]We use "if and only if" and "equivales" interchangeably, but most often the latter. We use "if and only if" when the proof is by mutual implication — as here. The abbreviation "iff" —pronounced "if"— is sometimes used in definitions.

That is, the fixed points of f are closed under g.

Now to establish the converse, assume the fixed points of f are closed under g. Then, for all x,

$$\begin{array}{ll} & f.(g.(f.x)) \;=\; g.(f.x) \\ \Leftarrow & \{ \quad \text{assumption: } \langle \forall x :: f.(g.x) = g.x \;\Leftarrow\; f.x = x \rangle \text{ with } x := f.x \quad \} \\ & f.(f.x) \;=\; f.x \\ = & \{ \quad f \text{ is a closure operator, idempotence property} \quad \} \\ & \mathsf{true} \;\; . \end{array}$$

That is (by extensionality) $f{\circ}g{\circ}f = g{\circ}f$.
□

The only use we have for definition 2.45 and lemma 2.46 is when the function g is pseudo-complementation. For ease of reference, we instantiate definition 2.45 and lemma 2.46 for this case below. In definition 2.47 and lemma 2.48, we assume that $\mathcal{A}$ is a pseudo-complemented, partially ordered set and f is an endofunction on $\mathcal{A}$.

Definition 2.47 The function f is said to be *pseudo-complementation fixed* iff

$$\langle \forall x :: f.({\sim}x) = {\sim}x \;\Leftarrow\; f.x = x \rangle \;\; .$$

The function f is said to be *pseudo-complementation idempotent* iff

$$\langle \forall x :: f.({\sim}(f.x)) \;=\; {\sim}(f.x) \rangle \;\; .$$

□

Lemma 2.48 Suppose f is a closure operator. Then f is pseudo-complementation fixed equivales f is pseudo-complementation idempotent.

Proof Instantiate the function g in lemma 2.46 in the obvious way.
□

2.6 Atoms, Saturation and Powersets

A powerset forms a complete, universally distributive, complemented lattice under the subset ordering. However, these properties do not characterise the properties of the *elements* of the sets in the powerset. For this, we need the notion of a "saturated", "atomic" lattice: elements of a set are modelled by so-called "atoms". We avoid the use of saturated atomicity wherever possible. However, there are some circumstances where its use is unavoidable.

Throughout this section, we assume that $\mathcal{A}$ is a complete lattice. (This means that we can use the supremum and infimum operators without caveats on their existence.) For brevity, we sometimes omit to say that $\mathcal{A}$ is complete. Variables p and q range over arbitrary elements of $\mathcal{A}$. For the moment, we continue to use $\sqsubseteq$ for the ordering relation on elements of $\mathcal{A}$. A *proper* element is an element different from $\perp\!\!\perp$.

Definition 2.49 (Atom and Atomicity) The element p is an *atom* iff

$$\langle \forall q :: q \sqsubseteq p \;\equiv\; q = p \vee q = \perp\!\!\perp \rangle \;.$$

Note that $\perp\!\!\perp$ is an atom according to this definition. If p is an atom that is different from $\perp\!\!\perp$ we say that it is a *proper* atom. A lattice is said to be *atomic* if

$$\langle \forall q :: q \neq \perp\!\!\perp \;\equiv\; \langle \exists a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : a \sqsubseteq q \rangle \rangle \;.$$

In words, a lattice is atomic if every proper element includes a proper atom.
□

Definition 2.50 (Saturated) A complete lattice is *saturated* iff

$$\langle \forall p :: p \;=\; \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq p : a \rangle \rangle \;.$$

□

Elsewhere the word “full” is sometimes used instead of our “saturated”. Other authors also sometimes use “atomic” to mean both atomic (according to definition 2.49) and saturated.

The following theorem [ABH$^+$92, theorem 6.43] is central to the use of saturated lattices as a model of powersets.

Theorem 2.51 Suppose $\mathcal{A}$ is a complete, universally distributive lattice. Then the following statements are equivalent.

(a) $\mathcal{A}$ is saturated,

(b) $\mathcal{A}$ is atomic and complemented,

(c) $\mathcal{A}$ is isomorphic to the powerset of its atoms.
□

We don't use theorem 2.51 directly. We use it indirectly in the sense that our axiomatisation of relation algebra postulates a complete, universally distributive, saturated lattice. In this section, we consider consequences of the definitions that allow point-wise reasoning akin to conventional reasoning about sets and, in particular, membership properties. Specifically, for lattice element p and proper atom a, the assertion $a \sqsubseteq p$

effectively means $a{\in}p$. For example, the booleans $\neg(a\sqsubseteq p)$ and $a\sqsubseteq{\sim}p$ are equal[4]; this models the commonly used property of set membership: the boolean $\neg(a{\in}p)$ is equal to $a\in{\sim}p$. See lemma 2.52. Other lemmas, such as lemmas 2.60 and 2.63, have a similar role. The section is concluded by a proof of theorem 2.51; hopefully, the proof clarifies how the notion of saturation models the notion of powerset in a way that avoids the use of the membership relation.

We begin by exploring the notion of saturation. First, the above-mentioned lemma expressing how we mimic the defining property of the complement of a set:

Lemma 2.52 Suppose $\mathcal{A}$ is a complete, pseudo-complemented lattice. Then for all elements p of $\mathcal{A}$ and all proper atoms a of $\mathcal{A}$,

(2.53) $\neg(a\sqsubseteq p) \;\equiv\; a\sqsubseteq{\sim}p$.

Proof Suppose a is an atom. Then, for all p,

$$\begin{array}{ll}
& \mathsf{true} \\
= & \{ \quad a \text{ is an atom and } a{\sqcap}p\sqsubseteq a\,, \\
& \quad\;\; \text{definition 2.49 with } p,q := a\,,a{\sqcap}p \quad \} \\
& a{\sqcap}p=\perp\!\!\perp \;\vee\; a{\sqcap}p=a \\
\Rightarrow & \{ \quad \mathcal{A} \text{ is a pseudo-complemented lattice, (2.9) with } p,q:=p,a \;; \\
& \quad\;\; \text{definition of infimum} \quad \} \\
& a\sqsubseteq{\sim}p \;\vee\; a\sqsubseteq p \quad .
\end{array}$$

That is, for all p, $\neg(a\sqsubseteq p)\Rightarrow a\sqsubseteq{\sim}p$. For the converse, we have:

$$\begin{array}{ll}
& a\sqsubseteq{\sim}p \Rightarrow \neg(a\sqsubseteq p) \\
= & \{ \quad \text{predicate calculus} \quad \} \\
& \neg(a\sqsubseteq{\sim}p \;\wedge\; a\sqsubseteq p) \\
= & \{ \quad \text{definition of infimum} \quad \} \\
& \neg(a \sqsubseteq {\sim}p\sqcap p) \\
= & \{ \quad (2.15)\!:\; {\sim}p\sqcap p = \perp\!\!\perp\,;\;\; \text{for all } a\,,\; \perp\!\!\perp\sqsubseteq a \quad \} \\
& a\neq\perp\!\!\perp \quad .
\end{array}$$

[4]In this informal introduction, ${\sim}p$ can be read as the complement of p. Later we prove the rule with ${\sim}p$ defined to be the pseudo-complement of p. The existence of complements is not required although, of course, for set membership complements do indeed exist.

Combining the two calculations, we get the lemma.
□

The universal quantification in the definition of saturated can be eliminated:

Lemma 2.54 A complete, universally distributive lattice is saturated iff its greatest element is saturated, i.e. iff

$$\top\!\!\top \;=\; \langle \sqcup a : \mathsf{atom}.a : a \rangle \;\; .$$

Proof The proof is by mutual implication. One implication is a straightforward consequence of the definition of saturation. (Just instantiate p to $\top\!\!\top$ in definition 2.50.) For the other, first note that a complete, universally distributive lattice is pseudo-complemented. (See corollary 2.32.) This means that lemma 2.52 is applicable. So, for all p,

$$\begin{array}{ll}
 & \top\!\!\top \;=\; \langle \sqcup a : \mathsf{atom}.a : a \rangle \\
= & \{ \quad \text{case analysis and range disjunction} \quad \} \\
 & \top\!\!\top \;=\; \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq p : a \rangle \sqcup \langle \sqcup a : \mathsf{atom}.a \wedge \neg(a \sqsubseteq p) : a \rangle \\
= & \{ \quad \text{lemma 2.52} \quad \} \\
 & \top\!\!\top \;=\; \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq p : a \rangle \sqcup \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq {\sim}p : a \rangle \\
\Rightarrow & \{ \quad p = \top\!\!\top \sqcap p \,,\ \text{universal distributivity} \quad \} \\
 & p \;=\; \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq p : a \sqcap p \rangle \sqcup \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq {\sim}p : a \sqcap p \rangle \\
= & \{ \qquad \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq {\sim}p : a \sqcap p \rangle \\
 & \qquad \sqsubseteq \quad \{ \quad \text{monotonicity} \quad \} \\
 & \qquad \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq {\sim}p : {\sim}p \sqcap p \rangle \\
 & \qquad = \quad \{ \quad \text{pseudo-complements: (2.10)} \quad \} \\
 & \qquad \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq {\sim}p : \bot\!\!\bot \rangle \\
 & \qquad = \quad \{ \quad \sqcup(\mathsf{K}.\bot\!\!\bot) = \bot\!\!\bot \ \text{(where } \mathsf{K} \text{ is the constant combinator)} \quad \} \\
 & \qquad \bot\!\!\bot \quad \} \\
 & p \;=\; \langle \sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq p : a \sqcap p \rangle \;\; .
\end{array}$$

That is, $\top\!\!\top$ is saturated implies p is saturated, for all p.
□

Another consequence of $\top\!\!\top$ being saturated is the existence of complements:

Lemma 2.55 Suppose $\mathcal{A}$ is a complete lattice, and both pseudo-complemented and pseudo-cocomplemented. Then it is complemented if its greatest element, $\top\!\!\top$, is saturated.

Proof We apply lemma 2.20.

$$\begin{array}{ll}
 & \mathcal{A} \text{ is complemented} \\
= & \{ \quad \text{assumption: } \mathcal{A} \text{ is complete, pseudo-complemented} \\
 & \quad\quad \text{and pseudo-cocomplemented, lemma 2.20} \quad \} \\
 & \langle \forall p :: p \sqcup {\sim}p = \top\!\top \rangle \\
= & \{ \quad \text{assumption: } \top\!\top \text{ is saturated; } \top\!\top \text{ is the greatest element} \quad \} \\
 & \langle \forall p :: \langle \sqcup a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : a \rangle \sqsubseteq p \sqcup {\sim}p \rangle \\
= & \{ \quad \text{definition of supremum} \quad \} \\
 & \langle \forall p,a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : a \sqsubseteq p \sqcup {\sim}p \rangle \ .
\end{array}$$

But, for all p and proper atoms a,

$$\begin{array}{ll}
 & a \sqsubseteq p \sqcup {\sim}p \\
= & \{ \quad \text{double negation} \quad \} \\
 & \neg\neg(a \sqsubseteq p \sqcup {\sim}p) \\
= & \{ \quad \text{lemma 2.52, specifically (2.53) with } p,a := p \sqcup {\sim}p \text{ , } a \quad \} \\
 & \neg(a \sqsubseteq {\sim}(p \sqcup {\sim}p)) \\
= & \{ \quad \text{pseudo-complement: (2.15)} \quad \} \\
 & \neg(a \sqsubseteq \perp\!\!\perp) \\
= & \{ \quad \text{assumption: } a \neq \perp\!\!\perp \quad \} \\
 & \mathsf{true} \ .
\end{array}$$

□

Now we turn to the notion of atomicity. The assumption of universal distributivity gives an alternative definition:

Lemma 2.56 Suppose $\mathcal{A}$ is universally distributive. Then $\mathcal{A}$ is atomic equivales

$$\langle \forall q :: q = \perp\!\!\perp \ \equiv \ \langle \sqcup a : \mathsf{atom}.a : a \rangle \sqcap q = \perp\!\!\perp \rangle \ .$$

Proof Comparing the lemma with the definition of atomicity (see definition 2.49), we have to prove that

$$\langle \forall q :: \neg\langle \exists a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : a \sqsubseteq q \rangle \ \equiv \ \langle \sqcup a : \mathsf{atom}.a : a \rangle \sqcap q = \perp\!\!\perp \rangle \ .$$

That is, we must show that

$$\langle \forall q :: \langle \forall a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : \neg(a \sqsubseteq q) \rangle \ \equiv \ \langle \sqcup a : \mathsf{atom}.a : a \rangle \sqcap q = \perp\!\!\perp \rangle \ .$$

We have, for all q,

$$
\begin{array}{ll}
 & \langle\forall a\ :\ \mathsf{atom}.a \wedge a \neq \perp\!\!\perp\ :\ \neg(a \sqsubseteq q)\rangle \\
= & \{\quad \text{trading}\quad\} \\
 & \langle\forall a\ :\ \mathsf{atom}.a \wedge a \sqsubseteq q\ :\ a = \perp\!\!\perp\rangle \\
= & \{\quad a = \perp\!\!\perp \equiv a \sqsubseteq \perp\!\!\perp\text{, definition of supremum}\quad\} \\
 & \langle\sqcup a : \mathsf{atom}.a \wedge a \sqsubseteq q : a\rangle\ =\ \perp\!\!\perp \\
= & \{\quad a \sqsubseteq q\ \equiv\ a = a\sqcap q\text{, Leibniz}\quad\} \\
 & \langle\sqcup a\ :\ \mathsf{atom}.a\ \wedge\ a = a\sqcap q\ :\ a\sqcap q\rangle = \perp\!\!\perp \\
= & \{\quad \sqcup(\mathsf{K}.\perp\!\!\perp) = \perp\!\!\perp\ \text{(where } \mathsf{K} \text{ is the constant combinator)}\quad\} \\
 & \qquad \langle\sqcup a\ :\ \mathsf{atom}.a\ \wedge\ a = a\sqcap q\ :\ a\sqcap q\rangle = \perp\!\!\perp \\
 & \wedge\quad \langle\sqcup a\ :\ \mathsf{atom}.a\ \wedge\ \perp\!\!\perp = a\sqcap q\ :\ a\sqcap q\rangle = \perp\!\!\perp \\
= & \{\quad a\sqcap q \sqsubseteq a\text{; so, by definition 2.49, } a = a\sqcap q\ \vee\ \perp\!\!\perp = a\sqcap q \\
 & \qquad (\Rightarrow)\text{ range disjunction and idempotence of supremum} \\
 & \qquad (\Leftarrow)\text{ range disjunction and } \perp\!\!\perp \text{ is the least element}\quad\} \\
 & \langle\sqcup a : \mathsf{atom}.a : a\sqcap q\rangle\ =\ \perp\!\!\perp \\
= & \{\quad \text{assumption: } \mathcal{A} \text{ is universally distributive}\quad\} \\
 & \langle\sqcup a : \mathsf{atom}.a : a\rangle \sqcap q\ =\ \perp\!\!\perp\ .
\end{array}
$$

□

We are now part way to establishing theorem 2.51:

Corollary 2.57 Suppose $\mathcal{A}$ is complete and universally distributive. Then $\mathcal{A}$ is atomic if $\mathcal{A}$ is saturated.

Proof

$$
\begin{array}{ll}
 & \mathcal{A} \text{ is atomic} \\
= & \{\quad \text{lemma 2.56}\quad\} \\
 & \langle\forall p\ ::\ p = \perp\!\!\perp\ \equiv\ \langle\sqcup a : \mathsf{atom}.a : a\rangle \sqcap p\ =\ \perp\!\!\perp\rangle \\
\Leftarrow & \{\quad \text{Leibniz}\quad\} \\
 & \langle\forall p\ ::\ \langle\sqcup a : \mathsf{atom}.a : a\rangle \sqcap p\ =\ p\rangle \\
= & \{\quad \text{assumption: } \mathcal{A} \text{ is complete, universally distributive and} \\
 & \qquad \text{saturated, lemma 2.54}\quad\} \\
 & \langle\forall p\ ::\ \top\!\!\top \sqcap p\ =\ p\rangle
\end{array}
$$

$=\quad\{\quad \top\!\top$ is the greatest element $\}$

$\quad \mathsf{true}$.

□

We continue with some more technical lemmas. The following lemma gives a useful characterisation of proper atoms.

Lemma 2.58

$$\mathsf{atom}.a \;\equiv\; \langle\forall q : a\sqcap q \neq \bot\!\!\bot : a \sqsubseteq q\rangle \;.$$

$$\mathsf{atom}.a \wedge a \neq \bot\!\!\bot \;\equiv\; \langle\forall q :: a\sqcap q \neq \bot\!\!\bot \equiv a \sqsubseteq q\rangle \;.$$

Proof First,

$\quad \mathsf{atom}.a$

$=\quad\{\quad$ definition 2.49 with $p := a$ $\}$

$\quad \langle\forall q :: q \sqsubseteq a \;\equiv\; q = a \vee q = \bot\!\!\bot\rangle$

$=\quad\{\quad \Leftarrow$ is trivial $\}$

$\quad \langle\forall q : q \sqsubseteq a : q = a \vee q = \bot\!\!\bot\rangle$

$=\quad\{\quad q \sqsubseteq a \;\equiv\; \langle\exists r :: q = a\sqcap r\rangle \quad\}$

$\quad \langle\forall q : \langle\exists r :: q = a\sqcap r\rangle : q = a \vee q = \bot\!\!\bot\rangle$

$=\quad\{\quad$ range disjunction $\}$

$\quad \langle\forall q,r : q = a\sqcap r : q = a \vee q = \bot\!\!\bot\rangle$

$=\quad\{\quad$ one-point rule $\}$

$\quad \langle\forall r :: a\sqcap r = a \vee a\sqcap r = \bot\!\!\bot\rangle$

$=\quad\{\quad$ trading rule and $a\sqcap r = a \equiv a \sqsubseteq r$ $\}$

$\quad \langle\forall r : a\sqcap r \neq \bot\!\!\bot : a \sqsubseteq r\rangle$.

The lemma follows by renaming the bound variable r. Second,

$\quad \mathsf{atom}.a \wedge a \neq \bot\!\!\bot$

$=\quad\{\quad$ above and $a \sqsubseteq q \equiv a\sqcap q = a$ $\}$

$\quad \langle\forall q : a\sqcap q \neq \bot\!\!\bot : a \sqsubseteq q\rangle \wedge \langle\forall q : a \sqsubseteq q : a\sqcap q \neq \bot\!\!\bot\rangle$

$=\quad\{\quad$ trading and mutual implication $\}$

$\quad \langle\forall q :: a\sqcap q \neq \bot\!\!\bot \equiv a \sqsubseteq q\rangle$.

□

Lemma 2.59 For all atoms a and all elements p, $p \sqcap a$ is an atom.

Proof We apply the definition: for all q,

$$\begin{array}{ll}
 & q \sqsubseteq p \sqcap a \\
= & \{ \quad \text{infima} \quad \} \\
 & q \sqsubseteq p \ \wedge \ q \sqsubseteq a \\
= & \{ \quad a \text{ is an atom, definition 2.49 with } p := a \quad \} \\
 & q \sqsubseteq p \ \wedge \ (q \sqsubseteq \perp\!\!\perp \vee q = a) \\
= & \{ \quad \text{distributivity, } q \sqsubseteq \perp\!\!\perp \ \equiv \ q = \perp\!\!\perp, \ \perp\!\!\perp \sqsubseteq p \quad \} \\
 & q = \perp\!\!\perp \ \vee \ (q \sqsubseteq p \ \wedge \ q = a) \\
\Rightarrow & \{ \quad \text{by Leibniz's rule, } q = a \Rightarrow (q \sqsubseteq p \equiv a = p \sqcap a) \quad \} \\
 & q = \perp\!\!\perp \ \vee \ q = p \sqcap a \\
\Rightarrow & \{ \quad \text{case analysis, } \perp\!\!\perp \sqsubseteq p \sqcap a \text{ and } p \sqcap a \sqsubseteq p \sqcap a \quad \} \\
 & q \sqsubseteq p \sqcap a \ .
\end{array}$$

It follows by mutual implication that, for all q, $q \sqsubseteq p \sqcap a \equiv q = \perp\!\!\perp \vee q = p \sqcap a$. The lemma follows by definition of an atom.
□

Lemma 2.60 If $p \neq \perp\!\!\perp$ and b is an atom, then $p = b \equiv p \sqsubseteq b$. Also, if a and b are both proper atoms, $a = b \ \equiv \ a \sqcap b \neq \perp\!\!\perp$.

Proof First,

$$\begin{array}{ll}
 & p = b \\
= & \{ \quad \text{assumption: } p \neq \perp\!\!\perp \text{, predicate calculus} \quad \} \\
 & (p = b \ \vee \ p = \perp\!\!\perp) \ \wedge \ p \neq \perp\!\!\perp \\
= & \{ \quad \text{assumption: } b \text{ is an atom, definition 2.49 with } p,q := p,b \quad \} \\
 & p \sqsubseteq b \ \wedge \ p \neq \perp\!\!\perp \\
= & \{ \quad \text{assumption: } p \neq \perp\!\!\perp \quad \} \\
 & p \sqsubseteq b \ .
\end{array}$$

Second,

$$
\begin{array}{ll}
 & a \sqcap b \neq \perp\!\!\perp \\
\Rightarrow & \{ \quad \text{assumption: } \mathsf{atom}.a \wedge a \neq \perp\!\!\perp \\
 & \quad\;\; \text{lemma 2.58 with } a,q := a,b \quad \} \\
 & a \sqsubseteq b \quad .
\end{array}
$$

Similarly, applying lemma 2.58 with $a,q := b,a$, we get the symmetric property

$$a \sqcap b \neq \perp\!\!\perp \Rightarrow b \sqsubseteq a \quad .$$

Combining the two using anti-symmetry of the partial ordering

$$a \sqcap b \neq \perp\!\!\perp \Rightarrow a = b \quad .$$

The converse implication is clearly true given the assumption that a and b are both proper atoms. So the lemma follows by mutual implication.
□

We are now well on the way to establishing theorem 2.51:

Corollary 2.61 Suppose $\mathcal{A}$ is complete and universally distributive. Then $\mathcal{A}$ is saturated if and only if $\mathcal{A}$ is atomic and complemented.

Proof Suppose $\mathcal{A}$ is complete and universally distributive.

First, by corollary 2.32, $\mathcal{A}$ is pseudo-complemented and pseudo-cocomplemented. So, by corollary 2.57 and lemma 2.55, it is atomic and complemented if it is saturated.

Conversely, suppose $\mathcal{A}$ is atomic and complemented. Then

$$
\begin{array}{ll}
 & \mathcal{A} \text{ is saturated} \\
\Leftarrow & \{ \quad \text{lemma 2.54} \quad \} \\
 & \top\!\!\top = \langle \sqcup a : \mathsf{atom}.a : a \rangle \\
\Leftarrow & \{ \quad \text{assumption: } \mathcal{A} \text{ is complemented,} \\
 & \quad\;\; \text{double negation: (2.24) and } -\perp\!\!\perp = \top\!\!\top \quad \} \\
 & \perp\!\!\perp = -\langle \sqcup a : \mathsf{atom}.a : a \rangle \\
= & \{ \quad \text{assumption: } \mathcal{A} \text{ is atomic, lemma 2.56 with } q := \perp\!\!\perp \quad \} \\
 & \langle \sqcup a : \mathsf{atom}.a : a \rangle \sqcap -\langle \sqcup a : \mathsf{atom}.a : a \rangle = \perp\!\!\perp \\
= & \{ \quad \text{complements are pseudo-complements, (2.10)} \quad \} \\
 & \mathsf{true} \quad .
\end{array}
$$

□

Lemma 2.52 establishes the existence of a Galois connection, albeit slightly disguised. Specifically, suppose $\mathcal{A}$ is a complete, complemented lattice. Then we have, for all elements p of $\mathcal{A}$, all proper atoms a of $\mathcal{A}$ and all booleans b,

$$(2.62)\quad (a \sqsubseteq p) \Rightarrow b \;\equiv\; p \sqsubseteq \textsf{if } b \rightarrow \top\!\top \;\Box\; \neg b \rightarrow {\sim}a \textsf{ fi} \quad .$$

(The simple proof that this is equivalent to (2.53) is left to the reader; of course, (2.25) must be invoked as well.) Applying theorem 2.30, we deduce that atoms are *irreducible* in the following sense.

Lemma 2.63 Suppose $\mathcal{A}$ is a complete, universally distributive, saturated lattice and a is a proper atom of $\mathcal{A}$. Then, for all subsets S of the proper atoms of $\mathcal{A}$,

$$a \sqsubseteq \langle \sqcup b : b \in S : b \rangle \;\equiv\; \langle \exists b : b \in S : a = b \rangle \quad .$$

Proof

$$\begin{array}{ll}
 & a \sqsubseteq \langle \sqcup b : b \in S : b \rangle \\
= & \{ \quad (2.62) \text{ and theorem 2.30 with } F := (a\sqsubseteq) \text{ and } \mathcal{A} := (\textsf{Bool}, \Rightarrow) \quad \} \\
 & \langle \exists b : b \in S : a \sqsubseteq b \rangle \\
= & \{ \quad a \text{ is a proper atom, dummy } b \text{ ranges over proper atoms,} \\
 & \quad \text{lemma 2.60} \quad \} \\
 & \langle \exists b : b \in S : a = b \vee a = \bot\!\bot \rangle \quad .
\end{array}$$

□

Let us now return to theorem 2.51. Corollary 2.61 establishes that 2.51(a) and 2.51(b) are equivalent. So it remains to establish that, if $\mathcal{A}$ is a complete and universally distributive, $\mathcal{A}$ is isomorphic to a powerset if and only if it is atomic, complemented and saturated.

If S is a set, the powerset 2^S is the set of all subsets of S. Set theory postulates that 2^S is a complete, universally distributive lattice under the usual subset ordering. The proper atoms of 2^S are the singleton sets $\{a\}$ where dummy a ranges over the elements of S; its top and bottom elements are S and the empty set $\emptyset$, and the supremum operator is set union. Set theory postulates that, for any subset p of S,

$$a \in p \;\equiv\; \{a\} \subseteq p$$

and

$$p \;=\; \langle \cup a : a \in p : \{a\} \rangle \quad .$$

That is, set theory postulates that 2^S is saturated. By corollary 2.61 it is thus atomic and complemented: the *complement* $\neg p$ of the set p is, of course,

$$\neg p \;=\; \langle \cup a : \neg(a \in p) : \{a\} \rangle \;\;.$$

Thus, if $\mathcal{A}$ is isomorphic to a powerset, it is atomic, complemented and saturated.

Conversely, if $\mathcal{A}$ is a complete, universally distributive, saturated lattice, define S to be the carrier set of $\mathcal{A}$. Define the mapping set from $\mathcal{A}$ to 2^S by, for all elements p of $\mathcal{A}$,

$$\mathsf{set}.p \;\;=\;\; \langle \cup a \;:\; \mathsf{atom}.a \wedge a \neq \bot\!\!\bot \wedge a \sqsubseteq p \;:\; \{a\} \rangle$$

(where a ranges over elements of $\mathcal{A}$). A straightforward consequence of the definition of atoms, definition 2.49, is that proper atoms a of $\mathcal{A}$ are then mapped to $\{a\}$, which is a proper atom of 2^S; the bottom element $\bot\!\!\bot$ is mapped to $\emptyset$ and the top element $\top\!\!\top$ is mapped to S. Then, assuming $\mathcal{A}$ is saturated, for all p in the carrier set of $\mathcal{A}$, we have

$$\begin{array}{ll}
 & \langle \cup a : \mathsf{atom}.a \wedge a \neq \bot\!\!\bot \wedge a \sqsubseteq p : \mathsf{set}.a \rangle \\
= & \{ \quad \text{by the definition of set and definition 2.49,} \\
 & \qquad a \neq \bot\!\!\bot \wedge a \sqsubseteq p \Rightarrow \mathsf{set}.a = \{a\} \quad \} \\
 & \langle \cup a : \mathsf{atom}.a \wedge a \neq \bot\!\!\bot \wedge a \sqsubseteq p : \{a\} \rangle \\
= & \{ \quad \text{definition of set} \quad \} \\
 & \mathsf{set}.p \\
= & \{ \quad \text{assumption: } \mathcal{A} \text{ is saturated} \quad \} \\
 & \mathsf{set}.\langle \sqcup a : \mathsf{atom}.a \wedge a \neq \bot\!\!\bot \wedge a \sqsubseteq p : a \rangle \;\;.
\end{array}$$

That is, assuming $\mathcal{A}$ is complete, universally distributive and saturated, the function set is an isomorphism of the lattice $\mathcal{A}$ and the powerset 2^S (ordered by set inclusion).

As mentioned at the beginning of this section, the exploitation of properties of atoms is a mechanism for mimicking pointwise reasoning within an axiomatic formulation of powersets. Because we want to avoid pointwise reasoning, we avoid the use of atoms except where this is absolutely essential (for example to show that every node in a graph is contained in a strongly connected component of the graph).

2.7 The Lattice of Fixed Points

Throughout this section, f is a monotonic endofunction on a partially ordered set $\mathcal{A}$. Recall that we use $\mathsf{Fix}.f$ to denote the fixed points of f. This section is about showing the extent to which $\mathsf{Fix}.f$ inherits algebraic properties of $\mathcal{A}$.

The set $\mathsf{Fix.f}$ is a subset of the set $\mathcal{A}$ and thus inherits its partial ordering. The following well-known lemma is often attributed to Alfred Tarski.

Lemma 2.64 Suppose the partially ordered set $\mathcal{A}$ is a complete lattice. Suppose f is a closure operator on the lattice $\mathcal{A}$. Then $\mathsf{Fix.f}$ is complete. Specifically, if h is a function with range $\mathsf{Fix.f}$ then

$$\sqcap_{\mathsf{Fix.f}}h \;=\; \sqcap_{\mathcal{A}}h \quad\wedge\quad \sqcup_{\mathsf{Fix.f}}h \;=\; f.\sqcup_{\mathcal{A}}h \quad .$$

In particular,

$$\top\!\!\top_{\mathsf{Fix.f}} \;=\; \top\!\!\top_{\mathcal{A}} \quad\wedge\quad \bot\!\!\bot_{\mathsf{Fix.f}} \;=\; f.\bot\!\!\bot_{\mathcal{A}} \quad .$$

Proof This is an application of the unity-of-opposites theorem (theorem 2.33). We note that a closure operator is the lower adjoint in a Galois connection: letting ι denote the "forgetful" function of type $\mathcal{A}\leftarrow\mathsf{Fix.f}$ (so called because it "forgets" that its argument is a fixed point), the definition 2.44 can be written as, for all $x{\in}\mathcal{A}$ and all $y\in\mathsf{Fix.f}$,

$$x\sqsubseteq\iota.y \;\equiv\; f.x\sqsubseteq y \quad .$$

The lemma follows by applying theorem 2.33 with the instantiations

$$\mathcal{A},\mathcal{B},F,G := \mathsf{Fix.f}\,,\mathcal{A}\,,f\,,\iota \quad .$$

□

Effectively, lemma 2.64 states that the infimum in $\mathcal{A}$ of a function h with range $\mathsf{Fix.f}$ is a fixed point of f. On the other hand, the supremum $\sqcup_{\mathcal{A}}h$ of a function h with range $\mathsf{Fix.f}$ is not necessarily a fixed point of f. Instantiating the unity-of-opposites theorem is complicated by the type information: the formulae given in the lemma for the infimum and the supremum "forget" the "forgetful" function. The lemma can easily be verified independently without reference to the unity-of-opposites theorem.

Our goal now is to show that if $\mathcal{A}$ is saturated then $\mathsf{Fix.f}$ is also saturated. Great care needs to be taken in doing so. The difficulty is that, although the partial ordering is the same for both sets, the supremum of a function with range $\mathsf{Fix.f}$ in $\mathcal{A}$ is not the same as the supremum of the function in $\mathsf{Fix.f}$. In particular, the least element of $\mathcal{A}$ is not the same as the least element of $\mathsf{Fix.f}$. Overloading the symbol "$\bot\!\!\bot$" is therefore ambiguous! Similarly, distributivity properties can also be ambiguous, or incorrect, if care is not taken to make clear which suprema are intended.

To avoid the clutter and the ambiguity, we use the unsubscripted symbol "$\bot\!\!\bot$" exclusively for $\bot\!\!\bot_{\mathcal{A}}$; similarly, occurrences of $\sqcup$ denote supremum in $\mathcal{A}$. Because the greatest elements of $\mathcal{A}$ and $\mathsf{Fix.f}$ coincide, subscripts are unnecessary for $\top\!\!\top$; similarly subscripts are unnecessary on occurrences of $\sqcap$. Occasionally it is necessary to recall that $f.\bot\!\!\bot$ denotes $\bot\!\!\bot_{\mathsf{Fix.f}}$ but sometimes we re-introduce subscripts for greater clarity.

Lemma 2.65 Suppose $\mathcal{A}$ is a complete, universally distributive lattice. Suppose f is a closure operator on the lattice $\mathcal{A}$ and suppose f is pseudo-complementation fixed. Then $f.a$ is an atom of $\mathsf{Fix}.f$ if a is an atom of $\mathcal{A}$.

Proof First, corollary 2.32 establishes that $\mathcal{A}$ is pseudo-complemented. Suppose that a is an atom of $\mathcal{A}$. By the idempotency property of closure operators, $f.a$ is a fixed point of f. Now, suppose p is an element of $\mathsf{Fix}.f$. That is, p is an element of $\mathcal{A}$ and $p = f.p$. Then,

$$\begin{array}{ll}
 & p \sqsubseteq f.a \\
= & \{ \quad \text{predicate calculus} \quad \} \\
 & a \sqsubseteq p \sqsubseteq f.a \;\vee\; (\neg(a \sqsubseteq p) \wedge p \sqsubseteq f.a) \quad .
\end{array}$$

We now consider each disjunct in turn. First,

$$\begin{array}{ll}
 & a \sqsubseteq p \sqsubseteq f.a \\
= & \{ \quad f \text{ is a closure operator and so is monotonic} \quad \} \\
 & a \sqsubseteq p \;\wedge\; f.a \sqsubseteq f.p \;\wedge\; p \sqsubseteq f.a \\
= & \{ \quad p = f.p \text{, Leibniz and anti-symmetry of } \sqsubseteq \quad \} \\
 & a \sqsubseteq p \;\wedge\; p = f.a \\
= & \{ \quad f \text{ is a closure operator, extensivity} \quad \} \\
 & p = f.a \quad .
\end{array}$$

(Note that no use has yet been made of the assumption that a is an atom of $\mathcal{A}$.) The second disjunct is split into two cases: $a = \perp\!\!\perp$ and $a \neq \perp\!\!\perp$. In the first case,

$$\neg(\perp\!\!\perp \sqsubseteq p) \wedge p \sqsubseteq f.\perp\!\!\perp \;\equiv\; \mathsf{false}$$

(since $\perp\!\!\perp \sqsubseteq p \equiv \mathsf{true}$). In the second case, a is, by definition, a proper atom of $\mathcal{A}$. So

$$\begin{array}{ll}
 & \neg(a \sqsubseteq p) \;\wedge\; p \sqsubseteq f.a \\
= & \{ \quad \text{assumption: } \mathcal{A} \text{ is complete, universally distributive and saturated;} \\
 & \quad\;\; a \text{ is a proper atom of } \mathcal{A}\text{, lemma 2.52} \quad \} \\
 & a \sqsubseteq {\sim}p \;\wedge\; p \sqsubseteq f.a \\
\Rightarrow & \{ \quad f \text{ is a closure operator and so is monotonic} \quad \} \\
 & f.a \sqsubseteq f.({\sim}p) \;\wedge\; p \sqsubseteq f.a
\end{array}$$

$=$ { $p = f.p$ and f is pseudo-complementation fixed, so $\sim p = f.(\sim p)$ }

$f.a \sqsubseteq \sim p \;\wedge\; p \sqsubseteq f.a$

$\Rightarrow$ { anti-monotonicity: (2.11) }

$\sim\sim p \sqsubseteq \sim(f.a) \;\wedge\; p \sqsubseteq f.a$

$\Rightarrow$ { double negation: (2.12), transitivity and infimum }

$p \sqsubseteq \sim(f.a) \sqcap f.a$

$=$ { pseudo-complement: (2.10) }

$p \sqsubseteq \perp\!\!\perp$

$\Rightarrow$ { $\perp\!\!\perp$ is least element, $a \neq \perp\!\!\perp$ }

$\neg(a \sqsubseteq p) \;\wedge\; p \sqsubseteq f.a$.

We conclude that, when $a \neq \perp\!\!\perp$,

$$\neg(a \sqsubseteq p) \;\wedge\; p \sqsubseteq f.a \;\equiv\; p \sqsubseteq \perp\!\!\perp \;.$$

Substituting the results of the three cases in the initial calculation, we have established that

$$p \sqsubseteq f.a \;\equiv\; p = f.a \vee p \sqsubseteq \perp\!\!\perp \;.$$

Since, as already mentioned, $f.a$ is a fixed point of f, it is, by definition 2.49, an atom of $\mathsf{Fix}.f$.
□

Lemma 2.66 Suppose $\mathcal{A}$ is a complete, universally distributive lattice. Suppose f is a closure operator on the lattice $\mathcal{A}$ and suppose f is pseudo-complementation fixed. Then

$$\top\!\!\top_{\mathsf{Fix.f}} \;=\; f.\top\!\!\top_{\mathcal{A}} \;=\; \top\!\!\top_{\mathcal{A}} \;\;\wedge\;\; \perp\!\!\perp_{\mathsf{Fix.f}} \;=\; f.\perp\!\!\perp_{\mathcal{A}} \;=\; \perp\!\!\perp_{\mathcal{A}} \;.$$

Proof The first conjunct is immediate from lemma 2.64 and the extensivity of a closure operator. For the second conjunct, we have:

$f.\perp\!\!\perp_{\mathcal{A}} = \perp\!\!\perp_{\mathcal{A}}$

$=$ { $\perp\!\!\perp = \sim\top\!\!\top$ (see (2.8)) }

$f.(\sim\top\!\!\top_{\mathcal{A}}) = \sim\top\!\!\top_{\mathcal{A}}$

$\Leftarrow$ { assumption: f is pseudo-complementation idempotent
lemma 2.48 and definition 2.47 }

$$\begin{array}{ll}
 & f.\top\!\!\top_{\mathcal{A}} = \top\!\!\top_{\mathcal{A}} \\
= & \{ \quad \top\!\!\top \text{ is the greatest element of } \mathcal{A}\text{, so } f.\top\!\!\top \sqsubseteq \top\!\!\top\text{, anti-symmetry} \quad \} \\
 & \top\!\!\top_{\mathcal{A}} \sqsubseteq f.\top\!\!\top_{\mathcal{A}} \\
= & \{ \quad \text{assumption: } f \text{ is a closure operator and hence extensive} \quad \} \\
 & \mathsf{true} \ .
\end{array}$$

□

Lemma 2.67 Suppose f is a pseudo-complementation idempotent closure operator. Then $\mathsf{Fix}.f$ is pseudo-complemented. Specifically, the pseudo-complement of fixed point x of f in $\mathsf{Fix}.f$ is its pseudo-complement in $\mathcal{A}$.

Proof Suppose y is a fixed point of f (i.e. $y = f.y$) and suppose $\sim\!y$ is the pseudo-complement of y in $\mathcal{A}$. We show that $\sim\!y$ is the pseudo-complement of y in $\mathsf{Fix}.f$.

Instantiating definition 2.6 with $\mathcal{A}$,p := $\mathsf{Fix}.f$, y, we must show that

$$\langle \forall q \ : \ q \in \mathsf{Fix}.f \ : \ q \sqsubseteq \sim\!y \equiv q \sqcap y = \bot\!\!\bot_{\mathsf{Fix}.f} \rangle \ .$$

But this is immediate from (2.9) (with $p := y$) and lemma 2.66 (specifically, the second conjunct). Note that implicit use is made of the fact that the ordering relation and infima are the same in $\mathcal{A}$ and $\mathsf{Fix}.f$.

□

Lemma 2.65 identifies a subset of the atoms of $\mathsf{Fix}.f$. We now strengthen the lemma to an equality.

Lemma 2.68 Suppose $\mathcal{A}$ is a complete, universally distributive, saturated lattice. Suppose f is a closure operator on the lattice $\mathcal{A}$ and suppose f is pseudo-complementation fixed. Then, for all a,

$$\mathsf{atom}_{\mathsf{Fix}.f}.a \ \equiv \ \langle \exists\, b : \mathsf{atom}_{\mathcal{A}}.b : a = f.b \rangle \ .$$

Moreover, $\mathsf{Fix}.f$ is saturated.

Proof Under the given assumptions, $\mathcal{A}$ is pseudo-complemented by corollary 2.32. Lemma 2.65 then establishes the implication

$$\mathsf{atom}_{\mathsf{Fix}.f}.a \Leftarrow \langle \exists\, b : \mathsf{atom}_{\mathcal{A}}.b : a = f.b \rangle \ .$$

For the converse, we first observe that

(2.69) $\langle \sqcup_{\mathcal{A}} b : \mathsf{atom}_{\mathcal{A}}.b : f.b \rangle \ = \ \top\!\!\top_{\mathsf{Fix}.f}$.

The proof is straightforward:

$$\begin{array}{ll}
& \top\!\!\top_{\mathcal{A}} \\
\sqsupseteq & \{ \quad \text{definition of top element} \quad \} \\
& \langle \sqcup_{\mathcal{A}} b : \mathsf{atom}_{\mathcal{A}}.b : f.b \rangle \\
\sqsupseteq & \{ \quad f \text{ is a closure operator, so } b \sqsubseteq f.b \quad \} \\
& \langle \sqcup_{\mathcal{A}} b : \mathsf{atom}_{\mathcal{A}}.b : b \rangle \\
= & \{ \quad \text{assumption: } \mathcal{A} \text{ is saturated} \quad \} \\
& \top\!\!\top_{\mathcal{A}} \quad .
\end{array}$$

Thus, by anti-symmetry, $\top\!\!\top_{\mathcal{A}} = \langle \sqcup_{\mathcal{A}} b : \mathsf{atom}_{\mathcal{A}}.b : f.b \rangle$. Property (2.69) follows immediately from the fact that $\top\!\!\top_{\mathcal{A}} = \top\!\!\top_{\mathsf{Fix.f}}$ (see lemma 2.64).

Now we can establish the converse implication. Suppose a is an atom in $\mathsf{Fix.f}$. There are two cases. If $a = \perp\!\!\perp_{\mathsf{Fix.f}}$, then $a = f.\perp\!\!\perp_{\mathcal{A}}$ by lemma 2.64. In the second case, $a \neq \perp\!\!\perp_{\mathsf{Fix.f}}$. Then

$$\begin{array}{ll}
& a \\
= & \{ \quad \text{definition of top, assumption: } a \in \mathsf{Fix.f} \quad \} \\
& a \sqcap \top\!\!\top_{\mathsf{Fix.f}} \\
= & \{ \quad (2.69) \quad \} \\
& a \sqcap \langle \sqcup_{\mathcal{A}} b : \mathsf{atom}_{\mathcal{A}}.b : f.b \rangle \\
= & \{ \quad \text{assumed universal distributivity property of } \mathcal{A} \quad \} \\
& \langle \sqcup_{\mathcal{A}} b \ : \ \mathsf{atom}_{\mathcal{A}}.b \ : \ a \sqcap f.b \rangle \\
= & \{ \quad \perp\!\!\perp_{\mathcal{A}} \text{ is zero of suprema} \quad \} \\
& \langle \sqcup_{\mathcal{A}} b \ : \ \mathsf{atom}_{\mathcal{A}}.b \ \wedge \ f.b \neq \perp\!\!\perp_{\mathcal{A}} \ : \ a \sqcap f.b \rangle \\
= & \{ \quad \text{assumption: } a \text{ is a proper atom of } \mathsf{Fix.f}, \\
& \quad\quad \text{by lemma 2.65, } f.b \text{ is an atom of } \mathsf{Fix.f}, \\
& \quad\quad \text{lemma 2.60 (applied to atoms of } \mathsf{Fix.f}) \quad \} \\
& \langle \sqcup_{\mathcal{A}} b \ : \ \mathsf{atom}_{\mathcal{A}}.b \ \wedge \ a = f.b \ : \ a \sqcap f.b \rangle \\
= & \{ \quad \text{Leibniz and idempotence of infimum} \quad \} \\
& \langle \sqcup_{\mathcal{A}} b \ : \ \mathsf{atom}_{\mathcal{A}}.b \ \wedge \ a = f.b \ : \ a \rangle \quad .
\end{array}$$

Summarising,

$$a \neq \perp\!\!\perp_{\mathsf{Fix.f}} \ \wedge \ a \ = \ \langle \sqcup_{\mathcal{A}} b \ : \ \mathsf{atom}_{\mathcal{A}}.b \ \wedge \ a = f.b \ : \ a \rangle \quad .$$

Now assume $\neg\langle\exists\, b : \mathsf{atom}_{\mathcal{A}}.b : a = f.b\rangle$. Then, by the definition of supremum,

$$a \neq \perp\!\!\perp_{\mathsf{Fix.f}} \quad\wedge\quad a = \perp\!\!\perp_{\mathcal{A}} \quad .$$

But $\perp\!\!\perp_{\mathsf{Fix.f}} = \perp\!\!\perp_{\mathcal{A}}$ (lemma 2.66). So we have a contradiction, and we conclude that $\langle\exists\, b : \mathsf{atom}_{\mathcal{A}}.b : a = f.b\rangle$ as required.

Finally, in order to show that Fix.f is saturated, it suffices to prove that $\top\!\!\top_{\mathsf{Fix.f}}$ is saturated. (See lemma 2.54.) That is, we have to prove that $\top\!\!\top_{\mathsf{Fix.f}}$ is the supremum of the atoms of Fix.f . Specifically, we have to prove that

(2.70) $\langle\sqcup_{\mathsf{Fix.f}}\, a : \mathsf{atom}_{\mathsf{Fix.f}}.a : a\rangle \;=\; \top\!\!\top_{\mathsf{Fix.f}} \quad .$

We have:

$$\begin{array}{ll}
 & \top\!\!\top_{\mathsf{Fix.f}} \\
\sqsupseteq & \{ \quad \text{definition of top and supremum} \quad \} \\
 & \langle\sqcup_{\mathsf{Fix.f}}\, a : \mathsf{atom}_{\mathsf{Fix.f}}.a : a\rangle \\
= & \{ \quad \text{lemma 2.64} \quad \} \\
 & f.\langle\sqcup_{\mathcal{A}}\, a : \mathsf{atom}_{\mathsf{Fix.f}}.a : a\rangle \\
\sqsupseteq & \{ \quad \text{properties of suprema, } f \text{ is monotonic} \quad \} \\
 & \langle\sqcup_{\mathcal{A}}\, a : \mathsf{atom}_{\mathsf{Fix.f}}.a : f.a\rangle \\
= & \{ \quad \mathsf{atom}_{\mathsf{Fix.f}}.a \Rightarrow f.a = a \quad \} \\
 & \langle\sqcup_{\mathcal{A}}\, a : \mathsf{atom}_{\mathsf{Fix.f}}.a : a\rangle \\
\sqsupseteq & \{ \quad \text{assumptions on } \mathcal{A} \text{ and } f\text{:} \\
 & \quad\;\; \text{so, by lemma 2.65, } \langle\forall b : \mathsf{atom}_{\mathcal{A}}.b : \mathsf{atom}_{\mathsf{Fix.f}}.(f.b)\rangle \quad \} \\
 & \langle\sqcup_{\mathcal{A}}\, b : \mathsf{atom}_{\mathcal{A}}.b : f.b\rangle \\
= & \{ \quad (2.69) \quad \} \\
 & \top\!\!\top_{\mathsf{Fix.f}} \quad .
\end{array}$$

The required property (2.70) now follows by anti-symmetry of the ordering relation.
□

We conclude this section with a summary of the properties we have established.

Theorem 2.71 Suppose $\mathcal{A}$ is a complete, universally distributive, saturated lattice. Suppose f is a closure operator on the lattice $\mathcal{A}$ and suppose f is pseudo-complementation fixed. Then Fix.f is a complete, saturated lattice. The atoms in Fix.f are given by $\{b : \mathsf{atom}_{\mathcal{A}}.b : f.b\}$.

Proof This is a combination of lemmas 2.68 and 2.64.
□

Note that we haven't proved that Fix.f is universally distributive. (Currently we do not know whether or not this is always the case.) So we cannot apply theorem 2.51 in order to claim that Fix.f is isomorphic to the powerset of its atoms.

Chapter 3

Regular Algebra

Regular algebra (sometimes also known as "Kleene algebra") is the algebra of three operators central to programming: composition, choice and iteration. As such, it is perhaps the most fundamental algebraic structure in computing science.

This chapter summarises fundamental properties of a regular algebra. Since the properties are typically well known, proofs are omitted. Some of the most important properties are given names for future reference

3.1 The Axioms

Algebraically, program composition is modelled by a monoid and choice by binary suprema in a lattice. Iteration is modelled by a particular form of fixed point.

Definition 3.1 A *monoid* is a triple $(\mathcal{A},\cdot,1)$, where $\mathcal{A}$ is a set, $\cdot$ is a binary operator and 1 is an element of $\mathcal{A}$, satisfying the properties:

(3.2) $1\cdot x = x = x\cdot 1$ for all $x\in\mathcal{A}$,

and

(3.3) $x\cdot(y\cdot z) = (x\cdot y)\cdot z$ for all $x,y,z\in\mathcal{A}$.

The element 1 is called the *unit* of the monoid, and the operator $\cdot$ is called the *product* operator.
□

(The raised dot used to denote a product operator throughout this chapter should not be confused with the non-raised dot used to denote function application. The only use of the non-raised dot in this chapter is in the expression "$\mathsf{red}.A$" introduced in (3.19).)

A monoid is such a simple algebraic structure that little can be said at this stage. (Perhaps one thing that can be said is that monoids are truly ubiquitous — but then a theory of everything is a theory of nothing!) Monoids only become interesting when combined with other algebraic structures.

Definition 3.4 (Regular Algebra) A *regular algebra* is a tuple $(\mathcal{A},\cdot,+,\leq,0,1)$ where

(a) $(\mathcal{A},\cdot,1)$ is a monoid,

(b) $(\mathcal{A},\leq,+,0)$ is a complete lattice with least element 0 and binary supremum operator $+$,

(c) for all $a{\in}\mathcal{A}$, the endofunctions $(a\cdot)$ and $(\cdot a)$ are both lower adjoints in Galois connections between $(\mathcal{A},\leq)$ and itself.

A regular algebra is said to be *universally distributive* if the underlying lattice (see (b)) is universally distributive.
□

Aside The assumption of a complete lattice means that all infima exist (as well as all suprema). However, discussions of regular algebra often ignore the existence of infima, and there is no standard notation for the infimum operator in a regular algebra or even the greatest element. In cases where infima are relevant, the choice of the "$+$" symbol for the binary supremum operator is unfortunate. Shortly, therefore, we switch to using set notation. **End of Aside**

Our definition of a regular algebra does not postulate the existence of a star operator. A universally distributive regular algebra is what Conway [Con71] calls a "standard Kleene algebra". (Instead of (c), Conway postulates a universal distributivity rule which, together with other axioms, is equivalent to (c).)

The upper adjoints of $(a\cdot)$ and $(\cdot a)$ are called the *factorisation* operators. Although these operators are important, we seldom use them directly; more often, we use only the fact that they exist.

Following Conway, we sometimes refer to the elements of the carrier set of a regular algebra as *events*.

3.2 Reflexive, Transitive Closure

In this section, properties of the "star" operator are briefly summarised. For more details on this section, see [Mat95]. The star operator models iteration.

There are several different definitions of the star operator in a regular algebra. Possibly the best known definition is

(3.5) $a^* = \langle \Sigma i : 0 \leq i : a^i \rangle$.

Another definition is

(3.6) $a^* = \langle \mu x :: 1 + a + x{\cdot}x \rangle$.

This definition states that a^* is the reflexive, transitive closure of a: specifically, the induction rule for least prefix points establishes that it is reflexive:

$$1 \leq \langle \mu x :: 1 + a + x{\cdot}x \rangle \ ,$$

it includes a:

$$a \leq \langle \mu x :: 1 + a + x{\cdot}x \rangle \ ,$$

and it is transitive:

$$\langle \mu x :: 1 + a + x{\cdot}x \rangle \cdot \langle \mu x :: 1 + a + x{\cdot}x \rangle \leq \langle \mu x :: 1 + a + x{\cdot}x \rangle \ .$$

It then follows that the star operator is a closure operator.

Two other commonly used definitions are in terms of left and right iteration. Specifically, left iteration is defined by

(3.7) $a^* = \langle \mu x :: 1 + a{\cdot}x \rangle$

and right iteration by

(3.8) $a^* = \langle \mu x :: 1 + x{\cdot}a \rangle$.

It is easily shown that all of these definitions are equivalent. Choosing one or other definition gives different induction rules; deciding which to use in specific circumstances requires some practice. We use all four different definitions at some stage below.

Note that the equivalence of (3.5) with, for example, (3.7) is proved using the universal distributivity of ($a\cdot$) over supremum (property (c) in definition 3.4). Othe axiomatisations of so-called “Kleene algebra” (for example, ones studied by Conway [Con71]) are oriented towards one particular application: the equality of regular expressions when interpreted as languages. To this end, they typically postulate properties of composition, choice and iteration but the properties of composition are not as strong as 3.4(c). It is not possible to prove this property with weaker axiom systems making them inadequate for reasoning about path problems in graphs.

Consequences of the above definitions are that

(3.9) $a^*{\cdot}b = \langle \mu x :: b + a{\cdot}x \rangle$

(which is most easily proved using (3.7), and

(3.10) $b{\cdot}a^* = \langle \mu x :: b + x{\cdot}a \rangle$

(which is most easily proved using (3.8)). An immediate corollary is that $a^*{\cdot}b$ and $b{\cdot}a^*$ are *fixed* points of the relevant functions:

(3.11) $b + a{\cdot}(a^*{\cdot}b) = a^*{\cdot}b$,

(3.12) $b + (b{\cdot}a^*){\cdot}a = b{\cdot}a^*$.

The *transitive closure* of a is denoted by a^+. Like the reflexive, transitive closure it has several equivalent definitions, the most commonly used being:

(3.13) $a^+ = \langle \Sigma i : 1 \leq i : a^i \rangle$

and

(3.14) $a^+ = \langle \mu x :: a + x{\cdot}x \rangle = \langle \mu x :: a + x{\cdot}a \rangle = \langle \mu x :: a + a{\cdot}x \rangle$.

Also like the reflexive, transitive closure, these different definitions give rise to different induction rules.

Other properties of the star operator are as follows:

(a) $a \cdot b^* \leq c^* \cdot a \Leftarrow a{\cdot}b \leq c{\cdot}a$

(b) $c^* \cdot a \leq a \cdot b^* \Leftarrow c{\cdot}a \leq a{\cdot}b$

(c) $a \cdot (b{\cdot}a)^* = (a{\cdot}b)^* \cdot a$

(d) $(a{+}b)^* = b^* \cdot (a \cdot b^*)^* = (b^* \cdot a)^* \cdot b^*$

Properties (a) and (b) are called *leapfrog* rules (because a “leapfrogs” from one side of a star term to the other). Both have the immediate corollary that * is monotonic (by taking a to be 1). Properties (c) and (d) are called the *mirror* rule and *star-decomposition* rule, respectively.

There are many other properties of the star operator that we use without further ado.

3.3 The Unique Extension Property

In the previous section we saw that $a{\cdot}b^*$ is the least solution of the equation

$$x{::}\ x = a + x{\cdot}b \ .$$

Here we consider its largest solution $\langle \nu x :: a + x{\cdot}b \rangle$. In particular, we do so for a lattice that is universally distributive, so that among other things $(y+)$ and $(+y)$ distribute over all infima and, hence, are upper adjoints. Then ν-fusion (the dual of μ-fusion) yields a simple proof of the following theorem.

Theorem 3.15 If $(y+)$ is an upper adjoint, then we have, for all a and b,

$$\langle \nu x :: a+x\cdot b\rangle \;=\; y+\langle \nu x :: x\cdot b\rangle \quad\Leftarrow\quad y = a+y\cdot b \quad .$$

□

As a consequence, in a universally distributive regular algebra, the *largest* solution of the equation $x{::}\; x = a+x\cdot b$ is the sum (i.e. supremum) of an arbitrary solution and the largest solution of the equation $x{::}\; x=x\cdot b$. Note that a special choice for y in theorem 3.15 is $y = a\cdot b^*$.

An immediate corollary of theorem 3.15 is that if $\langle \nu x :: x\cdot b\rangle = 0$, function $\langle x :: a+x\cdot b\rangle$ has a unique fixed point. The combination of this property and its converse is the rule we call the *unique extension property* (uep) of regular algebra.

Theorem 3.16 (The unique extension property (uep)) Suppose b is an element of a universally distributive regular algebra. Then

$$\langle \nu x :: x\cdot b\rangle = 0 \quad\equiv\quad \langle \forall x,a \;::\; x = a\cdot b^* \;\;\equiv\;\; x = a+x\cdot b\rangle \quad .$$

Proof Only-if is an immediate consequence of theorem 3.15. Specifically,

$$\begin{array}{ll}
 & \langle \nu x :: x\cdot b\rangle = 0 \\
\Rightarrow & \{ \quad \text{theorem 3.15} \quad \} \\
 & \langle \forall y,a \;::\; \langle \nu x :: a+x\cdot b\rangle \;=\; y \quad\Leftarrow\quad y = a+y\cdot b\rangle \\
= & \{ \quad y := a\cdot b^* \;\; ; \text{(3.12) with } a,b := b,a \quad \} \\
 & \qquad \langle \forall y,a \;::\; \langle \nu x :: a+x\cdot b\rangle \;=\; y \quad\Leftarrow\quad y = a+y\cdot b\rangle \\
 & \wedge \quad \langle \nu x :: a+x\cdot b\rangle \;=\; a\cdot b^* \\
\Rightarrow & \{ \quad \text{Leibniz (and dummy renaming)} \quad \} \\
 & \langle \forall x,a \;::\; a\cdot b^* = x \quad\Leftarrow\quad x = a+x\cdot b\rangle \\
= & \{ \quad \text{(3.12) with } a,b := b,a \quad \} \\
 & \langle \forall x,a \;::\; x = a\cdot b^* \;\;\equiv\;\; x = a+x\cdot b\rangle \quad .
\end{array}$$

The converse is straightforward:

$$\begin{array}{ll}
 & \langle \forall x,a \;::\; x = a\cdot b^* \;\;\equiv\;\; x = a+x\cdot b\rangle \\
\Rightarrow & \{ \quad a := 0 \text{, properties of } 0 \quad \} \\
 & \langle \forall x \;::\; x=0 \;\;\equiv\;\; x = x\cdot b\rangle \\
\Rightarrow & \{ \quad \text{by definition of } \nu, \quad \langle \nu x :: x\cdot b\rangle = \langle \nu x :: x\cdot b\rangle\cdot b \\
 & \qquad x := \langle \nu x :: x\cdot b\rangle \quad \} \\
 & \langle \nu x :: x\cdot b\rangle = 0 \quad .
\end{array}$$

□

Theorem 3.16 was postulated as an axiom of regular algebra in [Bac75, BC75]. Here, a proof is needed because the star operator is not a primitive but defined in terms of least fixed points.

The uep draws attention to the importance of property $\langle \nu x :: x \cdot b \rangle = 0$. In language theory, it is equivalent to $\varepsilon \notin b$ since if, on the contrary, x is a non-empty set such that $x = x \cdot b$, the length of the shortest word in x must be equal to the length of the shortest word in x plus the length of the shortest word in b . That is, the length of the shortest word in b is zero. The terminology that is often used is " b does not possess the *empty-word property*" . In relation algebra, we say " b is well-founded" : the property expresses that there are no infinite sequences of b -related elements (thus, if relation b represents a finite directed graph, $\langle \nu x :: x \cdot b \rangle = 0$ means that the graph is acyclic).

(We remarked earlier that other axiomatisations of regular algebra do not demand the existence of factorisation operators, making them inadequate for reasoning about path-finding problems. The requirement of universal distributivity is also commonly not made. However, the uep is a vital tool in the context of acyclic graphs and the omission of universal distributivity would render the theory inadequate for our purposes.)

In the context of relation algebra, there are several equivalent ways of defining well-foundedness, the one referred to above being perhaps less well known. This is discussed further in section 8.1.

3.4 Reflexive-Transitive Reduction

The reflexive-transitive reduction of a relation is an important concept. For example, it underlies the display of (small, finite) posets by means of a so-called Hasse diagram: the relation displayed in such a diagram is not the partial ordering but its reflexive-transitive reduction. The concept is important in other applications. For example, the basis of the Knuth-Morris-Pratt pattern matching algorithm (and its generalisations [KMP77, Wei73, AC75]) is the "factor graph" of a regular language defined by the pattern, and the "factor graph" is the reflexive-transitive reduction of Conway's "factor matrix" of the language [BL77, Bac16]. This section introduces the concept in this broader context; calculations update and expand on previously published work.

We assume that the algebra under consideration is complemented; we denote the complement operator by the prefix operator " $\neg$ ". (See theorem 2.51.)

Because the primary application is relation algebra, and because we want to make extensive use of the infimum operator, we now switch to using set notation: that is we use the symbol " $\subseteq$ " to denote the ordering, " $\cup$ " for the supremum operator and " $\cap$ "

for the infimum operator. Also, because it fits in with relation algebra, we use "I" for the unit.

It should not be supposed that relation algebra is the sole application of the results of this section. An important application is to a "matrix" (powerset) algebra. Such an algebra has carrier set the set of functions with source $A{\times}A$, for some A, and range a powerset algebra; the product operator is defined as is usual for matrices and the other operators are defined by a pointwise "lifting" of the operators of the powerset algebra. For more details see [Bac06].

Definition 3.17 (Starth Root) Suppose U is an event of a regular algebra. A *starth root* of U is any event V that satisfies $V^* = U^*$; it is *minimal* if no smaller event has this property. It is *least* if it is at most all starth roots. Formally, V is *a minimal starth root* of U if

$$V^* = U^* \;\wedge\; \langle \forall W : W \subseteq V \wedge W^* = U^* : W = V \rangle$$

and V is *the least starth root* of U if

$$V^* = U^* \;\wedge\; \langle \forall W : W^* = U^* : V \subseteq W \rangle \;\; .$$

□

Definition 3.18 and the lemmas and theorems that follow assume a complemented regular algebra. We use the notation $\neg U$ to denote the complement of event U. This should, of course, not be confused with the notation for the complement of a predicate: the context should make clear which is intended.

Definition 3.18 (Reflexive and Transitive Reduction) Let A and B be events in a complemented regular algebra with unit I. Then $A \cap \neg I$ is called the *reflexive reduction* of A and $B \cap \neg(B \cdot B^+)$ is called the *transitive reduction* of B. The transitive reduction of the reflexive reduction of A is called the *reflexive-transitive reduction* of A.

□

(Definitions 3.17 and 3.18 abstract from Brzozowski's theorem asserting the existence of a "unique irreducible generating set" of a "monoid with length" [Brz67, Theorem 2].)

We denote the reflexive-transitive reduction of A by $\mathsf{red}.A$. That is,

$$(3.19) \quad \mathsf{red}.A \;=\; A \cap \neg I \cap \neg((A \cap \neg I) \cdot (A \cap \neg I)^+) \;\; .$$

If G represents the edges of a graph, the reflexive-transitive reduction $\mathsf{red}.G$ "reduces" G by eliminating self-loops and edges connecting distinct nodes that are subsumed by paths of edge-length two or more and not involving self-loops. (Self-loops are edges from a node to itself. The multiple occurrences of "$\cap \neg I$" in (3.19) serve to eliminate such edges, leaving only edges connecting distinct nodes.)

A couple of lemmas on reflexive reduction prove useful later:

Lemma 3.20 Let X be an event in a complemented regular algebra with unit I. Then

$$X^* = (X\cap\neg I)^* \ .$$

(In words, the reflexive reduction of X is a starth root of X.)

Proof

$$\begin{array}{ll}
 & X^* = (X\cap\neg I)^* \\
\Leftarrow & \{ \quad X \supseteq X\cap\neg I\,,\ \text{monotonicity of star} \quad \} \\
 & X^* \subseteq (X\cap\neg I)^* \\
= & \{ \quad {}^*\ \text{is a closure operator} \quad \} \\
 & X \subseteq (X\cap\neg I)^* \\
\Leftarrow & \{ \quad I\cup Y\subseteq Y^*\ \text{with}\ Y := X\cap\neg I \quad \} \\
 & X \subseteq I\cup(X\cap\neg I) \\
= & \{ \quad \text{absorption rule} \quad \} \\
 & X \subseteq I\cup X \\
= & \{ \quad \text{set calculus} \quad \} \\
 & \text{true}\ .
\end{array}$$

□

Lemma 3.21 Let X and Y be events in a complemented regular algebra with unit I. Then

$$X^*\subseteq Y^* \ \equiv\ (X\cap\neg I)^+ \subseteq (Y\cap\neg I)^+ \ ,$$

$$X^* = Y^* \ \equiv\ (X\cap\neg I)^+ = (Y\cap\neg I)^+ \ .$$

Proof First,

$$\begin{array}{ll}
 & (X\cap\neg I)^+ \subseteq (Y\cap\neg I)^+ \\
= & \{ \quad {}^+\ \text{is a closure operator} \quad \} \\
 & X\cap\neg I \subseteq (Y\cap\neg I)^+ \\
= & \{ \quad \text{complements} \quad \} \\
 & X \subseteq (Y\cap\neg I)^+ \cup I \\
= & \{ \quad \text{for all}\ Z,\ Z^+\cup I = Z^*\ \text{with}\ Z := Y\cap\neg I,
\end{array}$$

$\qquad$ lemma 3.20 with $X{:=}Y$ $\}$

$X \subseteq Y^*$

$= \quad \{ \quad {}^*$ is a closure operator $\}$

$X^* \subseteq Y^*$.

The second property follows immediately from the anti-symmetry of set inclusion.
□

Theorem 3.22 (Least Starth Root) Let A be an event in a complemented regular algebra with unit I. Then

$$A^* = (\mathsf{red}.A)^* \;\Rightarrow\; \langle \forall X : X^* = A^* : \mathsf{red}.A \subseteq X \rangle \quad .$$

That is, if the reflexive-transitive reduction of A is a starth root of A, it is the least starth root of A.

Proof Assume that $A^* = (\mathsf{red}.A)^*$ and $X^* = A^*$. Let $B = A \cap \neg I$, $C = B \cap \neg(B \cdot B^+)$ and $Y = X \cap \neg I$. By applying lemma 3.20 and including the two assumptions, we have

$$A^* = B^* = C^* = X^* = Y^* \quad .$$

Next we note that

C

$= \quad \{ \quad$ definition of C and B $\}$

$A \cap \neg I \cap \neg(B \cdot B^+)$

$= \quad \{ \quad$ idempotency and symmetry of infimum $\}$

$(A \cap \neg I \cap \neg(B \cdot B^+)) \cap \neg I$

$= \quad \{ \quad$ definition of C and B $\}$

$C \cap \neg I$.

It follows that we can apply lemma 3.21 with $X,Y{:=}A,C$ and $X,Y{:=}C,X$ to deduce that

$$B^+ = C^+ = Y^+ \quad .$$

We can now proceed with the calculation.

$B \cap \neg(B \cdot B^+) \subseteq X$

$= \quad \{ \quad B \cap \neg(B \cdot B^+) = C = C \cap C^+ = C \cap Y^+ \quad \}$

$B \cap \neg(B \cdot B^+) \cap Y^+ \subseteq X$

$$
\begin{array}{ll}
= & \{ \quad \text{shunting rule (2.27)} \quad \} \\
& B \cap Y^+ \subseteq X \cup B \cdot B^+ \\
\Leftarrow & \{ \quad B \cap Y^+ \subseteq Y^+ \quad \} \\
& Y^+ \subseteq X \cup B \cdot B^+ \\
\Leftarrow & \{ \quad Y^+ = Y \cup Y \cdot Y^+ \quad \} \\
& Y \subseteq X \ \wedge \ Y \cdot Y^+ \subseteq B \cdot B^+ \\
= & \{ \quad Y = X \cap \neg I \quad \} \\
& Y \cdot Y^+ \subseteq B \cdot B^+ \\
= & \{ \quad [\ X \cdot X^+ = X^+ \cdot X^+ \ ] \text{ with } X{:=}B \\
& \quad \text{(well-known property, simple proof left to reader)} \quad \} \\
& Y^+ \cdot Y^+ \subseteq B^+ \cdot B^+ \\
= & \{ \quad B^+ = Y^+ : \text{ see above} \quad \} \\
& \mathsf{true} \ .
\end{array}
$$

□

Theorem 3.22 postulates a candidate for a least starth root. In some cases, the candidate is indeed a least starth root, as illustrated by example 3.23 below, but this is not always the case, as illustrated by example 3.24. (In the case of example 3.23, the "graph" is not infinite.) Fortunately, the candidate is indeed a starth root in the case relevant to the current discussion: when A is a finite acyclic graph.

Example 3.23 Consider the at-most relation on integers. This is normally denoted by the symbol "$\leq$" but it is more convenient here to use the symbol atmost. The at-most relation is, of course, reflexive and transitive. That is, $\mathsf{atmost} = \mathsf{atmost}^*$. Instantiating the variable A in theorem 3.22 with atmost, the relation B is the less-than relation. This is normally denoted by the symbol "$<$" but let us write less instead. The reader may easily verify that the relation $\mathsf{less} \cap \neg(\mathsf{less} \circ \mathsf{less}^+)$ is the predecessor relation, pred, given by, for all integers i and j,

$$i[\![\mathsf{pred}]\!]j \equiv i+1 = j \ .$$

The theorem states that, if the predecessor relation is a starth root of the at-most relation, then it is the least starth root of that relation. And, indeed, $\mathsf{pred}^* = \mathsf{atmost}$. So, we conclude that

$$\langle \forall R : R^* = \mathsf{atmost} : \mathsf{pred} \subseteq R \rangle \ .$$

□

Example 3.24 Suppose we consider the universal relation on the set $\{1,2,3\}$. Fig. 3.1(a) depicts the relation as a graph. Figs. 3.1(b) and (c) depict starth roots of the relation; they are both minimal but are distinct.

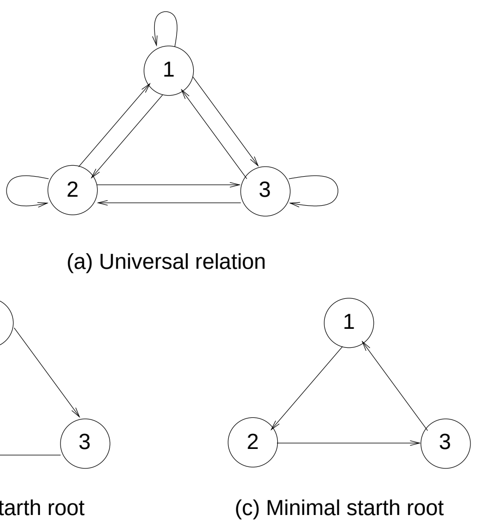

Figure 3.1: Distinct minimal starth roots of the universal relation

Denoting the universal relation on $\{1,2,3\}$ by $\top\!\!\top$ and the identity relation on $\{1,2,3\}$ by I, the relation $\top\!\!\top \cap \neg\mathrm{I} \cap \neg((\top\!\!\top \cap \neg\mathrm{I}) \circ (\top\!\!\top \cap \neg\mathrm{I})^{+})$ is the empty relation and the reflexive-transitive closure of the empty relation is the identity relation. Thus, it is not a starth root of the universal relation.
□

Example 3.25 The converse of theorem 3.22 is not valid since a relation may have a least starth root that is not its reflexive-transitive reduction. This is demonstrated by the following example.

Suppose R is the relation $\{(1,2),(2,1)\}$. Then R^{*} is the universal relation on $\{1,2\}$ and $\mathsf{red.R}$ is the empty relation. Thus, for all X, $\mathsf{red.R} \subseteq X$; however, $\mathsf{R}^{*} \neq (\mathsf{red.R})^{*}$. Indeed, the least starth root of R is R itself.
□

Example 3.26 The lexicographic ordering on words over a finite alphabet is well-founded. However, if the alphabet has at least two elements, it has no least starth root.

We can gain insight into why this is the case by considering a simpler case. Suppose we consider the alphabet $\{a,b\}$ and the set of words

$$\{k : 1 \leq k : a^{k}\} \cup \{k : 1 \leq k : b^{k}\} \quad .$$

That is, each word is either a string of as or a string of bs. Fig. 3.2(a) depicts the (reflexive reduction of the) lexicographic ordering on words in this set of length at most three. The transitive reduction of the latter relation is depicted in fig. 3.2(b). Note, in particular the diagonal edge from aaa to b.

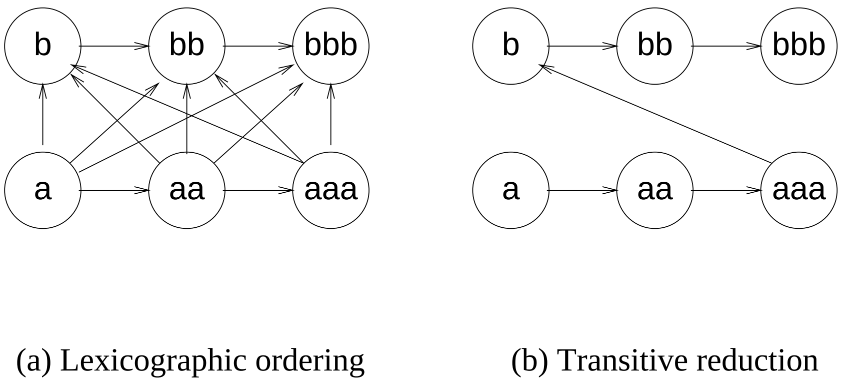

Figure 3.2: Subgraph of word order and its transitive reduction

Now imagine what happens when "three" is generalised to an arbitrary number and then consider what happens in the limit. The (reflexive-)transitive reduction of the lexicographic ordering on the infinite set of words relates a^k to a^{k+1} and b^k to b^{k+1} for each k but does not relate a^k to b^j for any values of j and k. It is thus not a starth root of the lexicographic ordering. Indeed, any starth root of the lexicographic ordering must relate a^k to b for an infinite number of values of k. But, given such a starth root, the removal of any one value of k is also a starth root. There is thus no least starth root.
□

Chapter 4

Relation Algebra

This chapter discusses the algebra of binary relations: *relation algebra* for short. Our axiomatisation is *point-free* as opposed to *pointwise*. A pointwise axiomatisation defines the operators of a relation algebra in terms of *Boolean* values xRy; the variables of the axiomatisation are thus relations, R, and *points*, x and y. This is the more conventional means of defining operators on relations. A point-free axiomatisation omits the points; the variables in the axiomatisation are exclusively relations.

The advantage of a point-free axiomatisation is increased concision. This is invaluable to the goal of establishing general properties of relations. A disadvantage is that when one comes to apply such general properties to particular relations, like the at-most relation, it is particular Boolean values, like $m \le n$, that are of interest. In addition to the point-free axioms we therefore need to give a pointwise *interpretation* of each of the operators. That is, we need to say, for each operator that we introduce, how the operator defines a set of pairs. Such an interpretation is often called a (set-theoretic) *model* of the axiom system. In giving the interpretation we use the notation $[\![E]\!]$ to mean "the interpretation of E". Thus we write $x[\![R]\!]y$ instead of xRy; this enhances readability and also emphasises the difference between the objects of an abstract relation algebra and the interpretation of such objects as binary relations. Note that the expression E is most often a relation, but is sometimes an ordering between relations.

A possible source of error is the interface between interpretation and the abstract algebra. That is, errors may be introduced either when formulating informal statements in the abstract algebra or, vice-versa, when interpreting expressions in the abstract algebra. It is impossible to avoid all such errors but, in order to minimise the risk, we formalise the process of interpreting point-free formulae in a way that narrows the gap between the formal and informal.

4.1 The Axioms

Relation algebra is a rich algebraic structure involving a large number of operators. There is a down-side as well as an up-side to its richness. On the one hand it is very expressive, on the other hand calculations within the algebra can be difficult because of the sheer abundance of calculational rules. In order to make the algebra more tractable we present it as a number of units with interfaces between the units. Each unit is a well-understood and well-documented mathematical structure of sufficiently small size to be easily comprehended.

The first unit in relation algebra is a lattice structure. Specifically, let $(\mathcal{A},\subseteq)$ be a partially-ordered set. We postulate that $\mathcal{A}$ forms a complete, universally distributive lattice. The infimum and supremum operators will be denoted by $\cap$ and $\cup$, respectively. The top and bottom elements of the lattice will be denoted by $\top\!\!\top$ and $\perp\!\!\perp$, respectively. We will call elements of $\mathcal{A}$ *relations* and denote them by variables R, S and T.

As suggested by the choice of notation, the interpretation of $\subseteq$ is the subset ordering, the interpretation of $\cap$ is set intersection, and the interpretation of $\cup$ is set union. Formally,

$$[\![R\subseteq S]\!] \equiv \langle\forall x{,}y : x[\![R]\!]y : x[\![S]\!]y\rangle \quad ,$$

$$x\,[\![R\cap S]\!]\,y \equiv x[\![R]\!]y \wedge x[\![S]\!]y \quad ,$$

and

$$x\,[\![R\cup S]\!]\,y \equiv x[\![R]\!]y \vee x[\![S]\!]y \quad .$$

This is the most complicated unit in the framework but one which should be familiar to the reader.

Every binary relation has a converse. At the point level the converse operator, denoted by a postfix "$^\cup$" symbol, is defined by

$$x\,[\![R^\cup]\!]\,y \;\equiv\; y[\![R]\!]x$$

for all x and y. At the point-free level we postulate the existence of a (total) unary function from relations to relations such that, for all relations R and S

$$(4.1)\qquad R^\cup\subseteq S \;\equiv\; R\subseteq S^\cup \quad .$$

The Galois connection (4.1) is all that is necessary to define the converse operator and its interface with the lattice structure. Its being a Galois connection makes it so attractive. Because the converse operator is its own upper and lower adjoint we can immediately infer that it is universally $\cap$-junctive (since it is its own upper adjoint) and universally

$\cup$-junctive (since it is its own lower adjoint). We most often use such distributivity properties in the case of finite suprema and infima. Specifically,

$$\top\!\!\top^{\cup} = \top\!\!\top \quad ,$$

and

$$\perp\!\!\perp^{\cup} = \perp\!\!\perp \quad ,$$

and, for all relations R and S,

$$(R{\cap}S)^{\cup} \;=\; R^{\cup}{\cap}S^{\cup} \quad \text{, and}$$

$$(R{\cup}S)^{\cup} \;=\; R^{\cup}{\cup}S^{\cup} \quad .$$

The fact that converse is its own upper and lower adjoint yields yet more. The two standard cancellation properties of Galois connections yield the inclusions $R\subseteq(R^{\cup})^{\cup}$ and $(R^{\cup})^{\cup}\subseteq R$ whence by anti-symmetry of the ordering relation we conclude

$$R=(R^{\cup})^{\cup} \quad .$$

Converse is thus a bijection from relations to relations that is its own inverse. Furthermore, it is a poset isomorphism; substituting $S^{\cup}$ for S in (4.1) and simplifying using $S=(S^{\cup})^{\cup}$ we have

$$R^{\cup}\subseteq S^{\cup} \;\equiv\; R\subseteq S \quad .$$

Finally, a property that often comes in handy is:

$$R=R^{\cup} \;\equiv\; R\subseteq R^{\cup} \quad .$$

The property is a trivial consequence of the defining Galois connection.

The set of binary relations over some universe includes the identity relation, I, defined at the point level by

$$x[\![I]\!]y \;\equiv\; x=y$$

for all x and y. Relations may also be composed via the binary composition operator, $\circ$, defined at the point level by

$$x\,[\![R{\circ}S]\!]\,z \;\equiv\; \langle\exists y :: x[\![R]\!]y \wedge y[\![S]\!]z\rangle \quad .$$

We capture these two notions in our algebraic framework by demanding the existence of a relation I and a binary operator, $\circ$, mapping a pair of relations to a relation, such that $(\mathcal{A},\circ,I)$ is a monoid. That is, composition is associative

$$(4.2) \qquad (R{\circ}S){\circ}T=R{\circ}(S{\circ}T) \quad ,$$

for all relations R, S and T, and I is a left and right unit of composition

(4.3) $\quad R{\circ}I = R = I{\circ}R \quad ,$

for all relations R.

There are two interfaces to be specified: that with the lattice structure and that with the converse operator. The interface with the converse operator is soon dealt with. Bearing in mind the intended relational interpretations of converse and composition we postulate

(4.4) $\quad (R{\circ}S)^{\cup} \;=\; S^{\cup}\circ R^{\cup} \quad ,$

for all relations R and S.

From (4.4), it is easy to deduce that

(4.5) $\quad I^{\cup} = I \quad .$

For the interface with the lattice structure we postulate that a relation algebra is a regular algebra. In particular, we postulate that for all relations R the functions $(R\circ)$ and $(\circ R)$ distribute universally over suprema.

By the fundamental theorem of Galois connections, this is equivalent to postulating the existence of two binary operators $\backslash$ and $/$ satisfying the properties

(4.6) $\quad R{\circ}S \subseteq T \;\equiv\; S \subseteq R\backslash T \quad ,$

and

(4.7) $\quad R{\circ}S \subseteq T \;\equiv\; R \subseteq T/S \quad .$

These two operators are called the *factoring*, or *division*, operators. We suggest that they be pronounced “under” and “over”, respectively.

The meaning of $R\backslash T$ expressed in terms of points can be recovered from (4.6) by instantiating S to the relation $\{(x,y)\}$. Formally, we have:

$$
\begin{array}{ll}
 & x\,[\![R\backslash T]\!]\,y \\
= & \{ \quad \text{definition} \quad \} \\
 & \{(x,y)\} \subseteq R\backslash T \\
= & \{ \quad (4.6) \quad \} \\
 & R\circ\{(x,y)\} \subseteq T \\
= & \{ \quad \text{interpretation of } \subseteq \quad \} \\
 & \langle \forall u,v : u\,[\![R\circ\{(x,y)\}]\!]\,v : u[\![T]\!]v \rangle
\end{array}
$$

$=\quad\{\quad$ interpretation of composition and the relation $\{(x,y)\}\quad\}$

$\quad\langle\forall\, u,v : \langle\exists w :: u[\![R]\!]w \wedge w = x \wedge v = y\rangle : u[\![T]\!]v\rangle$

$=\quad\{\quad$ one-point rule $\quad\}$

$\quad\langle\forall u : u[\![R]\!]x : u[\![T]\!]y\rangle\quad.$

That is,

$$x\,[\![R\backslash T]\!]\,y \;\equiv\; \langle\forall u{:}u[\![R]\!]x{:}u[\![T]\!]y\rangle\quad.$$

Similarly,

$$x\,[\![T/S]\!]\,y \;\equiv\; \langle\forall u{:}y[\![S]\!]u{:}x[\![T]\!]u\rangle\quad.$$

Just as the use of the composition operator avoids the use of existential quantifications, the use of the division operators avoids the use of universal quantifications in point-free reasoning.

4.1.1 Operator Precedence

We have now introduced quite a large number of operators. In order to reduce the number of parentheses in formulae we should agree on a precedence between the different operators.

A general rule we use throughout is that all prefix and postfix operators as well as subscripting and superscripting take precedence over infix operators and infix operators in turn take precedence over multifix operators. When both prefix and postfix operators are applied to an expression, we use parentheses to clarify the order of evaluation. Thus we only need to discuss the relative precedence of the infix operators.

For infix operators, the general rule is that metaoperators (operators like $\equiv$ and $\wedge$) have the lowest precedence. Next come relations like $\leq$ and $\subseteq$. The operators of relation algebra have the next highest precedence, and function application —when explicitly written as an infix operator— has the highest precedence of all.

Among the infix operators of relation algebra the precedence is: intersection and union have the same, lowest precedence, next is composition and the highest precedence is given to the division operators. Thus the expression $R \circ S\backslash T \cap U$ is parenthesised as $(R{\circ}(S\backslash T)){\cap}U$. (Note how white space is added in order to suggest the correct parsing.)

4.1.2 Modularity Rule and Cone Rule

We have postulated that composition distributes through suprema. We have *not* postulated that composition distributes through infima. Were we to do so then the binary relations would not form a model of our algebraic framework. The lack of such a law, however, poses severe problems. We know that, for each R, the function $(R\circ)$ is monotonic (since it is universally $\cup$-junctive) and hence

$$R\circ(S\cap T) \subseteq R\circ S \cap R\circ T \quad .$$

Thus we are in a position to reason with infima of compositions so long as they appear on the bigger side of an inclusion. But we have no means of working with such a term when it appears on the smaller side of an inclusion. Something more is needed to afford the manipulative freedom we need.

The rule we now introduce to overcome this difficulty acts as an interface between all three units of the framework. J. Riguet [Rig48] named the rule after the famous mathematician J.W.R. Dedekind (he called it "la relation de Dedekind") because of its resemblance to the modular identity, a property of normal subgroups discovered by Dedekind. Schmidt and Ströhlein [SS88, SS93] have adopted Riguet's terminology (they refer to "die Dedekind Formel", the Dedekind formula) whereas Freyd and Ščedrov [Fv90] call it the *law of modularity* (possibly for the same reason as Riguet). We call it the *modularity rule*.

The modularity rule is that, for all relations R, S and T,

(4.8) $\quad R\circ S\cap T \subseteq R\circ(S \cap R^{\cup}\circ T) \quad .$

At first sight, this is a difficult rule to appreciate and to use. A little analysis of its structure helps. Note that the term on the smaller side of the inclusion is an infimum of two terms and the term on the larger side is a composition of two terms. None of the rules given so far cater for either of these situations. Note also that R is the only repeated variable on the larger side. Viewing composition as a multiplication operator and infimum as addition, it is as if $R^{\cup}$ is the inverse of R, it being cancelled when R is multiplied through on the righthand side in order to obtain the lefthand side.

These hints may help the reader to understand and remember the rule. However, the best way to get to grips with it is to use it. Let's work through a few simple examples.

The easiest way to begin is to look for some obvious simplifications. Not all are interesting but some may prove to be.

One simplification is to eliminate the intersection operator on the right side. This we can do by the assignment $S{:=}\top\!\!\top$. We obtain

(4.9) $\quad R\circ\top\!\!\top\cap T \subseteq R\circ R^{\cup}\circ T \quad .$

This property has two interesting consequences. The right side can be simplified by instantiating T to I. We get

$$R\circ\top\!\!\top\cap I \subseteq R\circ R^{\cup} \quad .$$

Hence

$$R\circ\top\!\top\cap I \;\subseteq\; R\circ R^{\cup}\cap I \;\;.$$

But, by monotonicity, since $\top\!\top\supseteq R^{\cup}$ we have

$$R\circ\top\!\top\cap I \;\supseteq\; R\circ R^{\cup}\cap I \;\;.$$

We conclude

(4.10) $R\circ\top\!\top\cap I \;=\; R\circ R^{\cup}\cap I \;\;.$

Property (4.10) was obtained by choosing T so as to simplify the right side of (4.9). The second interesting consequence is obtained by choosing $T = R$ thus simplifying its left side. We obtain (since $R\circ\top\!\top\cap R = R$)

(4.11) $R\subseteq R\circ R^{\cup}\circ R \;\;.$

As a final, preliminary, example of the use of the modularity rule let us see what it predicts about the distribution of composition of cap. We have

$$\begin{array}{ll}
 & (R\circ S)\cap(R\circ T) = R\circ(S\cap T) \\
= & \{ \quad (R\circ S)\cap(R\circ T)\supseteq R\circ(S\cap T) \quad \} \\
 & (R\circ S)\cap(R\circ T)\subseteq R\circ(S\cap T) \\
\Leftarrow & \{ \quad \text{modularity rule: (4.8)} \\
 & \quad\;\; \text{with } R,S,T := R\,,S\,,R\circ T \quad \} \\
 & R\circ(S\cap R^{\cup}\circ R\circ T)\subseteq R\circ(S\cap T) \\
\Leftarrow & \{ \quad \text{monotonicity of composition} \quad \} \\
 & S\cap R^{\cup}\circ R\circ T \subseteq S\cap T \\
\Leftarrow & \{ \quad \text{monotonicity of } (S\cap) \quad \} \\
 & R^{\cup}\circ R\circ T\subseteq T \;\;.
\end{array}$$

By symmetry, S and T may be interchanged everywhere. So we conclude:

(4.12) $(R\circ S)\cap(R\circ T) = R\circ(S\cap T) \;\Leftarrow\; R^{\cup}\circ R\circ T\subseteq T \,\vee\, R^{\cup}\circ R\circ S\subseteq S \;\;.$

Because converse is a lattice isomorphism, all rules we obtain have a dual constructed by turning compositions around. The modularity rule itself has the dual form

(4.13) $S\circ R\cap T\subseteq (S\cap T\circ R^{\cup})\circ R \;\;.$

and the rules (4.10) and (4.12) have the duals

(4.14) $\top\!\top\circ R\cap I \;=\; R^{\cup}\circ R\cap I$.

and

(4.15) $(S\circ R)\cap(T\circ R) = (S\cap T)\circ R \;\Leftarrow\; T\circ R\circ R^{\cup}\subseteq T \vee S\circ R\circ R^{\cup}\subseteq S$.

(Property (4.11) is self-dual.) The reader is invited check these claims for themself. In the future, we sometimes make claims of the form "the converse-dual of x is y".

The rule sometimes called "Tarski's rule" is called the "cone rule" below: for all relations R,

(4.16) $\top\!\top\circ R\circ\top\!\top = \top\!\top \;\vee\; R = \perp\!\!\perp$.

The cone rule expresses the universality of the relation $\top\!\top$. Its significance becomes evident in section 5.2 where it is used in combination with the "all or nothing" rule to model reasoning about relations as sets of pairs.

The set of homogeneous binary relations on the empty set is, of course, the carrier set of a relation algebra. The empty relation, the identity relation and the universal relation are all equal and so the algebra is completely trivial. In order to exclude this model, the cone rule is sometimes reformulated as an exclusive-or rather than an inclusive-or (a disjunction). The rule is then, for all R:

$$R = \perp\!\!\perp \;\not\equiv\; \top\!\top\circ R\circ\top\!\top = \top\!\top \;.$$

(Equivalently,

$$R \neq \perp\!\!\perp \;\equiv\; \top\!\top\circ R\circ\top\!\top = \top\!\top \;.)$$

The reader can easily check that this is equivalent to the conjunction of the standard cone rule and $\perp\!\!\perp \neq \top\!\top$. Not excluding the trivial model becomes vital when the rule is extended to heterogeneous relations. See section 5.4.

Axiom systems for relation algebra often include a complementation (negation) operator and, instead of the modularity rule, the so-called Schröder rule is postulated. Our formulation of Schröder's rule is slightly different from standard accounts, as we now explain.

Suppose we consider an algebra that obeys all the axioms of relation algebra except for the modular identity. Suppose that the algebra is complemented (i.e. every relation has a complement). Consider the rule:

(4.17) $R^{\cup} = \neg I/\neg R$,

and the *middle-exchange rule*:

(4.18) $\mathsf{R}\circ\neg\mathsf{X}\circ\mathsf{S} \subseteq \neg\mathsf{Y} \;\equiv\; \mathsf{R}^{\cup}\circ\mathsf{Y}\circ\mathsf{S}^{\cup} \subseteq \mathsf{X}$.

The rules (4.17) *and* (4.18) are both equivalent to the modularity rule.

One way of proving the equivalence —left to the reader— is to show that (4.17) implies (4.18), that (4.18) implies the modularity rule and that the modularity rule implies (4.17). A step on the way is to prove the *divergence rule*,

(4.19) $\mathsf{R}\circ\mathsf{S} \subseteq \neg\mathsf{I} \;\equiv\; \mathsf{S}\circ\mathsf{R} \subseteq \neg\mathsf{I}$,

and the *not-* R *-verse* rule:

(4.20) $\neg(\mathsf{R}^{\cup}) = (\neg\mathsf{R})^{\cup}$.

The middle-exchange rule gets its name from the fact that the middle term in a composition is exchanged with the right side of an inclusion. It has an attractive, symmetric form, making it easy to remember in spite of having four free variables. The divergence rule gets its name from the interpretation of $\neg\mathsf{I}$: the relation that holds between two values if and only if they "diverge" from each other, i.e. are unequal. The name "not- R -verse" rule is borrowed from the way the famous mathematician Augustus de Morgan denoted the combination of complementation and converse. He literally wrote "not- R -verse" pointing out that it didn't matter whether one read this as (not R) converse or not (R converse). In our notation we would write $\neg\mathsf{R}^{\cup}$ and (deliberately) omit specifying a precedence of one operator over the other. This justifies the combination of a prefix operator for negation and a postfix operator for converse. In general, combining prefix operators with postfix operators is *not* to be recommended since, if the operand (R above) is any other than a variable or constant, it is extremely difficult to parse the formulae. Even so, we don't follow this recommendation and stick to standard notation — with the consequence that we have just warned about!

Instead of the middle-exchange rule, many publications state two rules, each with three variables, due to Schröder. The rules are equivalent to the conjunction of the two equivalences: for all R, S and T,

(4.21) $\mathsf{R}\circ\mathsf{S} \subseteq \neg\mathsf{T}^{\cup} \;\equiv\; \mathsf{S}\circ\mathsf{T} \subseteq \neg\mathsf{R}^{\cup}$

and

(4.22) $\mathsf{R}\circ\mathsf{S} \subseteq \mathsf{T} \;\equiv\; \mathsf{T}\circ\mathsf{R} \subseteq \neg\mathsf{S}^{\cup}$.

We call these rules the *rotation rules* (because of the way the variables are rotated).

4.2 Summary

This concludes our discussion of the algebraic framework. In a few sentences, a relation algebra is a complete, universally distributive lattice on which is defined a monoid structure and a unary converse operator. Composition on the left and on the right both have upper adjoints, the division operators. Converse is a lattice isomorphism that preserves the unit of composition and distributes contravariantly through composition. Finally, the lattice structure, converse and the monoid structure are all interrelated via the modularity and cone rules.

Chapter 5

Coreflexives, Heterogeneous Relations and Functions

When one writes a computer program there are many important details, mostly to do with efficiency, that play a major rôle. Ignoring all these details, the most primitive description that we can give of a sequential program is that it is a binary relation on the so-called "state space". (The state space of a program is the set of all values that can be assumed by the program variables.)

According to this view of programs, a programming language is a mechanism for describing and structuring binary relations that can be implemented: that is, descriptions of binary relations to which an "operational semantics" can be given detailing how the description can be interpreted as instructions controlling the execution of a machine.

Programming languages are normally so constrained that they *only* describe the relations that are implementable but, in order to support program construction, it is vital that an algebra be able to express relations that are not necessarily implementable or directly implementable. The notion of a "guard" on a guarded statement is an example. A guard acts as a filter on the domain of execution of a statement. Operationally it can be viewed as a partial skip. Mathematically, a guard is just a device that enables sets —subsets of the state space— to be incorporated into program statements.

In the relation calculus there are several mechanisms for viewing sets as relations, and thus modelling guards, each of which has its own merits. One is via "conditions" and another is via "coreflexives"[1]. Axiomatically these have the following definitions. First, we say that relation R is a *coreflexive* if and only if $R \subseteq I$. Second, we say that relation R is a *right condition* if and only if $R = \top\!\!\top \circ R$. Finally, we say that R is a *left condition* if and only if $R = R \circ \top\!\!\top$.

[1] "Coreflexives" are also called "monotypes" [ABH$^+$92, BW93, DBvdW97] or "tests" [Glü17], depending on the intended interpretation; the name "partial identity" is also used (eg. [Voe99]). We now prefer the application-neutral terminology used by Freyd and Ščedrov [Fv90].

In the relational model, we assume, for example, that the universe $\mathcal{U}$ contains two unequal values true and false. The *coreflexive* representation of the set boolean is then defined to be the relation

$$\{(\mathsf{true},\mathsf{true}),(\mathsf{false},\mathsf{false})\} \quad .$$

The *right condition* representation of the set boolean is the relation

$$\{x{:}\,x{\in}\mathcal{U}{:}\,(x,\mathsf{true})\} \cup \{x{:}\,x{\in}\mathcal{U}{:}\,(x,\mathsf{false})\}$$

It is clear that for any given universe $\mathcal{U}$ there is a one-to-one correspondence between the subsets of $\mathcal{U}$ and the coreflexives. Specifically, the set A is represented by the coreflexive p where $x[\![p]\!]y \equiv x{=}y \wedge y{\in}A$. Equally clear is the existence of a one-to-one correspondence between the subsets of $\mathcal{U}$ and the right conditions on $\mathcal{U}$. That is, if A is some set then the right condition defined by A is that relation A_r such that for all x and y, $x[\![A_r]\!]y \equiv y{\in}A$. Similarly, the left condition corresponding to A is that relation A_l such that for all x and y, $x[\![A_l]\!]y \equiv x{\in}A$.

Using coreflexives to represent subsets of $\mathcal{U}$ as relations, a guard on a relation is modelled by composition of the relation, either on the left or on the right, with such a coreflexive. Thus, if R and S are relations and p is a coreflexive then $p{\circ}R$ and $S{\circ}p$ are both relations, the first being relation R after restricting elements in its left domain to those in p and the second being the relation S after restricting elements in its right domain to those in p. Using conditions, a guard on the left domain of relation R is modelled by the intersection of R with a left condition, and a guard on the right domain of R by its intersection with a right condition. In principle, this poses a dilemma on the choice of representation of sets in the relation calculus. Should one choose coreflexives or conditions?

We choose coreflexives, there being several reasons for doing so. One is the simple fact that guarding both on the left and on the right of a relation is accomplished in one go with coreflexives. Moreover, coreflexives have simple and convenient properties. Specifically, for all coreflexives p and q

$$p = I{\cap}p = p^{\cup} = p{\circ}p$$

and

$$p{\circ}q = q{\circ}p = p{\cap}q \quad .$$

The most compelling reason, however, for choosing to represent sets by coreflexives is the dominant position occupied by composition among programming primitives. Introducing a guard in the middle of a sequential composition of relations is a frequent activity that is easy to express in terms of coreflexives but clumsy to express with conditions.

Nevertheless, conditions do have their place from time to time. They too have attractive calculational properties. In particular, they form a sublattice of the lattice of relations (that is they are closed under union and intersection) and —unlike the coreflexives— they are closed under negation. However, from the above it is clear that there is a one-to-one correspondence between coreflexives and both types of conditions which we document formally below. Exploitation of this correspondence is central to many calculations in the relation calculus. See [DBvdW97] for detailed examples.

Distributivity properties are used extensively in our calculations. In relation algebra, composition does not distribute through intersection — in general. In specific cases it does. One such case is composition with a coreflexive. Specifically, for all coreflexives p and all relations R and S,

(5.1) $\quad p\circ(R\cap S) \;=\; p\circ R\cap S \;=\; p\circ R\cap p\circ S \quad .$

The first equality is proved as follows.

$$\begin{array}{ll}
 & p\circ(R\cap S) = p\circ R\cap S \\
= & \{\quad \text{anti-symmetry} \quad\} \\
 & p\circ(R\cap S) \subseteq p\circ R\cap S \;\wedge\; p\circ(R\cap S) \supseteq p\circ R\cap S \\
= & \{\quad \text{1st conjunct: distributivity, } p\subseteq I \text{ and monotonicity} \quad\} \\
 & p\circ(R\cap S) \supseteq p\circ R\cap S \\
\Leftarrow & \{\quad \text{modularity rule: (4.8)} \quad\} \\
 & p\circ(R\cap S) \;\supseteq\; p\circ(R \cap p^{\cup}\circ S) \\
= & \{\quad p\subseteq I\text{, monotonicity} \quad\} \\
 & \text{true} \quad .
\end{array}$$

Now, for the second equality, we apply the first equality:

$$\begin{array}{ll}
 & p\circ R\cap S \\
= & \{\quad p = p\circ p \quad\} \\
 & p\circ p\circ R\cap S \\
= & \{\quad [\; p\circ(R\cap S) = p\circ R\cap S \;] \text{ with } R{:=}p\circ R \quad\} \\
 & p\circ(p\circ R\cap S) \\
= & \{\quad \text{symmetry of intersection} \quad\} \\
 & p\circ(S\cap p\circ R) \\
= & \{\quad [\; p\circ(R\cap S) = p\circ R\cap S \;] \text{ with } R,S := S\,,p\circ R
\end{array}$$

$\qquad\qquad$ symmetry of intersection $\quad\}$

$\quad p{\circ}R \cap p{\circ}S \quad .$

5.1 The Domain Operators

In this section, we introduce two operators mapping relations to coreflexives, the so-called *domain operators*. They play an extremely important rôle in the theory to follow.

We call the two operators the *left-domain operator* and the *right-domain operator*. We might have chosen to call one of them the "domain operator" and the other the "range operator", but this would have introduced an unwelcome direction in the interpretation of relations. (One of the elements in a pair satisfying a given relation would have to be designated the input and the other the output.) We prefer to make no commitment about the "direction" of a relation for as long as possible. The left- and right-domain operators are denoted by the postfix symbols " $<$ " and " $>$ ", respectively.

Definition 5.2 (Right Domain) The *right domain* of a relation R is the coreflexive denoted by $R{>}$ and defined by

$$R{>} = I \cap \top\!\!\top{\circ}R \quad .$$

Dually, the *left domain* of a relation R is the coreflexive denoted by $R{<}$ and defined by

$$R{<} = I \cap R{\circ}\top\!\!\top \quad .$$

□

We restrict our attention here to the right-domain operator. The reader is requested to dualise the results to the left-domain operator.

The intended interpretation of $R{>}$ (read R "right") for relation R is $\{x \mid \langle \exists y :: y[\![R]\!]x \rangle\}$. Two ways we can reformulate this requirement without recourse to points are formulated in the following theorem.

Theorem 5.3 (Right Domain) For all relations R and coreflexives p,

(5.4) $\quad R{>} \subseteq p \;\equiv\; R \subseteq \top\!\!\top{\circ}p \quad$ and

(5.5) $\quad R{>} \subseteq p \;\equiv\; R = R{\circ}p \quad .$

□

The characterisations (5.4) and (5.5) predict a number of useful calculational properties of the right domain operator. Some are immediate, some involve a little bit of work for their verification. Immediate from (5.4) —a Galois connection— is that the right domain operator is universally $\cup$-junctive, and $(\top\!\!\top\circ)$ is universally distributive over infima of coreflexives. In particular,

$$\top\!\!\top\circ(p\cap q) = (\top\!\!\top\circ p)\cap(\top\!\!\top\circ q) \quad ,$$

$$(R\cup S)_{>} = R_{>}\cup S_{>} \quad ,$$

and

$$\perp\!\!\perp_{>} = \perp\!\!\perp \quad .$$

The last of these can in fact be strengthened to

$$(5.6) \qquad R_{>} = \perp\!\!\perp \;\equiv\; R = \perp\!\!\perp \quad .$$

The proof is straightforward: use (5.4) in combination with $\top\!\!\top\circ\perp\!\!\perp = \perp\!\!\perp$.

From (5.4) we may also deduce a number of cancellation properties. But, in combination with the modularity rule, the cancellation properties can be strengthened. We leave their proofs together with a couple of other interesting applications of Galois connections as exercises.

Theorem 5.7 For all relations R, S and T

(a) $\top\!\!\top\circ R_{>} = \top\!\!\top\circ R$,

(b) $R\cap S\circ\top\!\!\top\circ T \;=\; S_{<}\circ R\circ T_{>}$,

(c) $(R^{\cup})_{>} = R_{<}$,

(d) $(R\cap S\circ T)_{>} = (S^{\cup}\circ R\cap T)_{>}$,

(e) $(R\circ\top\!\!\top\circ S)_{>} = S_{>} \Leftarrow R\neq\perp\!\!\perp$.

□

We complete this section by documenting the isomorphism between coreflexives and conditions. Recall that the right conditions are, by definition, the fixed points of the function $(\top\!\!\top\circ)$.

Theorem 5.8 The coreflexives are the fixed points of the right domain operator. That is, for all R,

(a) $R = R_{>} \equiv R \subseteq I$.

Also, for all coreflexives p and all right conditions C,

(b) $(\top\!\!\top \circ p)_{>} = p$, and
(c) $\top\!\!\top \circ C_{>} = C$.

Moreover, for all relations R and S,

(d) $R_{>} \subseteq S_{>} \equiv \top\!\!\top \circ R \subseteq \top\!\!\top \circ S$.

Hence,

(e) $R_{>} = S_{>} \equiv \top\!\!\top \circ R = \top\!\!\top \circ S$.

The right-domain operator is thus a poset isomorphism mapping the set of right conditions to the set of coreflexives and its inverse is the function $(\top\!\!\top \circ)$.
□

Some powerful and far from obvious theorems about coreflexives are proved by mapping the theorems to statements about conditionals and then exploiting the characteristic properties of $\top\!\!\top$ — $\top\!\!\top \supseteq R$ for all R, and $\top\!\!\top = \top\!\!\top^{\cup}$ — to prove these statements. An illustration of the technique is afforded by the proof of the following lemma.

(5.9) $(R \circ S)_{>} = (R_{>} \circ S)_{>}$.

We begin the proof by invoking theorem 5.8

$$
\begin{array}{ll}
 & (R \circ S)_{>} = (R_{>} \circ S)_{>} \\
= & \{ \quad \text{theorem 5.8(e)} \quad \} \\
 & \top\!\!\top \circ R \circ S = \top\!\!\top \circ R_{>} \circ S \\
= & \{ \quad \top\!\!\top \circ R_{>} = \top\!\!\top \circ R \quad \} \\
 & \top\!\!\top \circ R \circ S = \top\!\!\top \circ R \circ S \\
= & \{ \quad \text{reflexivity} \quad \} \\
 & \text{true} \ .
\end{array}
$$

Another useful property is:

(5.10) $X = \perp\!\!\perp \equiv X_{>} = \perp\!\!\perp$.

The proof is by mutual implication. First,

$$X = \perp\!\!\perp \ \Rightarrow \{\text{Leibniz}\} \quad X_{>} = \perp\!\!\perp_{>} \ \Rightarrow \{\perp\!\!\perp_{>} = \perp\!\!\perp\} \quad X_{>} = \perp\!\!\perp \ .$$

Second,

$$X_{>} = \perp\!\!\perp$$

$= \quad \{ \quad \perp\!\!\perp$ is least relation $\quad \}$

$$X_{>} \subseteq \perp\!\!\perp$$

$= \quad \{ \quad$ theorem 5.3 $\quad \}$

$$I \cap \top\!\!\top \circ X \subseteq \perp\!\!\perp$$

$\Rightarrow \quad \{ \quad$ monotonicity of composition,

preparing for use of the modularity rule $\quad \}$

$$(I \cap X \circ \top\!\!\top) \circ \top\!\!\top \subseteq \perp\!\!\perp$$

$\Rightarrow \quad \{ \quad$ modularity rule: (4.8), $\top\!\!\top = \top\!\!\top^{\cup}$ $\quad \}$

$$\top\!\!\top \cap X \subseteq \perp\!\!\perp$$

$= \quad \{ \quad \top\!\!\top$ is greatest relation, $\perp\!\!\perp$ is least relation $\quad \}$

$$X = \perp\!\!\perp \quad .$$

For modelling programming statements, in particular conditionals, *complemented domains* are necessary. We assume that the lattice of coreflexives is complemented and let $R_{>\bullet}$ denote the complement of $R_{>}$. That is,

$$R_{>} \cup R_{>\bullet} = I \qquad \text{and} \qquad R_{>} \cap R_{>\bullet} = \perp\!\!\perp \quad .$$

Then, for relations R and coreflexives p,

(5.11) $\quad R_{>\bullet} \supseteq p \;\equiv\; R \circ p = \perp\!\!\perp \quad .$

Moreover, for all R,

(5.12) $\quad (R_{>})_{>\bullet} = R_{>\bullet} = (R_{>\bullet})_{>} \quad .$

Note that (5.11) is a slightly disguised Galois connection since the right side can be rewritten as $R \subseteq \perp\!\!\perp / p$. (See (4.7).) The equation defines $R_{>\bullet}$ as the largest coreflexive p such that restricting the right domain of R to p yields the empty relation. A consequence is the distributivity property

$$(R \cup S)_{>\bullet} \;=\; R_{>\bullet} \cap S_{>\bullet} \quad .$$

Just as for the non-complemented domain operator, it is difficult to simplify $(R \cap S)_{>\bullet}$.

5.2 Points and Extensionality

In this section, our goal is to capture the notion that a relation is a set with elements pairs of points. We begin with the definition of a "point"[2] and then postulate an "extensionality" axiom similar to the notion of saturation discussed in section 2.6.

Definition 5.13 (Point) A homogeneous relation a of type A is a *point* iff it has the following three properties.

(a) $a \neq \perp\!\!\perp$,

(b) $a \subseteq I$, and

(c) $a = a \circ \top\!\!\top \circ a$.

In words, a point is a proper, coreflexive rectangle.
□

As in definition 5.13, we use lower case letters a, b and c to denote points.

Lemma 5.14 A point is an atom. That is, if a is a point then, for all b,

$$b \subseteq a \;\equiv\; b = \perp\!\!\perp \;\vee\; b = a \ .$$

Proof Suppose a is a point and b is a relation of the same type as a. The proof that a is an atom is by mutual implication. "If" is straightforward. For "only if", assume that $b \subseteq a$ and $b \neq \perp\!\!\perp$. We have to prove that $b = a$. This we do as follows.

$$
\begin{array}{cl}
 & b \\
= & \{ \quad \text{assumptions: } b \subseteq a \subseteq I \text{, property of coreflexives} \quad \} \\
 & a \circ b \circ a \\
= & \{ \quad \text{assumption: } a = a \circ \top\!\!\top \circ a \quad \} \\
 & a \circ \top\!\!\top \circ a \circ b \circ a \circ \top\!\!\top \circ a \\
= & \{ \quad \text{assumptions: } b \subseteq a \subseteq I \text{, property of coreflexives} \quad \} \\
 & a \circ \top\!\!\top \circ b \circ \top\!\!\top \circ a \\
= & \{ \quad \text{assumption: } b \neq \perp\!\!\perp \text{; cone rule} \quad \} \\
 & a \circ \top\!\!\top \circ a \\
= & \{ \quad \text{assumption: } a = a \circ \top\!\!\top \circ a \quad \} \\
 & a \ .
\end{array}
$$

[2]The definitions and lemmas in this section are due to Ed Voermans.

□

The following property was introduced by [Glü17]. Pairs (a,b) in classical formulations of relations are captured in our system by events of the form $a{\circ}\top\!\!\top{\circ}b$ where a and b are points (proper atomic coreflexives). The catchy name given to the lemma expresses the property that membership of a relation is a boolean.

Lemma 5.15 (All or Nothing) Suppose a and b are points. Then

$$\langle\forall R :: a{\circ}R{\circ}b = \perp\!\!\perp \vee a{\circ}R{\circ}b = a{\circ}\top\!\!\top{\circ}b\rangle \quad .$$

Proof Suppose $a{\circ}R{\circ}b \neq \perp\!\!\perp$. We have to prove that $a{\circ}R{\circ}b = a{\circ}\top\!\!\top{\circ}b$.

$$\begin{array}{ll} & a{\circ}R{\circ}b \\ = & \{ \quad \text{assumptions: } a = a{\circ}\top\!\!\top{\circ}a \text{ and } b = b{\circ}\top\!\!\top{\circ}b \quad \} \\ & a{\circ}\top\!\!\top{\circ}a{\circ}R{\circ}b{\circ}\top\!\!\top{\circ}b \\ = & \{ \quad \text{assumption: } a{\circ}R{\circ}b \neq \perp\!\!\perp \text{, cone rule} \quad \} \\ & a{\circ}\top\!\!\top{\circ}b \quad . \end{array}$$

□

In general, if a is a point of type A and b is a point of type B, the relation $a{\circ}\top\!\!\top{\circ}b$ represents the pair (a,b); given a relation R of type $A{\sim}B$ and points a and b of type A and B, respectively, the statement

$$a{\circ}\top\!\!\top{\circ}b \subseteq R$$

has the interpretation that the pair a and b are related by R. Specifically, for all relations R and points a and b of appropriate type,

$$(5.16)\quad (a{\circ}R{\circ}b \neq \perp\!\!\perp) \;=\; (a{\circ}\top\!\!\top{\circ}b \subseteq R) \;=\; (a{\circ}\top\!\!\top{\circ}b = a{\circ}R{\circ}b) \quad .$$

(In conformance with long-standing mathematical practice, property (5.16) should be read conjunctionally: that is as the equality of three terms. In this case, each term is boolean. The property is a straightforward corollary of the all-or-nothing rule.)

The following lemma motivates the all-or-nothing rule. That "pairs" $a{\circ}\top\!\!\top{\circ}b$ are atoms is equivalent to the all-or-nothing rule.

Lemma 5.17 Suppose a and b are atomic coreflexives. Then

$$\langle\forall R :: a{\circ}R{\circ}b = \perp\!\!\perp \vee a{\circ}R{\circ}b = a{\circ}\top\!\!\top{\circ}b\rangle \;\equiv\; \mathit{atomic}.(a{\circ}\top\!\!\top{\circ}b) \quad .$$

Proof Suppose p and q are coreflexives. Then, for all R,

$$
\begin{array}{ll}
 & R \subseteq p\circ\top\!\top\circ q \\
= & \{ \quad \text{set theory} \quad \} \\
 & R \;=\; R \cap p\circ\top\!\top\circ q \\
= & \{ \quad \text{domains (specifically theorem 5.7(b)),} \\
 & \quad\; p \text{ and } q \text{ are coreflexives, so } p = p_{<} \text{ and } q = q_{>} \quad \} \\
 & R \;=\; p\circ R\circ q \quad .
\end{array}
$$

We conclude that

$$R \subseteq p\circ\top\!\top\circ q \;\;\equiv\;\; R = p\circ R\circ q \quad .$$

We shall only need to apply this property in the case that p and q are atomic coreflexives. Now, assume a and b are atomic coreflexives and $a\circ R\circ b = \bot\!\!\bot \vee a\circ R\circ b = a\circ\top\!\top\circ b$. Then

$$
\begin{array}{ll}
 & R \subseteq a\circ\top\!\top\circ b \\
= & \{ \quad \text{above with } p{,}q := a{,}b \quad \} \\
 & R = a\circ R\circ b \\
\Rightarrow & \{ \quad \text{assumption and Leibniz} \quad \} \\
 & R = \bot\!\!\bot \vee R = a\circ\top\!\top\circ b \quad .
\end{array}
$$

We conclude, by definition 2.49 of atomic,

$$\langle \forall R :: a\circ R\circ b = \bot\!\!\bot \vee a\circ R\circ b = a\circ\top\!\top\circ b \rangle \;\Rightarrow\; \mathit{atomic}.(a\circ\top\!\top\circ b) \quad .$$

In words, if the all-or-nothing rule is universally valid for atomic coreflexives a and b, then $a\circ\top\!\top\circ b$ is atomic. Now, suppose $a\circ\top\!\top\circ b$ is an atom. Then, for all R,

$$
\begin{array}{ll}
 & a\circ R\circ b \subseteq a\circ\top\!\top\circ b \\
= & \{ \quad a\circ\top\!\top\circ b \text{ is an atom, definition 2.49} \quad \} \\
 & a\circ R\circ b = \bot\!\!\bot \;\vee\; a\circ\top\!\top\circ b = a\circ R\circ b \quad .
\end{array}
$$

That is, if $a\circ\top\!\top\circ b$ is an atom, the all-or-nothing rule applies to $a\circ R\circ b$, for all R.
□

Combining lemmas 5.15 and 5.17, we get:

Lemma 5.18 For all points a and b, $a\circ\top\!\top\circ b$ is atomic.

□

Lemma 5.19 For all proper coreflexives p and q,

$$p{\circ}\top\!\!\top{\circ}q \subseteq I \Rightarrow p=q \ .$$

Proof

$$
\begin{array}{ll}
 & p{\circ}\top\!\!\top{\circ}q \subseteq I \\
\Rightarrow & \{ \quad \text{monotonicity and unit of composition} \quad \} \\
 & p{\circ}p{\circ}\top\!\!\top{\circ}q \subseteq p \quad \wedge \quad p{\circ}\top\!\!\top{\circ}q{\circ}q \subseteq q \\
\Rightarrow & \{ \quad p \text{ is coreflexive, so } p=p{\circ}p=p{>}\text{, similarly for } q \\
 & \quad\ \ \text{monotonicity and domains} \quad \} \\
 & (p{\circ}\top\!\!\top{\circ}q){>} \subseteq p \quad \wedge \quad (p{\circ}\top\!\!\top{\circ}q){<} \subseteq q \\
= & \{ \quad \text{domains (specifically theorem 5.7(e)),} \\
 & \quad\ \ p \text{ and } q \text{ are non-empty coreflexives} \quad \} \\
 & q\subseteq p \ \wedge \ p\subseteq q \\
= & \{ \quad \text{anti-symmetry} \quad \} \\
 & p=q \ .
\end{array}
$$

□

An immediate corollary of lemma 5.19 is that, for all points a and b,

(5.20) $a{\circ}\top\!\!\top{\circ}b \subseteq I \equiv a=b \ .$

(Folows-from is immediate from the definition of a point. Implies is an instance of lemma 5.19.)

Definition 5.21 (Extensional) Suppose A is a type. The lattice of coreflexives of type A is said to be *extensional* iff for all coreflexives p of type A,

$$p \ = \ \langle \cup a : \mathsf{point}.a \wedge a\subseteq p : a \rangle \ .$$

□

We conclude with a theorem stating conditions under which the lattice of relations (of a given type) is saturated and atomic. The proper atoms are events of the form $a{\circ}\top\!\!\top{\circ}b$ where a and b are points; such an event models the pair (a,b) in conventional pointwise formulations of relation algebra.

Theorem 5.22 Suppose, for types A and B, the lattices of coreflexives of types A and B are both complete, universally distributive and extensional. Then the lattice of relations of type $A{\sim}B$ is a saturated, atomic lattice; the atoms are elements of the form

$a{\circ}\top\!\!\top{\circ}b$ where a and b are atoms of the lattice of coreflexives (of types A and B, respectively). It follows that, if the lattice of relations of type $A{\sim}B$ is complete and universally distributive, it is isomorphic to the powerset of the set of elements of the form $a{\circ}\top\!\!\top{\circ}b$ where a and b are atoms of the lattices of coreflexives of types A and B, respectively.

Proof By lemma 5.15, it suffices to prove that the lattice of relations of type $A{\sim}B$ is saturated. This is easy: for all R of type $A{\sim}B$,

$$\begin{array}{cl}
 & R \\
= & \{ \quad I \text{ is unit of composition,} \\
 & \quad \text{lattices of coreflexives of types } A \text{ and } B \text{ are extensional} \quad \} \\
 & \langle \cup a : \text{point}.a : a \rangle \circ R \circ \langle \cup b : \text{point}.b : b \rangle \\
= & \{ \quad \text{distributivity of composition over } \cup \quad \} \\
 & \langle \cup\, a,b : \text{point}.a \wedge \text{point}.b : a{\circ}R{\circ}b \rangle \\
= & \{ \quad \text{all-or-nothing rule: lemma 5.15, } \bot\!\!\bot \text{ is zero of supremum} \quad \} \\
 & \langle \cup\, a,b : \text{point}.a \wedge \text{point}.b \wedge a{\circ}R{\circ}b \neq \bot\!\!\bot : a{\circ}\top\!\!\top{\circ}b \rangle \quad .
\end{array}$$

That the lattice of relations is a powerset follows from theorem 2.51.
□

Henceforth, we assume that, for each type A, the lattice of coreflexives of type A is complete, universally distributive and saturated. That is, recalling theorem 2.51, we assume that the coreflexives of a given type form a powerset. Theorem 5.22 then states that, for each pair of types A and B, the lattice of relations of type $A{\sim}B$ is a powerset with atoms of the form $a{\circ}\top\!\!\top{\circ}b$ where a and b are points of type A and B, respectively. In view of theorem 2.51, we use $\subseteq$ for the ordering relation and $\sim$ for the complement operator on coreflexives. We use $\neg$ for the complement operator on relations. Thus, for coreflexive p, ${\sim}p = I \cap \neg p$. Later, when the relations represent graphs, we use "node" as a synonym for "point". Standard properties of powersets —the properties of set union, intersection and complementation— will be assumed, sometimes without specific mention and sometimes with the hint "set theory".

We use p and q to range over coreflexives and a and b to range over points.

Summarising, the *saturation* property is that

(5.23) $\langle \forall R :: R = \langle \cup\, a,b : a{\circ}\top\!\!\top{\circ}b \subseteq R : a{\circ}\top\!\!\top{\circ}b \rangle \rangle$.

The *irreducibility* property is that, if $\mathcal{R}$ is a function with range relations of type $A{\sim}B$ and source K, then, for all points a and b of appropriate type,

(5.24) $a{\circ}\top\!\!\top{\circ}b \subseteq \cup\mathcal{R} \equiv \langle \exists k : k{\in}K : a{\circ}\top\!\!\top{\circ}b \subseteq \mathcal{R}.k \rangle$.

The *identity relation* I_A of type A has the property that, for all points a and a' of type A,

(5.25) $a{\circ}\top\!\!\top{\circ}a' \subseteq \mathrm{I}_A \;\equiv\; a = a'$.

Other than its definition, the crucial property of the complement operator on coreflexives is that, for all points a and coreflexives p,

$$\neg(a \subseteq p) \;\equiv\; a \subseteq {\sim}p \;\; .$$

See lemma 2.52.

5.2.1 Properties of Points

This section documents properties of points with respect to domains and factors.

Lemma 5.26 For all relations R and points a and b (of appropriate type),

$$a \subseteq R{<} \;\equiv\; (a{\circ}R){>} \neq \bot\!\!\bot \;\;\text{, and}$$

$$b \subseteq R{>} \;\equiv\; (R{\circ}b){<} \neq \bot\!\!\bot \;\; .$$

Proof We prove the second equation.

$$
\begin{array}{ll}
 & (R{\circ}b){<} \neq \bot\!\!\bot \\
= & \{ \quad \text{cone rule: (4.16)} \quad \} \\
 & \top\!\!\top \circ (R{\circ}b){<} \circ \top\!\!\top \;=\; \top\!\!\top \\
= & \{ \quad [\; R{<} \circ \top\!\!\top \;=\; R{\circ}\top\!\!\top \;] \text{ with } R := R{\circ}b \quad \} \\
 & \top\!\!\top{\circ}R{\circ}b{\circ}\top\!\!\top = \top\!\!\top \\
= & \{ \quad [\; \top\!\!\top \circ R{>} \;=\; \top\!\!\top{\circ}R \;] \quad \} \\
 & \top\!\!\top \circ R{>} \circ b \circ \top\!\!\top \;=\; \top\!\!\top \\
= & \{ \quad \text{cone rule: (4.16)} \quad \} \\
 & R{>} \circ b \neq \bot\!\!\bot \\
= & \{ \quad R{>} \circ b \subseteq b \;\; ; \\
 & \qquad \text{so, by atomicity of } b\text{, } R{>} \circ b = b \vee R{>} \circ b = \bot\!\!\bot \;\; ; \\
 & \qquad \text{also, } b \neq \bot\!\!\bot \quad \} \\
 & R{>} \circ b = b \\
= & \{ \quad R{>} \circ b = R{>} \cap b \quad \} \\
 & b \subseteq R{>} \;\; .
\end{array}
$$

□

For a point b the square $R \circ b \circ R^{\cup}$ represents the set of all points a such that a and b are related by R. This is made precise in lemma 5.27 and its corollary, lemma 5.28.

Lemma 5.27 For all relations R of type $A \sim B$, all coreflexives p of type $A \sim A$ and all points b of type B,

$$p \subseteq R \circ b \circ R^{\cup} \quad \equiv \quad p \circ \top\!\!\top \circ b \subseteq R \ .$$

Symmetrically, for all relations R of type $A \sim B$, all coreflexives q of type $B \sim B$ and all points a of type A,

$$q \subseteq R^{\cup} \circ a \circ R \quad \equiv \quad a \circ \top\!\!\top \circ q \subseteq R \ .$$

Proof By mutual implication:

$$\begin{array}{ll}
 & p \subseteq R \circ b \circ R^{\cup} \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & p \circ \top\!\!\top \circ b \subseteq R \circ b \circ R^{\cup} \circ \top\!\!\top \circ b \\
\Rightarrow & \{ \quad R^{\cup} \circ \top\!\!\top \subseteq \top\!\!\top \quad \} \\
 & p \circ \top\!\!\top \circ b \subseteq R \circ b \circ \top\!\!\top \circ b \\
\Rightarrow & \{ \quad b \text{ is a point: so, by definition 5.13, } b \circ \top\!\!\top \circ b = b \text{ and } b \subseteq I \quad \} \\
 & p \circ \top\!\!\top \circ b \subseteq R \\
\Rightarrow & \{ \quad \text{converse and monotonicity} \quad \} \\
 & p \circ \top\!\!\top \circ b \circ b \circ \top\!\!\top \circ p^{\cup} \subseteq R \circ b \circ R^{\cup} \\
\Rightarrow & \{ \quad b \text{ is a point: so } b \circ b = b \text{ and } \top\!\!\top \circ b \circ \top\!\!\top = \top\!\!\top \\
 & \quad\ \ p \text{ is a coreflexive, so } p^{\cup} = p \text{; monotonicity} \quad \} \\
 & p \circ \top\!\!\top \circ p \subseteq R \circ b \circ R^{\cup} \\
\Rightarrow & \{ \quad I \subseteq \top\!\!\top \text{ and } p \circ p = p \quad \} \\
 & p \subseteq R \circ b \circ R^{\cup} \ .
\end{array}$$

□

Property (5.16) is the most basic formulation of membership of pairs in a relation. It can also be formulated in terms of squares and in terms of domains:

Lemma 5.28 For all relations R and points a and b (of appropriate type),

$$(a \subseteq R \circ b \circ R^{\cup}) \ = \ (a \circ \top\!\!\top \circ b \subseteq R) \ = \ (b \subseteq R^{\cup} \circ a \circ R) \ .$$

Proof Straightforward instantiation of lemma 5.27:

$$
\begin{array}{ll}
 & a \subseteq R \circ b \circ R^{\cup} \\
= & \{ \quad \text{lemma 5.27 with } p := a \quad \} \\
 & a \circ \top\!\top \circ b \subseteq R \\
= & \{ \quad \text{lemma 5.27 with } p := b \quad \} \\
 & b \subseteq R^{\cup} \circ b \circ R \ .
\end{array}
$$

□

Lemma 5.29 For all relations R and points a and b (of appropriate type),

$$(a \subseteq (R \circ b)_<) \ = \ (a \circ \top\!\top \circ b \subseteq R) \ = \ (b \subseteq (a \circ R)_>) \ .$$

Proof

$$
\begin{array}{ll}
 & a \circ \top\!\top \circ b \subseteq R \\
\Rightarrow & \{ \quad \text{monotonicity and } a \text{ is a coreflexive, so } a \circ a = a \quad \} \\
 & a \circ \top\!\top \circ b \subseteq a \circ R \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & (a \circ \top\!\top \circ b)_> \subseteq (a \circ R)_> \\
= & \{ \quad \text{domains abd} \quad \} \\
 & b \subseteq (a \circ R)_> \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & a \circ \top\!\top \circ b \ \subseteq \ a \circ \top\!\top \circ (a \circ R)_> \\
= & \{ \quad \text{domains: } [\ \top\!\top \circ R_> \ = \ \top\!\top \circ R \ ] \text{ with } R := a \circ R \quad \} \\
 & a \circ \top\!\top \circ b \ \subseteq \ a \circ \top\!\top \circ a \circ R \\
= & \{ \quad a \text{ is a point, so } a \circ \top\!\top \circ a = a \quad \} \\
 & a \circ \top\!\top \circ b \subseteq a \circ R \\
\Rightarrow & \{ \quad a \text{ is a coreflexive, monotonicity} \quad \} \\
 & a \circ \top\!\top \circ b \subseteq R \ .
\end{array}
$$

That is, we have shown by mutual implication that

$$a \circ \top\!\top \circ b \subseteq R \ \equiv \ b \subseteq (a \circ R)_> \ .$$

A symmetric calculation establishes that

$$a{\circ}\top\!\!\top{\circ}b \subseteq R \ \equiv\ a \subseteq (R{\circ}b)_{<} \quad .$$

□

Combined with property (5.16), lemmas 5.28 and 5.29 give six alternative ways of formulating the membership relation $a{\circ}\top\!\!\top{\circ}b \subseteq R$. All are useful.

Lemma 5.30 For all relations R and points a (of appropriate type),

$$a \subseteq R_{<} \ \equiv\ \langle\exists b : b \subseteq R_{>} : a{\circ}\top\!\!\top{\circ}b \subseteq R\rangle \quad .$$

Also, for all relations R and points b (of appropriate type),

$$b \subseteq R_{>} \ \equiv\ \langle\exists a : a \subseteq R_{<} : a{\circ}\top\!\!\top{\circ}b \subseteq R\rangle \quad .$$

Proof We prove the first equation:

$$\begin{array}{ll}
 & a \subseteq R_{<} \\
= & \quad\{ \quad \text{lemma 5.26} \quad \} \\
 & (a{\circ}R)_{>} \neq \perp\!\!\perp \\
= & \quad\{ \quad \text{lemma 5.32} \quad \} \\
 & \langle\exists b :: b \subseteq (a{\circ}R)_{>}\rangle \\
= & \quad\{ \quad \text{lemma 5.29} \quad \} \\
 & \langle\exists b :: a{\circ}\top\!\!\top{\circ}b \subseteq R\rangle \\
= & \quad\{ \quad \text{domains (specifically, } a{\circ}\top\!\!\top{\circ}b \subseteq R \Rightarrow b \subseteq R_{>}\text{)} \quad \} \\
 & \langle\exists b : b \subseteq R_{>} : a{\circ}\top\!\!\top{\circ}b \subseteq R\rangle \quad .
\end{array}$$

□

Lemma 5.31 gives a pointwise interpretations of the factor operators. Although we typically try to avoid pointwise reasoning, the lemma is sometimes indispensable.

Lemma 5.31 For all relations R of type $A{\sim}C$ and S of type $B{\sim}C$ (for some A, B and C) and all points a and b,

$$a{\circ}\top\!\!\top{\circ}b \subseteq R/S \ \equiv\ (b{\circ}S)_{>} \subseteq (a{\circ}R)_{>} \quad .$$

Dually, for all relations R of type $C{\sim}A$ and S of type $C{\sim}B$, and all points a and b,

$$a{\circ}\top\!\!\top{\circ}b \subseteq R\backslash S \ \equiv\ (R{\circ}a)_{<} \subseteq (S{\circ}b)_{<} \quad .$$

Proof By mutual implication:

$a\circ\top\!\!\top\circ b \subseteq R/S$

$=$ { definition of factor }

$a\circ\top\!\!\top\circ b\circ S \subseteq R$

$\Rightarrow$ { a and b are points, monotonicity and domains
(see initial steps in proof of lemma 5.29) }

$(b\circ S)_{>} \subseteq (a\circ R)_{>}$

$\Rightarrow$ { monotonicity }

$a\circ\top\!\!\top\circ(b\circ S)_{>} \subseteq a\circ\top\!\!\top\circ(a\circ R)_{>}$

$=$ { domains }

$a\circ\top\!\!\top\circ b\circ S \subseteq a\circ\top\!\!\top\circ a\circ R$

$=$ { a is a point (so $a\circ\top\!\!\top\circ a = a$) }

$a\circ\top\!\!\top\circ b\circ S \subseteq a\circ R$

$\Rightarrow$ { a is a coreflexive }

$a\circ\top\!\!\top\circ b\circ S \subseteq R$

$=$ { definition of factor }

$a\circ\top\!\!\top\circ b \subseteq R/S$.

The second equivalence is proved similarly.

$a\circ\top\!\!\top\circ b \subseteq R\backslash S$

$=$ { definition of factor }

$R\circ a\circ\top\!\!\top\circ b \subseteq S$

$\Rightarrow$ { monotonicity and coreflexives
(see initial steps in proof of lemma 5.29) }

$(R\circ a)_{<} \subseteq (S\circ b)_{<}$

$\Rightarrow$ { (as in above calculation) }

$a\circ\top\!\!\top\circ b \subseteq R\backslash S$.

□

5.2.2 Unicity

Sometimes we want to define functions indirectly via a property relating input and output values. The property is formalised and then it is shown that the formal specification relates each input value to exactly one output value. That is, the formal specification relates each input value to at most one and at least one output value. In order to reason within our axiom system, we then want to conclude that output values are points. See, for example, section 5.3, where we define the meaning of functionality and exhibit an expression that formulates, in very general terms, the result of applying a function to an argument.

Although the process seems to be obvious, we want to stick to our goal of validating every step within our axiom system. For this reason, we now present the technical justification. As just mentioned, we refer the reader to section 5.3 for a concrete example.

In the following lemmas, p is a coreflexive relation and dummies a and a' are points of the same type as p.

We begin with the consequence of showing that specification p has at least one solution.

Lemma 5.32

$$p \neq \perp\!\!\perp \;\equiv\; \langle \exists a :: a \subseteq p \rangle \;.$$

Proof

$p \neq \perp\!\!\perp$

$=$ { cone rule: (4.16) }

$\top\!\!\top \circ p \circ \top\!\!\top = \top\!\!\top$

$=$ { saturation property: (5.23) }

$\top\!\!\top \circ \langle \cup a : a \subseteq p : a \rangle \circ \top\!\!\top \;=\; \top\!\!\top$

$=$ { distributivity }

$\langle \cup a : a \subseteq p : \top\!\!\top \circ a \circ \top\!\!\top \rangle \;=\; \top\!\!\top$

$=$ { a ranges over points, so $a \neq \perp\!\!\perp$, cone rule: (4.16) }

$\langle \cup a : a \subseteq p : \top\!\!\top \rangle \;=\; \top\!\!\top$

$\Rightarrow$ { $\langle \cup a : \mathsf{false} : \top\!\!\top \rangle = \perp\!\!\perp$ and $\perp\!\!\perp \neq \top\!\!\top$ }

$\langle \exists a :: a \subseteq p \rangle$

$\Rightarrow$ { a ranges over points: so $\perp\!\!\perp \neq a$
predicate calculus, (details left to the reader) }

$p \neq \perp\!\!\perp \;.$

□

Next we formulate the consequence of showing that specification p has at most one solution.

Lemma 5.33

$$\langle\forall a : a\subseteq p : a=p\rangle \quad\equiv\quad \langle\forall\, a,a' \,:\, a\subseteq p \wedge a'\subseteq p \,:\, a=a'\rangle \;\;.$$

Proof

$\langle\forall a : a\subseteq p : a=p\rangle$

$=\qquad\{\qquad$ anti-symmetry $\quad\}$

$\langle\forall a : a\subseteq p : a\supseteq p\rangle$

$=\qquad\{\qquad$ extensionality assumption: definition 5.21 $\quad\}$

$\langle\forall a \,:\, a\subseteq p \,:\, a \supseteq \langle\cup a' : a'\subseteq p : a'\rangle\rangle$

$=\qquad\{\qquad$ suprema $\quad\}$

$\langle\forall a : a\subseteq p : \langle\forall a' : a'\subseteq p : a\supseteq a'\rangle\rangle$

$\Leftarrow\qquad\{\qquad$ reflexivity of the subset relation $\quad\}$

$\langle\forall a : a\subseteq p : \langle\forall a' : a'\subseteq p : a=a'\rangle\rangle$

$=\qquad\{\qquad$ nesting of quantifications $\quad\}$

$\langle\forall\, a,a' \,:\, a\subseteq p \wedge a'\subseteq p \,:\, a=a'\rangle$

$\Leftarrow\qquad\{\qquad$ Leibniz and predicate calculus $\quad\}$

$\langle\forall a : a\subseteq p : a=p\rangle\;\;.$

□

Theorem 5.34 Suppose p is a coreflexive relation. Then p is a point equivales

$$\langle\exists a :: a\subseteq p\rangle \;\;\wedge\;\; \langle\forall\, a,a' \,:\, a\subseteq p \wedge a'\subseteq p \,:\, a=a'\rangle \;\;.$$

(As above, dummies a and a' range over points of the same type as p.)

In words, a specification p defines a point iff it has at least one solution and at most one solution.

Proof In the following dummy q ranges over coreflexives of the same type as p and a ranges over points of the same type as p.

p is atomic

$=\qquad\{\qquad$ definition 2.49 $\quad\}$

$\langle\forall q : q\subseteq p : q=p \vee q=\perp\!\!\perp\rangle$

$=\qquad\{\qquad$ trading $\quad\}$

$\langle\forall q : q\subseteq p\wedge q\neq\perp\!\!\perp : q=p\rangle$

$=\qquad\{\qquad$ lemma 5.32 $\quad\}$

$\langle\forall q : q\subseteq p\wedge\langle\exists a :: a\subseteq q\rangle : q=p\rangle$

$=\qquad\{\qquad$ distributivity (of conjunction over disjunction),

range disjunction $\quad\}$

$\langle\forall q,a : a\subseteq q\subseteq p : q=p\rangle$

$\Leftarrow\qquad\{\qquad$ anti-symmetry $\quad\}$

$\langle\forall a : a\subseteq p : a=p\rangle$

$=\qquad\{\qquad$ lemma 5.33 $\quad\}$

$\langle\forall a,a' : a\subseteq p \wedge a'\subseteq p : a=a'\rangle$.

Also,

p is atomic

$=\qquad\{\qquad$ definition 2.49 $\quad\}$

$\langle\forall q : q\subseteq p : q=p \vee q=\perp\!\!\perp\rangle$

$\Rightarrow\qquad\{\qquad$ points a and a' are coreflexives, weakening $\quad\}$

$\langle\forall a,a' : a\subseteq p \wedge a'\subseteq p : (a=p \vee a=\perp\!\!\perp)\wedge(a'=p\vee a'=\perp\!\!\perp)\rangle$

$=\qquad\{\qquad$ points are proper (i.e. $a\neq\perp\!\!\perp$ and $a'\neq\perp\!\!\perp$) $\quad\}$

$\langle\forall a,a' : a\subseteq p \wedge a'\subseteq p : a=p\wedge a'=p\rangle$

$\Rightarrow\qquad\{\qquad$ transitivity of equality $\quad\}$

$\langle\forall a,a' : a\subseteq p \wedge a'\subseteq p : a=a'\rangle$.

Combining the two calculations, we have established by mutual implication that

(5.35) $\;$ p is atomic $\;\equiv\;\langle\forall a,a' : a\subseteq p\wedge a'\subseteq p : a=a'\rangle$.

It follows that, for all coreflexives p ,

p is a point

$=\qquad\{\qquad$ definitions 2.49 and 5.13, assumption: p is coreflexive $\quad\}$

$p\neq\perp\!\!\perp \;\wedge\; p$ is atomic

$=\qquad\{\qquad$ lemma 5.32 and (5.35) $\quad\}$

$\langle\exists a :: a\subseteq p\rangle \;\wedge\; \langle\forall a,a' : a\subseteq p\wedge a'\subseteq p : a=a'\rangle$.

□

5.3 Functionality and Totality

A subset of the relations is formed by the functions, which can be seen as deterministic relations. There are a number of ways to characterise them. Because we want to stress the importance of Galois connections we choose the following.

Definition 5.36 (Functional Relation) A relation f is said to be *functional* if and only if it has the property that for all relations R and S:

(5.37) $R \circ f{>} \subseteq S \circ f \equiv R \circ f^{\cup} \subseteq S$.

□

Note: The converse-dual of (5.37) could equally have been chosen as the definition of functional. It is at this point that we are obliged to commit to a "direction" when giving pointwise interpretations to relations. Specifically, we interpret the left domain of a relation as the possible "outputs" of the relation and the right domain as the possible "inputs". (See also section 5.4.) This choice is consistent with the use of the symbol "$\circ$" to denote both composition of relations and composition of functions.

The definition of functional is almost a Galois connection, but not quite: the right domain on the lefthand side spoils it. However, it is a Galois connection if we restrict our attention to total functions, that is functional relations with right domain the identity relation. Another way of turning the definition into a Galois connection is by considering the set of relations with right domain contained in $f{>}$. It can be shown that these relations form a complete lattice with $\perp\!\!\perp$ as bottom element, relation $\top\!\!\top \circ f{>}$ as top and the intersection and union operators as meet and join. It is not difficult to verify that the functions $(\circ f)$ and $(\circ(f^{\cup}))$ form a Galois connection between this lattice and the lattice of the relations. As a consequence, the function $(\circ f)$ distributes over non-empty intersections of relations, a property that is expected from pointwise considerations.

In this section, we deviated from our practice of starting with a pointwise interpretation. So, we now have to check whether the definition captures the idea of functionality. The characterising property of a function is that it is single-valued (also known as Leibniz's rule), i.e. if $y[\![f]\!]x$ and $z[\![f]\!]x$ then y is equal to z. This is written as:

$$\langle \forall y,z : \langle \exists x :: y[\![f]\!]x \wedge z[\![f]\!]x \rangle : y[\![I]\!]z \rangle \quad .$$

After rewriting the existential quantification using relation composition and subsequently the universal quantification using the definition of relation inclusion, we obtain (the much more concise):

(5.38) $f \circ f^{\cup} \subseteq I$.

Expression (5.38) follows easily from definition 5.36 by instantiation of R to f and S to I. It is also not difficult to derive condition (5.37) from (5.38), in other words, expression (5.38) is an alternative definition of the notion of functionality.

As is often the case with important concepts, there is a number of equivalent definitions of functionality. We mention a third:

(5.39) $f\circ f^{\cup} = f{<}$.

This is obtained by rewriting (5.38) as $f\circ f^{\cup} = I \cap f\circ f^{\cup}$ and noting that the righthand side of the latter formula is equal to the left domain of f (this uses the dual of (4.14)).

The notion dual to functionality, viz. injectivity, is now of course easy to define as: f is *injective* if and only if $f^{\cup}$ is functional. A relation that is both injective and functional is called a *bijection*.

The standard notion of a partial function is a relation that defines a unique output value for each input value in its domain. In our axiom system we have the following theorem.

Theorem 5.40 Suppose relation R has type $A{\sim}B$. Then

(5.41) $R\circ R^{\cup} \subseteq I_A \;\equiv\; \langle\forall b : b\subseteq R{>} : \mathsf{point}.(R\circ b\circ R^{\cup})\rangle$.

Moreover, if f is a relation of type $A{\sim}B$ and $f\circ f^{\cup} \subseteq I_A$, the relation $f\circ b\circ f^{\cup}$ is a point of type A and

(5.42) $\langle\forall a,b \;:\; b\subseteq f{>} \;:\; a\circ\top\!\!\top\circ b \subseteq f \;\equiv\; a = f\circ b\circ f^{\cup}\rangle$.

Proof We prove (5.41) by mutual implication. First,

$$\begin{array}{ll}
 & R\circ R^{\cup} \subseteq I_A \\
= & \{ \quad \text{domains} \quad \} \\
 & R\circ R{>}\circ R^{\cup} \subseteq I_A \\
= & \{ \quad \text{extensionality assumption: definition 5.21} \quad \} \\
 & R\circ\langle\cup b \;:\; b\subseteq R{>} \;:\; b\rangle\circ R^{\cup} \subseteq I_A \\
= & \{ \quad \text{distributivity} \quad \} \\
 & \langle\forall b \;:\; b\subseteq R{>} \;:\; R\circ b\circ R^{\cup} \subseteq I_A\rangle \\
\Leftarrow & \{ \quad \text{definition 5.13 of a point} \quad \} \\
 & \langle\forall b : b\subseteq R{>} : \mathsf{point}.(R\circ b\circ R^{\cup})\rangle \;.
\end{array}$$

Thus we have established the "if" part of the equivalence. Now, for the "only-if", assume $R\circ R^{\cup} \subseteq I_A$.

We first note that, for all b such that $b \subseteq R{>}$, the equation

(5.43) $a\colon\ \mathsf{point}.a\colon\ a \circ \top\!\!\top \circ b \subseteq R$

has *at most one* solution since, for all points a and a' of type A,

$$
\begin{array}{ll}
 & a \circ \top\!\!\top \circ b \subseteq R \;\wedge\; a' \circ \top\!\!\top \circ b \subseteq R \\
\Rightarrow & \{ \quad \text{converse and monotonicity} \quad \} \\
 & a \circ \top\!\!\top \circ b \circ b \circ \top\!\!\top \circ a' \subseteq R \circ R^{\cup} \\
= & \{ \quad b \text{ is a point, so } \top\!\!\top \circ b \circ b \circ \top\!\!\top = \top\!\!\top \quad \} \\
 & a \circ \top\!\!\top \circ a' \subseteq R \circ R^{\cup} \\
\Rightarrow & \{ \quad \text{assumption: } R \circ R^{\cup} \subseteq I_A \text{, transitivity of the subset relation} \quad \} \\
 & a \circ \top\!\!\top \circ a' \subseteq I_A \\
\Rightarrow & \{ \quad a \text{ and } a' \text{ are points: (5.25)} \quad \} \\
 & a = a' \ .
\end{array}
$$

That is,

(5.44) $\langle \forall b : b \subseteq R{>} : \langle \forall a, a' : a \circ \top\!\!\top \circ b \subseteq R \;\wedge\; a' \circ \top\!\!\top \circ b \subseteq R : a = a' \rangle\rangle$.

By lemma 5.26, equation (5.43) has *at least one* solution for all points b such that $b \subseteq R{>}$. That is,

(5.45) $\langle \forall b : b \subseteq R{>} : \langle \exists a :: a \circ \top\!\!\top \circ b \subseteq R \rangle\rangle$.

Thus equation (5.43) has *exactly one* solution for all points b such that $b \subseteq f{>}$. So:

$$
\begin{array}{ll}
 & \langle \forall b : b \subseteq R{>} : \mathsf{point}.(R \circ b \circ R^{\cup}) \rangle \\
= & \{ \quad \begin{array}{ll}
 & R \circ b \circ R^{\cup} \\
\subseteq & \{ \quad \text{assumption: } b \subseteq R{>} \text{, monotonicity} \quad \} \\
 & R \circ R{>} \circ R^{\cup} \\
= & \{ \quad \text{domains} \quad \} \\
 & R \circ R^{\cup} \\
\subseteq & \{ \quad \text{assumption: } R \circ R^{\cup} \subseteq I_A \quad \} \\
 & I_A \ ,
\end{array} \\
 & \quad \text{theorem 5.34 with } p := R \circ b \circ R^{\cup} \quad \} \\
 & \quad \langle \forall b : b \subseteq R{>} : \langle \exists a :: a \subseteq R \circ b \circ R^{\cup} \rangle\rangle
\end{array}
$$

$\wedge\quad \langle\forall b : b\subseteq R{>} : \langle\forall a,a' : a \subseteq R\circ b\circ R^{\cup} \wedge a' \subseteq R\circ b\circ R^{\cup} : a = a'\rangle\rangle$

$=\quad\{\quad$ lemma 5.28 $\quad\}$

$\langle\forall b : b\subseteq R{>} : \langle\exists a :: a\circ\top\!\top\circ b \subseteq R\rangle\rangle$

$\wedge\quad \langle\forall b : b\subseteq R{>} : \langle\forall a,a' : a\circ\top\!\top\circ b \subseteq R \wedge a'\circ\top\!\top\circ b \subseteq R : a = a'\rangle\rangle$

$=\quad\{\quad$ (5.44) and (5.45) $\quad\}$

true .

This concludes the proof of (5.41).

Now, assuming that $f\circ f^{\cup} \subseteq I$, it follows from (5.41) (with $R{:=}f$) that $f\circ b\circ f^{\cup}$ is a point. Also, for all points a and b (of types A and B, respectively),

$b\subseteq f{>} \wedge a\circ\top\!\top\circ b \subseteq f$

$=\quad\{\quad$ lemma 5.29 (aiming to eliminate first conjunct) $\quad\}$

$b\subseteq f{>} \wedge b\subseteq (a\circ f){>} \wedge a\circ\top\!\top\circ b \subseteq f$

$=\quad\{\quad$ monotonicity and lemma 5.29 $\quad\}$

$a\circ\top\!\top\circ b \subseteq f$

$=\quad\{\quad$ lemma 5.28 $\quad\}$

$a \subseteq f\circ b\circ f^{\cup}$

$=\quad\{\quad$ $f\circ b\circ f^{\cup}$ is a point, definitions 5.13 and 2.49 $\quad\}$

$a = f\circ b\circ f^{\cup}$.

□

In words, theorem 5.40 states that f is functional iff, for all points b in the right domain of f, the relation $f\circ b\circ f^{\cup}$ defines a unique point of type A. This is the point that we denote by $f.b$. The defining property of $f.b$ is thus

(5.46) $\langle\forall a,b : b\subseteq f{>} : a\circ\top\!\top\circ b \subseteq f \equiv a = f.b\rangle$.

A consequence of the unicity property expressed by (5.46) is the property that, for all functional relations f of type $C{\sim}A$ and g of type $C{\sim}B$, and all points a and b,

(5.47) $a\circ\top\!\top\circ b \subseteq f^{\cup}\circ g \equiv a\subseteq f{>} \wedge f.a = g.b \wedge b\subseteq g{>}$.

When introducing the modularity rule in section 4.1.2, we emphasised the importance of distributivity properties. A distributivity property that possibly goes unnoticed in pointwise calculations but must be used explicitly in point-free calculations is the distributivity of functions over intersection: for all relations R and S and all functional relations f,

(5.48) $(R\cap S)\circ f = R\circ f \cap S\circ f$.

The property is an application of (4.15) combined with (5.38).

Besides functionality and injectivity, there are two other dual notions which relations may enjoy: totality and surjectivity. We only spell out what it means for a relation to be total, because surjectivity can be defined in terms of totality: relation R is *surjective* iff its converse $R^{\cup}$ is total.

Relation R is total means that it can accept every element of the universe as an input. Formally, relation R is *total* iff $R{>} = I$. An equivalent formulation is: $I \subseteq R^{\cup} \circ R$. From this, it can be seen that surjectivity is, in a sense, also dual to injectivity: relation R is injective can be expressed as $I \supseteq R^{\cup} \circ R$.

We conclude this section with a useful lemma on establishing the equality of two functional relations.

Lemma 5.49 Suppose f and h are functional relations of the same type. Then

$$f = h \;\equiv\; f \subseteq h \wedge f{>} = h{>}$$

Proof Clearly, the left side implies the right side and it suffices to prove follows-from.

$$
\begin{array}{ll}
 & h \subseteq f \\
= & \{ \quad \text{domains: (5.5), and assumption: } f{>} = h{>} \quad \} \\
 & h \circ f{>} \subseteq f \\
= & \{ \quad \text{assumption: } f \text{ is a function and (5.37)} \quad \} \\
 & h \circ f^{\cup} \subseteq I \\
\Leftarrow & \{ \quad \text{assumption: } f \subseteq h\text{, monotonicity and transitivity} \quad \} \\
 & h \circ h^{\cup} \subseteq I \\
= & \{ \quad \text{assumption: } h \text{ is a function and (5.38)} \quad \} \\
 & \mathsf{true} \;\; .
\end{array}
$$

The required implication follows from the anti-symmetry of the subset relation.
□

5.4 Heterogeneous Relations

A *heterogeneous* relation R has a *type* given by two sets A and B, which we call the *target* and *source* of R. We use the notation $A{\sim}B$ to denote the type of a relation. Formally, a relation of type $A{\sim}B$ is a subset of $A{\times}B$. (Equivalently, it is a function with domain $A{\times}B$ and range Bool.) A *homogeneous* relation is a relation of type $A{\sim}A$ for some A.

The target and source of a relation should not be confused with its left domain and right domain. If R has type $A{\sim}B$ then its left domain $R{<}$ has type $A{\sim}A$ and its right domain $R{>}$ has type $B{\sim}B$. As always, $R{<}$ and $R{>}$ are coreflexives, but this property is expressed formally as $R{<}\subseteq I_A$ and $R{>}\subseteq I_B$, where I_A denotes the identity relation of type $A{\sim}A$ (and similarly for I_B).

The operators in the algebra of heterogeneous relations are typed. For example, the composition of two relations R and S, denoted as always by $R{\circ}S$, is only defined when the source of R equals the target of S. Moreover, the target of $R{\circ}S$ is the target of R and the source of $R{\circ}S$ is the source of S. That is, if R has type $A{\sim}B$ and S has type $B{\sim}C$ then $R{\circ}S$ has type $A{\sim}C$. We assume the reader is familiar with such rules.

As mentioned earlier, the rules of the untyped calculus are applicable in the typed calculus, with some restrictions on types. For example, the rule $R = R{<}\circ R$ remains valid without restriction. Restrictions are necessary on types for the middle-exchange and rotation rules (see section 4). For example, the inclusion $R{\circ}S \subseteq \neg T^{\cup}$ is only defined if R has type $A{\sim}B$, S has type $B{\sim}C$ and T has type $C{\sim}A$, for some sets A, B and C. (The converse $T^{\cup}$ of T then has type $A{\sim}C$, which equals the type of $\neg T^{\cup}$ and $R{\circ}S$.) With these type restrictions, $S{\circ}T \subseteq \neg R^{\cup}$ is also well-defined, and the two inclusions $R{\circ}S \subseteq \neg T^{\cup}$ and $S{\circ}T \subseteq \neg R^{\cup}$ are equal as per the rotation rule.

It is now possible to see why the choice of an inclusive-or in the statement of the cone rule (4.16) is vital: the rule, for all R:

$$R = \perp\!\!\perp \;\; \not\equiv \;\; \top\!\!\top{\circ}R{\circ}\top\!\!\top = \top\!\!\top$$

is invalid in the case that the type of R is $\emptyset{\sim}\emptyset$ and, as good programmers are very well aware, such extreme cases can and do occur in practice.

The care that must be exercised with overloading is exemplified by the rule

$$R{\circ}\top\!\!\top \;=\; R{<}\circ\top\!\!\top \quad .$$

Recall that, if R has type $A{\sim}B$, $R{<}$ has type $A{\sim}A$. Thus the notation "$\top\!\!\top$" on the left side of the equation denotes the universal relation of type $B{\sim}C$, for some type C; on the other hand, the notation "$\top\!\!\top$" on the right side of the equation denotes the universal relation of type $A{\sim}C$. Rather than overload the notation in this way, we could decorate every occurrence of $\top\!\!\top$ with its type. For example, we could rephrase the rule as

$$R \circ {}_B\top\!\!\top_C \;\;=\;\; R{<} \circ {}_A\top\!\!\top_C \quad .$$

We prefer not to do so because the type information is usually easy to infer. (An exception is that we occasionally decorate the identity relation I with its type: I_A

denotes the identity relation of type $A{\sim}A$.) Nevertheless, we urge the reader to check types, particularly where notation is overloaded.

Typed relation algebra, as briefly summarised above, extends category theory to what has been called *allegory theory*. See Freyd and Ščedrov [Fv90] for more details.

5.5 The Interface Between Formal and Informal

In order to narrow the gap between conventional pointwise reasoning and the formal axiomatic reasoning in this paper, this section explains how to proceed from one to the other.

Theorem 5.22 enables more familiar pointwise reasoning. For example, we can derive the standard pointwise definition of the composition of relations. With dummies a, b, c and d ranging over proper atomic coreflexives, we have:

$$
\begin{array}{ll}
 & R{\circ}S \\
= & \{ \quad \text{theorem 5.22} \quad \} \\
 & \langle \cup\, a,b : a{\circ}R{\circ}b \neq \bot\!\!\bot : a{\circ}\top\!\!\top{\circ}b \rangle \circ \langle \cup\, c,d : c{\circ}S{\circ}d \neq \bot\!\!\bot : c{\circ}\top\!\!\top{\circ}d \rangle \\
= & \{ \quad \text{distributivity and nesting of quantifications} \quad \} \\
 & \langle \cup\, a,b,c,d \;:\; a{\circ}R{\circ}b \neq \bot\!\!\bot \wedge c{\circ}S{\circ}d \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}b{\circ}c{\circ}\top\!\!\top{\circ}d \rangle \\
= & \{ \quad b{\circ}c = \bot\!\!\bot \Leftarrow b \neq c \text{, one-point rule and } b{\circ}b = b\,; \quad \} \\
 & \langle \cup\, a,b,d \;:\; a{\circ}R{\circ}b \neq \bot\!\!\bot \wedge b{\circ}S{\circ}d \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}b{\circ}\top\!\!\top{\circ}d \rangle \\
= & \{ \quad \text{cone rule: (4.16), and } b \neq \bot\!\!\bot \quad \} \\
 & \langle \cup\, a,b,d \;:\; a{\circ}R{\circ}b \neq \bot\!\!\bot \wedge b{\circ}S{\circ}d \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}d \rangle \\
= & \{ \quad \text{disjunction rule of the quantifier calculus} \quad \} \\
 & \langle \cup\, a,d \;:\; \langle \exists b \;::\; a{\circ}R{\circ}b \neq \bot\!\!\bot \wedge b{\circ}S{\circ}d \neq \bot\!\!\bot \rangle \;:\; a{\circ}\top\!\!\top{\circ}d \rangle \quad .
\end{array}
$$

That is, with dummies a, b and d ranging over proper atomic coreflexives,

(5.50) $R{\circ}S \;=\; \langle \cup\, a,d : \langle \exists b \;::\; a{\circ}R{\circ}b \neq \bot\!\!\bot \wedge b{\circ}S{\circ}d \neq \bot\!\!\bot \rangle : a{\circ}\top\!\!\top{\circ}d \rangle$

A similar calculation gives the standard pointwise definition of the converse of a relation.

(5.51) $R^{\cup} \;=\; \langle \cup\, a,b : a{\circ}R{\circ}b \neq \bot\!\!\bot : b{\circ}\top\!\!\top{\circ}a \rangle$.

In these calculations, the boolean $a{\circ}R{\circ}b \neq \bot\!\!\bot$ plays the role of $a[\![R]\!]b$ in conventional reasoning. Atomic coreflexives a and b thus play the role of points and an event of the form $a{\circ}\top\!\!\top{\circ}b$ models the pair (a, b) in conventional pointwise reasoning. In this way, pointwise statements in conventional reasoning can be mechanically translated into statements in our formal axiomatic system.

In the opposite direction, translating point-free statements into pointwise statements involves exploiting the fact that the lattice of relations is saturated and atomic. This allows a relation to be rewritten as the supremum of set of atoms $a{\circ}\top\!\!\top{\circ}b$ in the same way that in conventional reasoning a relation is expressed as the union of a set of pairs. Typically (as illustrated above) this involves the introduction of quantifiers, including universal and/or existential quantifiers.

As in example, this is how (5.38) is justified within the formal system we have presented.

$$\begin{array}{ll}
 & f \circ f^{\cup} \\
= & \{ \quad \text{theorem 5.22} \quad \} \\
 & \langle \cup\, a{,}b \;:\; a \circ f \circ f^{\cup} \circ b \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}b \rangle \\
= & \{ \quad \text{pointwise definitions: (5.50) and (5.51)} \\
 & \quad\;\; \text{and quantifier calculus (range disjunction)} \quad \} \\
 & \langle \cup\, a{,}b{,}c \;:\; a{\circ}f{\circ}c \neq \bot\!\!\bot \wedge b{\circ}f{\circ}c \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}b \rangle \quad .
\end{array}$$

So,

$$\begin{array}{ll}
 & \langle \forall\, a{,}b \;:\; \langle \exists c \;::\; a{\circ}f{\circ}c \neq \bot\!\!\bot \wedge b{\circ}f{\circ}c \neq \bot\!\!\bot \rangle \;:\; a = b \rangle \\
= & \{ \quad \text{range disjunction} \quad \} \\
 & \langle \forall\, a{,}b{,}c \;:\; a{\circ}f{\circ}c \neq \bot\!\!\bot \wedge b{\circ}f{\circ}c \neq \bot\!\!\bot \;:\; a = b \rangle \\
= & \{ \quad (5.20) \quad \} \\
 & \langle \forall\, a{,}b{,}c \;:\; a{\circ}f{\circ}c \neq \bot\!\!\bot \wedge b{\circ}f{\circ}c \neq \bot\!\!\bot \;:\; a{\circ}\top\!\!\top{\circ}b \subseteq I \rangle \\
= & \{ \quad \text{above and property of supremum} \quad \} \\
 & f \circ f^{\cup} \subseteq I \quad .
\end{array}$$

Of course, it is impossible to avoid pointwise reasoning. All the meta-reasoning we do is pointwise —the "points" are the events in our axiom system— and, within our axiom system, it is sometimes necessary to exploit saturation and atomicity.

When reasoning about algorithms in later sections, much of the reasoning becomes pointwise. The "points" are states of the program and properties of the states are expressed pointwise in terms of the values of the program variables. For this reason, it is important to consider the different ways that properties are formulated.

Just as, in conventional reasoning, $a[\![R]\!]b$ and $(a,b) \in R$ have the same meaning —implicitly exploiting the isomorphism between a subset of a power set and its characteristic (boolean-valued) function— the two expressions $a{\circ}R{\circ}b \neq \bot\!\!\bot$ and $a{\circ}R{\circ}b = a{\circ}\top\!\!\top{\circ}b$

have the same meaning. Indeed, there are many different but equivalent expressions in conventional pointwise reasoning; similarly, there are often different, but equivalent ways of translating informal expressions into the formal calculus.

When formulating proof rules for reasoning about algorithms, we typically choose to represent guards and assertions by coreflexives. See, for example, the induction theorem for reasoning about depth-first search presented in chapter 12. However, guards and assertions in programs are invariably expressed as boolean functions of the state space. Consequently, when applying the proof rules we need a formal mechanism for translating between the language of coreflexive relations and boolean functions. Below we formulate the translation.

The type Bool has two elements true and false. Let us use TRUE and FALSE to denote points of type $\mathsf{Bool}{\sim}\mathsf{Bool}$ representing the subsets $\{\mathsf{true}\}$ and $\{\mathsf{false}\}$, respectively. Suppose State is a set. The name is chosen on account of the application: State is the state space of a program segment. Then the function

$$\langle P :: (\mathsf{TRUE}\circ P){>}\rangle$$

maps a function P of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$ into a coreflexive of type $\mathsf{State}{\sim}\mathsf{State}$ that represents the set of states σ for which $P.\sigma$ is true. Conversely, the function

$$\langle p :: \mathsf{TRUE}\circ\top\!\!\top\circ p \cup \mathsf{FALSE}\circ\top\!\!\top\circ p{>}\!\bullet\rangle$$

maps a coreflexive p of type $\mathsf{State}{\sim}\mathsf{State}$ into a (total) function of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$. (Note that $p{>}\!\bullet = {\sim}(p{>}) = {\sim}p$.)

Lemma 5.52 Suppose P is a total function of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$. Then

$$(\mathsf{TRUE}\circ P){>}\!\bullet = (\mathsf{FALSE}\circ P){>}$$

and

$$(\mathsf{FALSE}\circ P){>}\!\bullet = (\mathsf{TRUE}\circ P){>} \ .$$

Proof By mutual inclusion. First,

$$
\begin{array}{cl}
 & (\mathsf{TRUE}\circ P){>}\!\bullet \supseteq (\mathsf{FALSE}\circ P){>} \\
= & \{ \quad \text{definition of complemented domain: (5.11)} \quad \} \\
 & \mathsf{TRUE}\circ P\circ(\mathsf{FALSE}\circ P){>} = \bot\!\!\bot \\
= & \{ \quad \text{(5.6) with “}{>}\text{” replaced by “}{<}\text{”} \quad \} \\
 & (\mathsf{TRUE}\circ P\circ(\mathsf{FALSE}\circ P){>}){<} = \bot\!\!\bot \\
= & \{ \quad \text{(5.9) and 5.7(c)} \quad \}
\end{array}
$$

$(\mathsf{TRUE}\circ \mathsf{P}\circ \mathsf{P}^{\cup}\circ \mathsf{FALSE})_{<} \;=\; \perp\!\!\perp$

$=\quad\{\quad$ (5.6) with “$>$” replaced by “$<$” $\quad\}$

$\mathsf{TRUE}\circ \mathsf{P}\circ \mathsf{P}^{\cup}\circ \mathsf{FALSE} \;=\; \perp\!\!\perp$

$\Leftarrow\quad\{\quad$ assumption: P is a function, so $\mathsf{P}\circ \mathsf{P}^{\cup}\subseteq I_{\mathsf{Bool}}$ $\quad\}$

$\mathsf{TRUE}\circ \mathsf{FALSE} \;=\; \perp\!\!\perp$

$=\quad\{\quad \mathsf{FALSE}=\sim\mathsf{TRUE} \quad\}$

true .

Second,

$(\mathsf{TRUE}\circ \mathsf{P})_{>\bullet} \;\subseteq\; (\mathsf{FALSE}\circ \mathsf{P})_{>}$

$=\quad\{\quad$ complements $\quad\}$

$(\mathsf{TRUE}\circ \mathsf{P})_{>\bullet}\circ(\mathsf{FALSE}\circ \mathsf{P})_{>\bullet} \;\subseteq\; \perp\!\!\perp$

$=\quad\{\quad$ complements, $\sim\perp\!\!\perp = I$ $\quad\}$

$(\mathsf{TRUE}\circ \mathsf{P})_{>}\cup(\mathsf{FALSE}\circ \mathsf{P})_{>} \;\supseteq\; I_{\mathsf{State}}$

$=\quad\{\quad$ distributivity $\quad\}$

$((\mathsf{TRUE}\cup \mathsf{FALSE})\circ \mathsf{P})_{>} \;\supseteq\; I_{\mathsf{State}}$

$=\quad\{\quad \mathsf{TRUE}\cup \mathsf{FALSE} = I_{\mathsf{Bool}} \quad\}$

$\mathsf{P}_{>} \;\supseteq\; I_{\mathsf{State}}$

$=\quad\{\quad$ assumption: P is total $\quad\}$

true .

Combining the two calculations, we have proved the first equation. The second is obtained by interchanging TRUE and FALSE.
□

Theorem 5.53 For all coreflexives p of type $\mathsf{State}{\sim}\mathsf{State}$,

$$p \;=\; (\mathsf{TRUE}\circ(\mathsf{TRUE}\circ\top\!\!\top\circ p \;\cup\; \mathsf{FALSE}\circ\top\!\!\top\circ p_{\bullet}))_{>}$$

and for all total functions P of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$

$$\mathsf{P} \;=\; \mathsf{TRUE}\circ\top\!\!\top\circ(\mathsf{TRUE}\circ \mathsf{P})_{>} \;\cup\; \mathsf{FALSE}\circ\top\!\!\top\circ(\mathsf{TRUE}\circ \mathsf{P})_{>\bullet} \;\; .$$

That is, the functions

$$\langle \mathsf{P} :: (\mathsf{TRUE}\circ \mathsf{P})_{>}\rangle \;\; ,$$

which maps a function P of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$ into a coreflexive of type $\mathsf{State}{\sim}\mathsf{State}$, and

$$\langle p \;::\; \mathsf{TRUE}{\circ}\top\!\!\top{\circ}p \,\cup\, \mathsf{FALSE}\circ\top\!\!\top\circ p{>\!\bullet}\rangle \;\;,$$

which maps a coreflexive p of type $\mathsf{State}{\sim}\mathsf{State}$ into a total function of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$, are inverses of each other.

Proof

$$
\begin{array}{ll}
 & (\mathsf{TRUE}\circ(\mathsf{TRUE}{\circ}\top\!\!\top{\circ}p \,\cup\, \mathsf{FALSE}\circ\top\!\!\top\circ p{>\!\bullet})){>} \\
= & \{ \quad \text{distributivity and } \mathsf{TRUE}{\circ}\mathsf{FALSE} = \bot\!\!\bot \quad \} \\
 & (\mathsf{TRUE}\circ\mathsf{TRUE}\circ\top\!\!\top\circ p){>} \\
= & \{ \quad \mathsf{TRUE} \text{ is a proper coreflexive, so } \mathsf{TRUE}\circ\mathsf{TRUE} = \mathsf{TRUE} \neq \bot\!\!\bot \\
 & \quad\;\; 5.7(e) \quad \} \\
 & p{>} \\
= & \{ \quad p \text{ is a coreflexive} \quad \} \\
 & p \;\;.
\end{array}
$$

Also,

$$
\begin{array}{ll}
 & \mathsf{TRUE}\circ\top\!\!\top\circ(\mathsf{TRUE}\circ P){>} \,\cup\, \mathsf{FALSE}\circ\top\!\!\top\circ(\mathsf{TRUE}\circ P){>\!\bullet} \\
= & \{ \quad 5.7(a) \text{ and lemma } 5.52 \quad \} \\
 & \mathsf{TRUE}\circ\top\!\!\top\circ\mathsf{TRUE}\circ P \,\cup\, \mathsf{FALSE}\circ\top\!\!\top\circ\mathsf{FALSE}\circ P \\
= & \{ \quad \mathsf{TRUE} \text{ and } \mathsf{FALSE} \text{ are points, definition } 5.13(c) \quad \} \\
 & \mathsf{TRUE}\circ P \,\cup\, \mathsf{FALSE}\circ P \\
= & \{ \quad \text{distributivity, } \mathsf{TRUE}\cup\mathsf{FALSE} = I_{\mathsf{Bool}} \quad \} \\
 & P \;\;.
\end{array}
$$

□

Theorem 5.53 is the formal basis for switching between functions of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$ and coreflexives of type $\mathsf{State}{\sim}\mathsf{State}$ to represent assertions and conditions in programs. See also section 6.8.5.

5.6 Bibliographic Remarks

Relation algebra was first developed in the 19th century by De Morgan [DM60], Peirce [Pei70] and Schröder [Sch95], and further developed in the mid 20th century by Tarski

[Tar41] and his students. Histories of its development are by Maddux [Mad91] and Pratt [Pra92].

Our presentation has its origins in a research project aimed at developing a relational theory of datatypes [ABH$^+$92]. See also [DBvdW97], [Hoo97] and [Voe99]. The all-or-nothing rule and the notions of complementation-idempotent and complementation-fixed closure operator are from [Glü17].

Part II

Semantics of Imperative Programs

Chapter 6

Imperative Programming

In later chapters, we derive several graph algorithms. The algorithms are presented as imperative programs and their correctness is formulated using standard techniques. We assume that the reader has already seen several examples. For introductions, see (for example) [Gri81, Bac03]. This chapter is about expressing the semantics of the programs in relation algebra. In order not to burden the reader with details that are not relevant later, some simplifications have been made, particularly with respect to the discussion of program termination and the difference between so-called "angelic" and "demonic" nondeterminism. See [BW93] for more details.

Programs are syntactic entities, the chosen syntax depending on the choice of programming language. Here we use a Pascal-like language comprising *assignment statements*, *sequential composition*, **while** *statements*, *conditional statements* and *choice statements*. For us *assertions* —a mechanism for documenting a program— also form an integral part of the syntax of a programming language. We often refer to components of a program as *program segments*. For example, the composition of an assignment statement and a **while** statement might be referred to as a program segment.

We also admit so-called "recursive" programs. A recursive program is a program that is defined by an "equation" in which the left side of the "equation" is the name of the program and the right side is a program segment that includes the name of the program. That is, the name of the program "recurs" in its definition.

The basic unit of syntax is an *identifier*; identifiers are the names given to *constants* and *variables*. Program segments are *parameterised* by constants, which include items that are normally understood as "constants", like the number 0, but also other items like the type Node, the less-than ordering relation on numbers and the subset ordering on subsets of Node, and functions like set union, etc. A "constant" is thus any entity named implicitly or explicitly in the program segment that is unchanged by execution of the program segment. Variables are named entities whose value changes during execution.

The variables that are *in scope* in a program segment determine its *state space*. For brevity, we omit explicit declarations of variables and their scope. Inspection of the variables referred to in a program is usually sufficient to determine the state space. For example, if a program segment refers to variables a of type Node and s of type $\mathsf{SetOfNode}$ then the state space of the program is the cartesian product $\mathsf{Node}\times\mathsf{SetOfNode}$. (The program may also refer to a constant G of type Graph.)

Choice statements augment the state space by introducing one or more variables that satisfy a given specification (the choice criterion). The *scope* of these variables is delimited by **begin-end** bracketing.

Assertions also sometimes extend the state space by the introduction of *ghost* variables. Ghost variables help to document the relation between input and output values at certain points in the execution of the code. In order to distinguish ghost variables from other variables we use subscripting as in, for example, σ_0. Typically, the ghost variable σ_0 would be used to relate the value of program variable σ at some point in the execution of the program to its initial value. Ghost variables σ_1, σ_2, etc. might be used to relate the value of σ to its value at certain intermediate points in the execution of the program.

Occasionally it is necessary to introduce additional *auxiliary* variables. Auxiliary variables play no role in the computation itself but, like ghost variables, are an aid to documenting a program. Auxiliary variables differ from ghost variables in that they do appear on the left side of assignment statements whereas ghost variables do not.

If a program segment $\mathcal{P}$ depends on variables xs, we often write $\mathcal{P}(xs)$. This notation does *not* denote function application. Instead, it is used to express *syntactic substitution*. For example, suppose we have an assertion $x+y=x^2$. In order to reason about the assertion, we might give it the name $p(x,y)$. Then by (for example) $p(x+1\,,y)$, we mean the syntactic entity $(x+1)+y=(x+1)^2$ obtained by substituting every occurrence of the symbol "x" in the assertion by "$(\,x+1\,)$".

6.1 Specifications

Programs are often described as defining an "input-output" relation. This suggests that the semantics of a program is a heterogeneous relation of type $\mathrm{Out}{\sim}\mathrm{In}$ for some types Out and In. This is not how we define the semantics of a program.

Programs are invariably *parameterised* by a number of entities which define the input of the program. Typically, some of the variables in a program are input parameters. The use of the word "variable" is then arguably misleading: the input "variables" are *constants* in the sense that their values are unchanged by execution of the program. A program may also be parameterised by other entities that are not normally called "variables"; these include types and relations. The *state space* of the program is defined by the variables that are not constants and the *output* of the program is specified as the final values of some subset of the state-space variables.

For example, we consider in section 6.9 an algorithm to calculate the least fixed point of a function of type $\mathcal{A}\leftarrow\mathcal{A}$ for some partially ordered set $(\mathcal{A},\preceq)$. The algorithm employs two variables, F and x, the value of F being constant whilst the value of x is continually updated during execution of the algorithm. Thus F is an input parameter and x defines the state space. The partially ordered set $(\mathcal{A},\preceq)$ is also a parameter of the algorithm.

A simple syntactic check enables the distinction between "constants" and truly varying "variables" in a program: the constants are the "variables" that do not occur on the left side of any assignment statement. Sometimes, however, programs are written that do assign to input variables, ghost variables then being necessitated in order to specify the program. We avoid this practice. Indeed, so that the reader can more easily distinguish constants from variables, our practice is to use lower-case identifiers (like "$seen$") to name variables; symbols (like "$\preceq$") and identifiers beginning with an upper-case letter are used to name constants.

When defining the semantics of a program, it is desirable to clearly separate the issue of termination from other issues. Whether or not a program terminates for given input values is governed by its so-called *operational semantics*: how the program is interpreted and executed. We do not present an operational semantics of programs but we do show how to determine whether or not individual program segments terminate. Termination of composite programs is defined to be *demonic*: that is, a program is guaranteed to terminate only if all segments of the program are guaranteed to terminate. We specify the semantics of programs only for *terminating* programs, by which we mean programs that are guaranteed to terminate for all inputs satisfying a given specification.

Formally, the semantics of a program segment is a function with *target*

$$\mathsf{State}\cup\{\bot\}\sim\mathsf{State}$$

where State is the type of the state space (typically a cartesian product of the types of the program variables) and $\bot$ expresses non-termination; the *source* of the function is a (typically quite complex) collection of types, operators, relations and values satisfying certain properties. A *terminating* program segment is one that is guaranteed to always terminate; the semantics of a terminating program segment is thus a function that maps the parameters of the program to a homogeneous relation on the state space. Less formally, a (terminating) program segment is a possibly non-deterministic, parameterised *state transformer*.

This view of program segments as parameterised state transformers allows us to restrict attention to homogeneous relations. In this way, we avoid the clutter of type checking. In what follows, the parameters will be implicit. Whenever we formulate a rule, it is to be understood that the rule is universally quantified over all possible values of the parameters. See section 6.4 on verification conditions for further discussion.

A *specification* is a triple $(\mathsf{Context},\mathsf{P},\mathsf{R})$ where $\mathsf{Context}$ specifies properties of the input parameters —including the state space State— , P is a (parameterised) predicate of type $\mathsf{Bool}\leftarrow\mathsf{State}$, and R is a (parameterised) homogeneous relation of type $\mathsf{State}\sim\mathsf{State}$.

Specifications are typically non-deterministic —for given input values, different output values may be acceptable— but program segments typically resolve some of the non-determinacy, if not all. (In the extreme cases, functional programs are deterministic: each input value yields exactly one output value.) Program segments are relations that can be expressed in a restricted language. A program segment Prog with meaning $[\![\mathsf{Prog}]\!]$ is said to *meet* specification $(\mathsf{Context},\mathsf{P},\mathsf{R})$ if, for all possible parameter values satisfying the predicate $\mathsf{Context}$, it is *conditionally correct*, i.e. $[\![\mathsf{Prog}]\!]\circ[\![\mathsf{P}]\!]\subseteq\mathsf{R}$ (where $[\![\mathsf{P}]\!]$ is the coreflexive corresponding to the predicate P) and it is guaranteed to terminate.

The type of the semantics of a program segment is most often a so-called "dependent type". For example, the fixed-point algorithm mentioned above has three inputs: a type $\mathcal{A}$, and an ordering relation and a function both of whose types depend on $\mathcal{A}$. The precise details form what we have called the "context" of the algorithm. Typically, the context embodies a great amount of detail most of which is implicit in informal accounts. When program verification is made formal and/or automated it becomes necessary to be explicit about the context. For example, programs that manipulate variables declared as "integers" often rely on properties of the less-than relation on natural numbers, these properties being implicit in the specification but vital to formal proof. The level of detail that is required is just too much for human consumption, and much of it is well known in any case. This is why we choose to define the semantics of program segments as parameterised state transformers whereby the parameters are left implicit.

We call P the *precondition* of the specification. It is common to combine the context and the precondition into one; previously, we have also done so. We now prefer to distinguish the two in order to emphasise that $\mathsf{Context}$ specifies properties that remain true throughout execution of the program segment. "Preconditions" (and "postconditions") are properties that hold only at certain points during the execution. Because the context is a constant of any implementation, it is most often an implicit parameter of the discussion that follows.

6.2 Structures

In order to present the semantics of program segments without making the context explicit, we exploit the insights on "structures" introduced by Dijkstra and Scholten

[DS90]. A "structure" is simply an expression that denotes a parameterised value where the parameters are not made explicit. As observed by Dijkstra and Scholten, reasoning about "structures" requires more care than is usual in traditional mathematics with regard to the overloading of operators, in particular the equality symbol. Specifically, given two structures A and B, the expression "$A = \mathrm{B}$" might denote a boolean structure or a boolean scalar.

For example, suppose $\mathbf{A}$ and $\mathbf{B}$ are vectors of the same (implicit) type and dimension. Then $\mathbf{A}=\mathbf{B}$ can be interpreted in two ways. Interpreting it as a structure, $\mathbf{A}=\mathbf{B}$ denotes a boolean vector of the same dimension as $\mathbf{A}$ and $\mathbf{B}$, the entries in the vectors being true or false depending on whether or not the corresponding entries in $\mathbf{A}$ and $\mathbf{B}$ are equal. Interpreting it as a scalar, $\mathbf{A}=\mathbf{B}$ is a boolean: it is true or false depending on whether or not $\mathbf{A}$ and $\mathbf{B}$ are *everywhere* equal (i.e. all corresponding entries in $\mathbf{A}$ and $\mathbf{B}$ are equal.)

Traditional practice in mathematics is to assume that $\mathbf{A}=\mathbf{B}$ denotes a boolean scalar. However, as argued by Dijkstra and Scholten [DS90], this practice is undesirable if the goal is to combine precision with concision in calculational reasoning. Their solution is straightforward as well as aesthetically pleasing. The expression $\mathbf{A}=\mathbf{B}$ is defined to be a structure (of the same shape as $\mathbf{A}$ and $\mathbf{B}$) and $[\mathbf{A}=\mathbf{B}]$ is the boolean scalar. The square brackets are called "everywhere" brackets, and $[\mathbf{A}=\mathbf{B}]$ is read as "$\mathbf{A}$ is everywhere equal to $\mathbf{B}$", or simply "everywhere $\mathbf{A}=\mathbf{B}$". The same device is applied to other operators. For example, if $\mathbf{A}$ and $\mathbf{B}$ are boolean vectors, $\mathbf{A}\Leftarrow\mathbf{B}$ is a boolean vector and $[\mathbf{A}\Leftarrow\mathbf{B}]$ is a boolean scalar. Similarly, if $\mathbf{A}$ and $\mathbf{B}$ are integer vectors, $\mathbf{A}<\mathbf{B}$ is a boolean vector and $[\mathbf{A}<\mathbf{B}]$ is a boolean scalar.

A potential drawback is that "everywhere" brackets become ubiquitous, particularly in formal calculations. Dijkstra and Scholten [DS90] avoid this by introducing conventions in the format of proofs that enable everywhere brackets to be omitted. We use the same conventions here. Another drawback is the unfamiliarity of most readers with the use of "everywhere" brackets. In order to avoid the dangers of misinterpretation that this may cause —see [BN98]— , we avoid the use of "everywhere" brackets when reasoning about concrete algorithms. This entails the introduction of new "pointed" operator symbols for each of the classical operator symbols in the context of the algorithm. For example, in our discussion of the semantics of depth-first search in section 11.2, we define the operators $\dot{\subseteq}$ and $\dot{\circ}$ as pointwise extensions of the subset relation and composition, respectively. See definition 6.6 for an example where the everywhere brackets eliminate this notational burden.

Dijkstra and Scholten presented a predicate-transformer semantics of imperative programs. We present a relational semantics. Returning to our earlier discussion, we formulate the definition of a program segment meeting a specification below. In doing so, we carry out our intention of making the specification of the input parameters —the

Context component of a specification— implicit. That is, in the definition, the meaning of a program segment is a function of the context, and the square "everywhere" brackets denote universal quantification over all input variables satisfying the Context predicate. (If P is a predicate on the context, many authors would prefer to write $\text{Context} \vDash P$ rather than $[P]$.)

Definition 6.1 Suppose $(\text{Context},P,R)$ is a specification. Suppose p denotes the coreflexive corresponding to precondition P. The *meaning* $[\![S]\!]$ of a program segment S with state space State in the context Context is a homogeneous relation on State. The program segment S *meets* the relation R *under precondition* P iff:

(i) It is *conditionally correct*. That is,

$$[\ [\![S]\!]\circ p \ \subseteq\ R\ ]\ \ .$$

(ii) It is *total*. That is,

$$[\ p \subseteq [\![S]\!]{>}\ ]\ \ .$$

(iii) It is (everywhere) *terminating*.

□

"Conditionally correct" is often called "partially correct". We prefer to use "conditional" because to say that something is "partially" correct suggests that it is also partially incorrect. Totality becomes an issue primarily when choice statements are used. See section 6.7. Termination becomes an issue when program segments involve loops and/or recursion. It is often established by introducing a so-called *bound function*. That is, some finiteness assumption is made about the input parameters and this is used to predict an upper bound on the number of operations used when executing the algorithm. In the case of the algorithms we present in this document, termination is relatively easy to verify and most effort is expended on establishing conditional correctness. For further discussion of the formal basis of bound functions, see section 6.8.6.

6.3 Assertions

We use *assertions* both to document programs and to document the structure of the verification that a program meets its specification.

Assertions are (parameterised) predicates on the state space. That is, they are functions of type $\text{Bool}{\leftarrow}\text{State}$. Given predicates P and Q (the so-called precondition and postcondition, respectively)

$$Q^{\cup}\circ(\Leftarrow)\circ P$$

is a (parameterised) relation of type $\mathsf{State}{\sim}\mathsf{State}$. Specifically, it is the relation R defined by

$$[\ \langle\forall\,\sigma,\sigma' :: \sigma'\ R\ \sigma\ \equiv\ (Q.\sigma' \Leftarrow P.\sigma)\rangle\ ]\quad .$$

The combination of two predicates P and Q (in a given context) thus determines a specification where the precondition is P and the relation is R as defined above.

Of course, not all homogeneous relations can be expressed in this way. So-called "ghost variables" are used to circumvent this limitation. See the example below.

Suppose S is a program segment. Suppose predicates P and Q are formulated by the expressions $pre(\sigma)$ and $post(\sigma)$, respectively. (So the meaning of $pre(\sigma)$ is the predicate P, and similarly for $post(\sigma)$.) Then the expression

$$\{\ \ pre(\sigma)\ \ \}$$
$$S$$
$$\{\ \ post(\sigma)\ \ \}$$

has meaning S meets the relation $Q^{\cup}\circ(\Leftarrow)\circ P$ under precondition P. That is, $[\![S]\!]$ is conditionally correct, total and terminating. (See definition 6.1.) Expressed pointwise, the conditional correctness of S is the theorem

$$[\ \langle\forall\,\sigma,\sigma'\ :\ [\![pre(\sigma)]\!] \wedge \sigma'[\![S]\!]\sigma\ :\ [\![post(\sigma')]\!] \Leftarrow [\![pre(\sigma)]\!]\rangle\ ]$$

which is equivalent to

$$[\ \langle\forall\,\sigma,\sigma'\ :\ [\![pre(\sigma)]\!] \wedge \sigma'[\![S]\!]\sigma\ :\ [\![post(\sigma')]\!]\rangle\ ]\quad .$$

(Note that σ' is on the left in $\sigma'[\![S]\!]\sigma$ and σ is on the right. This is a matter of convention. For us, the "output" of a relation is on the left and its "input" is on the right. The syntax of assertions is that the output is at the bottom and the input is at the top.) The point-free formulation of S meeting the relation $Q^{\cup}\circ(\Leftarrow)\circ P$ under precondition P is the theorem

$$[\ \ [\![S]\!]\circ(\mathsf{TRUE}{\circ}P)_{>}\ \subseteq\ Q^{\cup}\circ(\Leftarrow)\circ P\ \ ]\quad .$$

Just like the pointwise formulation, this has an equivalent form, namely:

$$(6.2)\qquad [\ \ ([\![S]\!]\circ P^{\cup}\circ\mathsf{TRUE})_{<}\ \subseteq\ (\mathsf{TRUE}{\circ}Q)_{>}\ \ ]\quad .$$

A program segment may meet several different specifications. For example, the assignment $i:=i+1$ meets the greater-than relation on numbers, as well as the at-least relation and the relation given by the pair $(1,0)$ (that is, if i has initial value 0, after the assignment it has value 1). Using the ghost variable i_0 to capture the initial state, we can document the first of these by

$\{ \quad i = i_0 \quad \}$

$i := i+1$

$\{ \quad i > i_0 \quad \}$

and the last by

$\{ \quad i = 0 \quad \}$

$i := i+1$

$\{ \quad i = 1 \quad \} \quad .$

The input parameters in this case are the integers, the addition operator and greater-than relation on integers, and the constants 0 and 1. We regard them as parameters because a formal verification of the program segment will necessarily be based on assumptions about their algebraic properties, thus allowing other interpretations of the parameters.

6.4 Verification Conditions

Suppose that we want to show that a program segment S meets a given specification. Often this involves establishing one or more *verification conditions*. Suppose the specification is expressed by the assertions $pre(\sigma)$ and $post(\sigma)$ and suppose we document the program segment as follows:

$\{ \quad pre(\sigma) \quad \}$

S

$\{ \quad post(\sigma) \quad \} \quad .$

Then, in the simpler cases, it is possible to compute a so-called "weakest precondition" $wp(\sigma)$ guaranteeing the postcondition $post$ after execution of S. By definition of "weakest precondition" that S meets the specification is equivalent to proving the theorem

(6.3) $\quad [\ \langle \forall \sigma : [\![pre(\sigma)]\!] : [\![wp(\sigma)]\!] \rangle\ ] \quad .$

The formula (6.3) is called a *verification condition*. We sometimes document the construction of verification conditions as follows:

$\{ \quad pre(\sigma) \quad \}$

$\{ \quad wp(\sigma) \quad \}$

S

$\{ \quad post(\sigma) \quad \} \quad .$

Where two assertions are juxtaposed as here, the meaning is that the upper assertion implies the lower assertion everywhere. That is, the meaning of a juxtaposition of assertions is the verification condition (6.3).

6.5 Assignment Statements

An assignment statement is the imperative syntax for a function. If xs is a list of distinct variables and Es is a list of expressions of the same length as xs, then a first approximation to the meaning of the assignment $xs := Es$ is the function $\langle xs :: Es\rangle$.

Recall, however, that the state space of a program segment is typically a cartesian product, and the individual variables of the segment refer to specific components of the product. An assignment statement is a convenient mechanism for specifying a function of type $\mathsf{State}{\leftarrow}\mathsf{State}$ that affects only certain components of the product. For example, the assignment $i := E$, where i is a variable of type $\mathbb{N}$ may be a segment in a program with state space $\mathbb{N}{\times}\mathbb{Z}$ whereby the second component is referenced by an additional variable, x say. In such a context, the assignment is equivalent to the assignment $i,x := E,x$ and its meaning is the function $\langle i :: E\rangle \times I_{\mathbb{Z}}$ (equivalently, $\langle (i,x) :: (E,x)\rangle$). That is, an assignment statement $xs := Es$ is the identity function on those components that are not named in the list xs and the function $\langle xs :: Es\rangle$ on the components that are named.

We don't give any guidance on what are allowable expressions on the right side of an assignment except to say that the expressions must be implementable in a conventional programming language, and their evaluation (for particular input values) must be guaranteed to terminate — typically, but not necessarily, in "constant time". We rely on the reader's programming experience to decide whether or not this is the case.

Most often assignment statements are total functions. In general, the right domain is the subset of the state space on which the right side of the assignment is defined. For example, the assignment $i := i{\div}j$ is defined on state spaces such that the value of variable j is non-zero.

Because assignments are functional, it is easy to derive the *assignment axiom.* (See below.) Specifically, the assignment axiom states that

$$\begin{array}{l} \{\ \ post(E)\ \ \} \\ \sigma := E \\ \{\ \ post(\sigma)\ \ \} \end{array}$$

is a theorem (i.e. it is true everywhere for all σ). Here $post(E)$ denotes the expression obtained by replacing all occurrences of σ in the expression $post(\sigma)$ by the expression "(E)". (We use quotation marks in order to emphasise that this is a syntactic substitution. The parentheses are necessary to avoid any error that might be caused by precedence conventions.)

Normally the assignment axiom is used to construct a verification condition. Suppose the assignment statement is documented as follows:

$$\begin{array}{l} \{\ \ pre(\sigma)\ \ \} \\ \sigma := E \\ \{\ \ post(\sigma)\ \ \}\ \ . \end{array}$$

Then we augment the documentation with the expression $post(E)$:

$$\begin{array}{l} \{\ \ pre(\sigma)\ \ \} \\ \{\ \ post(E)\ \ \} \\ \sigma := E \\ \{\ \ post(\sigma)\ \ \}\ \ . \end{array}$$

This then gives the verification condition:

$$(6.4)\qquad [\ \langle\forall\sigma : [\![pre(\sigma)]\!] : [\![post(E)]\!]\rangle\ ]\ \ .$$

(The point-free justification of the assignment axiom proceeds as follows. Suppose f is a function of type $State \leftarrow State$ and suppose P and Q are predicates on the state, i.e. functions of type $Bool \leftarrow State$. Then that f meets the relation $Q^{\cup} \circ (\Leftarrow) \circ P$ under precondition P is, by (6.2),

$$\left[\ (f \circ P^{\cup} \circ TRUE)_{<}\ \subseteq\ (TRUE \circ Q)_{>}\ \right]\ \ .$$

But,

$$\begin{array}{ll} & (f \circ P^{\cup} \circ TRUE)_{<}\ \subseteq\ (TRUE \circ Q)_{>} \\ = & \{\qquad \text{isomorphism of coreflexives and conditions}\quad \} \\ & f \circ P^{\cup} \circ TRUE \circ \top\!\!\top\ \subseteq\ Q^{\cup} \circ TRUE \circ \top\!\!\top \\ = & \{\qquad f \text{ is a function}\quad \} \\ & P^{\cup} \circ TRUE \circ \top\!\!\top\ \subseteq\ f^{\cup} \circ Q^{\cup} \circ TRUE \circ \top\!\!\top \\ = & \{\qquad \text{converse and isomorphism of coreflexives and conditions}\quad \} \\ & (TRUE \circ P)_{>}\ \subseteq\ (TRUE \circ Q \circ f)_{>}\ \ . \end{array}$$

The coreflexive $(TRUE \circ P)_{>}$ corresponds to the set of states for which P holds, and $(TRUE \circ Q \circ f)_{>}$ corresponds to the set of states σ for which Q holds of $f.\sigma$. The property

$$[\ (TRUE \circ P)_{>}\ \subseteq\ (TRUE \circ Q \circ f)_{>}\ ]$$

is the point-free formulation of the verification condition (6.4).)

6.6 Sequential Composition

The meaning of the sequential composition $S1 \,;\, S2$ is the so-called *demonic composition* of the meanings of $S1$ and $S2$. Formally,

$$[\![S1 \,;\, S2]\!] \quad = \quad [\![S2]\!] \circ [\![S1]\!] \circ [\![S1]\!] \backslash [\![S2]\!]_{>}$$

where $[\![S1]\!] \backslash [\![S2]\!]_{>}$ is a coreflexive. How this coreflexive is defined is not needed here. Its rôle is to restrict the right domain of $S1$ to values that guarantee that execution of $S1$ results in values that are elements of the right domain of $S2$. (See [BW93] for full details.)

In practice, the complications of the definition of demonic composition are avoided by establishing that $[\![S1]\!]_{<} \subseteq [\![S2]\!]_{>}$, in which case it equals the so-called *angelic composition*

$$[\![S2]\!] \circ [\![S1]\!] \quad .$$

The difference between demonic and angelic composition only becomes apparent when we consider choice statements.

Note the switch in the order of $S1$ and $S2$ ($S1 \,;\, S2$ versus $[\![S2]\!] \circ [\![S1]\!]$).

6.7 Choice Statements

Program segments in the algorithms we present commonly include *choice* statements, whereby a new variable is introduced and assigned —possibly non-deterministically— a value that satisfies some criterion.

As for composition, choice statements have a demonic (as opposed to angelic) semantics. (Again, see [BW93] for full details.) However, we avoid the complication by imposing a restriction on when the meaning of a choice statement is defined. Specifically, the meaning of the choice statement

```
begin
  choose $x$ such that $q(x,\sigma)$
; S
end
```

is a relation with right domain restricted to states that allow the *criterion* q to be satisfied; in this case, it is defined to be a supremum:

$$\langle \cup x : [\![q(x,\sigma)]\!] : [\![S]\!] \rangle \quad .$$

A choice statement introduces a new local variable with scope delimited by the **begin**-**end** bracketing; the state space of the statement S is thus assumed to be extended appropriately. This means that x is allowed to be a free variable in assertions about segments of S. However, assertions about the choice statement itself may not refer to the variable x.

Just as for expressions on the right side of assignment statements, choice criteria must be implementable in a conventional programming language, and their evaluation (for particular input values) must be guaranteed to terminate.

The operational meaning of a choice statement is that the variable x is assigned an initial value that satisfies the criterion q; then the program segment S is executed. We document a choice statement by adding assertions as shown below. The precondition pre and postcondition $post$ document the specification of the choice statement and are assumed to be given. Note that the precondition of the program segment S depends on the state σ and on x —reflecting the fact that the state space has been augmented— ; on the other hand, the postcondition does not depend on x.

```
{ $pre(\sigma)$ }
begin
  choose $x$ such that $q(x,\sigma)$
; { $pre(\sigma) \wedge q(x,\sigma)$ }
  $S$
  { $post(\sigma)$ }
end
{ $post(\sigma)$ }
```

In order to guarantee that such a choice statement meets a given specification, it is necessary to establish totality. (See definition 6.1(ii).) Supposing that the precondition is defined in the usual way by a predicate pre, the totality requirement becomes

$$[\,[\![pre(\sigma)]\!] \;\Rightarrow\; \langle \exists x :: [\![q(x,\sigma)]\!]\rangle\,] \quad .$$

The choice may be entirely deterministic: in particular, a statement of the form

```
begin
  choose $x$ such that $x = E$
; $S$
end
```

introduces a new local variable x that is initialised to the value of the expression E and has scope the program segment S. In this case, totality is immediate (except in less common cases where E may sometimes be undefined). When using such a deterministic choice statement, we omit the words “choose” and “such that” and write the choice in the standard way as an assignment statement. (A frequent occurrence is the initialisation of the program variables.)

6.8 Loops

So far, we have considered so-called “straight-line programs”: programs where termination is always guaranteed. In this section, we consider “loops” in the form of **while** statements.

The meaning of a **while** statement is the least fixed point of a so-called “recursive” equation. Depth-first search uses a more complex form of recursion; its meaning is discussed in section 11.2 and, more generally, in section 12. The loops we consider in this section are simpler because they are defined using the star operator of a regular algebra.

The meaning $[\![B]\!]$ of a guard B is a coreflexive, and the meaning of the statement **while** B **do** S is

$$\sim[\![B]\!] \circ ([\![S]\!] \circ [\![B]\!])^* \quad .$$

That is, it is the least solution of the equation

$$W\!:: \;\; [\; W \supseteq \; \sim[\![B]\!] \cup W \circ [\![S]\!] \circ [\![B]\!] \;] \quad .$$

This equation corresponds to the operational meaning of a **while** statement: the guard B is used to choose between terminating without a change of state —the operational meaning of the coreflexive $\sim[\![B]\!]$ —- or executing S and then “looping” back to execute the **while** statement again.

Termination of **while** statements is discussed in section 6.8.3.

(Although we haven’t discussed it here, a parameterised fixed point is a fixed point. This is a fundamental property of fixed points — so fundamental indeed that it is almost invariaby taken for granted. For reasons of expediency, we have omitted the relevant theory for now but we may include it at a later date.)

6.8.1 Invariant Relations

Given a specification comprising a relation R and a (coreflexive representation of a) precondition p, the key to constructing a loop implementing the specification is the invention of an invariant Inv. In the most general case, invariants are relations on the state space; in more specific cases, they are values or properties. In this subsection, we consider the most general case, whilst invariant values and properties are considered in subsection 6.8.4.

An invariant Inv is chosen in such a way that it satisfies three properties. First, the invariant can be "established" by some initialisation $Init$. Second, the combination of the initialisation the invariant, and some termination $Term$ satisfies the specification $Spec$. Third, the invariant is "maintained by" some loop body $Body$ whilst making progress towards termination.

These informal requirements can be made precise in a concise way. The components Inv, $Init$, $Term$ and $Body$ are all homogeneous binary relations on the (parameterised) state space, just like the specification $Spec$. Below we discuss how the tasks involved in showing that the implementations of these components meet the specification is achieved.

6.8.2 Conditional Correctness

The first requirement is that the invariant relation is total. That is,

$$[\ p \subseteq Inv{>}\ ] \quad .$$

Often this requirement is met trivially and needs no further discussion.

"Establishing" the invariant is the requirement that

$$[\ Init{\circ}p \subseteq Inv\ ] \quad .$$

In words, for all states σ' and σ such that σ satisfies the precondition p, if σ' is related by $Init$ to σ then σ' is also related by the invariant relation to σ.

That the combination of the termination and invariant satisfies the relation R is the requirement that

$$[\ Term{\circ}Inv \subseteq R\ ] \quad .$$

This is the requirement that for all states σ and σ'',

$$[\ \langle\forall\sigma' : \sigma''\ Term\ \sigma' \wedge \sigma'\ Inv\ \sigma : \sigma''\ Spec\ \sigma\rangle\ ]$$

(Here we see again the convention of placing input values on the right and output values on the left.)

Finally, that the invariant is maintained by the loop body is expressed by

$$[\ Body{\circ}Inv \subseteq Inv\ ] \quad .$$

Pointwise this is

$$[\ \langle\forall\sigma,\sigma',\sigma'' \ :\ \sigma''\ Body\ \sigma' \wedge \sigma'\ Inv\ \sigma \ :\ \sigma''\ Inv\ \sigma\rangle\ ] \quad .$$

So Body maps states σ' related by the invariant Inv to σ to states σ'' that are also related by Inv to σ.

Together these three properties guarantee that

$$[\ \mathit{Term}\circ\mathit{Body}^*\circ\mathit{Init}\circ p\ \subseteq\ R\ ]$$

since

$$\begin{array}{ll}
 & R \\
\supseteq & \{\quad [\ \mathit{Term}\circ\mathit{Inv}\subseteq R\ ]\quad\} \\
 & \mathit{Term}\circ\mathit{Inv} \\
\supseteq & \{\quad [\ \mathit{Body}\circ\mathit{Inv}\cup\mathit{Inv}\subseteq\mathit{Inv}\ ] \\
 & \qquad \text{hence }[\ \mathit{Body}^*\circ\mathit{Inv}\subseteq\mathit{Inv}\ ]\quad\} \\
 & \mathit{Term}\circ\mathit{Body}^*\circ\mathit{Inv} \\
\supseteq & \{\quad [\ \mathit{Init}\circ p\subseteq\mathit{Inv}\ ]\quad\} \\
 & \mathit{Term}\circ(\mathit{Body}\circ b)^*\circ\mathit{Init}\circ p\quad .
\end{array}$$

6.8.3 Totality and Termination

Our account of invariants needs to be further refined if we are to relate it to the implementation of loops by a **while** statement. Recall that Body specifies the body of the loop, and Term specifies the termination of the computation. The implementation of $\mathit{Term}\circ\mathit{Body}^*$ by a **while** statement demands that both relations Term and Body are partial and, more specifically, that their right domains are complementary.

Letting b denote the right domain of Body and $\sim b$ its complement (thus

$$[\ b\cup{\sim}b = I_{\mathsf{State}}\ \wedge\ b\cap{\sim}b = \perp\!\!\perp\ ]\quad ,$$

where I_{State} is the coreflexive of type $\mathsf{State}{\sim}\mathsf{State}$ representing the entire state space), we have

$$[\ \mathit{Term} = \mathit{Term}\circ{\sim}b\ \wedge\ \mathit{Body} = \mathit{Body}\circ b\ ]\quad .$$

Hence,

$$[\ \mathit{Term}\circ\mathit{Body}^*\circ\mathit{Init} = \mathit{Term}\circ{\sim}b\circ(\mathit{Body}\circ b)^*\circ\mathit{Init}\ ]\quad .$$

The statement

```
while b do
    S
```

is the implementation of $\sim b \circ (Body \circ b)^*$ provided that S implements the relation $Body$ under precondition[1] b. If $[\![S]\!] \circ b$ is (everywhere) well-founded, $\sim b \circ ([\![S]\!] \circ b)^*$ is, by the unique extension property of regular algebra, the unique solution of the equation:

$$W\text{::}\ [\ W\ =\ \sim b\ \cup\ W \circ [\![S]\!] \circ b\ ]\ \ .$$

Executing this equation is equivalent to executing the "recursive" program

$$W\ =\ \textbf{if}\ b\ \textbf{then}\ (S\,;W)\ \ .$$

The well-foundedness of $[\![S]\!] \circ b$ guarantees that the execution of the **while** statement will always terminate. It also guarantees that the implementation is total, provided that $Term$ and $Body$ have complementary right domains, and the initialisation $Init$ is total. Specifically, we have:

$$(Term \circ Body^* \circ Init)> \ =\ Init>$$

$=\qquad \{\qquad$ domain calculus $\quad\}$

$$((Term \circ Body^*)> \circ Init)> \ =\ Init>$$

$\Leftarrow\qquad \{\qquad I$ is the identity of composition $\quad\}$

$$(Term \circ Body^*)> \ =\ I$$

$=\qquad \{\qquad (Term \circ Body^*)>$ is the unique solution of the equation

$\qquad\qquad\qquad p\text{::}\ p = Term> \cup (p \circ Body)> \quad\}$

$$I\ =\ Term> \cup\ (I \circ Body)>$$

$=\qquad \{\qquad$ by assumption,

$\qquad\qquad\qquad Term$ and $Body$ have complementary right domains.

$\qquad\qquad\qquad$ In particular, $I = Term> \cup Body> \quad\}$

$$\text{true}\ \ .$$

The penultimate step needs further justification. The claim is that the equation

$$p\text{::}\ \ [\ p = Term> \cup (p \circ Body)>\ ]$$

has a unique solution provided that $Body$ is (everywhere) well-founded. This is easily derived from the uep of regular algebra (theorem 3.16). Specifically, for all homogeneous relations R, we have:

(6.5) $\quad R$ is well-founded $\equiv\ \langle\forall S,T\ ::\ T = S \cup T \circ R\ \equiv\ T = S \circ R^*\rangle\ \ .$

(See section 8.1 for more details.) Indeed, for all coreflexives p,

[1]Strictly, b is a coreflexive and what is meant here is the predicate corresponding to b.

$$
\begin{array}{ll}
& p = Term{>} \cup (p \circ Body){>} \\
= & \{ \quad \text{domain calculus.} \\
& \quad\quad \text{Specifically, } (\top\!\!\top \circ p){>} = p \text{ and } \top\!\!\top \circ R = \top\!\!\top \circ R{>} \quad \} \\
& \top\!\!\top \circ p \;=\; \top\!\!\top \circ Term \cup \top\!\!\top \circ p \circ Body \\
= & \{ \quad Body \text{ is well-founded, (6.5)} \quad \} \\
& \top\!\!\top \circ p = \top\!\!\top \circ Term \circ Body^* \\
= & \{ \quad \text{domain calculus (as above)} \quad \} \\
& p = (Term \circ Body^*){>} \quad .
\end{array}
$$

That is, $(Term \circ Body^*){>}$ is the unique solution of the above equation in p .

6.8.4 Invariant Properties and Invariant Values

In general, invariants are relations on the (parameterised) state space. Special cases of invariants are invariant *properties* and invariant *values*. Invariant properties will be familiar to many readers and invariant values possibly less so. Nevertheless, we begin with values because formally they are simpler.

Consider a simple example: Suppose the state space is a cartesian product of two sets ranged over by program variables x and y . Then, obviously, an assignment $x := E$ has no effect on the value of program variable y . We say that the value of y is an invariant of the assignment. A slightly more complex example is given by the assignment $x{,}y \;:=\; x{+}1\,{,}\,y{+}1$ (with state space $\mathsf{Int}{\times}\mathsf{Int}$); in this case the value of $x{-}y$ is an invariant of the assignment.

In general, a "value" is given by a total function on the state space. Let us denote such a function by h . Then that the "value" is an invariant of program segment S equivales the relation $h^{\cup} \circ h$ is an invariant of S . That is, $[\; S \subseteq h^{\cup} \circ h \;]$. Equivalently, $[\; h{\circ}S \subseteq h \;]$; alternatively, if σ and σ' denote successive states during execution of S (i.e. $\sigma'[\![S]\!]\sigma$), $h.\sigma' = h.\sigma$. For example, the "value" $x{-}y$ is given by the function mapping the pair (x,y) to $x{-}y$. This function is an invariant "value" of the assignment $x{,}y \;:=\; x{+}1\,{,}\,y{+}1$ because $(x{+}1){-}(y{+}1) = x{-}y$ is a theorem of arithmetic.

If h is a total function on the state space, the relation $h^{\cup} \circ h$ is reflexive and transitive. This is an important property of invariant values when reasoning about loops and recursion. (In fact, $h^{\cup} \circ h$ is an equivalence relation. However, in this context symmetry is not relevant.)

Let us now turn to invariant "properties". Suppose h is a boolean function of the state space (a function of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$) and let S be a relation (typically, the semantics of a program segment). Then h is an *invariant property* of S if

$$[\, S \subseteq h^{\cup} \circ (\Leftarrow) \circ h \,] \quad .$$

Expressed pointwise, h is an *invariant property* of S if

$$[\ \langle \forall \sigma',\sigma : \sigma' [\![S]\!] \sigma : h.\sigma' \Leftarrow h.\sigma \rangle\ ] \quad .$$

Follows-from of boolean-function values is obviously a reflexive and transitive relation. It follows that, if h is a total boolean function of the state space, the relation $h^{\cup} \circ (\Leftarrow) \circ h$ is also reflexive and transitive. More generally, if h is a total function and R is a homogeneous relation on the range of h, the relation $h^{\cup} \circ R \circ h$ is reflexive if R is reflexive and transitive if R is transitive. As for invariant values, this is important when reasoning about loops and recursion.

Invariant properties sometimes occur naturally but, more commonly, are introduced artificially through the use of so-called "ghost" variables. A "ghost" variable records the state before execution of a program segment S but, unlike auxiliary variables, a "ghost" variable is not made explicit in the program code. Instead, the convention is that a subscript "0" is used to denote the initial value of variable.

6.8.5 Truthifying and Maintaining Invariant Properties

When reasoning about loops, we often say that a property is "truthified" by the initialisation, and "maintained" by the body of the loop. Let us formulate these concepts.

Definition 6.6 Suppose P has type $\mathsf{Bool} \leftarrow \mathsf{State}$. Then a relation S *truthifies* P if

$$[\, S{<} \subseteq (\mathsf{TRUE} \circ P){>} \,] \quad .$$

The relation S *maintains* P if

$$[\, S \circ (\mathsf{TRUE} \circ P){>} \subseteq P^{\cup} \circ (\Leftarrow) \circ P \,] \quad .$$

(Equivalently, relation S *maintains* P if

$$[\, S \circ (\mathsf{TRUE} \circ P){>} \subseteq P^{\cup} \circ (\Leftarrow) \circ P \circ (\mathsf{TRUE} \circ P){>} \,]$$

since, for all relations S and R and all coreflexives p,

$$[\, S{\circ}p \subseteq R \ \equiv\ S{\circ}p \subseteq R{\circ}p \,] \quad .$$

The easy proof of the equivalence by mutual implication is left to the reader.)
□

Typically definition 6.6 is used with S instantiated to the meaning of a program segment. The square brackets denote universal quantification over the context of the program segment. Were we not to use the everywhere brackets, we would be obliged to introduce new symbols for all five operators in the definition. We would also have to write $\mathsf{K.TRUE}$ (the function that always returns TRUE) in the definition in order to distinguish it from TRUE (the scalar boolean value) as used, for example, in the hints in the proof of the lemma below. See the discussion of the semantics of depth-first search in section 11.2 for how so-called "lifted" operators are defined.

Lemma 6.7 Suppose P has type $\mathsf{Bool}\leftarrow\mathsf{State}$. Then

$$[\,(\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>})_{<}\;\subseteq\;(\mathsf{TRUE}\circ\mathsf{P})_{>}\,]\quad.$$

Proof

$$\begin{array}{ll}
 & (\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>})_{<} \\
= & \{\quad \text{domains: theorem 5.7(c), and dual of (5.9)}\quad\} \\
 & (\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\circ\mathsf{P}^{\cup}\circ\mathsf{TRUE})_{<} \\
\subseteq & \{\quad \mathsf{P}\ \text{is functional}\quad\} \\
 & (\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{TRUE})_{<} \\
= & \{\quad \mathrm{I}_{\mathsf{Bool}} = \mathsf{FALSE}\cup\mathsf{TRUE} \\
 & \qquad \mathsf{FALSE}\circ(\Leftarrow)\circ\mathsf{TRUE} = \bot\!\!\bot \\
 & \qquad \mathsf{TRUE}\circ(\Leftarrow)\circ\mathsf{TRUE} = \mathsf{TRUE}\quad\} \\
 & (\mathsf{P}^{\cup}\circ\mathsf{TRUE})_{<} \\
= & \{\quad \text{domains: theorem 5.7(c) and converse}\quad\} \\
 & (\mathsf{TRUE}\circ\mathsf{P})_{>}\quad.
\end{array}$$

□

Lemma 6.8 Suppose P has type $\mathsf{Bool}\leftarrow\mathsf{State}$. Suppose $\mathsf{S1}$ truthifies P and $\mathsf{S2}$ maintains P. Then $\mathsf{S2}\circ\mathsf{S1}$ truthifies P.

Proof We have:

$$\begin{array}{ll}
 & \mathsf{S2}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>}\;\subseteq\;\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\quad\wedge\quad\mathsf{S1}_{<}\;\subseteq\;(\mathsf{TRUE}\circ\mathsf{P})_{>} \\
\Rightarrow & \{\quad \mathsf{S1}_{<}\;\subseteq\;(\mathsf{TRUE}\circ\mathsf{P})_{>}\;\Rightarrow\;\mathsf{S1} = (\mathsf{TRUE}\circ\mathsf{P})_{>}\circ\mathsf{S1}\quad\} \\
 & \mathsf{S2}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>}\;\subseteq\;\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\quad\wedge\quad(\mathsf{S2}\circ\mathsf{S1})_{<}\;=\;(\mathsf{S2}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>}\circ\mathsf{S1})_{<} \\
\Rightarrow & \{\quad \text{monotonicity}\quad\}
\end{array}$$

$$
\begin{array}{ll}
 & (\mathsf{S2}{\circ}\mathsf{S1})_{<} \;\subseteq\; (\mathsf{P}^{\cup}\circ(\Leftarrow)\circ\mathsf{P}\circ(\mathsf{TRUE}\circ\mathsf{P})_{>})_{<} \\
\Rightarrow & \{\quad \text{lemma 6.7 and transitivity} \quad\} \\
 & (\mathsf{S2}{\circ}\mathsf{S1})_{<} \;\subseteq\; (\mathsf{TRUE}\circ\mathsf{P})_{>} \;\; .
\end{array}
$$

□

Lemma 6.9 Suppose P has type $\mathsf{Bool}{\leftarrow}\mathsf{State}$. Suppose $Init$ is a program segment that truthifies P. Suppose T is a function of type $\mathsf{Bool}{\leftarrow}\mathsf{State}$ and $t=(\mathsf{TRUE}\circ T)_{>}$. (So t is a coreflexive representing the set of all states satisfying the termination condition T.) Suppose $Body$ is a program segment such that $Body\circ{\sim}t$ maintains P. Then $(Body\circ{\sim}t)^*$ maintains P and $(Body\circ{\sim}t)^*\circ Init$ truthifies P. It follows that

$$t\circ(Body\circ{\sim}t)^*\circ Init$$

truthifies $P\wedge T$.

Proof The first step of the proof applies definition 6.6 and simultaneously "strengthens the induction hypothesis" ready for use of fixed-point induction

$$
\begin{array}{ll}
 & (Body\circ{\sim}t)^* \text{ maintains } P \\
= & \{\quad \text{definition 6.6 and domain calculus} \quad\} \\
 & [\; (Body\circ{\sim}t)^*\circ(\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \;] \;\; .
\end{array}
$$

But

$$
\begin{array}{ll}
 & (Body\circ{\sim}t)^*\circ(\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \\
\Leftarrow & \{\quad \text{fixed-point induction} \quad\} \\
 & \quad (\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \\
 & \wedge\;\; Body\circ{\sim}t\circ P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \\
= & \{\quad \text{domain calculus} \quad\} \\
 & \quad (\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P \\
 & \wedge\;\; Body\circ{\sim}t\circ P^{\cup}\circ(\Leftarrow)\circ P\circ(\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P \\
\Leftarrow & \{\quad \mathsf{TRUE}\subseteq I_{\mathsf{Bool}} \;;\; (\Leftarrow) \text{ is reflexive, i.e. } I_{\mathsf{Bool}} \subseteq (\Leftarrow) \;; \\
 & \quad \text{lemma 6.7} \quad\} \\
 & \quad P_{>} \;\subseteq\; P^{\cup}\circ P \\
 & \wedge\;\; Body\circ{\sim}t\circ(\mathsf{TRUE}\circ P)_{>} \;\subseteq\; P^{\cup}\circ(\Leftarrow)\circ P \\
\Leftarrow & \{\quad \text{domains (specifically } P_{>} \;=\; I\cap P^{\cup}\circ P\text{);}
\end{array}
$$

```
            assumption:  $\mathrm{Body} \circ \sim t$  maintains  $P$   }
    true  .
```

That $(\mathrm{Body} \circ \sim t)^* \circ \mathrm{Init}$ truthifies P now follows from lemma 6.8 and the assumption that Init truthifies P.

□

Lemma 6.9 justifies the way that invariant properties are used in practice. To document the code, we suppose that the specification is decomposed into precondition $pre(\sigma)$ and postcondition $post(\sigma)$; then we add assertions:

```
    {  $pre(\sigma)$  }
    Init
    {  **Invariant property**:  $P$  }
  ;  **while** $\sim t$ **do**
        {  $P \wedge \neg T$  }
        S
        {  $P$  }
    {  $P \wedge T$  }
    {  $post(\sigma)$  }
```

from which we can extract the three verification conditions:

```
    {  $pre(\sigma)$  }
    Init
    {  $P$  }
```

(the initialisation truthifies P),

```
    {  $P \wedge \neg T$  }
    S
    {  $P$  }
```

(the loop body maintains P), and

```
    {$P \wedge T$}
    {  $post(\sigma)$  }
```

(the postcondition is implied by the conjunction of the invariant and the condition for terminating the loop).

A final remark is that, although conditional correctness is most often established using invariant properties and values, *relations* are vital to establishing termination of loops and other forms of recursion. Section 6.9 gives an example.

6.8.6 Bound Functions

As we have seen in section 6.8.3, the use of **while** statements in programs entails establishing that the body of the loop maintains an invariant that is a well-founded relation. This is usually done by means of a so-called *bound function*.

Formally, the use of bound functions generalises the use of invariant *properties* to invariant *relations*. Suppose $\prec$ is a well-founded relation on some set $\mathcal{A}$, and suppose h is a function from the state space to $\mathcal{A}$ (i.e. a function of type $\mathcal{A}\leftarrow\mathsf{State}$). Then, establishing that a **while** statement with body Body and termination condition t is guaranteed to terminate is achieved by showing that

$$[\ \mathsf{Body}\circ{\sim}t\ \subseteq\ h^{\cup}\circ(\prec)\circ h\ ]\ \ .$$

The function h is called the *bound function*; the well-founded ordering is usually implicit in the type of h.

The theorem that is being exploited here is that $h^{\cup}\circ(\prec)\circ h$ is a well-founded relation if $\prec$ is well-founded. (See the discussion following lemma 8.36.) Note the resemblance of this expression to the notion of an invariant property: the ordering relation in the case of an invariant property is the "if" relation on predicates. When we discuss concrete algorithms, we see this pattern occurring repeatedly in the invariants we formulate.

6.9 Calculating a Least Fixed Point

As illustration, we present an iterative algorithm for calculating a least fixed point.

The least fixed point of a monotonic endofunction F on a finite, partially ordered set $(\mathcal{A},\preceq)$ with least element 0 can be computed by a simple iterative algorithm:

```
    $x := 0$
;   while $x \neq F.x$ do
        $x := F.x$
```

An invariant property of the algorithm is that $x \preceq \mu F$; the algorithm is guaranteed to terminate if the relation $\succ$ is well-founded (in particular, if $\mathcal{A}$ is finite) since x is strictly increased at each iteration. On termination, $x = F.x$. That is, x is a fixed point of F. Since μF is the least fixed point and, at all times, $x \preceq \mu F$, we conclude that, on termination, $x = \mu F$.

In order to relate this brief, informal account with the discussion above, we must identify the individual components of the algorithm. The program is parameterised by the partially ordered set $(\mathcal{A}, \preceq)$, the function F of type $(\mathcal{A}, \preceq) \leftarrow (\mathcal{A}, \preceq)$ (the set of monotonic endofunctions on $\mathcal{A}$) and the constant 0; its state space is $\mathcal{A}$. All of these constitute what we have called the context of the specification. Note that, even for such a simple algorithm, the context is quite complex. It includes, for example, the fact that the constant $\preceq$ is a reflexive, transitive and anti-symmetric relation.

The precondition of the specification is true and the relation is the relation R of type $\mathcal{A} \sim \mathcal{A}$ defined by, for all x and x_0 of type $\mathcal{A}$,

$$x \ R \ x_0 \ \equiv \ x = \mu F \ \ .$$

(The function F and the partially ordered set $(\mathcal{A}, \preceq)$ are implicit parameters of R, as explained above.) By the assumed anti-symmetry of the ordering relation, and the fixed-point induction rule,

$$[\ x = \mu F \ \Leftarrow \ x \preceq \mu F \wedge x = F.x \ ] \ \ .$$

(Here and elsewhere, the everywhere brackets denote a universal quantification over the input parameters and the state x.) Noting that $x = F.x$ is the condition for terminating the loop, conditional correctness thus amounts to showing that the algorithm truthifies the postcondition $x \preceq \mu F$, for all monotonic functions F of the given type. This suggests the use of lemma 6.9, with the property $x \preceq \mu F$ as invariant. That is, we show that the initialisation truthifies $x \preceq \mu F$ and the loop body maintains $x \preceq \mu F$.

Establishing that, in addition, the loop always terminates demands that we add the additional conjunct $x \preceq F.x$. By showing that this property is also invariant we infer that the loop body —which is just the function F— combined with the precondition for its execution

$$x \preceq F.x \wedge x \neq F.x \ \ ,$$

i.e. $x \prec F.x$, is a subset of the relation $\succ$. The guarantee of termination follows from the assumption that this relation is well-founded.

Thus the remaining task is to show that the initialisation truthifies, and the loop body maintains the property

$$x \preceq F.x \wedge x \preceq \mu F \ \ .$$

This gives rise to two verification conditions. Making use of the assignment axiom, the verification condition for the initialisation is:

$$[\ 0 \preceq F.0 \wedge 0 \preceq \mu F \ ]$$

and that for the loop body is:

$$[\ F.x \preceq F.(F.x) \wedge F.x \preceq \mu F \ \Leftarrow\ x \preceq F.x \wedge x \preceq \mu F \ \wedge\ x \neq F.x\ ]\quad .$$

Given the assumptions made about the input parameters (implicit in the everywhere brackets), both of these are true and the ("total") correctness of the algorithm has been established. (Note that the guard on executing the loop body, $x \neq F.x$, is not needed for the conditional correctness but is needed for the guarantee of termination.)

Part III

Components and Acyclicity

Chapter 7

Equivalence Relations and Partitions

In this chapter we explore properties of equivalence relations, some of which are well known. Section 7.1 formulates the well-known correspondence between partitions of a set and equivalence classes in a point-free style and section 7.2 explores properties of the equivalence-class function, in particular with respect to complementation.

7.1 Partitions

An *equivalence relation* is a relation that is reflexive, transitive and symmetric. As is well known, an equivalence relation *partitions* the set on which it is defined into a number of so-called *equivalence classes*. More formally, if R is an equivalence relation on a set A, there is a set C and a surjective function f of type $C \leftarrow A$, such that, for all a and b in A,

$$(7.1) \quad a[\![R]\!]b \;\equiv\; f.a = f.b \enspace .$$

(It is common to use square brackets to denote the function f. So, instead of writing $f.a$, one writes $[a]$, or $[a]_R$ if it is thought necessary to make the equivalence relation explicit.)

Conversely, given sets A and C and a total function f of type $C \leftarrow A$, we can use equation (7.1) to define a homogeneous relation R on A. The relation R is then an equivalence relation.

Equation (7.1) is expressed more succinctly by the point-free equation

$$(7.2) \quad R = f^{\cup} \circ f \enspace .$$

Point-free formulations of functionality, totality, surjectivity and injectivity then support effective point-free calculation. Here, for example, is the proof that it is transitive (in every detail, including the use of the associativity of composition).

$$
\begin{array}{ll}
 & (f^{\cup}\circ f)\circ(f^{\cup}\circ f) \\
= & \{ \quad \text{composition is associative} \quad \} \\
 & f^{\cup}\circ(f\circ f^{\cup})\circ f \\
\subseteq & \{ \quad f \text{ is functional, i.e. } f\circ f^{\cup}\subseteq I_C\text{ ,} \\
 & \quad\quad \text{monotonicity of composition} \quad \} \\
 & f^{\cup}\circ I_C\circ f \\
= & \{ \quad I_C \text{ is identity of composition} \quad \} \\
 & f^{\cup}\circ f \ \ .
\end{array}
$$

The converse proposition is that if R is an equivalence relation on set A, the function f of type $2^A \leftarrow A$ defined to be

$$\langle a \ :: \ \mathsf{Set}.(R\circ a)_< \rangle$$

maps (coreflexive) atoms a to equivalence classes of R (where Set is a so-called "cast" that maps a coreflexive of type $A\sim A$, for some A, to the atomic coreflexive of type 2^A representing the same subset of A). That is, $R = f^{\cup}\circ f$. The proof is straightforward, although somewhat long. See theorem 7.7 below.

Lemma 7.3 If R is an equivalence relation, then for all proper atomic coreflexives a and b,

$$(R\circ a)_< = (R\circ b)_< \ \equiv \ a\circ R\circ b = a\circ\top\!\top\circ b \ \ .$$

Proof By lemma 5.27 with $R{,}p{,}b := R{,}a{,}b$,

(7.4) $\quad a \subseteq (R\circ b)_< \ \equiv \ a\circ R\circ b = a\circ\top\!\top\circ b \ \ .$

Second,

$$
\begin{array}{ll}
 & (R\circ a)_< \subseteq (R\circ b)_< \\
\Rightarrow & \{ \quad \text{assuming } R \text{ is reflexive, } a\subseteq(R\circ a)_< \quad \} \\
 & a \subseteq (R\circ b)_< \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & (R\circ a)_< \subseteq (R\circ(R\circ b)_<)_< \\
= & \{ \quad \text{domains: (5.9)} \quad \} \\
 & (R\circ a)_< \subseteq (R\circ R\circ b)_< \\
\Rightarrow & \{ \quad \text{assuming } R \text{ is transitive, } R\circ R\subseteq R\text{; monotonicity} \quad \} \\
 & (R\circ a)_< \subseteq (R\circ b)_< \ \ .
\end{array}
$$

That is, if R is reflexive and transitive,

(7.5) $\quad a \subseteq (R{\circ}b)_< \;\equiv\; (R{\circ}a)_< \subseteq (R{\circ}b)_<$.

Moreover, if R is symmetric and a and b are coreflexives,

(7.6) $\quad a{\circ}R{\circ}b = a{\circ}\top\!\top{\circ}b \;\equiv\; b{\circ}R{\circ}a = b{\circ}\top\!\top{\circ}a$.

Thus, if R is an equivalence relation,

$$
\begin{array}{ll}
 & (R{\circ}a)_< = (R{\circ}b)_< \\
= & \{ \quad \text{anti-symmetry} \quad \} \\
 & (R{\circ}a)_< \subseteq (R{\circ}b)_< \;\wedge\; (R{\circ}b)_< \subseteq (R{\circ}a)_< \\
= & \{ \quad (7.5) \quad \} \\
 & a \subseteq (R{\circ}b)_< \;\wedge\; b \subseteq (R{\circ}a)_< \\
= & \{ \quad (7.4) \quad \} \\
 & a{\circ}R{\circ}b = a{\circ}\top\!\top{\circ}b \;\wedge\; b{\circ}R{\circ}a = b{\circ}\top\!\top{\circ}a \\
= & \{ \quad (7.6) \quad \} \\
 & a{\circ}R{\circ}b = a{\circ}\top\!\top{\circ}b \;.
\end{array}
$$

□

Theorem 7.7 Suppose R is an equivalence relation. Let the function f be defined to be

$$\langle a :: \mathsf{Set}.(R \circ a)_< \rangle \;.$$

Then

$$R = f^{\cup} \circ f$$

and

$$f = f \circ R \;.$$

Proof Suppose R is an equivalence relation on set A. By the definition of f, for all points a of type A and points c of type 2^A,

(7.8) $\quad c{\circ}\top\!\top{\circ}a \subseteq f \;\equiv\; c = \mathsf{Set}.(R{\circ}a)_<$.

Thus, with dummies a and b ranging over points of type A, and dummy c ranging over points of type 2^A, we have:

$$
\begin{array}{ll}
& R \\
= & \{ \quad \text{saturation assumption: theorem 5.22} \\
& \quad\ \ \text{and all-or-nothing rule} \quad \} \\
& \langle \cup a,b : a{\circ}R{\circ}b = a{\circ}\top\!\!\top{\circ}b : a{\circ}\top\!\!\top{\circ}b \rangle \\
= & \{ \quad \text{assumption: } R \text{ is an equivalence relation, corollary 7.3} \quad \} \\
& \langle \cup a,b \ :\ (R{\circ}a)_< \ =\ (R{\circ}b)_< \ :\ a{\circ}\top\!\!\top{\circ}b \rangle \\
= & \{ \quad \mathsf{Set} \text{ casts a coreflexive of type } A{\sim}A \text{ to a point of } 2^A \quad \} \\
& \langle \cup a,b \ :\ \mathsf{Set}.(R{\circ}a)_< \ =\ \mathsf{Set}.(R{\circ}b)_< \ :\ a{\circ}\top\!\!\top{\circ}b \rangle \\
= & \{ \quad \text{definition of } f\text{: (7.8), and (5.47)} \quad \} \\
& f^{\cup} \circ f
\end{array}
$$

and

$$
\begin{array}{ll}
& f \circ R \\
= & \{ \quad \text{above} \quad \} \\
& f \circ f^{\cup} \circ f \\
= & \{ \quad f \text{ is a function, so } f \circ f^{\cup} \subseteq I \text{ and hence } f \circ f^{\cup} = f_< \quad \} \\
& f_< \circ f \\
= & \{ \quad \text{domains (specifically theorem 5.3)} \quad \} \\
& f \ .
\end{array}
$$

□

7.2 Properties of the Partition Function

In section 7.1, the function $\langle a :: \mathsf{Set}.(R \circ a)_< \rangle$ was shown to map a proper atom a into the set of proper atoms equivalent to a under the (equivalence) relation R. This section is about exploring the properties of the endofunction $\langle p : p \subseteq I : (R{\circ}p)_< \rangle$. We show that it is a complementation-fixed closure operator.

To avoid clutter, we use the convention that lower case identifiers p and q range over coreflexives. So the function of interest is $\langle p :: (R{\circ}p)_< \rangle$.

Recalling definition 2.47 of complementation-fixed and noting that, for all relations R, $R_{\bullet<} = {\sim}(R_<)$, we explore conditions under which

$$(R \circ (R{\circ}S)_{\bullet<})_< \ =\ (R{\circ}S)_{\bullet<}$$

beginning with the inclusion.

Lemma 7.9 For arbitrary relations R and S,

$$(R\circ(R{\circ}S)_{\bullet<})_{<}\ \subseteq\ (R{\circ}S)_{\bullet<}\quad\Leftarrow\quad R^{\cup}\circ R\subseteq R\ \ .$$

Proof Note how the calculation below is used to determine simpler conditions on R for which the more complicated inclusion holds. The use of the isomorphism between conditionals and domains in the first step is driven by the fact that the negation of a condition is a condition. The use of middle-exchange then becomes obvious.

$$\begin{array}{ll}
 & (R\circ(R{\circ}S)_{\bullet<})_{<}\ \subseteq\ (R{\circ}S)_{\bullet<} \\
= & \{\quad \text{theorem 5.8(e)}\quad\} \\
 & R\circ(R{\circ}S)_{\bullet<}\circ\top\!\!\top\ \subseteq\ (R{\circ}S)_{\bullet<}\circ\top\!\!\top \\
= & \{\quad \text{property of negated left domain}\quad\} \\
 & R\circ\neg(R{\circ}S{\circ}\top\!\!\top)\ \subseteq\ \neg(R{\circ}S{\circ}\top\!\!\top) \\
= & \{\quad \text{middle-exchange rule (4.18)} \\
 & \qquad \text{with } R,X,S,Y := R\,,\,R{\circ}S{\circ}\top\!\!\top\,,\,I\,,\,R{\circ}S{\circ}\top\!\!\top\quad\} \\
 & R^{\cup}\circ R\circ S\circ\top\!\!\top\ \subseteq\ R{\circ}S{\circ}\top\!\!\top \\
\Leftarrow & \{\quad \text{monotonicity of composition}\quad\} \\
 & R^{\cup}\circ R\ \subseteq\ R\ \ .
\end{array}$$

□

Corollary 7.10 If R is an equivalence relation, for all S,

$$(R\circ(R{\circ}S)_{\bullet<})_{<}\ =\ (R{\circ}S)_{\bullet<}\ \ .$$

In words, if R is an equivalence relation, $(R{\circ}S)_{\bullet<}$ is a fixed point of the function mapping coreflexive p to $(R{\circ}p)_{<}$.

Proof We have:

$$\begin{array}{ll}
 & (R\circ(R{\circ}S)_{\bullet<})_{<} \\
\subseteq & \{\quad R \text{ is an equivalence relation, so } R^{\cup}{\circ}R\subseteq R\text{, lemma 7.9}\quad\} \\
 & (R{\circ}S)_{\bullet<} \\
= & \{\quad I \text{ is unit of composition; } (R_{\bullet<})_{<}=R_{\bullet<} \text{ for all } R\text{, with } R:=R{\circ}S\quad\} \\
 & (I\circ(R{\circ}S)_{\bullet<})_{<} \\
\subseteq & \{\quad R \text{ is an equivalence relation, so } I\subseteq R\,, \\
 & \qquad \text{monotonicity of composition and domains}\quad\} \\
 & (R\circ(R{\circ}S)_{\bullet<})_{<}\ \ .
\end{array}$$

The equality thus follows by the anti-symmetry of $\subseteq$.
□

Lemma 7.11 If R is reflexive and transitive, the function $\langle p :: (R{\circ}p)_< \rangle$ is a closure operator.

Proof The equivalence in definition 2.44 of a closure operator is established by mutual implication. Implication:

$$
\begin{array}{ll}
 & (R{\circ}q)_< \\
\subseteq & \{ \quad \text{assume } q \subseteq (R{\circ}p)_< \text{, monotonicity of } (R\circ) \text{ and } _< \quad \} \\
 & (R\circ(R{\circ}p)_<)_< \\
= & \{ \quad \text{domains (dual of theorem 5.9)} \quad \} \\
 & (R{\circ}R{\circ}p)_< \\
\subseteq & \{ \quad R \text{ is transitive} \quad \} \\
 & (R{\circ}p)_<
\end{array}
$$

and follows-from:

$$
\begin{array}{ll}
 & q \subseteq (R{\circ}p)_< \\
\Leftarrow & \{ \quad \text{assume } (R{\circ}q)_< \subseteq (R{\circ}p)_< \text{, transitivity of } \subseteq \quad \} \\
 & q \subseteq (R{\circ}q)_< \\
\Leftarrow & \{ \quad R \text{ is reflexive, i.e. } I \subseteq R \text{; monotonicity of } ({\circ}q) \text{ and } _< \quad \} \\
 & q \subseteq (I{\circ}q)_< \\
= & \{ \quad \text{domains: theorem 5.8, and assumption: } q \text{ is coreflexive} \quad \} \\
 & \text{true} \ .
\end{array}
$$
□

Theorem 7.12 If R is an equivalence relation, the function $\langle p : p \subseteq I : (R{\circ}p)_< \rangle$ is a complementation-fixed and complementation-idempotent closure operator.

Moreover, if R is an equivalence relation on a complete, universally distributive, saturated lattice, the set of coreflexives $\mathsf{Fix}.\langle p :: (R{\circ}p)_< \rangle$ is a complete, saturated lattice, its atoms being the set of coreflexives $(R{\circ}a)_<$ where a is an atom of the lattice of all coreflexives.

Proof An equivalence relation R is reflexive ($I \subseteq R$), symmetric ($R = R^{\cup}$) and transitive ($R{\circ}R \subseteq R$). So the function $\langle p :: (R{\circ}p)_< \rangle$ is a closure operator by lemma 7.11 and, hence, complementation-idempotent by corollary 7.10. It is thus also complementation-fixed by lemma 2.48.

If R is an equivalence relation on a complete, universally distributive lattice, the completeness and saturation properties are given by theorem 2.71.
□

Lemma 7.13 Suppose R is an equivalence relation on a saturated atomic lattice. Then

$$\langle \cup a : \mathsf{atom}.a : (R{\circ}a)_< \rangle \;=\; I \quad .$$

Proof

$$
\begin{array}{cl}
 & \langle \cup a : \mathsf{atom}.a : (R{\circ}a)_< \rangle \\
= & \{ \quad \text{the functions } {<} \text{ and } (R\circ) \text{ are lower adjoints} \\
 & \quad\;\; \text{and so are universally distributive} \quad \} \\
 & (R \circ \langle \cup a : \mathsf{atom}.a : a \rangle)_< \\
= & \{ \quad \text{the lattice of coreflexives is saturated, i.e. } \langle \cup a : \mathsf{atom}.a : a \rangle \;=\; I \quad \} \\
 & (R \circ I)_< \\
= & \{ \quad R_< \subseteq I \text{, for all } R; \\
 & \quad\;\; R \text{ is reflexive, i.e. } I \subseteq R;\ {<} \text{ is monotonic and } I_< = I. \quad \} \\
 & I \quad .
\end{array}
$$
□

Note that, as already observed, the function $\langle p :: (R{\circ}p)_< \rangle$ is the lower adjoint in a Galois connection of the coreflexives (ordered by the subset relation) with itself. Thus, if R is an equivalence relation, the function is universally distributive, as well as being a complementation-fixed and complementation-idempotent closure operator.

Finally, note that all the properties stated and proven in this section can be dualised to properties of the function $\langle p :: (p{\circ}R)_> \rangle$. This is important, for example when we consider the notions of left- and right-definiteness of a relation in section 8.1. The function $\langle R :: \mathsf{Set}.\langle p :: (R{\circ}p)_< \rangle \rangle$ is akin to what Bird and De Moor [BdM97] call the "existential image" functor. The function $\langle a :: \mathsf{Set}.(R \circ a)_< \rangle$ (where a ranges over proper atoms) is what they call the "power-transpose" of R. This terminology is more relevant to applications where relations are viewed as set-valued functions.

Chapter 8

Acyclic Graphs

This chapter begins our presentation of algorithmic graph theory in point-free relation algebra. From now on, a *graph* G is simply a homogeneous "edge" relation of type $\mathsf{Node}{\sim}\mathsf{Node}$ where Node is a *finite* set. A proper atom a in the lattice of coreflexives of type Node is a *node* of the graph. Then, if a and b are both nodes, the boolean $a{\circ}G{\circ}b \neq \perp\!\!\perp$ represents the existence of an *edge* from a to b; if indeed $a{\circ}G{\circ}b \neq \perp\!\!\perp$, the edge itself is the atom $a{\circ}\top\!\!\top{\circ}b$ (in the poset of relations of type $\mathsf{Node}{\sim}\mathsf{Node}$). The existence of a *path* from a to b is represented by the boolean $a \circ G^* \circ b \neq \perp\!\!\perp$. (The path itself is a sequence of nodes.) In this way, relation algebra is the appropriate vehicle for a study of the algorithmic properties of the *existence* of paths in graphs. (Regular algebra is the appropriate vehicle for studying more general properties of paths in graphs.)

Acyclic graphs (graphs without cyclic paths) form an important subclass of graphs. This is not just because they naturally occur in practical problems —they correspond to partial orderings on finite sets— but also because all graphs comprise a collection of so-called "strongly connected components" that are connected by an acyclic graph. This structural property of graphs —formalised in theorem 9.30— is important in path-finding algorithms as well as the seemingly unrelated problem of efficiently representing the inverse of a real matrix. (See the discussion following theorem 9.30 for further discussion.)

Subsection 8.1 defines acyclicity in the conventional way in terms of paths. At the same time, a less well-known property, which we call "definiteness" is introduced. Whereas acyclicity is particularly appropriate to reasoning about graphs, definiteness is more general. For finite graphs, the two notions coincide, as shown in this section.

Subsection 8.2 is about showing that the reflexive-transitive reduction of a definite relation is its least starth root. Equivalently, every partial ordering on a finite set has a unique so-called "Hasse diagram".

Subsection 8.3 develops a formal proof of the following fact from graph theory: in an acyclic graph, the nodes reachable from set A coincide with the nodes reachable from the minimal elements of A. The theorem is a corollary of a much more general theorem about "right-definiteness" of a relation. In more conventional terminology, it is the theorem that, given a well-founded relation on a set S, every non-empty subset of S has a minimal element (with respect to the well-founded relation).

The final subsection in this section, subsection 8.4, is about how a "topological search" of an acyclic graph assigns to the nodes of the graph a so-called "topological ordering". The definition of a topological ordering and the algorithm for topological search are formulated in point-free relation algebra.

Many properties we prove are valid for arbitrary relations and not just for graphs. That is, the assumption of finiteness is not required. Nevertheless, we sometimes use graph terminology — partly because this is the primary application here but also because it is more "graphic" in the sense of being easier to explain with the aid of diagrams. In order to make the level of generality clear, we use R to denote an arbitrary relation and G to denote a graph — that is, a relation over a finite set of nodes.

8.1 Definiteness and Acyclicity

We have to define the meaning of a graph being acyclic. Obviously, a cycle gives rise to an infinite path in the graph. But, conversely, an infinite path in a finite graph contains a cycle (because the number of vertices is finite). Therefore, acyclicity in finite graphs is the same as the absence of infinite paths, to which we give the name "(left- or right-) definite".

Definition 8.1 ((Right/Left) Definite) Relation R is said to be *right-definite* if and only if it satisfies

(8.2) $\langle\forall p :: p \subseteq \perp\!\!\perp \Leftarrow p \subseteq (p \circ R) \rangle \rangle$.

It is said to be *left-definite* if and only if it satisfies

(8.3) $\langle\forall p :: p \subseteq \perp\!\!\perp \Leftarrow p \subseteq (R \circ p) \lt \rangle$.

It is said to be *definite* if it is both left- and right-definite.
□

Informally, right-definiteness means the absence of infinite "descending" paths. That is, there is not a non-empty set of atoms, represented by the coreflexive p, such that, for all atoms a in p, it is always possible to find an atom b in p such a is in the set represented by $(b \circ R) \gt$, i.e. $b[\![R]\!]a$. Were this possible, the process can be repeated *ad infinitum*; in graphs, this means the existence of paths comprising an infinite number of edges. (See lemmas 8.17 and 8.19 for the formalisation of this argument.)

Note that R is right-definite equivales that its converse $R^{\cup}$ is left-definite. So left-definiteness means the absence of infinite "ascending" paths. A hint on how to remember which is which is that left-definiteness is defined in terms of the left domain operator and right-definiteness in terms of the right domain operator.

The importance of the concept of definiteness is what we have called the *unique extension property* (uep) of relation algebra.

Theorem 8.4 (Uep of Relation Algebra) Suppose R is a right-definite relation. Then, for all coreflexives p and q,

$$p = (p{\circ}R)_{>} \cup q \quad \equiv \quad p = (q \circ R^*)_{>} \;\; .$$

Also, for all relations X and S,

$$X = X{\circ}R \cup S \quad \equiv \quad X = S \circ R^* \;\; .$$

Dually, if R is a left-definite relation, for all coreflexives p and q,

$$p = (R{\circ}p)_{<} \cup q \quad \equiv \quad p = (R^* \circ q)_{<} \;\; ,$$

and, for all relations X and S,

$$X = R{\circ}X \cup S \quad \equiv \quad X = R^* \circ S \;\; .$$

□

A proof of theorem 8.4 can be found in [DBvdW97, section 7]. Effectively, in relation algebra theorem 8.4 is equivalent to the unique extension property of regular algebra presented in section 3.3. (See theorem 3.16.) Note that [DBvdW97] uses the terminology "well-founded" rather than "right-definite" in order to fit with the standard terminology of the principle application considered in the paper.

For later use, we note the following simple lemma.

Lemma 8.5 Suppose R is right-definite and $R \supseteq S$. Then S is right-definite. The same is true with "left" replacing "right".

Proof Immediate from the monotonicity of transitive closure, composition and the domain operators.

□

As mentioned earlier, in [DBvdW97] the better-known term "well-founded" was used instead of our "right-definite". An example of a well-founded relation is the less-than relation on the natural numbers. Expressed pointwise, (8.2) for this application is the property that, for all subsets p of the natural numbers,

$$p = \emptyset \;\Leftarrow\; \langle \forall m : m{\in}p : \langle \exists n : n{\in}p : n < m \rangle \rangle \;\; .$$

Expressed slightly differently, this is the property that for all subsets p of the natural numbers,

$$p = \emptyset \ \vee \ \langle \exists m : m{\in}p : \langle \forall n : n{\in}p : n \geq m \rangle \rangle \ \ .$$

In words, every non-empty set of natural numbers has a least element.

We mention this example because it illustrates the fact that left-definite and right-definite are *not* (in general) the same: the successor relation on the natural numbers (the converse of the predecessor relation) is not well-founded. Left- and right-definite are the same for *finite* graphs, as we shall see.

The less-than relation on natural numbers is the transitive closure of the predecessor relation (the converse of the successor function, where the successor of m is $m{+}1$). And, of course, the predecessor relation is well-founded. This exemplifies a (well-known) property, namely:

Lemma 8.6 Relation R is right-definite equivales relation R^+ is right-definite. Similarly for left-definite and for definite. Thus R is right-definite if and only if it satisfies

(8.7) $\langle \forall p :: p \subseteq \perp\!\!\perp \ \Leftarrow \ p \subseteq (p \circ R^+)_> \rangle \ \ .$

It is left-definite if and only if it satisfies

(8.8) $\langle \forall p :: p \subseteq \perp\!\!\perp \ \Leftarrow \ p \subseteq (R^+ \circ p)_< \rangle \ \ .$

Proof Obviously, $R \subseteq R^+$. So, by lemma 8.5, R is right definite if R^+ is right definite. For the converse, we have:

$$\begin{array}{cl}
 & p \ \subseteq \ (p \circ R^+)_> \\
= & \{ \quad \text{definition of set union} \quad \} \\
 & p \cup (p \circ R^+)_> \ \subseteq \ (p \circ R^+)_> \\
= & \{ \quad \text{distributivity, } R^* = 1 \cup R^+ \quad \} \\
 & (p \circ R^*)_> \ \subseteq \ (p \circ R^+)_> \\
= & \{ \quad R^+ = R^* \circ R \text{, domains} \quad \} \\
 & (p \circ R^*)_> \ \subseteq \ ((p \circ R^*)_> \circ R)_> \ \ .
\end{array}$$

Moreover, since $\perp\!\!\perp$ is the zero of composition and $p \subseteq (p \circ R^*)_>$,

$$p \subseteq \perp\!\!\perp \ \equiv \ (p \circ R^*)_> \subseteq \perp\!\!\perp \ \ .$$

Thus

$$\langle\forall p :: p\subseteq\perp\!\!\perp \;\Leftarrow\; p\subseteq (p\circ R)_{>}\rangle$$
$$\Rightarrow\qquad\{\qquad p := (p\circ R^{*})_{>}\qquad\}$$
$$\langle\forall p :: (p\circ R^{*})_{>}\subseteq\perp\!\!\perp \;\Leftarrow\; (p\circ R^{*})_{>}\subseteq ((p\circ R^{*})_{>}\circ R)_{>}\rangle$$
$$=\qquad\{\qquad \text{above}\qquad\}$$
$$\langle\forall p :: p\subseteq\perp\!\!\perp \;\Leftarrow\; p\subseteq (p\circ R^{+})_{>}\rangle\ \ .$$

That is, R^{+} is right definite if R is right definite.
□

Because the antecedent of (8.2) is formally stronger than the antecedent of (8.7), it can be easier to use definition 8.1 to establish that a relation is right-definite. On the other hand, when it is known that a relation is right-definite, definition 8.6 may be easier to use.

See [DBvdW97] for a detailed study of properties of R and R^{+} of which lemma 8.6 is an instance.

Anticipating the definition of acyclicity (definition 8.11), we rephrase right-definiteness in terms of atomic coreflexives.

Lemma 8.9 For all R and all atomic coreflexives a,

$$a\subseteq R \;\equiv\; a\subseteq (a\circ R)_{>}\ \ .$$

Proof Suppose a is an atomic coreflexive. Then, for all R,

$$a\subseteq R$$
$$\Rightarrow\qquad\{\qquad a \text{ is coreflexive, so } (a\circ a)_{>} = a\,;\ \text{monotonicity}\qquad\}$$
$$a\subseteq (a\circ R)_{>}$$
$$\Rightarrow\qquad\{\qquad a\circ\top\!\!\top\circ a = a\,,\ \text{monotonicity}\qquad\}$$
$$a\subseteq a\circ\top\!\!\top\circ(a\circ R)_{>}$$
$$=\qquad\{\qquad \text{domains (specifically, theorem 5.7(a))}\qquad\}$$
$$a\subseteq a\circ\top\!\!\top\circ a\circ R$$
$$\Rightarrow\qquad\{\qquad a\circ\top\!\!\top\circ a = a\subseteq I\,,\ \text{monotonicity and transitivity}\qquad\}$$
$$a\subseteq R\ \ .$$

The lemma follows by mutual implication.
□

Lemma 8.10 If R is right-definite, then, for all atomic coreflexives a,

$$a\subseteq\perp\!\!\perp \;\Leftarrow\; a\subseteq R^{+}\ \ .$$

Proof Assume that R is right-definite. Then,

$$
\begin{array}{ll}
 & a \subseteq R^{+} \\
= & \{ \quad \text{lemma 8.9 with } R := R^{+} \quad \} \\
 & a \subseteq (a \circ R^{+})_{>} \\
\Rightarrow & \{ \quad \text{assumption: } R \text{ is right-definite, lemma 8.6} \quad \} \\
 & a \subseteq \perp\!\!\perp \ .
\end{array}
$$

□

We now define acyclicity:

Definition 8.11 (Acyclicity) A relation R is said to be *acyclic* if

$$I \cap R^{+} \;=\; \perp\!\!\perp \ .$$

A proper atomic coreflexive a is said to be *in a cycle* of R if $a \subseteq R^{+}$.
□

A proper atomic coreflexive a that is in a cycle of R "witnesses" the fact that R is not acyclic. Formally, we have:

Lemma 8.12

$$I \cap R^{+} \neq \perp\!\!\perp \;\equiv\; \langle \exists a : \mathsf{AC}.a \wedge a \neq \perp\!\!\perp : a \subseteq R^{+} \rangle \ .$$

Proof

$$
\begin{array}{ll}
 & I \cap R^{+} \neq \perp\!\!\perp \\
= & \{ \quad \text{lattice of relations is atomic, definition 2.49} \quad \} \\
 & \langle \exists a : \mathsf{atom}.a \wedge a \neq \perp\!\!\perp : a \subseteq I \cap R^{+} \rangle \\
= & \{ \quad a \subseteq I \cap R^{+} \equiv a \subseteq I \wedge a \subseteq R^{+} \\
 & \quad\; \text{trading and definition of atomic coreflexive, } \mathsf{AC} \quad \} \\
 & \langle \exists a : \mathsf{AC}.a \wedge a \neq \perp\!\!\perp : a \subseteq R^{+} \rangle \ .
\end{array}
$$

□

A straightforward calculation shows that

$$I \cap R^{+} \;=\; I \cap (R^{\cup})^{+} \ .$$

It follows that R is acyclic equivales $R^{\cup}$ is acyclic.

An alternative definition of relation R being acyclic is —essentially— that the relation R^{*} is a partial ordering (i.e. anti-symmetric as well as transitive and reflexive). To be precise:

Lemma 8.13 For all R, the relation R^* is anti-symmetric (i.e. $R^* \cap (R^*)^\cup = I$) if R is acyclic. Conversely, $\neg I \cap R$ is acyclic if R^* is anti-symmetric. (Equivalently —since R^* is reflexive and transitive— R^* is a partial ordering if R is acyclic and, conversely, $\neg I \cap R$ is acyclic if R^* is a partial ordering.)

Proof Suppose R is acylic. Then

$$\begin{array}{ll}
 & R^* \cap (R^*)^\cup = I \\
= & \{ \quad [\, R^* = I \cup R^+ \,] \text{, distributivity} \quad \} \\
 & (I \cap (R^*)^\cup) \;\cup\; (R^+ \cap (R^*)^\cup) \;=\; I \\
= & \{ \quad [\, (R^*)^\cup = (R^\cup)^* \,],\ [\, I \cap S^* = I \,] \text{ with } S := R^\cup \quad \} \\
 & I \;\cup\; (R^+ \cap (R^*)^\cup) \;=\; I \\
\Leftarrow & \{ \quad \bot\!\!\bot \text{ is zero of supremum} \quad \} \\
 & R^+ \cap (R^*)^\cup \;\subseteq\; \bot\!\!\bot \\
\Leftarrow & \{ \quad \text{modular law} \quad \} \\
 & (R^+ \circ R^* \cap I) \circ (R^*)^\cup \;\subseteq\; \bot\!\!\bot \\
= & \{ \quad [\, R^+ \circ R^* = R^+ \,] \text{, symmetry of } \cap, \\
 & \quad\;\; \bot\!\!\bot \text{ is zero of supremum} \quad \} \\
 & (I \cap R^+) \circ (R^*)^\cup \;=\; \bot\!\!\bot \\
= & \{ \quad R \text{ is acyclic, definition 8.11} \quad \} \\
 & \mathsf{true} \;\; .
\end{array}$$

That is, R^* is anti-symmetric if R is acyclic.

For the converse, suppose that R^* is anti-symmetric. Then

$$\begin{array}{ll}
 & I \cap (\neg I \cap R)^+ \;\subseteq\; \bot\!\!\bot \\
\Leftarrow & \{ \quad [\, (\neg I \cap R)^+ = (\neg I \cap R) \circ R^* \,], \\
 & \quad\;\; \text{modular law and } \bot\!\!\bot \text{ is zero of composition} \quad \} \\
 & (\neg I \cap R)^\cup \cap R^* \;\subseteq\; \bot\!\!\bot \\
= & \{ \quad \text{distributivity, } (\neg I)^\cup = \neg I \quad \} \\
 & \neg I \cap R^\cup \cap R^* \;\subseteq\; \bot\!\!\bot \\
\Leftarrow & \{ \quad [\, R^\cup \subseteq (R^*)^\cup \,] \text{, symmetry of union} \quad \} \\
 & \neg I \cap (R^*)^\cup \cap R^* \;\subseteq\; \bot\!\!\bot
\end{array}$$

$\Leftarrow$ { assumption: R^* is anti-symmetric
(i.e. $(R^*)^{\cup} \cap R^* = I$) }

$\neg I \cap I \subseteq \perp\!\!\perp$

$=$ { complement }

true .

□

Definition 8.11 is meaningful for arbitrary relations but we instantiate it primarily for finite graphs. Recall that nodes are points. (See definition 5.13.) So identifying a node in a cycle of graph G establishes that G is not acyclic. Formally, we have:

Lemma 8.14 Suppose a is a point. Then $a \circ R^+ \circ a = \perp\!\!\perp$ if relation R is acyclic. Conversely, if $a \circ R^+ \circ a \neq \perp\!\!\perp$, R is not acyclic, as witnessed by a ; that is, a is a point in a cycle of R .

Proof By lemma 5.15 and definition 5.13(c),

(8.15) $a \circ R^+ \circ a = a \quad \vee \quad a \circ R^+ \circ a = \perp\!\!\perp$

for all points a .

Assume R is acyclic. Then

$a \circ R^+ \circ a = a$

$=$ { a is coreflexive, lemmas 5.27 (with $p{,}b := a{,}a$) and 8.9 }

$a \subseteq I \cap R^+$

$\Rightarrow$ { assumption: R is acyclic (definition 8.11) }

$a = \perp\!\!\perp$

$\Rightarrow$ { a is a point so $a \neq \perp\!\!\perp$ }

false .

We conclude that, if R is acyclic, $a \circ R^+ \circ a = \perp\!\!\perp$ for all points a .

For the converse, suppose a is an atomic coreflexive and $a \circ R^+ \circ a \neq \perp\!\!\perp$. Then, by (8.15), $a \circ R^+ \circ a = a$. It follows that a is proper and, applying definition 8.11, a is in a cycle of R .

□

We now show that, for *finite* graphs, right- (or left-) definiteness equivales acyclicity. Lemma 8.16 shows that finiteness is not required to show that right- (or left-) definiteness implies acyclicity but the converse is not always true for relations on infinite sets. For example, the less-than ordering on real numbers is acyclic but it is not well-founded.

Lemma 8.16 A right-definite relation is acyclic. Symmetrically, a left-definite relation is acyclic.

Proof With a ranging over atomic coreflexives, we have

$$\begin{array}{ll}
 & \mathsf{rightdefinite}.R \\
\Rightarrow & \{ \quad \text{definition 8.1 (with } p := a \text{) and lemma 8.10} \quad \} \\
 & \langle \forall a :: a \subseteq \perp\!\!\!\perp \Leftarrow a \subseteq R^{+} \rangle \\
\Rightarrow & \{ \quad \text{the lattice of coreflexives is saturated, i.e. } \langle \cup a :: a \rangle = I \quad \} \\
 & I \cap R^{+} = \perp\!\!\!\perp \\
= & \{ \quad \text{definition 8.11} \quad \} \\
 & \mathsf{acyclic}.R \ .
\end{array}$$

The symmetric property of left-definiteness follows straightforwardly. (See the remarks above about the relation between left-definiteness of $R^{\cup}$ and right-definiteness of R.)
□

We turn now to the proof that definiteness follows from acyclicity. Like lemma 8.16, lemma 8.17 and corollary 8.18 below do not require finiteness of the relation R; however, their application in lemma 8.19, will force the restriction to finite graphs.

Earlier we argued informally that right-definiteness means the absence of infinite "descending" paths. Formally, we have:

Lemma 8.17 Suppose p is a coreflexive such that $p \neq \perp\!\!\!\perp$ and $p \subseteq (p \circ R^{+})_{>}$. Suppose a is a proper atomic coreflexive such that $a \subseteq p$. Then, with dummy b ranging over atomic coreflexives, we have

$$\langle \exists b :: b \neq \perp\!\!\!\perp \ \wedge \ b \subseteq p \ \wedge \ a \subseteq (b \circ R^{+})_{>} \ \wedge \ (a \circ R^{+})_{>} \subseteq (b \circ R^{+})_{>} \rangle \ .$$

Proof The proof of (8.17) is in two stages. First,

$$\begin{array}{ll}
 & a \subseteq p \\
\Rightarrow & \{ \quad \text{assumption: } p \subseteq (p \circ R^{+})_{>} \text{, transitivity} \quad \} \\
 & a \subseteq (p \circ R^{+})_{>} \\
= & \{ \quad \text{saturation assumption: definition 2.50, distributivity} \quad \} \\
 & a \subseteq \langle \cup b : b \subseteq p : (b \circ R^{+})_{>} \rangle \\
\Rightarrow & \{ \quad a \text{ is a proper atom, irreducibility: lemma 2.63} \quad \} \\
 & \langle \exists b : b \neq \perp\!\!\!\perp \wedge b \subseteq p : a \subseteq (b \circ R^{+})_{>} \rangle \ .
\end{array}$$

Second, assuming $a \subseteq (b \circ R^+)_>$,

$$
\begin{array}{ll}
 & (a \circ R^+)_> \\
\subseteq & \{ \quad \text{assumption, monotonicity} \quad \} \\
 & ((b \circ R^+)_> \circ R^+)_> \\
= & \{ \quad \text{domains (specifically theorem 5.9)} \quad \} \\
 & (b \circ R^+ \circ R^+)_> \\
\subseteq & \{ \quad R^+ \text{ is transitive, monotonicity} \quad \} \\
 & (b \circ R^+)_> \quad .
\end{array}
$$

□

Corollary 8.18 Suppose p is a coreflexive such that $p \neq \perp\!\!\perp$ and $p \subseteq (p \circ R^+)_>$. Then it is possible to construct an infinite sequence of proper atomic coreflexives a_i such that

$$\langle \forall i : 0 \leq i : a_i \subseteq p \rangle \ \wedge \ \langle \forall i,j : 0 \leq i < j : a_i \subseteq (a_j \circ R^+)_> \rangle \quad .$$

Proof The initial term a_0 is an arbitrary element of p. That is, $a_0 \subseteq p$. (Formally, we exploit the assumption that the lattice of coreflexives is atomic: see definition 2.49.) Subsequent nodes are constructed by exploiting lemma 8.17 (with $a,b := a_i, a_{i+1}$). Because, for all i,

$$(a_i \circ R^+)_> \subseteq (a_{i+1} \circ R^+)_>$$

it follows, by transitivity, that

$$\langle \forall i,j : i < j : (a_i \circ R^+)_> \subseteq (a_j \circ R^+)_> \rangle \quad .$$

Combining this with the fact that, for all i, $a_i \subseteq (a_{i+1} \circ R^+)_>$, we have:

$$\langle \forall i,j : 0 \leq i < j : a_i \subseteq (a_j \circ R^+)_> \rangle \quad .$$

□

This is the point at which we are obliged to introduce the finiteness assumption.

Lemma 8.19 Suppose G is a finite graph. Then G is right-definite if G is acyclic.

Proof We prove the contrapositive: if G is a finite graph that is not right-definite, then G is not acyclic.

Suppose G is not right-definite. Then there is a coreflexive p such that $p \neq \perp\!\!\perp$ and $p \subseteq (p \circ G^+)_>$. Applying corollary 8.18 with $R := G$, it is possible to construct an infinite sequence of nodes a_i such that

$$\langle \forall i,j : 0 \leq i < j : a_i \subseteq (a_j \circ G^+)_> \rangle \quad .$$

There is only a finite number of nodes; so, for some m and n, $m<n$ and $a_m = a_n$. Thus

$$a_m \subseteq (a_m \circ G^+)_> \quad .$$

Hence,

$$
\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{lemma 8.9 (with } a,R := a_m,G \text{)} \quad \} \\
 & a_m \subseteq G^+ \\
\Rightarrow & \{ \quad a_m = I \cap a_m \text{, monotonicity} \quad \} \\
 & a_m \subseteq I \cap G^+ \\
\Rightarrow & \{ \quad \perp\!\!\perp \neq a_m \text{, } \perp\!\!\perp \text{ is the least element} \quad \} \\
 & \perp\!\!\perp \neq I \cap G^+ \quad .
\end{array}
$$

That is, G is not acyclic.
□

Corollary 8.20 Suppose G is a finite graph. Then G is definite if G is acyclic.

Proof Straightforward combination of lemma 8.19 and properties of converse. First,

$$
\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{lemma 8.19} \quad \} \\
 & \langle \forall G : \mathsf{finite}.G : \mathsf{leftdefinite}.G \Leftarrow \mathsf{acyclic}.G \rangle \\
= & \{ \quad \text{converse is a bijection} \quad \} \\
 & \langle \forall G : \mathsf{finite}.G^{\cup} : \mathsf{leftdefinite}.G^{\cup} \Leftarrow \mathsf{acyclic}.G^{\cup} \rangle \\
= & \{ \quad \mathsf{finite}.G^{\cup} = \mathsf{finite}.G \text{, } \mathsf{leftdefinite}.G^{\cup} = \mathsf{rightdefinite}.G \text{,} \\
 & \quad\;\; \mathsf{acyclic}.G^{\cup} = \mathsf{acyclic}.G \quad \} \\
 & \langle \forall G : \mathsf{finite}.G : \mathsf{rightdefinite}.G \Leftarrow \mathsf{acyclic}.G \rangle \quad .
\end{array}
$$

So

$$
\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{lemma 8.19 and above} \quad \} \\
 & \quad \langle \forall G : \mathsf{finite}.G : \mathsf{leftdefinite}.G \Leftarrow \mathsf{acyclic}.G \rangle \\
 & \wedge \quad \langle \forall G : \mathsf{finite}.G : \mathsf{rightdefinite}.G \Leftarrow \mathsf{acyclic}.G \rangle
\end{array}
$$

$=$ { predicate caculus }

$\langle \forall \mathsf{G} : \mathsf{finite.G} : \mathsf{leftdefinite.G} \wedge \mathsf{rightdefinite.G} \Leftarrow \mathsf{acyclic.G} \rangle$

$=$ { $\mathsf{leftdefinite.G} \wedge \mathsf{rightdefinite.G} \equiv \mathsf{definite.G}$ }

$\langle \forall \mathsf{G} : \mathsf{finite.G} : \mathsf{definite.G} \Leftarrow \mathsf{acyclic.G} \rangle$.

□

To summarise, we have the following theorem.

Theorem 8.21 If G is a finite graph, G is acyclic equivales G is definite.

Proof Straightforward combination of corollary 8.16 and corollary 8.20.
□

Remark The term "definite" was, to our knowledge, first used by Carré [Car71]. Inspired by Conway's maxim [Con71, p.40] that any axiomatisation of a regular algebra should extend to (finite) matrices, an algebraic formulation of "definiteness" in a regular algebra was introduced in [Bac75, BC75]. (See section 3.3 and, in particular theorem 3.16.) This made it possible to establish a link between the notion of the "empty word property" [Sal69] —of both languages and matrices of languages— and the (well-known) notion of singularity of matrices in linear algebra. At that time no distinction was made between the notions of left- and right-definiteness, the sole application under consideration being finite matrices (equivalently, finite graphs) where —in view of theorem 8.21— the two notions are indistinguishable..

The distinction between left- and right-definiteness only emerged with the recognition that a regular algebra is an important subcomponent of a relation algebra, and that, in a relation algebra, right-definiteness corresponds to the fundamental concept of well-foundedness [DBvdW97] (and is distinct from left-definiteness). In the same way that definiteness in a regular algebra formulates Salomaa's (absence of the) empty word property of a matrix of languages, the notion offers an alternative but equivalent algebraic formulation of the acyclicity of a finite graph. The notions of left- and right-definiteness are, however, more general than acyclicity, as we have seen in this section. **End of Remark**

8.2 Starth Root and Reflexive-Transitive Reduction

In this section, we show that the reflexive-transitive reduction of a definite relation is the least starth root of the graph. It follows that the same is true of a finite, acyclic graph.

Recall the definition of reflexive-transitive reduction: definition 3.18. The definition of the function red is quite complicated, much of the complication being due to the need to eliminate self-loops. An acyclic relation has no self-loops so the definition can be simplified:

Lemma 8.22 If R is acyclic, then $R = R \cap \neg I$. So

$$\mathsf{red}.R \;=\; R \cap \neg(R \circ R^+) \;\; .$$

Proof

$R = R \cap \neg I$

$=$ { $R \supseteq R \cap \neg I$, anti-symmetry, $R \subseteq R$ }

$R \subseteq \neg I$

$=$ { shunting rule (2.27) }

$R \cap I \subseteq \perp\!\!\perp$

$\Leftarrow$ { $R \subseteq R^+$, monotonicity and transitivity }

$R^+ \cap I \subseteq \perp\!\!\perp$

$=$ { R is acyclic }

true .

The formula for $\mathsf{red}.R$ follows by instantiating (3.19) and replacing R by $R \cap \neg I$.
□

Theorem 8.23 The least starth root of a definite relation is its reflexive-transitive reduction. That is, for all definite relations R,

$$(\mathsf{red}.R)^* = R^* \quad \wedge \quad \langle \forall X : X^* = R^* : \mathsf{red}.R \subseteq X \rangle \;\; .$$

In particular, the least starth root of a finite, acyclic graph is its reflexive-transitive reduction.

Proof Assume that R is definite. By theorem 3.22, it suffices to prove the lefthand conjunct.

$(\mathsf{red}.R)^* \;=\; R^*$

$=$ { $\mathsf{red}.R \subseteq R$ and R is definite, so $\mathsf{red}.R$ is right-definite (theorem 8.21)
uep of relation algebra: theorem 8.4 }

$R^* \;=\; I \cup \mathsf{red}.R \circ R^*$

$\Leftarrow$ { $R^* = I \cup R^+$, Leibniz }

$R^+ \;=\; \mathsf{red}.R \circ R^*$

$=$ { R is left-definite,

$\quad\quad$ uep of relation algebra: theorem 8.4 $\}$

$\quad R^+ \;=\; \mathsf{red}.R \cup R^+ \circ R$

$=\quad\{\quad$ by lemma 8.16, R is acyclic; lemma 8.22 $\quad\}$

$\quad R^+ \;=\; (R \cap \neg(R \circ R^+)) \;\cup\; R^+ \circ R$

$=\quad\{\quad R \circ R^+ = R^+ \circ R$ and absorption rule of set calculus $\quad\}$

$\quad R^+ \;=\; R \cup R^+ \circ R$

$=\quad\{\quad$ fixed-point definition of transitive closure $\quad\}$

$\quad \mathsf{true}\;\;.$

The particular case of a finite, acyclic graph follows from corollary 8.20.
□

Observe that the proof of theorem 8.23 uses both left- and right-definiteness. The lexicographic ordering on words over an alphabet of size at least two demonstrates that just one of left- or right-definiteness is not sufficient: it is right-definite (i.e. well-founded) but it is not left-definite (i.e. its converse is not well-founded) and it does not have a least starth root: see example 3.26.

Examples of non-finite relations that are definite can be constructed using one's understanding of bound functions (section 6.8.6). An illustrative case is the relation R on integers defined by

$$\langle \forall m,n \;::\; m[\![R]\!]n \;\equiv\; \mathsf{even}.m \wedge \mathsf{odd}.n \wedge m<n \rangle \;\;.$$

The bound function is the function even. Indeed,

$$R \;=\; \mathsf{even}^{\cup} \circ (\not\Leftarrow) \circ \mathsf{even} \quad\wedge\quad \langle \forall m,n \;::\; m[\![\mathsf{red}.R]\!]n \;\equiv\; \mathsf{even}.m \wedge n=m+1 \rangle \;\;.$$

where the symbol $\not\Leftarrow$ denotes the complement of the only-if relation on booleans.

8.3 Minimal Nodes and Reachability

This section is about formulating and proving the property that, given a right-definite relation, the set of nodes "reachable" from a given set of nodes equals the set of nodes "reachable" from a minimal subset of the given set of nodes.

Suppose G is a graph. To define reachability we observe that node x is reachable from a set of nodes A if there exists $y{\in}A$ such that there is a path from y to x. That there is a path from y to x can of course be expressed as $y[\![G^*]\!]x$, so reachability of x from A becomes $\langle \exists y : y{\in}A : y[\![G^*]\!]x \rangle$ or by definition of composition: $\langle \exists y :: y[\![A{\circ}G^*]\!]x \rangle$. In the last expression we recognise the pointwise definition of the domain operator: if set

A is represented by the coreflexive p, the expression is equivalent to $x \in (p\circ G^*)_{>}$. Generalising from graph G to an arbitrary relation R, the point-free definition of $\mathsf{reachable.R.p}$ is therefore:

(8.24) $\quad \mathsf{reachable.R.p} \;=\; (p\circ R^*)_{>} \quad .$

That a node x is a minimal element of a set of nodes A means that x is an element of A and that, furthermore, there is no edge from a node in A to x. This is more formally expressed as $x{\in}A \wedge \neg\langle\exists y : y{\in}A : y[\![R]\!]x\rangle$. Alternatively, by again introducing the domain operator and representing set A by the coreflexive p, as $x \in p \cap (p\circ R)_{>\bullet}$. Replacing the intersection by a composition of coreflexives, the set $\mathsf{minimal.R.p}$ of minimal elements of p is thus defined as:

(8.25) $\quad \mathsf{minimal.R.p} \;=\; p \circ (p\circ R)_{>\bullet}$

The formal statement of the fact that the nodes reachable from set A coincide with the nodes reachable from the minimal elements of A now becomes:

Lemma 8.26 Suppose relation R is right-definite. Then, for all coreflexives p,

(8.27) $\quad \mathsf{reachable.R.p} \;=\; \mathsf{reachable.R.(minimal.R.p)} \quad .$

More generally, for all coreflexives p and q,

(8.28) $\quad \mathsf{reachable.R.p} \subseteq \mathsf{reachable.R.q} \quad \Leftarrow \quad \mathsf{minimal.R.p} \subseteq q \quad .$

Proof Assume that R is right-definite. We prove (8.27) by mutual inclusion. One inclusion is easy. From the definition (8.24) it is clear that $\mathsf{reachable.R}$ is a monotone function. Furthermore from (8.25) we see that p contains $\mathsf{minimal.R.p}$. Therefore

$$\mathsf{reachable.R.p} \supseteq \mathsf{reachable.R.(minimal.R.p)} \quad .$$

It remains to prove the other inclusion. Somewhere we have to use the assumption of right-definiteness, but how? We have to prove that

$$\mathsf{reachable.R.p} \subseteq \mathsf{reachable.R.(minimal.R.p)} \quad ,$$

whereof the *righthand* occurrence of $\mathsf{reachable}$ involves a reflexive-transitive closure. This suggests that we use the uep of relation algebra. Furthermore, it turns out that the expression $\mathsf{minimal.R.p}$ does not play a role. Therefore, we begin by deriving a condition implying

$$\mathsf{reachable.R.p} \subseteq \mathsf{reachable.R.q}$$

for arbitrary coreflexive q. (This turns out to be the property (8.28).) We begin by exploiting (the dual of) lemma 7.11:

$$
\begin{array}{ll}
 & \mathsf{reachable.R.p} \;\subseteq\; \mathsf{reachable.R.q} \\
= & \{ \quad \text{definition reachables: (8.24)} \quad \} \\
 & (p{\circ}R^*)_{>} \;\subseteq\; (q{\circ}R^*)_{>} \\
= & \{ \quad \text{the function } \langle p :: (p{\circ}R^*)_{>} \rangle \text{ is a closure operator} \\
 & \quad\;\; \text{(dual of lemma 7.11) and definition 2.44} \quad \} \\
 & p \;\subseteq\; (q{\circ}R^*)_{>} \quad .
\end{array}
$$

Now we can invoke the right-definiteness of R. From the discussion of theorem 8.4 on the uep of relation algebra it follows that, for right-definite relation R, relation $(q{\circ}R^*)_{>}$ is the greatest fixed point of the function $\langle X :: q \cup (X{\circ}R)_{>} \rangle$. Exploitation of this fact is the main step in the following calculation.

$$
\begin{array}{ll}
 & p \;\subseteq\; (q{\circ}R^*)_{>} \\
= & \{ \quad R \text{ is right-definite: } (q{\circ}R^*)_{>} = \langle \nu X :: q \cup (X{\circ}R)_{>} \rangle \,; \\
 & \quad\;\; \text{fixed-point induction} \quad \} \\
 & p \;\subseteq\; (q \cup p{\circ}R)_{>} \\
= & \{ \quad \text{domain operator is } \cup\text{-junctive} \quad \} \\
 & p \;\subseteq\; q \cup (p{\circ}R)_{>} \\
= & \{ \quad \text{shunting (2.27) in the coreflexive lattice} \quad \} \\
 & p \circ (p{\circ}R)_{>\bullet} \;\subseteq\; q \\
= & \{ \quad \text{definition (8.25)} \quad \} \\
 & \mathsf{minimal.R.p} \subseteq q \quad .
\end{array}
$$

With this calculation we have established the property (8.28). Instantiating q with $\mathsf{minimal.R.p}$ in this formula then gives the desired result:

$$\mathsf{reachable.R.p} \subseteq \mathsf{reachable.R.(minimal.R.p)} \qquad .$$

This completes the proof of the theorem.
□

An interesting observation can be made if we take a closer look at the antecedent of formula (8.28). After instantiating q to the empty relation and writing out the definition of $\mathsf{minimal.R}$ it reads: $p \circ (p{\circ}R)_{>\bullet} \subseteq \perp\!\!\perp$. Now we can apply shunting in the coreflexive lattice and we get $p \subseteq (p{\circ}R)_{>}$. This expression is the antecedent in (8.2). So, another formulation of a relation R being right-definite is: for all coreflexives p,

(8.29) $\quad p \subseteq \perp\!\!\perp \quad \Leftarrow \quad \mathsf{minimal.R.p} \subseteq \perp\!\!\perp \qquad ,$

or the equivalent contrapositive (using that $\perp\!\!\perp$ is the bottom of the lattice): for all coreflexives p,

(8.30) $p \neq \perp\!\!\perp \quad \Rightarrow \quad \mathsf{minimal.R.p} \neq \perp\!\!\perp$.

This is the familiar characterisation "every non-empty set has a minimal element" of well-foundedness.

Now we consider the converse of lemma 8.26. Is it true that a graph with property (8.27) is right-definite? This question can be answered affirmatively and the proof is simple. We show that a relation satisfying (8.27) also satisfies (8.29).

$$\begin{array}{ll}
 & \mathsf{minimal.R.p} \;=\; \perp\!\!\perp \\
\Rightarrow & \{ \quad \text{Leibniz} \quad \} \\
 & \mathsf{reachable.R.(minimal.R.p)} = \mathsf{reachable.R.}\perp\!\!\perp \\
= & \{ \quad \text{assumption: } \mathsf{reachable.R.(minimal.R.p)} \;=\; \mathsf{reachable.R.p}\,; \\
 & \quad\;\; \text{definition of } \mathsf{reachable}:\ (8.24) \quad \} \\
 & \mathsf{reachable.R.p} = (\perp\!\!\perp \circ R^*)_{>} \\
= & \{ \quad \text{definition of } \mathsf{reachable}:\ (8.24); \\
 & \quad\;\; \perp\!\!\perp \text{ is zero of composition} \quad \} \\
 & (p \circ R^*)_{>} = \perp\!\!\perp \\
\Rightarrow & \{ \quad I \subseteq R^* \quad \} \\
 & p \subseteq \perp\!\!\perp \;\; .
\end{array}$$

We thus conclude:

Theorem 8.31 Relation R is right-definite equivales for all coreflexives p,

$$\mathsf{reachable.R.p} \;=\; \mathsf{reachable.R.(minimal.R.p)} \;\; .$$

In particular, that (finite) graph G is acyclic equivales for all coreflexives p

$$\mathsf{reachable.G.p} \;=\; \mathsf{reachable.G.(minimal.G.p)} \;\; .$$

□

8.4 Topological Search

"Topological" search is an algorithm for visiting all the nodes in an acyclic graph in so-called "topological" order.

Definition 8.32 (Topological Order) A *topological ordering* of a homogeneous relation R of type A is a total, injective function ord from A to the natural numbers with the property that, for all elements a and b of A, $\mathrm{ord}.a < \mathrm{ord}.b$ if $a[\![R^+]\!]b$.
□

Expressed as a point-free formula, the requirement for the function ord to be a topological ordering of R is as follows:

$$(8.33)\quad I_A = \mathrm{ord}^{\cup} \circ \mathrm{ord} \;\;\wedge\;\; \mathrm{ord} \circ \mathrm{ord}^{\cup} \subseteq I_{\mathbb{N}} \;\;\wedge\;\; R^+ \subseteq \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \;\; .$$

Here we have used "less" to denote the less-than ordering on natural numbers rather than the symbol "$<$".

In order to verify the property of being a topological ordering, or —more importantly— to *construct* a topological ordering, it is useful to weaken the requirement, replacing R^+ by R:

Lemma 8.34 Suppose ord is a total, injective function of type $\mathbb{N} \leftarrow A$ and R is a homogeneous relation of type A. Then ord is a topological ordering of R equivales

$$R \subseteq \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \;\; .$$

Proof The proof is a straightforward application of the definition of transitive closure and fixed-point induction:

$$\begin{array}{ll}
 & R^+ \subseteq \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \\
\Leftarrow & \{ \quad R^+ = \langle \mu x :: R \cup x \circ x \rangle \text{ ; fixed-point induction} \quad \} \\
 & R \cup \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \circ \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \;\subseteq\; \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \\
= & \{ \quad \begin{array}{ll} & \mathsf{less} \circ \mathrm{ord} \circ \mathrm{ord}^{\cup} \circ \mathsf{less} \\ \subseteq & \{ \quad \mathrm{ord} \circ \mathrm{ord}^{\cup} \subseteq I \text{, monotonocity} \quad \} \\ & \mathsf{less} \circ \mathsf{less} \\ \subseteq & \{ \quad \mathsf{less} \text{ is transitive} \quad \} \\ & \mathsf{less} \quad ; \end{array} \\
 & \quad \text{definition of set union and monotonicity of composition} \quad \} \\
 & R \subseteq \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \\
\Leftarrow & \{ \quad R \subseteq R^+ \quad \} \\
 & R^+ \subseteq \mathrm{ord}^{\cup} \circ \mathsf{less} \circ \mathrm{ord} \;\; .
\end{array}$$

□

Lemma 8.34 means that the requirement (8.33) for the function ord to be a topological ordering of R can be simplified to

(8.35) $\quad I_A = \mathsf{ord}^{\cup} \circ \mathsf{ord} \quad \wedge \quad \mathsf{ord} \circ \mathsf{ord}^{\cup} \subseteq I_{\mathbb{N}} \quad \wedge \quad R \subseteq \mathsf{ord}^{\cup} \circ \mathsf{less} \circ \mathsf{ord} \quad .$

The less-than relation on natural numbers is, of course, well-founded — that is, right-definite in the terminology used here. The function ord in the definition of a topological ordering thus acts like a so-called *bound* function for establishing termination of a loop in a program. The relevant property is the following.

Lemma 8.36 Suppose ord is a total function of type $\mathbb{N} \leftarrow A$ for some A. Then the homogeneous relation $\mathsf{ord}^{\cup} \circ \mathsf{less} \circ \mathsf{ord}$ (where less denotes the less-than relation on natural numbers) is right-definite.

Proof Suppose p is a coreflexive of type A. Then

$$
\begin{array}{ll}
 & p \subseteq (p \circ \mathsf{ord}^{\cup} \circ \mathsf{less} \circ \mathsf{ord})_{>} \\
\Rightarrow & \{ \quad \text{monotonicity (aiming to exploit the functionality of } \mathsf{ord}) \quad \} \\
 & (p \circ \mathsf{ord}^{\cup})_{>} \subseteq ((p \circ \mathsf{ord}^{\cup} \circ \mathsf{less} \circ \mathsf{ord})_{>} \circ \mathsf{ord}^{\cup})_{>} \\
= & \{ \quad \text{domains: (5.9)} \quad \} \\
 & (p \circ \mathsf{ord}^{\cup})_{>} \subseteq (p \circ \mathsf{ord}^{\cup} \circ \mathsf{less} \circ \mathsf{ord} \circ \mathsf{ord}^{\cup})_{>} \\
\Rightarrow & \{ \quad \mathsf{ord} \text{ is functional, i.e. } \mathsf{ord} \circ \mathsf{ord}^{\cup} \subseteq I\text{, monotonicity} \quad \} \\
 & (p \circ \mathsf{ord}^{\cup})_{>} \subseteq (p \circ \mathsf{ord}^{\cup} \circ \mathsf{less})_{>} \\
\Rightarrow & \{ \quad \text{preparing for use of (8.2): (5.9)} \quad \} \\
 & (p \circ \mathsf{ord}^{\cup})_{>} \subseteq ((p \circ \mathsf{ord}^{\cup})_{>} \circ \mathsf{less}^{+})_{>} \\
\Rightarrow & \{ \quad \mathsf{less} \text{ is well-founded, i.e. right-definite: (8.2)} \quad \} \\
 & (p \circ \mathsf{ord}^{\cup})_{>} \subseteq \perp\!\!\perp \\
= & \{ \quad \text{domains: (5.6)} \quad \} \\
 & p \circ \mathsf{ord}^{\cup} \subseteq \perp\!\!\perp \\
\Rightarrow & \{ \quad \mathsf{ord} \text{ is total (i.e. } I \subseteq \mathsf{ord}^{\cup} \circ \mathsf{ord}), \\
 & \quad\;\; \text{monotonicity and } \perp\!\!\perp \text{ is zero of composition} \quad \} \\
 & p \subseteq \perp\!\!\perp \quad .
\end{array}
$$

Thus, by definition, $ord^{\cup} \circ less \circ ord$ is right-definite.
□

Lemma 8.36 is the basis of the use of so-called "bound functions" to establish termination of loops and recursion: the function ord "bounds" the number of iterations. The only property of the relation $less$ that is used in the proof of lemma 8.36 is that it is well-founded (right-definite). So "bound functions" can be used in conjunction with other well-founded relations although in some cases it would be difficult to interpret the function ord as a "bound". For example, the relation $less$ could be taken to be the lexicographic ordering on words; the function ord would then map a state to a word.

Corollary 8.37 Suppose ord is a topological ordering of the homogeneous relation R. Then R is right-definite.

Proof Immediate from lemmas 8.5, 8.34 and 8.36.
□

We now want to consider the converse of corollary 8.37. Is it the case that every right-definite relation can be topologically ordered? The answer is: no, not in general. (For example, the lexicographical ordering of words over a finite alphabet is well-founded but it is not possible to assign a number to each word that defines its position in the ordering.) The answer is, however, yes if we restrict attention to finite graphs. The proof is constructive. We assume that G is a finite graph that is acyclic and we present an algorithm that constructs a topological ordering of the nodes of G.

The development of the algorithm proceeds as follows. Given a finite graph G, the requirement is to construct a topological ordering ord of all the nodes of G: specifically, the postcondition that must be satisfied is given by (8.35).

The obvious strategy is to order the nodes one-by-one, beginning with the empty set of nodes and ending with all the nodes of G. In order to guarantee injectivity, an obvious choice is to assign to each node the number of nodes that have already been ordered. (Thus, the first node to be ordered is assigned the number 0, the second 1, and so on.) Introducing the coreflexive variable $seen$ to represent the nodes that have been ordered (the nodes that have been "seen" in the search of the graph) and the integer variable k to count the number of nodes in the set represented by $seen$, we design a loop that has invariants

(8.38) $seen = ord^{\cup} \circ ord \;\;\wedge\;\; ord \circ ord^{\cup} = \overline{\{j \mid 0 \leq j < k\}}$, and

(8.39) $seen \circ G \circ seen \subseteq ord^{\cup} \circ less \circ ord$.

The overbar notation used in (8.38) denotes the mapping from a set to its representation as a coreflexive. The invariant (8.38) states that ord is functional with right domain

$seen$ and it is injective with left domain the set of natural numbers less than k. The invariant (8.39) states that if there is an edge in G from a node a that has been "seen" to a node b that has also been "seen" then $ord.a < ord.b$.

The invariants (8.38) and (8.39) are clearly derived from (8.35) by the well-known, correct-by-construction design method of replacing a constant by a variable: in this case, several occurrences of the (sometimes invisible) identity relation are replaced.

The development thus far is summarised below. The property (8.40) listed as an invariant has yet to be derived. Also, queries ("???") have been added to indicate that the criterion for choosing node b is incomplete.

```
    { acyclic.G }
    $seen, ord, k := \perp\!\!\perp , \perp\!\!\perp , 0$
  ; { Invariant: (8.38) ∧ (8.39) ∧ (8.40) }
    while $I_A \neq seen$ do
       begin
         choose arbitrary node $b$ such that $b \subseteq {\sim}seen \wedge$ ???
       ; $seen := seen \cup b$
       ; $ord, k := ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b \ , \ k+1$
       end
    { $I_A = seen = ord^{\cup} \circ ord \ \wedge \ ord \circ ord^{\cup} \subseteq I_{\mathbb{N}} \ \wedge \ G \subseteq ord^{\cup} \circ less \circ ord$ }
```

The key element of the algorithm is how to choose the next node to be ordered. It is straightforward to verify that (8.38) is an invariant of the algorithm as shown. (See lemma 8.41 below.) The choice of node b must guarantee that (8.39) is maintained. That is, we require that, for all b and $seen$,

$$(seen \cup b) \circ G \circ (seen \cup b) \ \subseteq \ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ less \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)$$
$$\Leftarrow \quad seen \circ G \circ seen \subseteq ord^{\cup} \circ less \circ ord \ \wedge \ (8.38) \ \wedge \ b \subseteq {\sim}seen \ \wedge \ (8.40)$$

where (8.40) has yet to be derived.

Using distributivity properties, the left side of the topmost subset ordering expands to

$$seen \circ G \circ (seen \cup b) \ \cup \ b \circ G \circ (seen \cup b)$$

and, omitting two terms, the right side of this ordering expands to

$$ord^{\cup} \circ less \circ ord \ \cup \ ord^{\cup} \circ less \circ \overline{\{k\}} \circ \top\!\!\top \circ b \ .$$

(The two omitted terms are, in fact, equal to $\perp\!\!\perp$ but this fact is not needed.) Taking account of domains (specifically, $seen = ord^{\cup} \circ ord$ and $b \subseteq {\sim}seen$), the invariant (8.39) is thus maintained if

$$\begin{array}{ll} & seen \circ G \circ seen \;\subseteq\; ord^{\cup} \circ \mathsf{less} \circ ord \\ \wedge & seen \circ G \circ b \;\subseteq\; ord^{\cup} \circ \mathsf{less} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \\ \wedge & b \circ G \circ b \;=\; \perp\!\!\perp \\ \wedge & b \circ G \circ seen \;=\; \perp\!\!\perp \quad . \end{array}$$

The first conjunct is identical to the first conjunct on the right side of the implication; so it can be eliminated. The second conjunct follows from (8.38) and properties of the less-than ordering. The third conjunct is $true$ because G is assumed to be acyclic and hence has no self-loops. Finally, the fourth conjunct enables us to identify the as-yet-undefined invariant (8.40): specifically,

$$(8.40) \quad {\sim}seen \circ G \circ seen \;=\; \perp\!\!\perp \quad .$$

Of course, the introduction of a new invariant implies a new design obligation: property (8.40) is clearly established by the initialisation $seen{:=}\perp\!\!\perp$ but we must guarantee that it is maintained by the loop body. Doing so gives us the condition for choosing node b: maintaining the invariant demands that, for all b, G and $seen$,

$${\sim}(seen \cup b) \circ G \circ (seen \cup b) \;=\; \perp\!\!\perp \;\;\Leftarrow\;\; {\sim}seen \circ G \circ seen = \perp\!\!\perp \wedge \text{choice of } b \quad .$$

An easy calculation gives the condition for choosing b as:

$${\sim}(seen \cup b) \circ G \circ b \;=\; \perp\!\!\perp \quad .$$

This condition can be strengthened to:

$${\sim}seen \circ G \circ b \;=\; \perp\!\!\perp \quad .$$

In words, there are no edges in the graph G from an unseen node to node b. This completes the derivation of the algorithm:

```
    {  acyclic.G  }
    $seen, ord, k \;:=\; \perp\!\!\perp, \perp\!\!\perp, 0$
 ;  {  Invariant:  (8.38) $\wedge$ (8.39) $\wedge$ (8.40)  }
    while $I_A \neq seen$ do
       begin
```

```
      choose arbitrary node $b$ such that $b \subseteq \sim seen \;\wedge\; \sim seen \circ G \circ b = \perp\!\!\perp$
    ; $seen := seen \cup b$
    ; $ord,k \;:=\; ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b \;,\; k{+}1$
    end
{ $I_A = seen = ord^{\cup} \circ ord \;\wedge\; ord \circ ord^{\cup} \subseteq I_{\mathbb{N}} \;\wedge\; G \subseteq ord^{\cup} \circ less \circ ord$ }
```

There is one more —vital— proof obligation: we have to verify that the condition for choosing b can be satisfied. This is where the assumption that G is acyclic, and hence right-definite, is crucial: see lemma 8.45 below. (So far, we have only used the property that G has no self-loops.) The formal verification of all the informal claims made above now follows.

The algorithm clearly terminates since the size of the set represented by $seen$ increases by one at each iteration.

In order to verify that the algorithm meets its specification, there are three tasks remaining.

1. Establish that each of (8.38), (8.40) and (8.39) is truthified by the initialisation, and that the truth of each is invariant under execution of the loop body.

2. Prove that it is possible to choose a node b in accordance with the criterion for its choice.

3. Prove that the stated postcondition is a logical consequence of the invariant property and the condition for termination of the loop.

The first of these splits into three tasks, one for each of the stated properties. These tasks form lemmas 8.41, 8.42, and 8.43 below. The second —the central task both literally and figuratively— is the topic of lemma 8.45, and the third is the topic of lemma 8.46.

Lemma 8.41 Property (8.38) is an invariant of the algorithm.

Proof Property (8.38) is clearly true after the initial assignment $seen,ord := \perp\!\!\perp,\perp\!\!\perp$. The verification condition

$$
\begin{array}{ll}
 & (8.38)\,[seen,ord,k \;:=\; seen\cup b \;,\; ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b \;,\; k{+}1] \\
\Leftarrow & (8.38) \;\wedge\; b \subseteq \sim seen \;\wedge\; b \neq \perp\!\!\perp
\end{array}
$$

is a straightforward consequence of the proper atomicity of $\overline{\{k\}}$ and b (viz. $\overline{\{k\}} \circ \top\!\!\top \circ \overline{\{k\}} = \overline{\{k\}}$, $\top\!\!\top \circ \overline{\{k\}} \circ \top\!\!\top = \top\!\!\top$, and similarly for b). Specifically,

$$
\begin{array}{ll}
 & (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b) \\
= & \{ \quad \text{distributivity} \quad \} \\
 & ord^{\cup} \circ ord \;\cup\; ord^{\cup} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \;\cup\; b \circ \top\!\!\top \circ \overline{\{k\}} \circ ord \;\cup\; b \circ \top\!\!\top \circ \overline{\{k\}} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \\
= & \{ \qquad ord^{\cup} \circ \overline{\{k\}} \\
 & \qquad = \quad \{ \quad \text{assumption: (8.38) and domains} \quad \} \\
 & \qquad\quad ord \circ \overline{\{j \mid 0 \leq j < k\}} \circ \overline{\{k\}} \\
 & \qquad = \quad \{ \quad \overline{\{j \mid 0 \leq j < k\}} \circ \overline{\{k\}} \;=\; \bot\!\!\bot \quad \} \\
 & \qquad\quad \bot\!\!\bot \quad \} \\
 & ord^{\cup} \circ ord \;\cup\; b \circ \top\!\!\top \circ \overline{\{k\}} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \\
= & \{ \quad \text{assumption: (8.38);} \\
 & \quad\;\; \text{by cone rule: (4.16), and assumption: } b \neq \bot\!\!\bot\,, \\
 & \quad\;\; \top\!\!\top \circ \overline{\{k\}} \circ \overline{\{k\}} \circ \top\!\!\top = \top\!\!\top\,,\;\; b \circ \top\!\!\top \circ b = b \quad \} \\
 & seen \cup b \quad .
\end{array}
$$

The verification of the second conjunct is very similar:

$$
\begin{array}{ll}
 & (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b) \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \\
= & \{ \quad \text{distributivity,} \quad \} \\
 & ord \circ ord^{\cup} \;\cup\; ord \circ b \circ \top\!\!\top \circ \overline{\{k\}} \;\cup\; \overline{\{k\}} \circ \top\!\!\top \circ b \circ ord^{\cup} \;\cup\; \overline{\{k\}} \circ \top\!\!\top \circ b \circ b \circ \top\!\!\top \circ \overline{\{k\}} \\
= & \{ \qquad ord \circ b \\
 & \qquad = \quad \{ \quad \text{assumption: (8.38) and domains} \quad \} \\
 & \qquad\quad ord \circ seen \circ b \\
 & \qquad = \quad \{ \quad \text{assumption: } b \subseteq {\sim}seen\text{, i.e. } b = {\sim}seen \circ b \\
 & \qquad\qquad\quad seen \circ {\sim}seen = \bot\!\!\bot \quad \} \\
 & \qquad\quad \bot\!\!\bot \quad , \\
 & \quad\;\; \text{so, also } b \circ ord^{\cup} = \bot\!\!\bot \quad \} \\
 & ord \circ ord^{\cup} \;\cup\; \overline{\{k\}} \circ \top\!\!\top \circ b \circ b \circ \top\!\!\top \circ \overline{\{k\}} \\
= & \{ \quad \text{assumption: (8.38); } b \neq \bot\!\!\bot\text{, cone rule: (4.16)} \quad \} \\
 & \overline{\{j \mid 0 \leq j < k\}} \;\cup\; \overline{\{k\}} \circ \top\!\!\top \circ \overline{\{k\}} \\
= & \{ \quad \overline{\{k\}} \text{ is an atomic coreflexive, so } \overline{\{k\}} \circ \top\!\!\top \circ \overline{\{k\}} = \overline{\{k\}}\,, \\
 & \quad\;\; \text{properties of } < \text{ relation on natural numbers} \quad \}
\end{array}
$$

$$\overline{\{j \mid 0 \le j < k+1\}} \ .$$

□

Lemma 8.42 Property (8.40) is an invariant of the algorithm.

Proof Property (8.40) is clearly truthified by the initial assignment $seen := \perp\!\!\perp$. For the loop body, we verify that

$$(8.40)\,[seen := seen \cup b] \quad \Leftarrow \quad (8.40) \;\wedge\; \sim seen \circ G \circ b = \perp\!\!\perp$$

is a theorem for all coreflexives b and $seen$ and all relations G.

$$\begin{array}{ll}
 & \sim(seen \cup b) \circ G \circ (seen \cup b) \;=\; \perp\!\!\perp \\
= & \{ \quad \text{distributivity and } \perp\!\!\perp \text{ is least element} \quad \} \\
 & \sim(seen \cup b) \circ G \circ seen \;=\; \perp\!\!\perp \;\;\wedge\;\; \sim(seen \cup b) \circ G \circ b \;=\; \perp\!\!\perp \\
= & \{ \quad \text{assumption (8.40): } \sim seen \circ G \circ seen \;=\; \perp\!\!\perp \\
 & \quad\;\; \text{choice of } b\text{: } \sim seen \circ G \circ b \;=\; \perp\!\!\perp \\
 & \quad\;\; \sim(seen \cup b) \subseteq \sim seen \text{ and monotonicity} \quad \} \\
 & \mathsf{true} \ .
\end{array}$$

□

Lemma 8.43 Property (8.39) is an invariant of the algorithm.

Proof Property (8.39) is clearly true after the initial assignment $seen, ord := \perp\!\!\perp, \perp\!\!\perp$. The verification condition

$$\begin{array}{ll}
 & (8.39)\,[seen,ord \;:=\; seen \cup b \;,\; ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b] \\
\Leftarrow & b \subseteq \sim seen \;\wedge\; (8.40) \;\wedge\; \mathsf{acyclic}.G \;\wedge\; (8.38) \;\wedge\; (8.39)
\end{array}$$

is shown to be true for all $seen$, ord, b, k and G in several steps. First,

$$\begin{array}{ll}
 & (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ \mathsf{less} \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b) \\
\supseteq & \{ \quad \text{distributivity and ignoring two of the four terms} \quad \} \\
 & ord^{\cup} \circ \mathsf{less} \circ ord \;\cup\; ord^{\cup} \circ \mathsf{less} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \\
\supseteq & \{ \quad \text{assumption: (8.39)} \quad \} \\
 & seen \circ G \circ seen \;\cup\; ord^{\cup} \circ \mathsf{less} \circ \overline{\{k\}} \circ \top\!\!\top \circ b \ .
\end{array}$$

Second,

$$\begin{array}{ll}
& ord^{\cup} \circ less \circ \overline{\{k\}} \circ \top\!\!\top \\
= & \{ \quad \text{assumption: (8.38), and domains} \quad \} \\
& ord^{\cup} \circ \overline{\{j \mid 0 \leq j < k\}} \circ less \circ \overline{\{k\}} \circ \top\!\!\top \\
= & \{ \quad \text{property of } less \text{ (specifically } [\, 0 \leq j < k \Rightarrow j < k \,]) \\
& \quad\ \text{all-or-nothing rule} \quad \} \\
& ord^{\cup} \circ \overline{\{j \mid 0 \leq j < k\}} \circ \top\!\!\top \circ \overline{\{k\}} \circ \top\!\!\top \\
= & \{ \quad \overline{\{k\}} \neq \perp\!\!\perp \text{, cone rule: (4.16)} \quad \} \\
& ord^{\cup} \circ \overline{\{j \mid 0 \leq j < k\}} \circ \top\!\!\top \\
= & \{ \quad \text{assumption: (8.38), and domains} \quad \} \\
& seen \circ \top\!\!\top \\
\supseteq & \{ \quad \top\!\!\top \supseteq G \text{ and monotonicity} \quad \} \\
& seen \circ G \quad .
\end{array}$$

Putting the calculations together, we get that

$$\begin{array}{ll}
& (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ less \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b) \\
\supseteq & \{ \quad \text{first calculation (assuming (8.39))} \quad \} \\
& seen \circ G \circ seen \ \cup \ ord^{\cup} \circ less \circ \overline{\{k\}} \circ \top\!\!\top \circ b \\
\supseteq & \{ \quad \text{second calculation (assuming (8.38))} \quad \} \\
& seen \circ G \circ seen \ \cup \ seen \circ G \circ b \\
= & \{ \quad \text{distributivity} \quad \} \\
& seen \circ G \circ (seen \cup b) \quad .
\end{array}$$

That is, assuming (8.39) and (8.38),

(8.44) $(ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ less \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b) \ \supseteq \ seen \circ G \circ (seen \cup b) \quad .$

Our goal has thus become to complify the right side of (8.44) to

$$(seen \cup b) \circ G \circ (seen \cup b)$$

which, by distributivity, equals

$$seen \circ G \circ (seen \cup b) \ \cup \ b \circ G \circ (seen \cup b) \quad .$$

In order to achieve this goal, we must show that the rightmost term equals the empty relation, $\perp\!\!\perp$. Lemma 8.14 (with the instantiation $a,R := b,G$) and the precondition that G is acyclic establishes that $b \circ G \circ b = \perp\!\!\perp$. That $b \circ G \circ seen = \perp\!\!\perp$ is a consequence of (8.40) and $b \subseteq \sim seen$. Specifically,

$b \circ G \circ seen \;=\; \perp\!\!\perp$

$\Leftarrow \quad \{ \quad$ assumption: $b \subseteq {\sim}seen$, $\perp\!\!\perp$ is least element $\quad \}$

${\sim}seen \circ G \circ seen \;\subseteq\; \perp\!\!\perp$

$= \quad \{ \quad$ assumption: (8.40) $\quad \}$

true .

In summary, we have:

$(ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)^{\cup} \circ less \circ (ord \cup \overline{\{k\}} \circ \top\!\!\top \circ b)$

$\supseteq \quad \{ \quad$ (8.44) (assuming (8.39) and (8.38)) $\quad \}$

$seen \circ G \circ (seen \cup b)$

$= \quad \{ \quad$ assumption: $acyclic.G \wedge$ (8.40). hence

$\quad\quad b \circ G \circ b = \perp\!\!\perp$ and $b \circ G \circ seen = \perp\!\!\perp$ (see above) $\quad \}$

$seen \circ G \circ (seen \cup b) \;\cup\; b \circ G \circ b \;\cup\; b \circ G \circ seen$

$= \quad \{ \quad$ distributivity $\quad \}$

$(seen \cup b) \circ G \circ (seen \cup b)$.

We have thus verified that (8.39) is an invariant of the loop body.
□

We now establish that it is always possible to choose a node b as specified by the algorithm. We exploit the precondition that the graph G is acyclic, and hence definite.

Lemma 8.45 The set of nodes b such that $b \subseteq {\sim}seen$ and ${\sim}seen \circ G \circ b \;=\; \perp\!\!\perp$ is non-empty.

Proof For brevity, let g denote ${\sim}seen \circ G$. Then the choice criteria become $b \subseteq {\sim}seen$ and $g \circ b = \perp\!\!\perp$.

We show that the node b can always be chosen to be any element of $minimal.g.{\sim}seen$ and the latter is non-empty. First, note that g is acyclic since G is acyclic and $g \subseteq G$.

Applying theorem 8.21, it follows that g is definite and, in particular, right-definite. Thus

$minimal.g.{\sim}seen \neq \perp\!\!\perp$

$\Leftarrow \quad \{ \quad$ (8.30) with $R,p := g , {\sim}seen$; g is right-definite $\quad \}$

${\sim}seen \neq \perp\!\!\perp$

$= \quad \{ \quad$ condition for executing loop body: $seen \neq I_A$, i.e. ${\sim}seen \neq \perp\!\!\perp \quad \}$

true .

Clearly, by definition (8.25), $\mathsf{minimal}.R.p \subseteq p$, for all relations R and coreflexives p. So, we conclude that

$$\bot\!\!\bot \neq \mathsf{minimal}.g.{\sim}seen \subseteq {\sim}seen \quad .$$

This means that $\mathsf{minimal}.g.{\sim}seen$ is (the coreflexive representation of) a non-empty set of nodes b such that $b \subseteq {\sim}seen$. Also,

$$\begin{array}{ll}
 & g \circ \mathsf{minimal}.g.{\sim}seen \\
= & \{ \quad \text{definition of } \mathsf{minimal}: (8.25) \quad \} \\
 & g \circ {\sim}seen \circ ({\sim}seen \circ g)\!\succ \\
= & \{ \quad g = {\sim}seen \circ g \quad \} \\
 & g \circ {\sim}seen \circ g\!\succ \\
\subseteq & \{ \quad {\sim}seen \subseteq I \text{, monotonicity} \quad \} \\
 & g \circ g\!\succ \\
= & \{ \quad \text{domains: (5.11)} \quad \} \\
 & \bot\!\!\bot \quad .
\end{array}$$

That is, nodes b in (the set represented by) $\mathsf{minimal}.g.{\sim}seen$ also satisfy the choice criterion $g \circ b = \bot\!\!\bot$. (Formally, this is an application of the saturation axiom.)
□

Lemma 8.46 The postcondition

$$I_A = ord^{\cup} \circ ord \;\wedge\; ord \circ ord^{\cup} \subseteq I_{\mathbb{N}} \;\wedge\; G^{+} \subseteq ord^{\cup} \circ \mathsf{less} \circ ord$$

is implied by the conjunction of (8.38) and (8.39) (the second two conjuncts of the loop invariant) and $I_A = seen$ (the condition for terminating the loop).

Proof It is obvious, from the definition of $\overline{\{j \,|\, 0 \leq j < k\}}$ and the transitivity of equality, that the conjunction of (8.38), (8.39) and $I_A = seen$ implies

$$I_A = ord^{\cup} \circ ord \;\wedge\; ord \circ ord^{\cup} \subseteq I_{\mathbb{N}} \;\wedge\; G \subseteq ord^{\cup} \circ \mathsf{less} \circ ord \quad .$$

That this implies the postcondition follows from lemma 8.34.
□

The conclusion of this section is the following theorem.

Theorem 8.47 Suppose G is a finite graph. Then that there is a topological ordering of G equivales G is acyclic.

Proof The proof is by mutual implication. The algorithm just discussed establishes constructively that there is a topological ordering of G if G is acyclic. For the converse, suppose that ord is a topological ordering of G. Then

$$\begin{array}{ll}
 & I_A \cap G^{+} \\
\subseteq & \{ \quad \text{definition of topological ordering: (8.33), and monotonicity} \quad \} \\
 & \mathit{ord}^{\cup} \circ \mathit{ord} \cap \mathit{ord}^{\cup} \circ \mathsf{less} \circ \mathit{ord} \\
= & \{ \quad \text{by definition (8.33), } \mathit{ord} \text{ is a total function; distributivity} \quad \} \\
 & \mathit{ord}^{\cup} \circ (I_{\mathbb{N}} \cap \mathsf{less}) \circ \mathit{ord} \\
= & \{ \quad I_{\mathbb{N}} \cap \mathsf{less} = \perp\!\!\perp \text{ ; } \perp\!\!\perp \text{ is zero of composition} \quad \} \\
 & \perp\!\!\perp \ .
\end{array}$$

That is, G is acyclic.
□

Chapter 9

Components

This chapter is a preliminary to later discussion of the calculation of the so-called "strongly connected components" of a graph. The focus is on the algebraic properties, whilst later sections present an algorithm to calculate strongly connected components.

The strongly connected components of graph G are the equivalence classes of the relation $G^* \cap (G^{\cup})^*$. The algebraic properties that we present in this section are valid for arbitrary (homogeneous binary) relations and not just for finite graphs. However, we sometimes provide informal interpretations in terms of (paths in) graphs.

We begin by giving a definition of a "component" of a relation (definition 9.1) and then explore its properties, first for relations in general, then for transitive relations (section 9.1), and finally for transitive and symmetric relations (section 9.2).

"Strongly connected components" are defined in section 9.3. Properties of strongly connected components are derived in sections 9.4, 9.5, 9.6 and 9.7. Section 9.4 is about connectivity properties of nodes within and without the same strongly connected component. Section 9.5 records the well-known property that every node is an element of exactly one strongly connected component. Finally, section 9.7 formalises the structural decomposition of a graph into a collection of strongly connected components and an acyclic graph that is "pathwise homomorphic" to the given graph. The non-trivial proof of this property is enabled by a lemma on starth roots of a given graph formulated and presented in section 9.6.

Definition 9.1 Suppose p is a coreflexive and R is a relation. We say that p is *connected by* R iff $p \circ \top\!\top \circ p \subseteq R$. We say that p is a *component* of R iff p is connected by R and $\langle \forall q : q \circ \top\!\top \circ q \subseteq R : p \subseteq q \equiv p = q \rangle$.
□

Note that $\perp\!\perp$ is, by definition, connected by R. It is also a component of R in the case that the carrier of the lattice of coreflexives is the empty set.

An obvious corollary of definition 9.1 is the following:

Lemma 9.2

(a) Suppose q is a coreflexive and S is a relation. Then, q is connected by S if ($q \subseteq p$ and p is connected by R and $R \subseteq S$).

(b) p is connected by $R \cap S$ equivales p is connected by both R and S.

(c) The following are all equivalent:

- **(i)** p is connected by R
- **(ii)** p is connected by $R^{\cup}$
- **(iii)** p is connected by $R \cap R^{\cup}$

(d) The following are all equivalent:

- **(i)** p is a component of R
- **(ii)** p is a component of $R^{\cup}$
- **(iii)** p is a component of $R \cap R^{\cup}$

Proof (a) is obvious from the monotonicity of composition.
(b) is immediate from the definition of infima, in particular:

$$p \circ \top\!\top \circ p \subseteq R \cap S \quad \equiv \quad p \circ \top\!\top \circ p \subseteq R \wedge p \circ \top\!\top \circ p \subseteq S \quad .$$

(c) is obvious from the fact that p and $p \circ \top\!\top \circ p$ are symmetric. More specifically:

$$\begin{array}{ll} & p \circ \top\!\top \circ p \subseteq R \\ = & \{ \quad \text{converse} \quad \} \\ & (p \circ \top\!\top \circ p)^{\cup} \subseteq R^{\cup} \\ = & \{ \quad [\ (R \circ S)^{\cup} = S^{\cup} \circ R^{\cup}\ ]\ ,\ p^{\cup} = p\ ,\ \top\!\top^{\cup} = \top\!\top \quad \} \\ & p \circ \top\!\top \circ p \subseteq R^{\cup} \quad . \end{array}$$

This establishes the equivalence of (i) and (ii). That (i) implies (iii) is then established by (b) (with S instantiated to $R^{\cup}$) and the converse (iii) implies (i) is established by (a).
(d) Trivial consequence of (c) and the definition of component.
□

Informally, p is connected by R means that, when restricted to p, R equals the universal relation. Formally:

Lemma 9.3 For all coreflexives p and relations R,

$$p{\circ}\top\!\top{\circ}p \subseteq R \;\;\equiv\;\; p{\circ}\top\!\top{\circ}p = p{\circ}R{\circ}p \quad .$$

Proof This is proved by mutual implication as follows.

$$\begin{array}{ll}
 & p{\circ}\top\!\top{\circ}p \subseteq R \\
\Rightarrow & \{ \quad p{\circ}p = p \text{ , monotonicity of composition} \quad \} \\
 & p{\circ}\top\!\top{\circ}p \subseteq p{\circ}R{\circ}p \\
= & \{ \quad R \subseteq \top\!\top \text{ , monotonicity of composition, anti-symmetry} \quad \} \\
 & p{\circ}\top\!\top{\circ}p = p{\circ}R{\circ}p \\
\Rightarrow & \{ \quad p \subseteq I \text{ , monotonicity of composition, transitivity of } \subseteq \quad \} \\
 & p{\circ}\top\!\top{\circ}p \subseteq R \quad .
\end{array}$$

□

9.1 Transitive Relations

Lemma 9.4 Distinct components of a transitive relation are disjoint. Formally, suppose T is a transitive relation and p and q are both components of T. Then

$$p = q \;\;\vee\;\; p \cap q = \bot\!\!\bot \quad .$$

Proof For coreflexives p and q, $p{\circ}q = p{\cap}q = q{\circ}p$. This suggests applying the definition of a component in a way that introduces their product:

$$\begin{array}{ll}
 & p{\cap}q = \bot\!\!\bot \;\;\vee\;\; p = q \\
= & \{ \quad \text{idempotency of } \cup \quad \} \\
 & p{\cap}q = \bot\!\!\bot \;\;\vee\;\; p = p{\cup}q = q \\
\Leftarrow & \{ \quad p \subseteq p{\cup}q \text{ , } p \text{ is a component of } T \text{ ,} \\
 & \quad\;\; q \subseteq p{\cup}q \text{ , } q \text{ is a component of } T \quad \} \\
 & p{\cap}q = \bot\!\!\bot \;\;\vee\;\; (p{\cup}q){\circ}\top\!\top{\circ}(p{\cup}q) \subseteq T \\
= & \{ \quad \text{distributivity; } p \text{ and } q \text{ are both connected by } T \quad \} \\
 & p{\cap}q = \bot\!\!\bot \;\;\vee\;\; (p{\circ}\top\!\top{\circ}q \subseteq T \;\wedge\; q{\circ}\top\!\top{\circ}p \subseteq T) \\
= & \{ \quad \text{distributivity, } p{\circ}q = p{\cap}q = q{\circ}p \quad \} \\
 & (p{\circ}q = \bot\!\!\bot \vee p{\circ}\top\!\top{\circ}q \subseteq T) \;\wedge\; (q{\circ}p = \bot\!\!\bot \vee q{\circ}\top\!\top{\circ}p \subseteq T)
\end{array}$$

$\Leftarrow$ { cone rule: (4.16) with $R := p\circ q$ and $R := q\circ p$:

i.e. $p\circ q = \perp\!\!\perp \;\vee\; \top\!\!\top\circ p\circ q\circ\top\!\!\top = \top\!\!\top$,

and $q\circ p = \perp\!\!\perp \;\vee\; \top\!\!\top\circ q\circ p\circ\top\!\!\top = \top\!\!\top$ }

$p\circ\top\!\!\top\circ p\circ q\circ\top\!\!\top\circ q \subseteq T \;\wedge\; q\circ\top\!\!\top\circ q\circ p\circ\top\!\!\top\circ p \subseteq T$

$\Leftarrow$ { T is transitive, transitivity of $\subseteq$ }

$p\circ\top\!\!\top\circ p\circ q\circ\top\!\!\top\circ q \subseteq T\circ T \;\wedge\; q\circ\top\!\!\top\circ q\circ p\circ\top\!\!\top\circ p \subseteq T\circ T$

$\Leftarrow$ { p and q are connected by T , composition is monotonic }

true .

□

Lemma 9.5 Suppose T is a transitive relation and p and q are both components of T . Then

$$p\circ T\circ q \neq \perp\!\!\perp \wedge q\circ T\circ p \neq \perp\!\!\perp \;\Rightarrow\; p = q \quad .$$

Proof

$p\circ T\circ q \neq \perp\!\!\perp$

$\Rightarrow$ { cone rule: (4.16) }

$\top\!\!\top\circ p\circ T\circ q\circ\top\!\!\top = \top\!\!\top$

$\Rightarrow$ { Leibniz }

$p\circ\top\!\!\top\circ p\circ T\circ q\circ\top\!\!\top\circ q = p\circ\top\!\!\top\circ q$

$=$ { p and q are both connected by T ,

so, by lemma 9.3, $p\circ\top\!\!\top\circ p = p\circ T\circ p$ and $q\circ\top\!\!\top\circ q = q\circ T\circ q$ }

$p\circ T\circ p\circ T\circ q\circ T\circ q = p\circ\top\!\!\top\circ q$

$\Rightarrow$ { p and q are coreflexives, so $I \supseteq p$ and $I \supseteq q$

monotonicity and I is unit of composition }

$p\circ T\circ T\circ T\circ q \supseteq p\circ\top\!\!\top\circ q$

$\Rightarrow$ { T is a transitive relation, transitivity of $\supseteq$ }

$p\circ T\circ q \supseteq p\circ\top\!\!\top\circ q$

$=$ { $T \subseteq \top\!\!\top$, monotonicity of composition and anti-symmetry of $\subseteq$ }

$p\circ T\circ q = p\circ\top\!\!\top\circ q$.

In summary,

$$p\circ T\circ q \neq \perp\!\!\perp \Rightarrow p\circ T\circ q = p\circ\top\!\!\top\circ q \quad .$$

Interchanging p and q, we get

$$q\circ T\circ p \neq \perp\!\!\perp \Rightarrow q\circ T\circ p = q\circ\top\!\!\top\circ p \quad .$$

So,

$$\begin{array}{ll}
 & p\circ T\circ q \neq \perp\!\!\perp \wedge q\circ T\circ p \neq \perp\!\!\perp \\
\Rightarrow & \{ \quad \text{above, and } p \text{ and } q \text{ are both connected by } T \quad \} \\
 & \quad p\circ T\circ q = p\circ\top\!\!\top\circ q \wedge q\circ T\circ p = q\circ\top\!\!\top\circ p \\
 & \wedge \quad p\circ T\circ p = p\circ\top\!\!\top\circ p \wedge q\circ T\circ q = q\circ\top\!\!\top\circ q \\
\Rightarrow & \{ \quad \text{distributivity of composition over } \cup\text{, Leibniz} \quad \} \\
 & (p\cup q)\circ T\circ(p\cup q) = (p\cup q)\circ\top\!\!\top\circ(p\cup q) \\
\Rightarrow & \{ \quad \text{definition of connected and lemma 9.3,} \\
 & \qquad p\subseteq p\cup q \text{ and } q\subseteq p\cup q\text{, } p \text{ and } q \text{ are components of } T\text{,} \\
 & \qquad \text{definition 9.1} \quad \} \\
 & p = p\cup q = q \quad .
\end{array}$$

□

(The above proof parallels a pointwise proof. A pointwise proof would begin by assuming that there are points u,v in p and x, y in q such that $u\,T\,x$ and $y\,T\,v$. Then the argument would be made that u is connected by T to all points in q and, similarly x is connected by T to all points in p. In the point-free proof, it is not necessary to introduce four additional variables.)

Taking the contrapositive of lemma 9.5, we get:

Corollary 9.6 Suppose T is a transitive relation and p and q are both components of T. Then

$$p\circ T\circ q = \perp\!\!\perp \vee q\circ T\circ p = \perp\!\!\perp \Leftarrow p\neq q \quad .$$

□

Corollary 9.6 is the basis of the construction of a directed acyclic graph from the strongly connected components of a graph.

9.2 Transitive and Symmetric Relations

Undirected graphs correspond to symmetric relations. The transitive closure of relation R, denoted by R^+, has the property that

$$(R^+)^\cup = (R^\cup)^+ \quad .$$

(The proof of this property is a nice illustration of the fusion theorem: R^+ is a least fixed point and converse is Galois connected to itself and commutes with the function mapping x to $x{\circ}x$.) It follows that

$$(R^+)^\cup = R^+ \;\Leftarrow\; R^\cup = R \quad .$$

Here we consider properties of transitive and symmetric relations.

A remarkable (and perhaps surprising) property is that every undirected graph or its (undirected) complement is connected. We don't know any practical significance of this property but its proof is an interesting application of point-free reasoning. So, as an aside to the main development, this is proved in theorem 9.8 below.

Lemma 9.7 For all symmetric and transitive relations S and T,

$$S = \top\!\!\top \;\vee\; T = \top\!\!\top \;\Leftarrow\; S{\cup}T = \top\!\!\top \quad .$$

Proof Assume that S and T are symmetric and transitive, and $S{\cup}T = \top\!\!\top$. Then $S = S^\cup$, $T = T^\cup$, $S \supseteq S{\circ}S$, $T \supseteq T{\circ}T$, $S \supseteq \neg T$ and $T \supseteq \neg S$. So,

$$
\begin{array}{ll}
 & S = \top\!\!\top \;\vee\; T = \top\!\!\top \\
= & \{ \quad \text{complements (preparing for cone rule)} \quad \} \\
 & S = \top\!\!\top \;\vee\; \neg T = \bot\!\!\bot \\
\Leftarrow & \{ \quad \text{cone rule: (4.16)} \quad \} \\
 & S = \top\!\!\top \;\vee\; \top\!\!\top \circ \neg T \circ \top\!\!\top \neq \top\!\!\top \\
\Leftarrow & \{ \quad \text{boolean algebra and } S = \top\!\!\top \equiv S \supseteq \top\!\!\top \quad \} \\
 & S \supseteq \top\!\!\top \circ \neg T \circ \top\!\!\top \\
= & \{ \quad \text{assumption: } S{\cup}T = \top\!\!\top \quad \} \\
 & S \supseteq (S{\cup}T) \circ \neg T \circ (S{\cup}T) \\
= & \{ \quad \text{distributivity} \quad \} \\
 & S \supseteq S \circ \neg T \circ S \,\cup\, S \circ \neg T \circ T \,\cup\, T \circ \neg T \circ S \,\cup\, T \circ \neg T \circ T \\
\Leftarrow & \{ \quad S \text{ is transitive, so } S \supseteq S{\circ}S \text{ and } S \supseteq S{\circ}S{\circ}S,
\end{array}
$$

monotonicity of composition }

$S \supseteq \neg T \cup \neg T\circ T \cup T\circ\neg T \cup T\circ\neg T\circ T$

$=$ { by assumption: $S\supseteq\neg T$, suprema }

$S\supseteq\neg T\circ T \;\wedge\; S\supseteq T\circ\neg T \;\wedge\; S\supseteq T\circ\neg T\circ T$

$=$ { middle exchange rule, $S=S^{\cup}$, $T=T^{\cup}$ }

$T\supseteq\neg S\circ T \;\wedge\; T\supseteq T\circ\neg S \;\wedge\; T\supseteq T\circ\neg S\circ T$

$\Leftarrow$ { T is transitive, so $T \supseteq T\circ T$ and $T \supseteq T\circ T\circ T$,

monotonicity of composition }

$T\supseteq\neg S$

$=$ { shunting rule (2.27) }

$S\cup T=\top\!\top$.

□

Theorem 9.8 Suppose R is a symmetric relation. Then

$$R^*=\top\!\top \;\vee\; (\neg R)^*=\top\!\top \;.$$

Proof Suppose R is symmetric. If $\top\!\top=\bot\!\bot$ then $\top\!\top=S=\bot\!\bot$ for all relations S and the theorem is trivial. So assume that $\bot\!\bot\neq\top\!\top$. Then

$R^*=\top\!\top \;\vee\; (\neg R)^*=\top\!\top$

$\Leftarrow$ { $(\neg R)^{\cup}=\neg(R^{\cup})$ and $(R^*)^{\cup}=(R^{\cup})^*$;

lemma 9.7 with $S,T := R^*, (\neg R)^*$ }

$R^*\cup(\neg R)^* = \top\!\top$

$\Leftarrow$ { for all S, $S=\top\!\top \equiv S\supseteq\top\!\top$

$R^*\supseteq R$, $(\neg R)^*\supseteq\neg R$, transitivity of $\supseteq$ }

$R\cup\neg R = \top\!\top$

$=$ { complements }

true .

□

We now continue our investigation.

Lemma 9.9 Suppose T is a transitive and symmetric relation. Then $(T\circ p)_<$ is connected by T if p is connected by T.

Proof We have

$(T\circ p)_< \circ \top\!\!\top \circ (T\circ p)_<$

$=$ { theorem 5.7(a) and (c) }

$T\circ p\circ \top\!\!\top \circ p\circ T^\cup$

$\subseteq$ { assume p is connected by T ;
definition 9.1 and monotonicity of composition }

$T\circ T\circ T^\cup$

$\subseteq$ { T is transitive and symmetric }

T .

The lemma follows by definition of is-connected-by.
□

Theorem 9.10 Suppose T is a transitive and symmetric relation. Then $p = (T\circ p)_<$ if p is a component of T .

Proof Assume T is transitive and symmetric and p is a component of T .

$p = (T\circ p)_<$

$\Leftarrow$ { assumptions, lemma 9.9, and definition 9.1 of component }

$p \subseteq (T\circ p)_<$

$=$ { coreflexive-condition isomorphism }

$p\circ\top\!\!\top \subseteq T\circ p\circ\top\!\!\top$

$\Leftarrow$ { p is a component of T , so p is connected by T
i.e. $p\circ\top\!\!\top\circ p \subseteq T$
monotonicity of composition and transitivity of $\subseteq$ }

$p\circ\top\!\!\top \subseteq p\circ\top\!\!\top\circ p\circ p\circ\top\!\!\top$

$=$ { p is a coreflexive, so $p\circ p = p$, cone rule: (4.16) }

$p\circ\top\!\!\top \subseteq p\circ\top\!\!\top\circ p\circ\top\!\!\top \;\wedge\; (\top\!\!\top\circ p\circ\top\!\!\top = \top\!\!\top \;\vee\; p = \bot\!\!\bot)$

$=$ { distributivity of conjunction over disjunction
Leibniz and $\bot\!\!\bot$ is zero of composition and least element }

true .
□

Corollary 9.11 The components of an equivalence relation T are atoms in the lattice of fixed points of the function that maps coreflexive q to $(T{\circ}q)_<$. That is, if T is an equivalence relation and p is a component of T,

$$(q \subseteq p \ \equiv\ q = p \lor q = \bot\!\!\bot) \ \Leftarrow\ q = (T{\circ}q)_< \ .$$

Proof Apply lemma 2.65 with f instantiated to the function that maps coreflexive q to $(T{\circ}q)_<$. This function is a complementation-fixed closure operator by theorem 7.12.
□

Theorem 9.12 Suppose p is a coreflexive, T is a transitive and symmetric relation and q is a component of T. Then

$$p{\circ}T{\circ}q = \bot\!\!\bot \Leftarrow p{\circ}q = \bot\!\!\bot \ .$$

In particular, the property holds when p and q are both components of T.

Proof

$$\begin{array}{ll}
& p{\circ}T{\circ}q \\
= & \{ \quad \text{property of domains: } [\ R = R_< \circ R\ ] \text{ with } R := T{\circ}q \quad \} \\
& p \circ (T{\circ}q)_< \circ T \circ q \\
= & \{ \quad \text{theorem 9.10 with } p := q \quad \} \\
& p{\circ}q{\circ}T{\circ}q \\
= & \{ \quad \text{assume } p{\circ}q = \bot\!\!\bot,\ \bot\!\!\bot \text{ is zero of composition} \quad \} \\
& \bot\!\!\bot \ .
\end{array}$$

□

9.3 Strongly Connected Components

The notion of a “strongly connected component” of a finite graph is prominent in algorithmic graph theory. This section is about fundamental properties of strongly connected components. Since the properties do not depend on the finiteness of graphs, we present them for arbitrary relations.

Definition 9.13 (Strongly Connected Component) Coreflexive p is said to be a *strongly connected component* of relation R if p is a component of R^*.
□

Definition 9.14 The function equiv mapping arbitrary relations to equivalence relations is defined by, for all R,

$$\mathsf{equiv}.R \;=\; R^* \cap (R^*)^{\cup} \quad .$$

It is a well-known fact that $\mathsf{equiv}.R$ is an equivalence relation (i.e. it is reflexive, transitive and symmetric). The straightforward (point-free) proof is omitted.
□

Theorem 9.15 Suppose p is a strongly connected component of R. Then p is a component of $\mathsf{equiv}.R$. Conversely, every component of $\mathsf{equiv}.R$ is a strongly connected component of R.

Proof Immediate from the definition of strongly-connected and lemma 9.2(d).
□

Theorem 9.16 Suppose p is a strongly connected component of R. Then

$$p \;=\; (\mathsf{equiv}.R \circ p)_<$$

Moreover, p is an atom in the lattice of fixed points of the function that maps p to $(\mathsf{equiv}.R \circ p)_<$.

Proof Immediate from the definition of strongly-connected, lemma 9.2, theorem 9.10 and corollary 9.11.
□

9.4 Absolute Connectivity

This section is about paths in a graph connecting two nodes in one and the same strongly connected component of the graph. We show that all nodes on such paths are elements of the strongly connected component.

As in section 9.3, the finiteness of graphs is not used and the stated properties are valid for arbitrary relations; nevertheless, we interpret the properties in terms of graphs.

Recall that $\sim p$ denotes the negation of p in the lattice of coreflexives. For a finite graph, lemma 9.18 states that there are no paths from component p to itself that pass through nodes not in p. The lemma is a corollary of lemma 9.17.

Lemma 9.17 Suppose p is a strongly connected component of relation R. Then

$$p \;=\; (p \circ R^*)_> \;\cap\; (R^* \circ p)_< \quad .$$

Proof Let us abbreviate $(p\circ R^*)_> \cap (R^*\circ p)_<$ to q. We have to prove that $q = p$. In order to exploit the assumption that p is a a strongly connected component of R, the goal is to prove that q is connected by R^*.

$$
\begin{array}{ll}
 & q\circ\top\!\top\circ q \\
\subseteq & \{ \quad q = (p\circ R^*)_> \cap (R^*\circ p)_<\text{, monotonicity} \quad \} \\
 & (R^*\circ p)_<\circ\top\!\top\circ(p\circ R^*)_> \\
= & \{ \quad [\ R_<\circ\top\!\top = R\circ\top\!\top\ ]\text{ with } R := R^*\circ p\,, \\
 & \quad\ \ [\ \top\!\top\circ R_> = \top\!\top\circ R\ ]\text{ with } R := p\circ R^* \quad \} \\
 & R^*\circ p\circ\top\!\top\circ p\circ R^* \\
= & \{ \quad p\text{ is strongly connected by } R\,, \\
 & \quad\ \ \text{definitions 9.1 and 9.13, and lemma 9.3} \quad \} \\
 & R^*\circ p\circ R^*\circ p\circ R^* \\
\subseteq & \{ \quad p\subseteq I\text{, monotonicity of composition} \quad \} \\
 & R^*\circ R^*\circ R^* \\
= & \{ \quad R^* = R^*\circ R^* \quad \} \\
 & R^* \quad .
\end{array}
$$

That is, by definition 9.1, q is connected by R^*. Hence

$$
\begin{array}{ll}
 & p = q \\
\Leftarrow & \{ \quad p\text{ is strongly connected by } R\text{, definitions 9.13 and 9.1} \quad \} \\
 & q\text{ is connected by } R^* \ \wedge\ p\subseteq q \\
= & \{ \quad \text{above, definition of } q \quad \} \\
 & p \subseteq (p\circ R^*)_> \cap (R^*\circ p)_< \\
= & \{ \quad I\subseteq R^*\text{, monotonicity and properties of coreflexives} \quad \} \\
 & \text{true} \quad .
\end{array}
$$

□

Lemma 9.18 Suppose R is a relation and p is a strongly connected component of R. Then

$$p\circ R^*\circ{\sim}p\circ R^*\circ p = \perp\!\!\perp \quad .$$

Proof We have:

$$p\circ R^*\circ{\sim}p\circ R^*\circ p \;=\; \bot\!\!\bot$$

$=\quad\{\quad$ domains $\quad\}$

$$p\circ R^*\circ(p\circ R^*)_>\circ{\sim}p\circ(R^*\circ p)_<\circ R^*\circ p \;=\; \bot\!\!\bot$$

$\Leftarrow\quad\{\quad \bot\!\!\bot$ is zero of composition $\quad\}$

$$(p\circ R^*)_>\circ{\sim}p\circ(R^*\circ p)_< \;=\; \bot\!\!\bot$$

$\Leftarrow\quad\{\quad [\,p\circ q = p\cap q\,]$ (for coreflexives p and q), properities of intersection $\quad\}$

$$(p\circ R^*)_> \cap (R^*\circ p)_< \;\subseteq\; p$$

$=\quad\{\quad$ lemma 9.17 $\quad\}$

true .

□

Like lemma 9.18, lemma 9.19 below is valid for all relations but, for finite graphs, it formulates a property of paths between nodes in the same strongly connected component: in this case, in terms of the edges that form the paths. The first term, $p\circ\top\!\!\top\circ p$, is the relation that holds between all nodes in the same component p. The second and third terms capture the existence of paths defined by edges from the component p. The third term is more complex than the second term; it is included because it expresses more directly that elements of strongly connected component p are connected by paths formed of edges connecting elements of p. Specifically, the term $p\circ R$ represents the edges in R from a node in p, and the term $p\circ R\circ p$ represents the edges of R that connect nodes in p. So $(p\circ R)^*\circ p$ is interpreted as the relation between two nodes of which the second is in p that are connected by edges that are from nodes in p; similarly, $p\circ(p\circ R\circ p)^*$ represents the relation between two nodes of which the first is in p and that are connected by edges that connect nodes in p. The outer occurrences of "p" are necessary because (for all R) R^* includes the identity relation.

Lemma 9.19 Suppose R is a relation and p is a strongly connected component of R. Then

$$p\circ\top\!\!\top\circ p \;=\; (p\circ R)^*\circ p \;=\; p\circ(p\circ R\circ p)^*\circ p \quad .$$

Proof The equality between the second and third terms is straightforward:

$$p\circ(p\circ R\circ p)^*$$

$=\quad\{\quad$ mirror rule: $[\; R\circ(S\circ R)^* = (R\circ S)^*\circ R \;]$ with $R,S := p\,,p\circ R \quad\}$

$$(p\circ p\circ R)^*\circ p$$

$=\quad\{\quad p$ is a coreflexive, so $p\circ p = p \quad\}$

$$(p\circ R)^*\circ p \quad .$$

It is somewhat more difficult to establish the equality between the first and second terms, which we now do.

The relation R^* represents paths to and from all nodes and not just nodes in p. In order to separate out paths that are not to or not from nodes in p we begin by complifying R^*:

$$
\begin{array}{ll}
 & R^* \\
= & \{ \quad p \cup {\sim}p = I \quad \} \\
 & ((p \cup {\sim}p) \circ R \circ (p \cup {\sim}p))^* \\
= & \{ \quad \text{distributivity of composition over union,} \\
 & \quad\ \ \text{idempotency of set union and } p \cup {\sim}p = I \quad \} \\
 & (p \circ R \circ p \ \cup \ R \circ {\sim}p \ \cup \ {\sim}p \circ R)^* \\
= & \{ \quad \text{star decomposition} \quad \} \\
 & (p \circ R \circ p)^* \circ ((R \circ {\sim}p \ \cup \ {\sim}p \circ R) \circ (p \circ R \circ p)^*)^* \quad .
\end{array}
$$

We have indeed constructed a complicated expression for R^*. It is the composition of two terms; our goal is to show that the second term can be eliminated when we consider $p \circ R^* \circ p$. So that the expressions don't become too long, let us write the second term in the composition as S^*. That is,

(9.20) $S = (R \circ {\sim}p \ \cup \ {\sim}p \circ R) \circ (p \circ R \circ p)^* \ \ \wedge \ \ R^* = (p \circ R \circ p)^* \circ S^*$.

We show that

(9.21) $p \circ S^* \circ p = p$.

We have:

$$
\begin{array}{ll}
 & p \circ S^* \circ p \\
= & \{ \quad S^* = I \cup S \circ S^*, \\
 & \quad\ \ \text{distributivity of composition over union, etc.} \quad \} \\
 & p \cup p \circ S \circ S^* \circ p \\
= & \{ \quad \text{1st conjunct of (9.20), distributivity and } p \circ {\sim}p = \perp\!\!\perp \quad \} \\
 & p \cup p \circ R \circ {\sim}p \circ (p \circ R \circ p)^* \circ S^* \circ p \\
= & \{ \quad \text{2nd conjunct of (9.20)} \quad \} \\
 & p \cup p \circ R \circ {\sim}p \circ R^* \circ p \\
= & \{ \quad R \subseteq R^*, \text{ lemma 9.18} \quad \} \\
 & p \quad .
\end{array}
$$

We can now complete the calculation.

$$
\begin{array}{ll}
 & p\circ\top\!\top\circ p \\
= & \{ \quad p \text{ is a strongly connected component of } R\,, \\
 & \quad\;\; \text{definition 9.13 and lemma 9.3} \quad \} \\
 & p\circ R^*\circ p \\
= & \{ \quad (9.20) \quad \} \\
 & p\circ(p\circ R\circ p)^*\circ S^*\circ p \\
= & \{ \quad \text{mirror rule: } [\ R\circ(S\circ R)^* = (R\circ S)^*\circ R\ ] \text{ with } R,S := p\,,p\circ R\,,\ p\circ p = p \quad \} \\
 & (p\circ R)^*\circ p\circ S^*\circ p \\
= & \{ \quad (9.21) \quad \} \\
 & (p\circ R)^*\circ p \quad .
\end{array}
$$

□

9.5 Saturation

Note that atomicity has not been used anywhere above. Saturated atomicity is necessary to show that all nodes in a graph are elements of a strongly connected component of the graph. The calculations are straightforward:

Lemma 9.22 For all points a and relations R, $(\text{equiv}.R\circ a)_<$ is a strongly connected component of R. (Recall definition 5.13 of a point.)

Proof We exploit theorem 9.15. Accordingly, we have to show that $(\text{equiv}.R\circ a)_<$ is a component of $\text{equiv}.R$. That is, $(\text{equiv}.R\circ a)_<$ is connected by $\text{equiv}.R$ and it is maximal among such coreflexives.

First, we show that $(\text{equiv}.R\circ a)_<$ is connected by $\text{equiv}.R$. For all points a and all relations R, we have:

$$
\begin{array}{ll}
 & (\text{equiv}.R\circ a)_<\circ\top\!\top\circ(\text{equiv}.R\circ a)_< \\
= & \{ \quad \text{domains (specifically theorem 5.7(a))} \quad \} \\
 & \text{equiv}.R\circ a\circ\top\!\top\circ(\text{equiv}.R\circ a)^\cup \\
= & \{ \quad \text{converse} \quad \} \\
 & \text{equiv}.R\circ a\circ\top\!\top\circ a\circ\text{equiv}.R \\
= & \{ \quad a\circ\top\!\top\circ a = a:\ \text{definition 5.13(c)} \quad \}
\end{array}
$$

$$
\begin{array}{ll}
 & \mathsf{equiv.R} \circ a \circ \mathsf{equiv.R} \\
\subseteq & \{ \quad a \subseteq I \text{, monotonicity} \quad \} \\
 & \mathsf{equiv.R} \circ \mathsf{equiv.R} \\
\subseteq & \{ \quad \mathsf{equiv.R} \text{ is transitive} \quad \} \\
 & \mathsf{equiv.R} \quad .
\end{array}
$$

Now we must show that, if a is a point,

$$\langle \forall q : q \circ \top\!\top \circ q \subseteq \mathsf{equiv.R} : (\mathsf{equiv.R} \circ a)_< \subseteq q \;\equiv\; (\mathsf{equiv.R} \circ a)_< = q \rangle \quad .$$

Suppose q is a coreflexive such that $q \circ \top\!\top \circ q \subseteq \mathsf{equiv.R}$. Then, by lemma 9.3,

$$q \circ \top\!\top \circ q \;=\; q \circ \mathsf{equiv.R} \circ q \quad .$$

So,

$$
\begin{array}{ll}
 & (\mathsf{equiv.R} \circ a)_< \supseteq q \\
= & \{ \quad \text{coreflexive-condition isomorphism} \quad \} \\
 & \mathsf{equiv.R} \circ a \circ \top\!\top \supseteq q \circ \top\!\top \\
\Leftarrow & \{ \quad q \circ \top\!\top \circ q \subseteq \mathsf{equiv.R} \quad \} \\
 & q \circ \top\!\top \circ q \circ a \circ \top\!\top \supseteq q \circ \top\!\top \\
\Leftarrow & \{ \quad \text{monotonicity} \quad \} \\
 & \top\!\top \circ q \circ a \circ \top\!\top \supseteq \top\!\top \\
= & \{ \quad \text{assume } (\mathsf{equiv.R} \circ a)_< \subseteq q \\
 & \quad\quad \text{then, since } I \subseteq \mathsf{equiv.R},\ (I \circ a)_< \subseteq q \\
 & \quad\quad \text{i.e. } a \subseteq q \text{ and } q \circ a = a \quad \} \\
 & \top\!\top \circ a \circ \top\!\top \supseteq \top\!\top \\
= & \{ \quad a \text{ is a point, cone rule (4.16)} \quad \} \\
 & \mathsf{true} \quad .
\end{array}
$$

We have thus shown that, if a is a point,

$$\langle \forall q : q \circ \top\!\top \circ q \subseteq \mathsf{equiv.R} : (\mathsf{equiv.R} \circ a)_< \supseteq q \;\Leftarrow\; (\mathsf{equiv.R} \circ a)_< \subseteq q \rangle \quad .$$

The required equivalence is a straightforward consequence of the anti-symmetry and reflexivity of the subset relation.
□

The converse of lemma 9.22 is the following:

Lemma 9.23 If p is a strongly connected component of R, and a is a point such that $a \subseteq p$, then $p = (\mathsf{equiv}.R \circ a)_<$.

Proof Assume p is a strongly connected component of R, and a is a point such that $a \subseteq p$. Then,

$$\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{theorem 9.16} \quad \} \\
 & p \;=\; (\mathsf{equiv}.R \circ p)_< \\
\Rightarrow & \{ \quad a \subseteq p\text{, monotonicity of composition and domains} \quad \} \\
 & p \;\supseteq\; (\mathsf{equiv}.R \circ a)_< \\
\Rightarrow & \{ \quad \text{lemma 9.22, theorem 9.15 and definition 9.1} \quad \} \\
 & p \;=\; (\mathsf{equiv}.R \circ a)_< \quad .
\end{array}$$

□

Summarising, we have:

Theorem 9.24 Suppose R is a homogeneous relation. Then the strongly connected components of R are given by $\langle \cup a\!:\!\mathsf{point}.a\!:\!\{(\mathsf{equiv}.R \circ a)_<\}\rangle$. The strongly connected components partition the set of all points[1]. That is, distinct strongly connected components are disjoint and each point is an element of a strongly connected component (specifically, a is an element of $(\mathsf{equiv}.R \circ a)_<$).

Proof Lemmas 9.22, 9.23, 9.4 and 7.13 (with $R := \mathsf{equiv}.R$).

□

9.6 Starth Roots of the Equivalence Relation

We have defined $\mathsf{equiv}.R$ as $R^* \cap (R^*)^\cup$. (See definition 9.14.) It is useful to express it as E^* where (for graph R) E represents the edges in R that connect nodes in the same strongly connected component (i.e. nodes that are "E"quivalent under the relation $\mathsf{equiv}.R$). This is the content of theorem 9.26.

One application of theorem 9.26 is theorem 9.28, which states —with a minor qualification— that a graph G being acyclic is equivalent to the relation $\mathsf{equiv}.G$ being the identity relation. Application of theorem 9.26 is also an important step in the proof of theorem 9.30 below, which decomposes paths in a graph into paths in an acyclic graph connecting strongly connected components of the graph. First, a lemma:

[1]When applied to graphs, "points" are "nodes".

Lemma 9.25 For all relations R, U, V and W,

$$R^* \cap U{\circ}V{\circ}W \;=\; R^* \cap U\circ(R^*\cap V)\circ W \quad\Leftarrow\quad U\cup W\subseteq(R^\cup)^* \ .$$

(Note that composition has precedence over intersection. The spacing of our formulae is designed to make this clear.)

Proof We calculate the condition on U and W as follows.

$$\begin{array}{ll}
 & R^* \cap U{\circ}V{\circ}W \;=\; R^* \cap U\circ(R^*\cap V)\circ W \\
= & \{ \quad V\supseteq R^*\cap V \text{, monotonicity and anti-symmetry} \quad \} \\
 & R^* \cap U{\circ}V{\circ}W \;\subseteq\; R^* \cap U\circ(R^*\cap V)\circ W \\
= & \{ \quad \text{properties of } \cap \quad \} \\
 & R^* \cap U{\circ}V{\circ}W \;\subseteq\; U\circ(R^*\cap V)\circ W \\
\Leftarrow & \{ \quad \text{modularity rule (4.8) with } R,S,T := U\,,V{\circ}W\,,R^*\,, \\
 & \quad\;\; \text{and symmetric rule with } R,S,T := W\,,V\,,U^\cup\circ R^* \quad \} \\
 & U\circ(U^\cup\circ R^*\circ W^\cup \;\cap\; V)\circ W \;\subseteq\; U\circ(R^*\cap V)\circ W \\
\Leftarrow & \{ \quad \text{monotonicity} \quad \} \\
 & U^\cup\circ R^*\circ W^\cup \;\cap\; V \;\subseteq\; R^*\cap V \\
\Leftarrow & \{ \quad R^*\circ R^*\circ R^* \;=\; R^* \text{, monotonicity} \quad \} \\
 & U^\cup\subseteq R^* \;\wedge\; W^\cup\subseteq R^* \\
= & \{ \quad (4.1) \text{ and } (R^\cup)^* \;=\; (R^*)^\cup \quad \} \\
 & U\subseteq(R^\cup)^* \;\wedge\; W\subseteq(R^\cup)^* \ .
\end{array}$$

(The antecedent in the statement of the lemma is, of course, equivalent to the last line of the calculation.)
□

Now, the theorem:

Theorem 9.26 For all relations R,

$$\text{equiv}.R \;=\; (R^\cup\cap R^*)^* \;=\; (R\cap(R^\cup)^*)^* \ .$$

Proof We begin by proving, by induction on k, that, for all U and W,

$$(9.27)\quad R^*\cap U{\circ}(R^\cup)^k{\circ}W \;=\; R^*\cap U{\circ}(R^\cup\cap R^*)^k{\circ}W \;\Leftarrow\; U\cup W\subseteq(R^\cup)^* \ .$$

The basis, $k=0$ is trivial since $X^0=I$, for all X. For the induction step, assume U and W are such that $U\cup W\subseteq(R^\cup)^*$. Then,

$R^* \;\cap\; U\circ(R^{\cup}\cap R^*)^{k+1}\circ W$

$=\quad\{\quad$ definition of $(R^{\cup}\cap R^*)^{k+1}\quad\}$

$R^* \;\cap\; U\circ(R^{\cup}\cap R^*)^{k}\circ(R^{\cup}\cap R^*)\circ W$

$=\quad\{\quad$ by assumption, $U\subseteq(R^{\cup})^*$; so $U\circ(R^{\cup}\cap R^*)^k\subseteq(R^{\cup})^*$,

also, by assumption, $W\subseteq(R^{\cup})^*$

lemma 9.25 with $U,V,W := U\circ(R^{\cup}\cap R^*)^k , R^{\cup} , W\quad\}$

$R^* \;\cap\; U\circ(R^{\cup}\cap R^*)^{k}\circ R^{\cup}\circ W$

$=\quad\{\quad$ by assumption, $W\subseteq(R^{\cup})^*$; so $R^{\cup}\circ W \;\subseteq\; (R^{\cup})^*$

also, by assumption, $U\subseteq(R^{\cup})^*$

induction hypothesis (9.27) with $W \;:=\; R^{\cup}\circ W\quad\}$

$R^* \;\cap\; U\circ(R^{\cup})^{k}\circ R^{\cup}\circ W$

$=\quad\{\quad$ definition of $(R^{\cup})^{k+1}\quad\}$

$R^* \;\cap\; U\circ(R^{\cup})^{k+1}\circ W$.

By induction, we have established (9.27) for all natural numbers k. Hence,

$\mathsf{equiv}.R$

$=\quad\{\quad$ definition 9.14 $\quad\}$

$R^*\cap(R^*)^{\cup}$

$=\quad\{\quad (R^*)^{\cup}=(R^{\cup})^*$, definition of star as a sum of powers $\quad\}$

$R^*\cap\langle\cup k : 0\le k : (R^{\cup})^k\rangle$

$=\quad\{\quad$ distributivity $\quad\}$

$\langle\cup k \,:\, 0\le k \,:\, R^*\cap(R^{\cup})^k\rangle$

$=\quad\{\quad$ (9.27) with $U,W := I,I\quad\}$

$\langle\cup k \,:\, 0\le k \,:\, R^*\cap(R^{\cup}\cap R^*)^k\rangle$

$=\quad\{\quad$ distributivity $\quad\}$

$R^*\cap\langle\cup k \,:\, 0\le k \,:\, (R^{\cup}\cap R^*)^k\rangle$

$=\quad\{\quad$ definition of star as a sum of powers, $R^*\supseteq(R^{\cup}\cap R^*)^*\quad\}$

$(R^{\cup}\cap R^*)^*$.

The final equality in the statement of the lemma follows by symmetry (formally, by replacing R by $R^{\cup}$ in the first equality and using the properties of converse).

$\square$

Given that theorem 9.26 expresses a property that some might regard as obvious, the proof is surprisingly complicated: the induction hypothesis is non-trivial. It is also unfortunate that the proof uses the definition of the star operator as a sum of powers (and not as a least fixed point). A proof using fixed-point fusion would be preferable —albeit by mutual inclusion— but, so far, has eluded us.

The following theorem exploits theorem 9.26.

Theorem 9.28 If R is acyclic, $\mathsf{equiv.R}$ is the identity relation. That is,

$$I \cap \mathsf{R}^+ = \perp\!\!\perp \quad\Rightarrow\quad \mathsf{equiv.R} = I \ .$$

Conversely, if $\mathsf{equiv.R}$ is the identity relation, $\mathsf{R} \cap \neg I$ is acyclic. That is,

$$\mathsf{equiv.R} = I \quad\Rightarrow\quad I \cap (\mathsf{R} \cap \neg I)^+ = \perp\!\!\perp \ .$$

(In terms of graphs, $\mathsf{R} \cap \neg I$ is the graph R with "self-loops" removed.)

Proof Suppose $I \cap \mathsf{R}^+ = \perp\!\!\perp$. Then

$$\begin{array}{cl}
 & \mathsf{equiv.R} \\
= & \{ \quad \text{theorem } 9.26 \quad \} \\
 & (\mathsf{R}^\cup \cap \mathsf{R}^*)^* \\
\subseteq & \{ \quad \text{modularity rule: (4.8), monotonicity} \quad \} \\
 & (\mathsf{R}^\cup \circ (I \ \cap \ \mathsf{R} \circ \mathsf{R}^*))^* \\
= & \{ \quad \mathsf{R} \circ \mathsf{R}^* = \mathsf{R}^+ \text{, assumption: } I \cap \mathsf{R}^+ = \perp\!\!\perp \quad \} \\
 & (\mathsf{R}^\cup \circ \perp\!\!\perp)^* \\
= & \{ \quad \perp\!\!\perp \text{ is zero of composition, } \perp\!\!\perp^* = I \quad \} \\
 & I \ .
\end{array}$$

That is, $\mathsf{equiv.R} \subseteq I$. Since, $I \subseteq \mathsf{equiv.R}$, it follows by anti-symmetry of set inclusion that $\mathsf{equiv.R} = I$.

For the converse, we have:

$$\begin{array}{cl}
 & I \cap (\mathsf{R} \cap \neg I)^+ \\
= & \{ \quad [\ \mathsf{R}^+ = \mathsf{R} \circ \mathsf{R}^*\ ] \text{ with } \mathsf{R} := \mathsf{R} \cap \neg I,\ \mathsf{R}^* = (\mathsf{R} \cap \neg I)^* \quad \} \\
 & I \ \cap \ (\mathsf{R} \cap \neg I) \circ \mathsf{R}^* \\
\subseteq & \{ \quad \text{modularity rule: (4.8), } I \text{ is unit of composition} \quad \}
\end{array}$$

$$
\begin{array}{ll}
& (R\cap\neg I)\circ((R\cap\neg I)^{\cup}\cap R^{*}) \\
\subseteq & \{ \quad R\cap\neg I\subseteq\neg I\,,\ (R\cap\neg I)^{\cup}\subseteq R^{\cup}\,,\text{ theorem 9.26,} \\
& \quad [\,R\subseteq R^{*}\,]\ \text{ with } R:=R^{\cup}\cap R^{*} \\
& \quad \text{monotonicity (of converse, composition and star)} \quad \} \\
& \neg I\circ \mathsf{equiv}.R \quad .
\end{array}
$$

Thus

$$
\begin{array}{ll}
& \mathsf{equiv}.R\subseteq I \\
\Rightarrow & \{ \quad \text{above, monotonicity of composition and transitivity of } \subseteq \quad \} \\
& I\cap(R\cap\neg I)^{+}\ \subseteq\ \neg I\circ I \\
= & \{ \quad I \text{ is unit of composition, complements,} \\
& \quad \text{idempotency of intersection} \quad \} \\
& I\cap(R\cap\neg I)^{+}=\perp\!\!\perp \quad .
\end{array}
$$

□

Note that, although theorem 9.28 is valid for all relations, its significance is primarily when applied to finite graphs; the more significant property of a non-finite relation is whether or not it is left- or right-definite (or both).

9.7 A Pathwise Homomorphism

A well-known property is that the strongly connected components of a graph G define an acyclic graph G'. The nodes of the graph G' are the strongly connected components of G, and the edges of G' are the edges of G that connect nodes of G in distinct strongly connected components. Moreover, there is a path in G from node u to node v equivales there is a path in G' from the strongly connected component containing u to the strongly connected component containing v. The primary purpose of this section is to formalise this theorem.

Because the nodes of G and G' are different, it is necessary to use a *typed* algebra of *heterogeneous* relations rather than the *untyped* algebra of *homogeneous* relations. As remarked earlier, the rules that we have been using remain valid provided some caution is exercised when overloading notation.

Suppose N is a set (of "nodes") and G is a relation of type $N{\sim}N$ (the "edges" of the "graph"). As we have seen the function

$$\langle a \ :\ a{\in}N\ :\ \mathsf{Set}.(\mathsf{equiv}.G\circ a)_{<}\rangle$$

maps nodes to strongly connected components. Let us denote this function by sc and the set of strongly connected components of G by C. Then sc has type $C\leftarrow N$ and, by theorem 7.7,

(9.29) $\mathsf{equiv}.G \;=\; \mathsf{sc}^{\cup}\circ\mathsf{sc}$.

The relation

$$\mathsf{sc}\circ G\circ\mathsf{sc}^{\cup} \;\cap\; \neg I_C$$

is a homogeneous relation on the strongly connected components of G, i.e. a relation of type $C{\sim}C$. Informally, it is a graph obtained from the graph G by coalescing the nodes in a strongly connected component of G into a single node whilst retaining the edges of G that connect nodes in distinct strongly connected components. Theorem 9.30 establishes the formal relationship between its reflexive-transitive closure and G^*.

Theorem 9.30 Let $\mathcal{A}$ denote $\mathsf{sc}\circ G\circ\mathsf{sc}^{\cup} \cap \neg I_C$. Then,

(9.31) $G^* \;=\; \mathsf{sc}^{\cup}\circ\mathcal{A}^*\circ\mathsf{sc}$.

Moreover, $\mathcal{A}$ is acyclic. That is,

(9.32) $I_C\cap\mathcal{A}^+ \;=\; \perp\!\!\perp$.

It follows that $\mathcal{A}^*$ is a partial ordering of the strongly connected components of G.

Proof With theorem 9.26 in mind, we split G into two relations: D and E where D is defined by

$$D \;=\; G\cap\neg((G^{\cup})^*)$$

and E is defined by

$$E \;=\; G\cap(G^{\cup})^* \;\; .$$

The relation D captures the edges of G that connect “D”istinct strongly connected components. To be precise:

(9.33) $\mathsf{sc}\circ D\circ\mathsf{sc}^{\cup} \subseteq \neg I_C$,

since

$$\begin{array}{ll} & \mathsf{sc}\circ D\circ\mathsf{sc}^{\cup} \subseteq \neg I_C \\ = & \{ \quad \text{definition of } D \quad \} \\ & \mathsf{sc}\circ(G\cap\neg((G^{\cup})^*))\circ\mathsf{sc}^{\cup} \subseteq \neg I_C \end{array}$$

$=$ { middle-exchange (4.18),
I_C is unit of composition, and complements }

$sc^{\cup} \circ sc \subseteq \neg G \cup (G^{\cup})^*$

$\Leftarrow$ { (9.29) }

$equiv.G \subseteq (G^{\cup})^*$

$=$ { $equiv.G = G^* \cap (G^*)^{\cup}$ and $(G^*)^{\cup} = (G^{\cup})^*$ }

true .

Conversely, the relation E captures the edges of G that are in “E”qual strongly connected components:

(9.34) $sc \circ E \circ sc^{\cup} \subseteq I_C$,

since

$sc \circ E \circ sc^{\cup}$

$\subseteq$ { $E \subseteq E^*$ and monotonicity }

$sc \circ E^* \circ sc^{\cup}$

$=$ { by (9.29) and theorem 7.7 with $R := G$,
$E^* = equiv.G = sc^{\cup} \circ sc$ }

$sc \circ sc^{\cup} \circ sc \circ sc^{\cup}$

$\subseteq$ { sc is a function }

I_C .

In order to prove (9.31) and (9.32) we need three additional properties of D. The first,

(9.35) $D^{\cup} \cap G^* = \perp\!\!\perp$,

is obvious from the definition of D and properties of converse and complement:

$D^{\cup} \cap G^*$

$=$ { definition of D }

$(G \cap \neg((G^{\cup})^*))^{\cup} \cap G^*$

$=$ { distributivity properties of converse and $[\, (G^{\cup})^{\cup} = G \,]$ }

$G^{\cup} \cap \neg(G^*) \cap G^*$

$=$ { $[\, R \cap \neg S \cap S = \perp\!\!\perp \,]$ with $R,S := G^{\cup}, G^*$ }

$\perp\!\!\perp$.

The second,

(9.36) $\quad \mathsf{G}^* \;=\; \mathsf{equiv.G} \circ (\mathsf{D} \circ \mathsf{equiv.G})^* \quad ,$

is proved as follows:

$$\begin{array}{ll}
 & \mathsf{G}^* \\
= & \{ \quad \mathsf{D} \cup \mathsf{E} = \mathsf{G} \quad \} \\
 & (\mathsf{D} \cup \mathsf{E})^* \\
= & \{ \quad \text{star decomposition} \quad \} \\
 & \mathsf{E}^* \circ (\mathsf{D} \circ \mathsf{E}^*)^* \\
= & \{ \quad \text{by theorem 9.26 with } \mathsf{R} := \mathsf{G}\,,\ \mathsf{E}^* = \mathsf{equiv.G} \quad \} \\
 & \mathsf{equiv.G} \circ (\mathsf{D} \circ \mathsf{equiv.G})^* \quad .
\end{array}$$

The third property,

(9.37) $\quad \mathcal{A} \;=\; \mathsf{sc} \circ \mathsf{D} \circ \mathsf{sc}^{\cup} \quad ,$

is a combination of (9.33) and (9.34):

$$\begin{array}{ll}
 & \mathcal{A} \\
= & \{ \quad \text{definition of } \mathcal{A}\,,\ \mathsf{D} \cup \mathsf{E} = \mathsf{G} \quad \} \\
 & \mathsf{sc} \circ (\mathsf{D} \cup \mathsf{E}) \circ \mathsf{sc}^{\cup} \;\cap\; \neg I_C \\
= & \{ \quad \text{distributivity} \quad \} \\
 & (\mathsf{sc} \circ \mathsf{D} \circ \mathsf{sc}^{\cup} \cap \neg I_C) \;\cup\; (\mathsf{sc} \circ \mathsf{E} \circ \mathsf{sc}^{\cup} \;\cap\; \neg I_C) \\
= & \{ \quad (9.33) \text{ and } (9.34) \quad \} \\
 & \mathsf{sc} \circ \mathsf{D} \circ \mathsf{sc}^{\cup} \quad .
\end{array}$$

We now prove (9.31).

$$\begin{array}{ll}
 & \mathsf{G}^* \\
= & \{ \quad (9.36) \quad \} \\
 & \mathsf{equiv.G} \circ (\mathsf{D} \circ \mathsf{equiv.G})^* \\
= & \{ \quad (9.29) \quad \} \\
 & \mathsf{sc}^{\cup} \circ \mathsf{sc} \circ (\mathsf{D} \circ \mathsf{sc}^{\cup} \circ \mathsf{sc})^* \\
= & \{ \quad \text{mirror rule} \quad \}
\end{array}$$

$$\mathsf{sc}^{\cup}\circ(\mathsf{sc}\circ \mathrm{D}\circ \mathsf{sc}^{\cup})^{*}\circ \mathsf{sc}$$
$$= \qquad \{ \qquad (9.37) \qquad \}$$
$$\mathsf{sc}^{\cup}\circ \mathcal{A}^{*}\circ \mathsf{sc} \quad .$$

It remains to prove that $\mathcal{A}$ is acyclic. We have:

$$\mathrm{I}_{\mathrm{C}}\cap \mathcal{A}^{+}$$
$$= \qquad \{ \qquad \mathcal{A}^{+} = \mathcal{A}\circ\mathcal{A}^{*} \text{ and } (9.37) \qquad \}$$
$$\mathrm{I}_{\mathrm{C}}\ \cap\ \mathsf{sc}\circ \mathrm{D}\circ \mathsf{sc}^{\cup}\circ(\mathsf{sc}\circ \mathrm{D}\circ \mathsf{sc}^{\cup})^{*}$$
$$= \qquad \{ \qquad \text{mirror rule and } (9.29) \qquad \}$$
$$\mathrm{I}_{\mathrm{C}}\ \cap\ \mathsf{sc}\circ(\mathrm{D}\circ \mathsf{equiv.G})^{*}\circ \mathrm{D}\circ \mathsf{sc}^{\cup}$$
$$\subseteq \qquad \{ \qquad \text{modularity rule: (4.8) (applied in both forms)} \qquad \}$$
$$\mathsf{sc}\circ(\mathsf{sc}^{\cup}\circ \mathsf{sc}\circ \mathrm{D}^{\cup}\ \cap\ (\mathrm{D}\circ \mathsf{equiv.G})^{*})\circ \mathrm{D}\circ \mathsf{sc}^{\cup}$$
$$= \qquad \{ \qquad (9.29) \qquad \}$$
$$\mathsf{sc}\circ(\mathsf{equiv.G}\circ \mathrm{D}^{\cup}\ \cap\ (\mathrm{D}\circ \mathsf{equiv.G})^{*})\circ \mathrm{D}\circ \mathsf{sc}^{\cup}$$
$$\subseteq \qquad \{ \qquad \text{modularity rule: (4.8), and } \mathsf{equiv.G} = (\mathsf{equiv.G})^{\cup} \qquad \}$$
$$\mathsf{sc}\circ \mathsf{equiv.G}\circ(\mathrm{D}^{\cup}\ \cap\ \mathsf{equiv.G}\circ(\mathrm{D}\circ \mathsf{equiv.G})^{*})\circ \mathrm{D}\circ \mathsf{sc}^{\cup}$$
$$= \qquad \{ \qquad (9.36) \qquad \}$$
$$\mathsf{sc}\circ \mathsf{equiv.G}\circ(\mathrm{D}^{\cup}\cap \mathrm{G}^{*})\circ \mathrm{D}\circ \mathsf{sc}^{\cup}$$
$$= \qquad \{ \qquad (9.35) \text{ and } \perp\!\!\perp \text{ is zero of composition} \qquad \}$$
$$\perp\!\!\perp \quad .$$

Property (9.32) follows from the fact that $\perp\!\!\perp \subseteq \mathrm{R}$, for all R, and anti-symmetry of the subset relation.
□

Theorem 9.30 is valid for all relations G and not just for graphs. (Nowhere have we used the assumption that the set of nodes is finite.) Its primary importance, however, is that solving path problems can be decomposed into solving the problems for each individual strongly connected component and then combining the results using a topological search of an acyclic graph. Perhaps surprisingly, it is also used when inverting real matrices in order to preserve sparsity. As shown in [BC75], the standard so-called elimination techniques for inverting a matrix are algebraically identical to algorithms for constructing paths in a graph. (Essentially, $\mathbf{A}^{-1} = (\mathbf{1}-(\mathbf{1}-\mathbf{A}))^{-1} = (\mathbf{1}-\mathbf{A})^{*}$. The elimination algorithms exploit the star-decomposition rule to decompose the computation of $\mathbf{A}^{-1}$ into smaller components; the mirror rule is then used to evaluate $\mathbf{A}^{-1}$ for row/column matrices.) In this application, a topological search is often called "forward substitution". See also [BC82] for more detailed discussion of sparsity considerations.

(Of course, this does not mean that theorem 9.30 is valid for other interpretations of the star operator. For example, if G is a matrix of languages, it is not valid. Many steps in the calculation are valid in other interpretations but lemma 9.25 relies on the modularity rule, which is valid for relations but not for languages.)

Part IV

Graph Searching

Chapter 10

Generic Algorithms

In chapters 11 and 13, we show how depth-first search is used to compute the strongly connected components of a finite graph. There are two ways that depth-first search can be implemented. The first is an iterative algorithm that explicitly maintains a stack representing incomplete searches. The second is a recursive algorithm.

In general, the implementation of a recursive algorithm involves maintaining a stack representing incomplete computations but the algorithm itself typically does not make explicit use of the stack. One way to reason about recursive algorithms is to make the stack explicit. In our analysis of depth-first search we do not adopt that approach; instead, we choose to tackle the recursion head on using fixed-point induction as the primary tool. This poses significant challenges concerning how to present the calculations in a way that truly supports understanding of the algorithms.

This chapter is a prelude to chapters 11 and 13 intended as a gentle introduction to the more complex calculations of those chapters. In section 10.1, we present a generic graph-searching algorithm of which depth-first search is an instance. The algorithm determines the set of nodes that can be reached in a graph from a given set of nodes. Reasoning about the algorithm is a combination of fixed-point induction and a lemma, lemma 10.1, that helps to characterise when a search is complete. As mentioned earlier, depth-first search is an instance of the generic algorithm; in this way, section 10.1 is an introduction to chapter 11.

Then, in section 10.2, we consider repeated graph searches: that is, starting from an empty set of nodes, repeatedly initiating a new search from a node that has not already been "seen". Just as in section 10.1, we consider a generic algorithm whereby new searches are initiated using a choice *function*. The choice is recorded in the algorithm by the construction of a function that we call the "delegate" function: the "delegate" of a node a is the node b from which the search that "sees" a was initiated.

Apart from being total and functional, no other requirements are placed on the choice of initiating nodes. We see, however, in chapter 13 how depth-first search is used to construct a choice function with the property that the "delegate" of a node a is a representative of the strongly connected component of which a is a member. In this way, section 10.2 is a necessary preliminary to section 13.

10.1 A Generic Graph-Searching Algorithm

In section 6.9, we presented a simple iterative algorithm for computing the least fixed point of a monotonic endofunction on a finite, partially ordered set with a given least element. This small theory is immediately applicable to graph-searching.

Given a set of nodes s and a graph G the nodes reachable from a node in s are given by $(s\circ G^*)\!>$. Since this is a least fixed point of the function $\langle x :: s\cup(x\circ G)\!>\rangle$, the reachable nodes can be computed as follows:

```
    $seen := \perp\!\!\perp$
;   while $seen \neq s\cup(seen\circ G)\!>$ do
        $seen := s\cup(seen\circ G)\!>$
```

The name "$seen$" conveys an operational interpretation of the algorithm: initially no nodes have been "seen"; subsequently nodes that are reachable by a single edge from nodes that have already been "seen" are also "seen" .

Remark As always, the use of common English words to name variables can be misleading. Elsewhere (for example, [AHU82, pp.222–226]) the word "visited" is used instead of our "seen". We have chosen to use "seen" primarily because it is shorter. However, another reason is that "visited" suggests an action that has been completed. In the second phase of the strongly-connected-components algorithm, it is important to distinguish between when a search from a given node has "started" and when it has "finished". Our use of the word "seen" rather than "visited" is intended to suggest that the search has started but may not have finished. Nevertheless, it may be interpreted differently by different readers. Formal statements clarify the precise functions of the variables. **End of Remark**

The invariant property is that $seen\subseteq(s\circ G^*)\!>$ and the loop is guaranteed to terminate whenever G is a finite graph. On termination, $seen=(s\circ G^*)\!>$. That is, $seen$ is (a coreflexive representing) the set of nodes reachable from s .

This simple algorithm has the drawback that it is not very efficient: it involves the computation of $(x\circ G)\!>$ for an increasingly large set of nodes x and much of this computation just repeats what has already been computed. In order to eliminate this recomputation, we need a property that separates the new from the old. Such a property is the following:

Lemma 10.1 Let p be a coreflexive and R a homogeneous relation. Then

$$(p\circ R^*)_> \;=\; p \;\cup\; (p\circ R\circ{\sim}p\circ R^*)_>$$

(We use variables p and R to emphasise that no assumption of finiteness is made.)

Proof

$$\begin{array}{ll}
 & (p\circ R^*)_> \;=\; p \cup (p\circ R\circ{\sim}p\circ R^*)_> \\
= & \{ \quad \text{definition of } R^*\text{, distributivity, } p_> = p \quad \} \\
 & p \;\cup\; (p\circ R\circ R^*)_> \;=\; p \;\cup\; (p\circ R\circ{\sim}p\circ R^*)_> \\
= & \{ \quad {\sim}p \subseteq I\text{, monotonicity and reversing first step} \quad \} \\
 & (p\circ R^*)_> \;\subseteq\; p \cup (p\circ R\circ{\sim}p\circ R^*)_> \\
\Leftarrow & \{ \quad \text{fixed-point induction} \quad \} \\
 & p \cup ((p \;\cup\; p\circ R\circ{\sim}p\circ R^*)\circ R)_> \;\subseteq\; p \cup (p\circ R\circ{\sim}p\circ R^*)_> \\
\Leftarrow & \{ \quad \text{distributivity and monotonicity} \quad \} \\
 & \quad (p\circ R)_> \;\subseteq\; p \cup (p\circ R\circ{\sim}p\circ R^*)_> \\
 & \wedge \quad (p\circ R\circ{\sim}p\circ R^*\circ R)_> \;\subseteq\; (p\circ R\circ{\sim}p\circ R^*)_> \\
= & \{ \quad R^*\circ R \subseteq R^* \text{ and monotonicity} \quad \} \\
 & (p\circ R)_> \;\subseteq\; p \cup (p\circ R\circ{\sim}p\circ R^*)_> \\
\Leftarrow & \{ \quad \text{case analysis: } p\cup{\sim}p = I \quad \} \\
 & \quad (p\circ R)_>\circ p \;\subseteq\; p \\
 & \wedge \quad (p\circ R)_>\circ{\sim}p \;\subseteq\; (p\circ R\circ{\sim}p\circ R^*)_> \\
= & \{ \quad (p\circ R)_> \subseteq I \text{ and monotonicity,} \\
 & \quad\quad I \subseteq R^* \text{ and domains (theorems 5.9 and 5.8)} \quad \} \\
 & \mathsf{true} \;\;.
\end{array}$$

□

Lemma 10.1 suggests an alternative iterative algorithm for computing reachable nodes. Applying it to finite graph G and set of nodes $seen$, we have:

$$(seen\circ G^*)_> \;=\; seen \;\cup\; (seen\circ G\circ{\sim}seen\circ G^*)_> \;\;.$$

The subexpression $seen\circ G\circ{\sim}seen$ represents a set of “unexplored” edges of G in the sense that they are edges from a node that has been “seen” to a node that has not been “seen”. The nodes reachable from s can thus be computed by initialising $seen$ to s and then subsequently choosing an edge (a,b) in the set of “unexplored” edges; the node b has not previously been “seen” and so can be added to $seen$.

```
{   G is a finite graph and s is a coreflexive representing a subset of the nodes   }
$seen := s$
; { **Invariant:** $(s\circ G^*)_> \;=\; (seen\circ G^*)_>$ }
**while** $seen\circ G\circ{\sim}seen \neq \perp\!\!\perp$ **do**
   **begin**
      choose nodes $a$ and $b$ such that $a\circ\top\!\!\top\circ b \;\subseteq\; seen\circ G\circ{\sim}seen$
      { $b\subseteq{\sim}seen$ }
   ; $seen := seen\cup b$
   **end**
{ $(s\circ G^*)_> \;=\; seen$ }
```

Obviously, the invariant is truthified by the initialisation. Termination is guaranteed by the fact that $b\subseteq{\sim}seen$ is a precondition of the assignment $seen := seen\cup b$ in the loop body. (Thus the assignment increases the number of nodes in $seen$ by 1 and so the number of times the loop body is executed is at most the number of nodes in the graph.) We prove this fact as follows:

$$\begin{array}{ll}
& \mathsf{true} \\
= & \{ \quad \text{choice of } a \text{ and } b \quad \} \\
& a\circ\top\!\!\top\circ b \;\subseteq\; seen\circ G\circ{\sim}seen \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
& (a\circ\top\!\!\top\circ b)_> \;\subseteq\; (seen\circ G\circ{\sim}seen)_> \\
\Rightarrow & \{ \quad a \text{ and } b \text{ are atoms, so } (a\circ\top\!\!\top\circ b)_> = b \\
& \qquad (seen\circ G\circ{\sim}seen)_> \;\subseteq\; {\sim}seen \quad \} \\
& b \;\subseteq\; {\sim}seen \;\; .
\end{array}$$

Execution of the loop body demands that there exist nodes a and b satisfying the condition

$$a\circ\top\!\!\top\circ b \;\subseteq\; seen\circ G\circ{\sim}seen \;\; .$$

This is an immediate consequence of the condition for executing the loop body. (Formally, we exploit the fact that the lattice of relations is saturated.) It remains to show that the invariant is maintained by the body of the loop, and the claimed postcondition is implied by the conjunction of the invariant and the condition for terminating the loop. We verify the invariant in the following lemma.

Lemma 10.2 For all s, G, a and b,

$$(((s\circ G^*)_> = (seen\circ G^*)_>)[seen := seen\cup b] \;\equiv\; (s\circ G^*)_> = (seen\circ G^*)_>)$$
$$\Leftarrow \quad a\circ\top\!\!\top\circ b \subseteq seen\circ G\circ{\sim}seen \ .$$

Proof We assume that nodes a and b satisfy $a\circ\top\!\!\top\circ b \subseteq seen\circ G\circ{\sim}seen$.

$$\begin{array}{ll}
 & ((seen\circ G^*)_>)[seen := seen\cup b] \\
= & \{ \quad \text{definition of substitution and distributivity} \quad \} \\
 & (seen\circ G^*)_> \cup (b\circ G^*)_> \\
= & \{ \quad \begin{array}{ll} & (seen\circ G^*)_> \\ \supseteq & \{ \quad \text{lemma 10.1 with } p,R{:=}seen,G \quad \} \\ & (seen\circ G\circ{\sim}seen\circ G^*)_> \\ \supseteq & \{ \quad \text{choice of } a \text{ and } b\text{:} \quad a\circ\top\!\!\top\circ b \subseteq seen\circ G\circ{\sim}seen \quad \} \\ & (a\circ\top\!\!\top\circ b\circ G^*)_> \\ = & \{ \quad \text{domains (specifically } (R\circ S)_> = (R_>\circ S)_>\text{)} \\ & \quad\quad (a\circ\top\!\!\top\circ b)_> = b \quad \} \\ & (b\circ G^*)_> \quad \} \end{array} \\
 & (seen\circ G^*)_> \ .
\end{array}$$

The lemma follows by the definition of substitution.
□

That the postcondition $(s\circ G^*)_> = seen$ is valid is an immediate consequence of lemma 10.1:

$$\begin{array}{ll}
 & (s\circ G^*)_> \;=\; (seen\circ G^*)_> \;\wedge\; seen\circ G\circ{\sim}seen = \bot\!\!\bot \\
= & \{ \quad \text{lemma 10.1 with } p,R{:=}seen,G \quad \} \\
 & (s\circ G^*)_> \;=\; seen \,\cup\, (seen\circ G\circ{\sim}seen\circ G^*)_> \;\wedge\; seen\circ G\circ{\sim}seen = \bot\!\!\bot \\
\Rightarrow & \{ \quad \text{Leibniz, } \bot\!\!\bot \text{ is zero of composition, } \bot\!\!\bot_> = \bot\!\!\bot \quad \} \\
 & (s\circ G^*)_> = seen \ .
\end{array}$$

A concrete implementation of the above graph-searching algorithm involves choosing a suitable data structure in which to store the unexplored edges represented by $seen\circ G\circ{\sim}seen$. Breadth-first search stores the edges in a queue (so newly added edges are chosen in the order that they are added), whilst depth-first search stores the edges in a stack (so the most recently added edge is chosen first). Other variations enable the solution of more specific path-finding problems. For example, if edges are labelled by distances, shortest paths from a given source can be found by storing edges in a priority queue. Topological search (section 8.4) is also an instance: edges from each node are grouped together and an edge from a given node is chosen when the node has no unexplored incoming edges. We do not go into details any further.

10.2 Repeated Search and Delegates

In this section, we explore a property of repeated application of graph-searching starting with an empty set of "seen" nodes until all nodes have been seen.

The algorithm we consider is introduced in section 10.2.2 and further refined in section 10.2.3. Roughly speaking, the algorithm repeatedly searches a given graph starting from a node chosen from among the nodes not yet seen so as to maximise a "choice function"; at each iteration, the graph searched is the given graph but restricted to edges connecting nodes not yet seen. The algorithm records the chosen nodes in a function that we call a "delegate function", the "delegate" of a node a being the node from which the search that "sees" a is initiated. The formal specification of the delegate function is given in section 10.2.1.

Our formulation of the notion of a "delegate" is inspired by Cormen, Leiserson and Rivest's [CLR90, p.490] discussion of a "forefather" function as used in depth-first search to compute strongly connected components of a graph. However, our presentation is much more general than theirs. In particular, Cormen, Leiserson and Rivest assume that the choice function is injective. We establish some consequences of this assumption in section 10.2.4; this is followed in section 10.2.5 by a comparative discussion of our account and that of Cormen, Leiserson and Rivest.

10.2.1 Delegate Function

Suppose f is a total function of type $\mathbb{N}\leftarrow\mathsf{Node}$ and suppose G is a graph. We call f the *choice function*.

A *delegate function on* G *according to* f is a relation φ of type $\mathsf{Node}\sim\mathsf{Node}$ with the properties that

(10.3) $\varphi\circ\varphi^{\cup}\subseteq I_{\mathsf{Node}}\subseteq\varphi^{\cup}\circ\varphi$, and

(10.4) $\varphi\subseteq(G^{*})^{\cup}\;\;\wedge\;\;G^{*}\subseteq(f\circ\varphi)^{\cup}\circ\geq\circ f$.

The property (10.3) states that φ is a total function. Property (10.4), expressed pointwise and in words, states that for all nodes a and b, node a is the delegate of node b equivales the conjunction of (i) there is a path in G from b to a and (ii) among all nodes c such that there is a path from b to c, node a maximises the value of the choice function f.

Delegate functions have a couple of additional properties that we exploit later. These are formulated and proved in the lemma below.

Lemma 10.5 If φ is a delegate function on G according to f,

$$I \subseteq G^* \circ \varphi \ \wedge \ G^* \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f \circ \varphi \quad .$$

In words, there is a path in G from each node to its delegate, and if there is a path in G from node b to node c, the value of f at the delegate of b is at least the value of f at the delegate of c.

Proof First,

$$\begin{array}{ll}
 & I \subseteq G^* \circ \varphi \\
\Leftarrow & \{ \quad \varphi \text{ is total, i.e. } I \subseteq \varphi^{\cup} \circ \varphi \quad \} \\
 & \varphi^{\cup} \subseteq G^* \\
= & \{ \quad \text{converse} \quad \} \\
 & \varphi \subseteq (G^*)^{\cup} \\
= & \{ \quad (G^*)^{\cup} = (G^{\cup})^* \text{ and definition of delegate: (10.4)} \quad \} \\
 & \text{true} \quad .
\end{array}$$

Second,

$$\begin{array}{ll}
 & G^* \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f \circ \varphi \\
\Leftarrow & \{ \quad I \subseteq G^* \circ \varphi \ \text{(see above)} \quad \} \\
 & G^* \circ G^* \circ \varphi \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f \circ \varphi \\
\Leftarrow & \{ \quad G^* \circ G^* = G^* \text{, monotonicity} \quad \} \\
 & G^* \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f \\
= & \{ \quad \text{definition of delegate: (10.4)} \quad \} \\
 & \text{true} \quad .
\end{array}$$

□

Lemma 10.6 If φ is a delegate function on G according to f,

$$\varphi \subseteq f^{\cup} \circ \geq \circ f \quad .$$

In words, the delegate of a node has f-value that is at least that of the node.

Proof

$$
\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{definition: (10.3) and (10.4)} \quad \} \\
 & \varphi \circ \varphi^{\cup} \subseteq I \ \wedge\ G^* \subseteq (f{\circ}\varphi)^{\cup} \circ \geq \circ f \\
\Rightarrow & \{ \quad I \subseteq G^* \text{ and transitivity; converse} \quad \} \\
 & \varphi \circ \varphi^{\cup} \subseteq I \ \wedge\ I \subseteq \varphi^{\cup} \circ f^{\cup} \circ \geq \circ f \\
\Rightarrow & \{ \quad \varphi{\circ}I = \varphi \text{, monotonicity of composition and transitivity} \quad \} \\
 & \varphi \subseteq f^{\cup} \circ \geq \circ f \ .
\end{array}
$$

□

10.2.2 Assigning Delegates

The basic structure of the algorithm for computing a delegate function is shown in fig. 10.1. It is a simple loop that initialises the coreflexive $seen$ (representing a set of nodes) to $\perp\!\!\perp$ and then repeatedly chooses a node a that has the largest f-value among the nodes that do not have a delegate and adds to $seen$ the coreflexive $\sim\!seen \circ (G^* \circ a)_<$; this coreflexive represents the nodes that do not have a delegate and from which there is a path to a in the graph. Simultaneous with the assignments to $seen$, the variable φ is initialised to $\perp\!\!\perp$ and subsequently updated by assigning the value of φ to a at all newly "delegated" nodes.

For brevity in the calculations below, the temporary variable s (short for "seen") has been introduced. The sequence of assignments

$$
\begin{array}{ll}
 & s \ := \ \sim\!seen \circ (G^* \circ a)_< \\
; & \varphi, seen \ := \ \varphi \cup a{\circ}\top\!\!\top{\circ}s \ , \ seen \cup s
\end{array}
$$

is implemented by an adaptation of the graph-searching algorithm discussed in section 10.1. The details of how this is done are given in section 10.2.3.

Apart from being a total function, we impose no restrictions on f. If f is a constant function (for example, if $f.a = 0$ for all nodes a), the "choice" is completely arbitrary.

The property

$$\varphi \subseteq (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^*$$

in the postcondition is stronger than the requirement $\varphi \subseteq (G^*)^{\cup}$ in (10.4). It states that there is a path from each node to its delegate comprising nodes that all have the same delegate. (More precisely, it states that there is a path from each node to its delegate

```
{ $f \circ f^{\cup} \subseteq I_{\mathbb{N}} \wedge I_{\mathsf{Node}} \subseteq f^{\cup} \circ f$ }
$\varphi, seen := \perp\!\!\perp, \perp\!\!\perp$ ;
{ Invariant: (10.7) thru (10.14) }
while $seen \neq I_{\mathsf{Node}}$ do
   begin
     choose node $a$ such that $a \circ seen = \perp\!\!\perp$ and $\sim seen \circ \top\!\!\top \circ a \subseteq f^{\cup} \circ \leq \circ f$
   ; $s := \sim seen \circ (G^{*} \circ a)_{<}$
   ; $\varphi, seen := \varphi \cup a \circ \top\!\!\top \circ s \, , \, seen \cup s$
   end
{      $\varphi \circ \varphi^{\cup} \subseteq I_{\mathsf{Node}} \subseteq \mathsf{equiv}.G \subseteq \varphi^{\cup} \circ \varphi$
   $\wedge$  $\varphi \subseteq (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^{*}$ $\wedge$ $G^{*} \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f$
   $\wedge$  $\varphi = \varphi \circ \varphi$ }
```

Figure 10.1: Repeated Search. Outer Loop

such that successive nodes on the path have the same delegate. The equivalence of these two informal interpretations is formulated in lemma 10.36.)

Note the property $\varphi = \varphi \circ \varphi$ in the postcondition. Cormen, Leiserson and Rivest [CLR90, p.490] require that the function f is injective and use this to derive the property from the definition of a delegate ("forefather" in their terminology). We don't require that f is injective but show instead that it is a consequence of the algorithm used to calculate delegates. For completeness, we also show that it is a consequence of the definition of delegate under the assumption that f is injective: see lemma 10.30. Similarly, the property $\mathsf{equiv}.G \subseteq \varphi^{\cup} \circ \varphi$ can be derived from the definition of a delegate if f is assumed to be injective. Again for completeness, we also show that it is a consequence of the definition of delegate under the assumption that f is injective: see lemma 10.31.

Termination of the loop is obvious: the coreflexive $seen$ represents a set of nodes that increases strictly in size at each iteration. (The chosen node a is added at each iteration.) The number of iterations of the loop body is thus at most the number of nodes in the graph, which is assumed to be finite. The principle task is thus to verify conditional correctness (correctness assuming termination, often called "partial" correctness).

The invariant properties of the algorithm are as follows:

(10.7) $\varphi_{>} = seen$,

(10.8) $\varphi \circ \varphi^{\cup} \subseteq seen$,

(10.9) $\varphi \subseteq (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^*$,

(10.10) $\varphi = \varphi \circ \varphi$,

(10.11) $seen = (G^* \circ seen)_<$,

(10.12) $seen \circ \top\!\!\top \circ \sim seen \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f$,

(10.13) $seen \circ G^* \circ seen \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f$,

(10.14) $seen \circ \mathsf{equiv}.G \circ seen \subseteq \varphi^{\cup} \circ \varphi$.

Before verifying the invariant properties, let us consider the postcondition. The post-condition

$$\varphi \circ \varphi^{\cup} \subseteq I_{\mathsf{Node}} \subseteq \varphi^{\cup} \circ \varphi$$

expresses the fact that, on termination, φ is total and functional; the claimed invariants (10.7) and (10.8) state that intermediate values of φ are total on $seen$ and functional. The invariants (10.9) and (10.10) are both conjuncts of the postcondition. The additional conjunct

$$\mathsf{equiv}.G \subseteq \varphi^{\cup} \circ \varphi$$

states that strongly connected nodes have the same delegate. The invariant (10.14) states that this is the case for nodes that have been assigned a delegate. Like (10.7) and (10.8), invariant (10.13) states that intermediate values of φ maximise f for those nodes for which a delegate has been assigned. It is therefore obvious that the postcondition is implied by the conjunction of the invariant and the termination condition. The additional invariants (10.11) and (10.12) are needed in order to establish the invariance of (10.13).

Since, for all coreflexives p and q,

$$p \cup q = p \cup \sim p \circ q \quad ,$$

it is the case that

(10.15) $seen \cup \sim seen \circ (G^* \circ a)_< = seen \cup (G^* \circ a)_<$.

Note that the left side of this equality is the right side of the assignment to $seen$ in the above algorithm. The right side is slightly simpler. Accordingly, we use the right side when reasoning about the invariant properties of $seen$. (The right side of the assignment to φ cannot be simplified in this way.)

As usually happens, it is obvious that all of the claimed invariant properties are true on initialisation (since $\bot\!\!\bot$ is the zero of composition). It remains to establish the verification condition: for all φ, $seen$ and a,

$$
\begin{array}{ll}
& ((10.7)\text{ thru }(10.14))\,[\varphi,seen \;:=\; \varphi\cup a\circ\top\!\top\circ s\;,\;seen\cup s] \\
\Leftarrow & \quad ((10.7)\text{ thru }(10.14)) \\
& \wedge\quad a\circ seen = \bot\!\bot \;\wedge\; {\sim}seen\circ\top\!\top\circ a \subseteq f^{\cup}\circ\leq\circ f
\end{array}
$$

where $s = {\sim}seen\circ(G^*\circ a)_<$. We consider each of the conjuncts in the consequent in turn, invoking the premises when necessary. (The first line of premises is the conjunction of all the claimed invariant properties and the second line is the criterion used to choose a .) Because of the large number of properties, the remainder of this section is quite long. Most of the calculations are, however, straightforward. An exception is, perhaps, the invariance of (10.9), proved in lemma 10.22.

We begin with a few lemmas on the consequences of the invariant properties.

Lemma 10.16 Assuming properties (10.7), (10.8) and $a\circ seen = \bot\!\bot$, the following properties also hold:

$$\varphi\circ s = \bot\!\bot = s\circ\varphi \;\wedge\; \varphi\circ a = \bot\!\bot \quad .$$

Proof These are all straightforward. First,

$$
\begin{array}{ll}
& \varphi\circ s \\
= & \{\quad \text{domains} \quad\} \\
& \varphi\circ\varphi_>\circ s \\
= & \{\quad (10.7) \quad\} \\
& \varphi\circ seen\circ s \\
= & \{\quad s = {\sim}seen\circ(G^*\circ a)_< \quad\} \\
& \varphi\circ seen\circ{\sim}seen\circ(G^*\circ a)_< \\
= & \{\quad \text{complements: } seen\circ{\sim}seen = \bot\!\bot \quad\} \\
& \bot\!\bot \quad .
\end{array}
$$

Second,

$$
\begin{array}{ll}
& s\circ\varphi \\
= & \{\quad \text{domains} \quad\} \\
& s\circ\varphi_<\circ\varphi \\
\subseteq & \{\quad (10.8)\text{: } \varphi_< \subseteq seen \quad\} \\
& s\circ seen\circ\varphi
\end{array}
$$

$$= \quad \{ \quad s{\circ}seen \subseteq {\sim}seen \circ seen \subseteq \perp\!\!\perp \quad \}$$

$$\perp\!\!\perp \ .$$

The third property is an immediate consequence of $\varphi{\circ}s = \perp\!\!\perp$ (the first property) and $a \subseteq s$ (which is a consequence of the choice of a).
□

Lemma 10.17 Assuming $seen = (G^* \circ seen)_<$ (i.e. (10.11)) and $a{\circ}seen = \perp\!\!\perp$, the following properties also hold:

$${\sim}seen \circ G^* \circ seen = \perp\!\!\perp \ \wedge \ {\sim}seen \circ G^* \circ a = ({\sim}seen \circ G)^* \circ a \ .$$

Proof First,

$${\sim}seen \circ G^* \circ seen$$

$$= \quad \{ \quad \text{domains: } [\ R = R_< \circ R\ ] \text{ with } R := G^* \circ seen\ ;$$
$$seen = (G^* \circ seen)_< \quad \}$$

$${\sim}seen \circ seen \circ G^* \circ seen$$

$$= \quad \{ \quad {\sim}seen \circ seen = \perp\!\!\perp \quad \}$$

$$\perp\!\!\perp \ .$$

Second,

$${\sim}seen \circ G^* \circ a$$

$$= \quad \{ \quad I = seen \cup {\sim}seen\text{; distributivity and star decomposition:}$$
$$[\ (R{\cup}S)^* = R^* \circ (S \circ R^*)^*\ ] \quad \text{with } R,S := seen \circ G\ , {\sim}seen \circ G \quad \}$$

$${\sim}seen \circ (seen \circ G)^* \circ ({\sim}seen \circ G \circ (seen \circ G)^*)^* \circ a$$

$$= \quad \{ \quad (seen{\circ}G)^* = I \cup seen{\circ}G{\circ}(seen{\circ}G)^*$$
$$\text{distributivity and } {\sim}seen \circ seen = \perp\!\!\perp \quad \}$$

$${\sim}seen \circ ({\sim}seen \circ G \circ (seen \circ G)^*)^* \circ a$$

$$= \quad \{ \quad (seen{\circ}G)^* = I \cup seen{\circ}G{\circ}(seen{\circ}G)^*$$
$$\text{distributivity and } {\sim}seen \circ G^* \circ seen = \perp\!\!\perp$$
$$\text{(whence } {\sim}seen \circ G \circ seen = \perp\!\!\perp) \quad \}$$

$${\sim}seen \circ ({\sim}seen \circ G)^* \circ a$$

$$= \quad \{ \quad ({\sim}seen \circ G)^* = I \cup {\sim}seen \circ G \circ ({\sim}seen \circ G)^*$$

distributivity }

$\sim\!seen \circ a \cup \sim\!seen \circ \sim\!seen \circ G \circ (\sim\!seen \circ G)^* \circ a$

$=$ { $\sim\!seen \circ a = a$ and $\sim\!seen \circ \sim\!seen = \sim\!seen$,

$(\sim\!seen \circ G)^* = I \cup \sim\!seen \circ G \circ (\sim\!seen \circ G)^*$

distributivity }

$(\sim\!seen \circ G)^* \circ a$.

□

Lemma 10.18 Assuming properties (10.7) thru (10.14) and $a \circ seen = \perp\!\!\perp$,

$$s = ((\sim\!seen \circ G)^* \circ a)_<\ .$$

Proof

s

$=$ { definition }

$\sim\!seen \circ (G^* \circ a)_<$

$=$ { domains: for all coreflexives p and all relations R,

$p \circ R_< = (p \circ R)_<$ with $p,R := \sim\!seen\ ,\ G^* \circ a$ }

$(\sim\!seen \circ G^* \circ a)_<$

$=$ { lemma 10.17 }

$((\sim\!seen \circ G)^* \circ a)_<$.

□

Lemma 10.19 Assuming properties (10.7) thru (10.14) and $a \circ seen = \perp\!\!\perp$,

$$s = ((s \circ G)^* \circ a)_<\ .$$

Proof Applying lemma 10.18, the task is to prove that

$$((\sim\!seen \circ G)^* \circ a)_< = ((s \circ G)^* \circ a)_<\ .$$

Clearly, since $s \subseteq \sim\!seen$, the left side of this equation is at least the right side. So it suffices to prove the inclusion. This we do as follows.

$((\sim\!seen \circ G)^* \circ a)_< \subseteq ((s \circ G)^* \circ a)_<$

$\Leftarrow$ { fixed-point fusion }

$\quad\quad a \subseteq ((s{\circ}G)^* \circ a)_<$

$\quad \wedge \quad (\sim\!seen \circ G \circ ((s{\circ}G)^* \circ a)_<)_< \ \subseteq \ ((s{\circ}G)^* \circ a)_<$

$=$ $\quad\{$ first conjunct is clearly true ;

$\quad\quad\quad \sim\!seen$

$\quad\quad =$ $\quad\{$ case analysis: $I \ = \ (G^* \circ a)_< \cup (G^* \circ a)_{\bullet<}$ $\}$

$\quad\quad\quad \sim\!seen \circ (G^* \circ a)_< \cup \sim\!seen \circ (G^* \circ a)_{\bullet<}$

$\quad\quad =$ $\quad\{$ definition of s $\}$

$\quad\quad\quad s \cup \sim\!seen \circ (G^* \circ a)_{\bullet<}$ $\quad\}$

$\quad ((s \cup \sim\!seen \circ (G^* \circ a)_{\bullet<}) \circ G \circ ((s{\circ}G)^* \circ a)_<)_< \ \subseteq \ ((s{\circ}G)^* \circ a)_<$

$=$ $\quad\{$ domains: $[\,(R \circ S_<)_< = (R{\circ}S)_<\,]$

$\quad\quad$ with $R,S := (s \cup \sim\!seen \circ (G^* \circ a)_{\bullet<}) \circ G \ , \ (s{\circ}G)^* \circ a$;

$\quad\quad$ distributivity $\}$

$\quad\quad (s \circ G \circ (s{\circ}G)^* \circ a)_< \ \subseteq \ ((s{\circ}G)^* \circ a)_<$

$\quad \wedge \quad (\sim\!seen \circ (G^* \circ a)_{\bullet<} \circ G \circ (s{\circ}G)^* \circ a)_< \ \subseteq \ ((s{\circ}G)^* \circ a)_<$

$\Leftarrow$ $\quad\{$ first conjunct is true (since $[\ R{\circ}R^* \subseteq R^*\ ]$ with $R{:=}\,s{\circ}G$);

$\quad\quad$ second conjunct: $G \circ (s{\circ}G)^* \subseteq G^*$ and domains $\}$

$\quad (\sim\!seen \circ (G^* \circ a)_{\bullet<} \circ (G^* \circ a)_<)_< \ \subseteq \ ((s{\circ}G)^* \circ a)_<$

$=$ $\quad\{$ complements: $(G^* \circ a)_{\bullet<} \circ (G^* \circ a)_< = \perp\!\!\perp$ $\}$

$\quad$ true .

□

We can now proceed to the verification of each of the invariant properties.

Lemma 10.20 Property (10.7) is an invariant of the algorithm.

Proof It is clearly true after initialisation of φ and *seen*. For the loop body, assume (10.7) is true. Let s denote $\sim\!seen \circ (G^* \circ a)_<$. Then we have:

$\quad (\varphi \cup a{\circ}\top\!\!\top{\circ}s)_>$

$=$ $\quad\{$ distributivity $\}$

$\quad \varphi_> \cup (a{\circ}\top\!\!\top{\circ}s)_>$

$=$ $\quad\{$ assumption: (10.7) $\}$

$\quad seen \cup (a{\circ}\top\!\!\top{\circ}s)_>$

$= \quad \{ \quad s = {\sim}seen \circ (G^* \circ a)_< \text{, domains} \quad \}$

$\quad seen \cup (a \circ \top\!\!\top \circ a \circ (G^*)^\cup \circ {\sim}seen)_>$

$= \quad \{ \quad a \text{ is an atom, so } a \circ \top\!\!\top \circ a = a \quad \}$

$\quad seen \cup (a \circ (G^*)^\cup \circ {\sim}seen)_>$

$= \quad \{ \quad \text{domains, } s = {\sim}seen \circ (G^* \circ a)_< \quad \}$

$\quad seen \cup s \quad .$

That is, (10.7) is an invariant of the algorithm.
□

Lemma 10.21 Property (10.8) is an invariant of the algorithm.

Proof It is clearly true after initialisation of φ and $seen$. For the loop body, assume (10.8) is true. Let $s = {\sim}seen \circ (G^* \circ a)_<$. Then

$\quad (\varphi \cup a \circ \top\!\!\top \circ s) \circ (\varphi \cup a \circ \top\!\!\top \circ s)^\cup \subseteq seen \cup s$

$= \quad \{ \quad \text{distributivity} \quad \}$

$\quad \varphi \circ \varphi^\cup \cup a \circ \top\!\!\top \circ s \circ \varphi^\cup \cup \varphi \circ s \circ \top\!\!\top \circ a \cup a \circ \top\!\!\top \circ s \circ s \circ \top\!\!\top \circ a \subseteq seen \cup s$

$\Leftarrow \quad \{ \quad \text{definition of set union} \quad \}$

$\qquad \varphi \circ \varphi^\cup \subseteq seen$

$\quad \wedge \quad a \circ \top\!\!\top \circ s \circ \varphi^\cup \subseteq \perp\!\!\perp$

$\quad \wedge \quad \varphi \circ s \circ \top\!\!\top \circ a \subseteq \perp\!\!\perp$

$\quad \wedge \quad a \circ \top\!\!\top \circ s \circ s \circ \top\!\!\top \circ a \subseteq s \quad .$

We consider each of the conjuncts in turn. The first is (10.8) which we assume to be true. The second and third are clearly equivalent (because $a = a^\cup$, $s = s^\cup$ and $\perp\!\!\perp = \perp\!\!\perp^\cup$) and the third is obviously an immediate consequence of lemma 10.16 (in particular, $\varphi \circ s = \perp\!\!\perp$). Finally,

$\quad a \circ \top\!\!\top \circ s \circ s \circ \top\!\!\top \circ a$

$\subseteq \quad \{ \quad \top\!\!\top \circ s \circ s \circ \top\!\!\top \subseteq \top\!\!\top \quad \}$

$\quad a \circ \top\!\!\top \circ a$

$= \quad \{ \quad a \text{ is a node (an atomic coreflexive)} \quad \}$

$\quad a$

$\subseteq \quad \{ \quad I \subseteq G^* \text{, monotonicity;}$

$$
\begin{array}{ll}
 & \quad\text{choice of } a:\ a\circ seen = \perp\!\!\perp \quad \} \\
 & \sim seen \circ (G^* \circ a)_< \\
= & \quad \{ \quad \text{definition} \quad \} \\
 & s\ .
\end{array}
$$

We have thus verified that (10.8) is an invariant of the algorithm.
□

Lemma 10.22 Property (10.9) is an invariant of the algorithm.

Proof We have to prove that

$$\varphi \cup a\circ\top\!\!\top\circ s \ \subseteq\ (G^{\cup} \cap (\varphi \cup a\circ\top\!\!\top\circ s)^{\cup} \circ (\varphi \cup a\circ\top\!\!\top\circ s))^*$$

assuming (10.7) thru (10.14).

Clearly (by monotonicity)

$$\varphi \ \subseteq\ (G^{\cup} \cap (\varphi \cup a\circ\top\!\!\top\circ s)^{\cup} \circ (\varphi \cup a\circ\top\!\!\top\circ s))^* \ \Leftarrow\ (10.9)$$

so it suffices to prove that

$$a\circ\top\!\!\top\circ s \ \subseteq\ (G^{\cup} \cap (\varphi \cup a\circ\top\!\!\top\circ s)^{\cup} \circ (\varphi \cup a\circ\top\!\!\top\circ s))^* \quad .$$

As usual, we begin with the more complicated side.

$$
\begin{array}{ll}
 & (G^{\cup} \cap (\varphi \cup a\circ\top\!\!\top\circ s)^{\cup} \circ (\varphi \cup a\circ\top\!\!\top\circ s))^* \\
\supseteq & \quad \{ \quad \text{monotonicity} \quad \} \\
 & (G^{\cup} \cap (a\circ\top\!\!\top\circ s)^{\cup} \circ a \circ \top\!\!\top \circ s)^* \\
= & \quad \{ \quad \text{converse; } a \text{ is a node, so } \top\!\!\top \circ a^{\cup} \circ a \circ \top\!\!\top \ =\ \top\!\!\top \quad \} \\
 & (G^{\cup} \cap s\circ\top\!\!\top\circ s)^* \\
= & \quad \{ \quad s \text{ is a coreflexive, so } s = s_< = s_> \text{ ; domains} \quad \} \\
 & (s \circ G^{\cup} \circ s)^* \\
\supseteq & \quad \{ \quad I \supseteq s \quad \} \\
 & (s \circ G^{\cup} \circ s)^* \circ s \\
= & \quad \{ \quad \text{mirror rule, } s\circ s = s \quad \} \\
 & s \circ (G^{\cup} \circ s)^* \\
\supseteq & \quad \{ \quad s \supseteq a\text{, monotonicity} \quad \}
\end{array}
$$

$a \circ (G^{\cup} \circ s)^*$

$=$ { a is a node, so $a \circ \top\!\!\top \circ a = a$

distributivity properties of converse }

$a \circ \top\!\!\top \circ a \circ ((s \circ G)^*)^{\cup}$

$=$ { lemma 10.19 and domains }

$a \circ \top\!\!\top \circ s$.

□

Lemma 10.23 Property (10.10) is an invariant of the algorithm.

Proof It is clearly true after initialisation of φ . For the loop body, assume (10.10) is true. Then

$(\varphi \cup a \circ \top\!\!\top \circ s) \circ (\varphi \cup a \circ \top\!\!\top \circ s)$

$=$ { distributivity }

$\varphi \circ \varphi \cup \varphi \circ a \circ \top\!\!\top \circ s \cup a \circ \top\!\!\top \circ s \circ \varphi \cup a \circ \top\!\!\top \circ s \circ a \circ \top\!\!\top \circ s$

$=$ { lemma 10.16 }

$\varphi \circ \varphi \cup a \circ \top\!\!\top \circ s \circ a \circ \top\!\!\top \circ s$

$=$ { by lemma 10.18, $s \circ a = a$;

so, by cone rule (4.16): $\top\!\!\top \circ s \circ a \circ \top\!\!\top = \top\!\!\top$

hypothesis (10.10): $\varphi \circ \varphi = \varphi$ }

$\varphi \cup a \circ \top\!\!\top \circ s$.

The property (10.10) is thus maintained by the body of the loop.

□

Lemma 10.24 Property (10.11) is an invariant of the algorithm.

Proof It is clearly true after initialisation of $seen$. For the loop body, assume (10.11) is true. Then

$(G^* \circ (seen \cup (G^* \circ a)_<))_<$

$=$ { distributivity }

$(G^* \circ seen)_< \;\cup\; (G^* \circ (G^* \circ a)_<)_<$

$=$ { assumption: (10.11) and domains }

$$\begin{array}{ll}
& seen \ \cup \ (G^* \circ G^* \circ a)_< \\
= & \{ \quad G^* \circ G^* = G^* \quad \} \\
& seen \ \cup \ (G^* \circ a)_< \ .
\end{array}$$

Recalling property (10.15), it follows that (10.11) is an invariant of the algorithm.
□

Lemma 10.25 Property (10.12) is an invariant of the algorithm.

Proof It is clearly true after the initialisation of $seen$ and φ. For the loop body, assume (10.12) is true.

$$\begin{array}{ll}
& (seen \cup s) \circ \top\!\!\top \circ {\sim}(seen \cup s) \\
= & \{ \quad \text{distributivity} \quad \} \\
& seen \circ \top\!\!\top \circ {\sim}(seen \cup s) \ \cup \ s \circ \top\!\!\top \circ {\sim}(seen \cup s) \\
\subseteq & \{ \quad \text{assumption: (10.12) and domains} \quad \} \\
& (f \circ \varphi)^\cup \circ \geq \circ f \ \cup \ s \circ \top\!\!\top \circ {\sim}seen \ .
\end{array}$$

We continue with the second term:

$$\begin{array}{ll}
& s \circ \top\!\!\top \circ {\sim}seen \ \subseteq \ s \circ \top\!\!\top \circ a \circ f^\cup \circ \geq \circ f \\
\Leftarrow & \{ \quad \text{choice of } a\text{: } {\sim}seen \circ \top\!\!\top \circ a \subseteq f^\cup \circ \leq \circ f \\
& \qquad \text{i.e. } a \circ \top\!\!\top \circ {\sim}seen \subseteq f^\cup \circ \geq \circ f \quad \} \\
& s \circ \top\!\!\top \circ {\sim}seen \ \subseteq \ s \circ \top\!\!\top \circ a \circ a \circ \top\!\!\top \circ {\sim}seen \\
= & \{ \quad a \text{ is a node (a non-empty atom), cone rule: } \top\!\!\top \circ a \circ a \circ \top\!\!\top = \top\!\!\top \quad \} \\
& \mathsf{true} \ .
\end{array}$$

Combining the two calculations:

$$\begin{array}{ll}
& (seen \cup s) \circ \top\!\!\top \circ {\sim}(seen \cup s) \\
\subseteq & \{ \quad \text{monotonicity of set union} \quad \} \\
& (f \circ \varphi)^\cup \circ \geq \circ f \ \cup \ s \circ \top\!\!\top \circ a \circ f^\cup \circ \geq \circ f \\
= & \{ \quad \text{distributivity} \quad \} \\
& (f \circ (\varphi \cup a \circ \top\!\!\top \circ s))^\cup \circ \geq \circ f \ .
\end{array}$$

That is, (10.12) is invariant under the assignment.
□

Lemma 10.26 Property (10.13) is an invariant of the algorithm.

Proof It is clearly true after initialisation of $seen$. For the loop body, assume (10.13) is true. The left side of (10.13) $[seen := seen \cup s]$ expands into the union of four terms. We consider each in turn.

First,

$$
\begin{array}{ll}
 & seen \circ G^* \circ seen \\
\subseteq & \{ \quad \text{assumption: (10.13)} \quad \} \\
 & (f \circ \varphi)^{\cup} \circ \geq \circ f \quad .
\end{array}
$$

Second,

$$
\begin{array}{ll}
 & s \circ G^* \circ s \\
\subseteq & \{ \quad G^* \subseteq \top\!\top \text{, monotonicity} \quad \} \\
 & s \circ \top\!\top \circ s \\
= & \{ \quad \text{definition of } s \text{, domains} \quad \} \\
 & s \circ \top\!\top \circ a \circ (G^*)^{\cup} \circ {\sim}seen \\
\subseteq & \{ \quad (G^*)^{\cup} \subseteq \top\!\top \text{, monotonicity} \quad \} \\
 & s \circ \top\!\top \circ a \circ \top\!\top \circ {\sim}seen \\
\subseteq & \{ \quad \text{choice of } a\text{: } {\sim}seen \circ \top\!\top \circ a \subseteq f^{\cup} \circ \leq \circ f \text{; converse and } a \circ a = a \quad \} \\
 & s \circ \top\!\top \circ a \circ f^{\cup} \circ \geq \circ f \quad .
\end{array}
$$

Third,

$$
\begin{array}{ll}
 & s \circ G^* \circ seen \\
= & \{ \quad \text{domains} \quad \} \\
 & s \circ (G^* \circ seen)_< \circ G^* \circ seen \\
= & \{ \quad \text{invariant: (10.11)} \quad \} \\
 & s \circ seen \circ G^* \circ seen \\
= & \{ \quad \text{definition of } s \text{, coreflexives commute} \quad \} \\
 & (G^* \circ a)_< \circ {\sim}seen \circ seen \circ G^* \circ seen \\
= & \{ \quad {\sim}seen \circ seen = \perp\!\perp \quad \} \\
 & \perp\!\perp \quad .
\end{array}
$$

Finally,

$$
\begin{array}{ll}
 & seen \circ G^* \circ s \\
\subseteq & \{ \quad \text{by definition and propertie of coreflexives, } s \subseteq {\sim} seen \quad \} \\
 & seen \circ G^* \circ {\sim} seen \\
\subseteq & \{ \quad G^* \subseteq \top\!\!\top \text{ and lemma 10.25} \quad \} \\
 & (f \circ \varphi)^{\cup} \circ \geq \circ f \ .
\end{array}
$$

Putting the calculations together, we have:

$$
\begin{array}{ll}
 & (seen \cup s) \circ G^* \circ (seen \cup s) \\
\subseteq & \{ \quad \text{distributivity and above calculations} \quad \} \\
 & (f \circ \varphi)^{\cup} \circ \geq \circ f \ \cup \ s \circ \top\!\!\top \circ a \circ f^{\cup} \circ \geq \circ f \\
= & \{ \quad \text{distributivity} \quad \} \\
 & (f \circ (\varphi \cup a \circ \top\!\!\top \circ s))^{\cup} \circ \geq \circ f
\end{array}
$$

That is, (10.13) is invariant under the assignment.
□

Lemma 10.27 Property (10.14) is an invariant of the algorithm.

Proof It is clearly true after initialisation of $seen$ and φ. For the loop body, assume (10.14) is true. Then

$$
\begin{array}{ll}
 & (\varphi \cup a \circ \top\!\!\top \circ s)^{\cup} \circ (\varphi \cup a \circ \top\!\!\top \circ s) \\
= & \{ \quad \text{distributivity} \quad \} \\
 & \varphi^{\cup} \circ \varphi \ \cup \ \varphi^{\cup} \circ a \circ \top\!\!\top \circ s \ \cup \ s \circ \top\!\!\top \circ a \circ \varphi \ \cup \ s \circ \top\!\!\top \circ a \circ a \circ \top\!\!\top \circ s \\
= & \{ \quad \text{lemma 10.16} \quad \} \\
 & \varphi^{\cup} \circ \varphi \ \cup \ s \circ \top\!\!\top \circ a \circ a \circ \top\!\!\top \circ s \\
\supseteq & \{ \quad \text{cone rule: } \top\!\!\top \circ a \circ a \circ \top\!\!\top = \top\!\!\top \\
 & \quad\ \ \text{hypothesis (10.14)} \quad \} \\
 & seen \circ \text{equiv.G} \circ seen \ \cup \ s \circ \top\!\!\top \circ s \\
\supseteq & \{ \quad \text{by lemma 10.17, equiv.G} \subseteq G^* \text{, and } s \subseteq {\sim} seen \\
 & \quad\ \ seen \circ \text{equiv.G} \circ s = \bot\!\!\bot \ ; \\
 & \quad\ \ \text{so, using properties of converse, } s \circ \text{equiv.G} \circ seen = \bot\!\!\bot \quad \} \\
 & \quad seen \circ \text{equiv.G} \circ seen \ \cup \ s \circ \text{equiv.G} \circ s
\end{array}
$$

$$\begin{array}{ll} & \cup \quad s \circ \mathsf{equiv.G} \circ seen \cup seen \circ \mathsf{equiv.G} \circ s \\ = & \{ \quad \text{distributivity} \quad \} \\ & (seen \cup s) \circ \mathsf{equiv.G} \circ (seen \cup s) \quad . \end{array}$$

The property (10.14) is thus maintained by the body of the loop.
□

This completes the verification of the algorithm.

10.2.3 Incremental Computation

The algorithm shown in fig. 10.1 assigns to the variable s (the coreflexive representing) all the nodes that do not yet have a delegate and can reach the node a. The variable φ is also updated so that a becomes the delegate of all the nodes in the set represented by s. As mentioned then, the assignments are implemented by an adaptation of the graph-searching algorithm discussed in section 10.1. Fig. 10.2 shows the details.

The consecutive assignments in the body of the loop in fig. 10.1 (to s, and to φ and $seen$) are implemented by an inner loop together with initialising assignments. The assertions should enable the reader to verify that the two algorithms are equivalent: the variables s, $seen_0$ and φ_0 are auxiliary variables used to express the property that the inner loop correctly implements the two assignments that they replace in the outer loop; in an actual implementation the assignments to these variables may be omitted (or, preferably, included but identified as auxiliary statements that can be ignored by the computation proper).

It is straightforward to verify the correctness of this algorithm. For completeness, we give the details below.

The auxiliary variable $seen_0$ records the initial value of $seen$. That

$$\sim seen_0 \circ G \circ seen_0 = \perp\!\!\perp$$

is truthified is a straightforward consequence of (10.11):

Lemma 10.28

$$(\sim seen_0 \circ G \circ seen_0 = \perp\!\!\perp)[seen_0 := seen] \quad \Leftarrow \quad seen = (G^* \circ seen)_< \quad .$$

Proof

$$\begin{array}{ll} & (\sim seen_0 \circ G \circ seen_0)[seen_0 := seen] \\ = & \{ \quad \text{substitution} \quad \} \\ & \sim seen \circ G \circ seen \end{array}$$

```
    { $a{\circ}seen = \perp\!\!\perp \wedge$ (10.7) thru (10.14) }
    /* $s$ , $seen_0$ and $\varphi_0$ are auxiliary variables */
    $s, seen_0, \varphi_0 \;:=\; a, seen, \varphi$
    { $\sim seen_0 \circ G \circ seen_0 \;=\; \perp\!\!\perp$ }
; $seen, \varphi \;:=\; seen \cup a \,,\; \varphi \cup a{\circ}\top\!\!\top{\circ}a$
; { Invariant: $seen \;=\; s \cup seen_0 \;\wedge\; \varphi \;=\; \varphi_0 \cup a{\circ}\top\!\!\top{\circ}s$
      Invariant: $a \subseteq s \subseteq \sim seen_0 \circ (G^* \circ a)_<$ }
   while $\sim seen \circ G \circ seen \neq \perp\!\!\perp$ do
      begin
         choose node $b$ such that $b \subseteq \sim seen \circ (G{\circ}seen)_<$
         { $b \subseteq \sim seen_0 \circ (G^* \circ a)_<$ }
      ; $s \;:=\; s \cup b$
      ; $seen, \varphi \;:=\; seen \cup b \,,\; \varphi \cup a{\circ}\top\!\!\top{\circ}b$
      end
   { $s \;=\; \sim seen_0 \circ (G^* \circ a)_< \;\wedge\; seen \;=\; s \cup seen_0 \;\wedge\; \varphi \;=\; \varphi_0 \cup a{\circ}\top\!\!\top{\circ}s$ }
   { $seen \;=\; seen_0 \cup (G^* \circ a)_< \;\wedge\; \varphi \;=\; \varphi_0 \cup a \circ \top\!\!\top \circ \sim seen_0 \circ (G^* \circ a)_<$ }
```

Figure 10.2: Repeated Search. Inner Loop.

$=$ { domains }

$\sim seen \circ (G{\circ}seen)_< \circ G \circ seen$

$\subseteq$ { $G \subseteq G^*$, monotonicity }

$\sim seen \circ (G^* \circ seen)_< \circ G \circ seen$

$=$ { assume: $seen \;=\; (G^* \circ seen)_<$ (i.e. (10.11)) }

$\sim seen \circ seen \circ G \circ seen$

$=$ { $\sim seen \circ seen = \perp\!\!\perp$ }

$\perp\!\!\perp$

$=$ { substitution }

$\perp\!\!\perp[seen_0 := seen]$.

□

Now we must show that the invariants are truthified by the initialisation. That

$$seen \;=\; seen_0 \cup s \;\;\wedge\;\; \varphi \;=\; \varphi_0 \cup a{\circ}\top\!\!\top{\circ}s$$

is truthified is a straightforward application of the assignment axiom. That

$$a \subseteq s$$

is truthified is obvious. Finally, that

$$s \;\subseteq\; {\sim}seen_0 \circ (G^* \circ a)_<$$

is truthified is a straightforward consequence of the precondition $a{\circ}seen = \perp\!\!\perp$:

Lemma 10.29

$$(s \;\subseteq\; {\sim}seen_0 \circ (G^* \circ a)_<)[seen_0,s \;:=\; seen,a] \;\;\Leftarrow\;\; a{\circ}seen = \perp\!\!\perp \;\;.$$

Proof

$$\begin{array}{ll}
 & (s \;\subseteq\; {\sim}seen_0 \circ (G^* \circ a)_<)[seen_0,s \;:=\; seen,a] \\
= & \{ \quad \text{substitution} \quad \} \\
 & a \;\subseteq\; {\sim}seen \circ (G^* \circ a)_< \\
= & \{ \quad \text{coreflexives} \quad \} \\
 & a \subseteq {\sim}seen \;\;\wedge\;\; a \;\subseteq\; (G^* \circ a)_< \\
\Leftarrow & \{ \quad \text{domains and } I \subseteq G^* \quad \} \\
 & seen{\circ}a \;=\; \perp\!\!\perp \;\wedge\; a \;\subseteq\; a_< \\
= & \{ \quad \text{precondition and } a \text{ is a coreflexive} \quad \} \\
 & \text{true} \;\;.
\end{array}$$

□

The next step is to show that the invariants are maintained by the loop body. To do this, we establish the assertion

$$b \subseteq {\sim}seen_0 \circ (G^* \circ a)_<$$

that follows the choice of b. First, we must note that b can always be chosen, since

$${\sim}seen \circ G \circ seen \neq \perp\!\!\perp \;\;\equiv\;\; {\sim}seen \circ (G{\circ}seen)_< \neq \perp\!\!\perp \;\;.$$

Then,,

$$
\begin{array}{ll}
 & b \\
\subseteq & \{ \quad \text{choice of } b \quad \} \\
 & {\sim}seen \circ (G \circ seen)_< \\
\subseteq & \{ \quad \text{invariant: } seen = s \cup seen_0 \text{ and monotonicity} \quad \} \\
 & {\sim}seen_0 \circ (G \circ (s \cup seen_0))_< \\
= & \{ \quad \text{distributivity and lemma 10.28} \quad \} \\
 & {\sim}seen_0 \circ (G \circ s)_< \\
\subseteq & \{ \quad \text{invariant: } s \subseteq {\sim}seen_0 \circ (G^* \circ a)_< \text{ and monotonicity} \quad \} \\
 & {\sim}seen_0 \circ (G \circ (G^* \circ a)_<)_< \\
\subseteq & \{ \quad \text{domains, } G \circ G^* \subseteq G^* \text{ and monotonicity} \quad \} \\
 & {\sim}seen_0 \circ (G^* \circ a)_< \ .
\end{array}
$$

It is now straightforward to check that each of the invariants is maintained by the loop body. We leave this task to the reader.

The final task is to verify that the postcondition is a consequence of the invariants and the condition for terminating the loop. Clearly it is only necessary to verify the postcondtion

$$s = {\sim}seen_0 \circ (G^* \circ a)_< \ .$$

This we do as follows:

$$
\begin{array}{ll}
 & s = {\sim}seen_0 \circ (G^* \circ a)_< \\
= & \{ \quad \text{invariant: } s \subseteq {\sim}seen_0 \circ (G^* \circ a)_< \text{ and anti-symmetry} \quad \} \\
 & s \supseteq {\sim}seen_0 \circ (G^* \circ a)_< \\
= & \{ \quad \text{shunting rule (2.27)} \quad \} \\
 & s \cup seen_0 \supseteq (G^* \circ a)_< \\
\Leftarrow & \{ \quad \text{fixed-point induction} \quad \} \\
 & s \cup seen_0 \supseteq a \ \wedge \ s \cup seen_0 \supseteq (G \circ (s \cup seen_0))_< \\
\Leftarrow & \{ \quad \text{invariants: } a \subseteq s \quad \text{and} \quad seen = s \cup seen_0 \quad \} \\
 & seen \supseteq (G \circ seen)_< \\
= & \{ \quad \text{domains} \quad \} \\
 & {\sim}seen \circ G \circ seen = \perp\!\!\perp \ .
\end{array}
$$

Since the property $\sim seen \circ G \circ seen \; = \; \perp\!\!\perp$ is the condition for terminating the loop, we are done.

10.2.4 Injective Choice

This section is a preliminary to the discussion in section 10.2.5. Throughout the section, we assume that f has type $\mathbb{N} \leftarrow \mathsf{Node}$. Also, the symbol I denotes I_{Node}: the identity relation on nodes.

Previous sections have established the existence of a delegate function φ according to choice function f with the only proviso being that f is total and functional. Moreover, the property $\varphi \circ \varphi = \varphi$ is an invariant of the algorithm for computing delegates. Cormen, Leiserson and Rivest [CLR90] derive it from the other requirements assuming that f is also injective. For completeness, this is the point-free rendition of their proof.

Lemma 10.30 If f is a total, *injective* function and φ is a delegate function according to f, then

$$\varphi \circ \varphi = \varphi \quad .$$

Proof

$$\begin{array}{ll}
 & \varphi \circ \varphi = \varphi \\
\Leftarrow & \{ \quad \text{assumption: } f \text{ is total and injective, i.e. } f^{\cup} \circ f = I \quad \} \\
 & f \circ \varphi \circ \varphi = f \circ \varphi \\
= & \{ \quad \text{antisymmetry of } \geq \\
 & \quad\;\; \text{and distributivity properties of total functions} \quad \} \\
 & \quad I \subseteq (f \circ \varphi \circ \varphi)^{\cup} \circ \leq \circ f \circ \varphi \\
 & \wedge \quad I \subseteq (f \circ \varphi \circ \varphi)^{\cup} \circ \geq \circ f \circ \varphi \quad .
\end{array}$$

We establish the truth of both conjuncts as follows. First,

$$\begin{array}{ll}
 & (f \circ \varphi \circ \varphi)^{\cup} \circ \leq \circ f \circ \varphi \\
= & \{ \quad \text{converse} \quad \} \\
 & \varphi^{\cup} \circ (f \circ \varphi)^{\cup} \circ \leq \circ f \circ \varphi \\
\supseteq & \{ \quad G^{*} \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f \circ \varphi \;\; (\text{lemma 10.5}) \\
 & \quad\;\; \text{i.e. } (G^{*})^{\cup} \subseteq (f \circ \varphi)^{\cup} \circ \leq \circ f \circ \varphi \\
 & \quad\;\; (\text{distributivity properties of converse and } (\geq)^{\cup} = (\leq)) \quad \}
\end{array}$$

$$\varphi^\cup \circ (G^*)^\cup$$
$$\supseteq \quad \{ \quad I \subseteq G^* \circ \varphi \ \text{(lemma 10.5) and converse} \quad \}$$
$$I \ .$$

Second,

$$(f\circ\varphi\circ\varphi)^\cup \circ \geq \circ f \circ \varphi$$
$$= \quad \{ \quad \text{converse} \quad \}$$
$$\varphi^\cup \circ (f\circ\varphi)^\cup \circ \geq \circ f \circ \varphi$$
$$\supseteq \quad \{ \quad \text{definition of delegate: (10.4) and monotonicity} \quad \}$$
$$\varphi^\cup \circ G^* \circ \varphi$$
$$\supseteq \quad \{ \quad I \subseteq G^* \quad \}$$
$$\varphi^\cup \circ \varphi$$
$$\supseteq \quad \{ \quad \varphi \ \text{is total (by definition: (10.3))} \quad \}$$
$$I \ .$$

□

As also shown above, the property $\mathsf{equiv}.G \subseteq \varphi^\cup \circ \varphi$ is an invariant of the algorithm. However, if f is a total, injective function, the property follows from the definition of a delegate, as we show below.

Lemma 10.31 If f is a total, injective function and φ is a delegate function according to f, strongly connected nodes have the same delegate. That is

$$\mathsf{equiv}.G \subseteq \varphi^\cup \circ \varphi \ .$$

Proof

$$\mathsf{equiv}.G$$
$$= \quad \{ \quad \text{definition} \quad \}$$
$$G^* \cap (G^*)^\cup$$
$$\subseteq \quad \{ \quad \text{lemma 10.5} \quad \}$$
$$(f\circ\varphi)^\cup \circ \geq \circ f \circ \varphi \ \cap \ ((f\circ\varphi)^\cup \circ \geq \circ f \circ \varphi)^\cup$$
$$= \quad \{ \quad \text{converse} \quad \}$$
$$(f\circ\varphi)^\cup \circ \geq \circ f \circ \varphi \ \cap \ (f\circ\varphi)^\cup \circ \leq \circ f \circ \varphi$$
$$= \quad \{ \quad f \ \text{and} \ \varphi \ \text{are total functions, distributivity} \quad \}$$

$(f{\circ}\varphi)^{\cup} \circ (\geq \cap \leq) \circ f \circ \varphi$

$=$ { $\leq$ is antisymmetric }

$\varphi^{\cup} \circ f^{\cup} \circ f \circ \varphi$

$=$ { f is injective and total, i.e. $f^{\cup} \circ f = I$ }

$\varphi^{\cup} \circ \varphi$.

□

The relation $\varphi \circ G^{\cup} \circ \varphi^{\cup}$ is a relation on delegates. Viewed as a graph, it is a homomorphic image of the graph $G^{\cup}$ formed by coalescing all the nodes with the same delegate into one node. Excluding self-loops, this graph is acyclic and topologically ordered by f, as we now show.

Lemma 10.32 If f is a total, injective function and φ is a delegate function according to f, the graph $\varphi \circ G^{\cup} \circ \varphi^{\cup} \cap \neg I$ is acyclic with f as a topological ordering.

Proof By theorem 8.47, it suffices to show that f is a topological ordering. The function f is, by assumption, a total, injective function of type $\mathbb{N} \leftarrow \mathrm{Node}$. Thus, by assumption, f satisfies the first requirement of being a topological ordering. (See definition 8.32.) Applying lemma 8.34, establishing the second requirement is achieved by the following calculation.

$\varphi \circ G^{\cup} \circ \varphi^{\cup} \cap \neg I \;\subseteq\; f^{\cup} \circ < \circ f$

$=$ { shunting rule (2.27) }

$\varphi \circ G^{\cup} \circ \varphi^{\cup} \;\subseteq\; f^{\cup} \circ < \circ f \;\cup\; I$

$=$ { f is total and injective, i.e. $I = f^{\cup} \circ f$
distributivity and definition of $\leq$ }

$\varphi \circ G^{\cup} \circ \varphi^{\cup} \;\subseteq\; f^{\cup} \circ \leq \circ f$

$\Leftarrow$ { φ is functional, i.e. $\varphi \circ \varphi^{\cup} \subseteq I$
monotonicity, converse and transitivity }

$G^{\cup} \;\subseteq\; (f{\circ}\varphi)^{\cup} \circ \leq \circ f \circ \varphi$

$=$ { converse }

$G \;\subseteq\; (f{\circ}\varphi)^{\cup} \circ \geq \circ f \circ \varphi$

$\Leftarrow$ { $G \subseteq G^{*}$, transitivity }

$G^{*} \;\subseteq\; (f{\circ}\varphi)^{\cup} \circ \geq \circ f \circ \varphi$

$=$ { lemma 10.5 }

true .

□

The algorithm presented in fig. 10.1 shows that, viewed as a specification of the function φ, the equation (10.4) always has at least one solution. However, the algorithm is non-deterministic, which means that there may be more than one solution. We now prove that (10.4) has a unique solution in unknown φ if the function f is total and injective.

Lemma 10.33 Suppose f of type $\mathbb{N}\leftarrow\mathsf{Node}$ is a total and injective function, and φ and ψ are both total functions of type $\mathsf{Node}\leftarrow\mathsf{Node}$. Then

$$\begin{array}{ll} & \varphi=\psi \\ \Leftarrow & \quad (\varphi \subseteq (G^*)^\cup \;\wedge\; G^* \subseteq (f\circ\varphi)^\cup\circ\geq\circ f) \\ & \wedge\;\; (\psi \subseteq (G^*)^\cup \;\wedge\; G^* \subseteq (f\circ\psi)^\cup\circ\geq\circ f) \quad . \end{array}$$

Proof Suppose ψ is a total function of type $\mathsf{Node}\leftarrow\mathsf{Node}$. Then

$$\begin{array}{ll} & \psi \subseteq (G^*)^\cup \;\wedge\; G^* \subseteq (f\circ\varphi)^\cup\circ\geq\circ f \\ \Rightarrow & \{ \quad \text{converse and transitivity} \quad \} \\ & \psi^\cup \subseteq (f\circ\varphi)^\cup\circ\geq\circ f \\ = & \{ \quad \psi \text{ is total, i.e. } I \subseteq \psi^\cup\circ\psi \quad \} \\ & I \subseteq (f\circ\varphi)^\cup\circ\geq\circ f\circ\psi \quad . \end{array}$$

Interchanging φ and ψ, and combining the two properties thus obtained, we get that, if φ and ψ are both total functions of type $\mathsf{Node}\leftarrow\mathsf{Node}$,

$$\begin{array}{ll} & \quad (\varphi \subseteq (G^*)^\cup \;\wedge\; G^* \subseteq (f\circ\psi)^\cup\circ\geq\circ f) \\ & \wedge\;\; (\psi \subseteq (G^*)^\cup \;\wedge\; G^* \subseteq (f\circ\varphi)^\cup\circ\geq\circ f) \\ \Rightarrow & \{ \quad \text{see above} \quad \} \\ & \quad I \subseteq (f\circ\psi)^\cup\circ\geq\circ f\circ\varphi \\ & \wedge\;\; I \subseteq (f\circ\varphi)^\cup\circ\geq\circ f\circ\psi \\ = & \{ \quad f,\ \varphi \text{ and } \psi \text{ are all total functions,} \\ & \quad\; \text{converse and distributivity} \quad \} \\ & \quad I \subseteq (f\circ\psi)^\cup\circ((\leq)\cap(\geq))\circ f\circ\varphi \\ & \wedge\;\; I \subseteq (f\circ\varphi)^\cup\circ((\leq)\cap(\geq))\circ f\circ\psi \\ = & \{ \quad \text{anti-symmetry of } (\leq) \quad \} \end{array}$$

$$\begin{array}{ll}
& I \subseteq (f\circ\psi)^{\cup}\circ f\circ\varphi \;\wedge\; I \subseteq (f\circ\varphi)^{\cup}\circ f\circ\psi \\
= & \{ \quad f \text{ and } \psi \text{ are total functions, anti-symmetry of subset} \quad \} \\
& f\circ\psi = f\circ\varphi \\
= & \{ \quad f \text{ is an injective, total function} \quad \} \\
& \psi = \varphi \;\; .
\end{array}$$

The lemma follows by symmetry and associativity of conjunction.
□

Earlier, we stated that (10.9) formulates the property that there is a path from each node to its delegate on which successive nodes have the same delegate. Combined with (10.10) and the transitivity of equality, this means that there is a path from each node to its delegate on which all nodes have the same delegate. We conclude this section with a point-free proof of this claim. Since the claim is not specific to the delegate function, we formulate the underlying lemmas (lemmas 10.34 and 10.35) in general terms. The relevant property of the delegate function, lemma 10.36, is then a simple instance.

Should one wish to interpret lemma 10.34 pointwise, the key is to note that, for total function h and arbitrary relation S, $h^{\cup}\circ h \cap S$ relates two points x and y if they are related by S and $h.x = h.y$. However, it is not necessary to do so: completion of the calculation in lemma 10.35 demands the proof of lemma 10.34 and this is best achieved by uninterpreted calculation. In turn, lemma 10.35 is driven by lemma 10.36 which expresses the delegate function φ as a least fixed point; crucially, this enables the use of fixed-point induction to reason about φ.

Lemma 10.34 If h is a total function,

$$h \cap R\circ(h^{\cup}\circ h \cap S) \;=\; h \cap (h{\cap}R)\circ S$$

for all relations R and S.

Proof By mutual inclusion:

$$\begin{array}{ll}
& h \cap (h{\cap}R)\circ S \\
\subseteq & \{ \quad \text{modularity rule: (4.8)} \quad \} \\
& (h{\cap}R)\circ((h{\cap}R)^{\cup}\circ h \;\cap\; S) \\
\subseteq & \{ \quad h{\cap}R \subseteq h \text{, monotonicity} \quad \} \\
& (h{\cap}R)\circ(h^{\cup}\circ h \;\cap\; S) \\
\subseteq & \{ \quad h \text{ is a total function, so } h\circ h^{\cup}\circ h = h \\
& \quad\;\; h{\cap}R \subseteq h \text{, distributivity and monotonicity} \quad \}
\end{array}$$

$$
\begin{array}{ll}
 & h \cap R\circ(h^{\cup}\circ h \cap S) \\
= & \{ \quad \text{idempotency (preparatory to next step)} \quad \} \\
 & h \cap h \cap R\circ(h^{\cup}\circ h \cap S) \\
\subseteq & \{ \quad \text{modularity rule: (4.8)} \quad \} \\
 & h \cap (h\circ(h^{\cup}\circ h \cap S)^{\cup} \ \cap \ R)\circ(h^{\cup}\circ h \cap S) \\
\subseteq & \{ \quad h \text{ is a total function, so } h\circ h^{\cup}\circ h = h \\
 & \quad\ (h^{\cup}\circ h \cap S)^{\cup} \subseteq h^{\cup}\circ h\,, \\
 & \quad\ \text{distributivity and monotonicity} \quad \} \\
 & h \cap (h{\cap}R)\circ S \quad .
\end{array}
$$

□

Lemma 10.35 If h is a total function,

$$h \cap (h^{\cup}\circ h \cap R)^{*} \;=\; \langle \mu X :: h\cap(I \cup X\circ R)\rangle$$

for all relations R.

Proof We derive the right side as follows.

$$
\begin{array}{ll}
 & h \cap (h^{\cup}\circ h \cap R)^{*} \;=\; \mu g \\
\Leftarrow & \{ \quad \text{fusion theorem} \quad \} \\
 & \langle \forall X :: h\cap(I \cup X\circ(h^{\cup}\circ h \cap R)) \;=\; g.(h\cap X)\rangle \\
= & \{ \quad \text{distributivity, lemma 10.34 with } R,S := X,R \quad \} \\
 & \langle \forall X :: (h\cap I)\cup(h \cap (h{\cap}X)\circ R) \;=\; g.(h\cap X)\rangle \\
\Leftarrow & \{ \quad \text{strengthening: } X := h\cap X \quad \} \\
 & \langle \forall X :: (h\cap I)\cup(h \cap X\circ R) \;=\; g.X\rangle \\
= & \{ \quad \text{distributivity} \quad \} \\
 & \langle \forall X :: h\cap(I \cup X\circ R) \;=\; g.X\rangle \quad .
\end{array}
$$

□

Lemma 10.36

$$\varphi \;=\; \langle \mu X :: \varphi\cap(I \,\cup\, X\circ G^{\cup})\rangle \quad .$$

Proof

$$
\begin{array}{ll}
 & \varphi \\
= & \{ \quad \text{lemma 10.22 (specifically, } \varphi \subseteq (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^{*}) \quad \} \\
 & \varphi \cap (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^{*} \\
= & \{ \quad \text{lemma 10.35} \quad \} \\
 & \langle \mu X :: \varphi \cap (I \cup X \circ G^{\cup}) \rangle \quad .
\end{array}
$$

□

The significance of the equality in lemma 10.36 is the inclusion of the left side in the right side. (The converse is trivial.) Thus, in words, the lemma states that there is a path from each node to its delegate on which every node has the same delegate.

10.2.5 Summary and Discussion

We summarise the results of this section with the following theorem.

Theorem 10.37 (Delegate Function) Suppose f of type $\mathbb{N} \leftarrow \mathsf{Node}$ is a total function and G is a finite graph. Then the equation

$$\varphi \;::\; \varphi \circ \varphi^{\cup} \subseteq I_{\mathsf{Node}} \subseteq \varphi^{\cup} \circ \varphi \;\;\wedge\;\; \varphi \subseteq (G^{*})^{\cup} \;\;\wedge\;\; G^{*} \subseteq (f \circ \varphi)^{\cup} \circ \geq \circ f$$

has a solution with the additional properties that the solution is a closure operator (i.e. a delegate is its own delegate):

$$\varphi \circ \varphi = \varphi \quad ,$$

strongly connected nodes have the same delegate:

$$\mathsf{equiv}.G \subseteq \varphi^{\cup} \circ \varphi$$

and there is a path from each node to its delegate on which successive nodes have the same delegate:

$$\varphi \subseteq (G^{\cup} \cap \varphi^{\cup} \circ \varphi)^{*} \quad .$$

More precisely, there is a path from each node to its delegate on which all nodes have the same delegate:

$$\varphi = \langle \mu X :: \varphi \cap (I_{\mathsf{Node}} \cup X \circ G^{\cup}) \rangle \quad .$$

Moreover, a delegate has the largest f value

$$\varphi \subseteq f^{\cup} \circ \geq \circ f \quad .$$

If the function f is injective, the solution is unique; in this case, we call the unique solution *the* delegate function on G according to f. Moreover, f is a topological ordering of the nodes of the graph

$$\varphi \circ G^{\cup} \circ \varphi^{\cup} \cap \neg I_{\mathsf{Node}}$$

(the graph obtained from $G^{\cup}$ by coalescing all nodes with the same delegate and removing self-loops). This graph is therefore acyclic.

Proof As discussed prior to lemma 10.33, the algorithm establishes the existence of at least one solution, and lemma 10.33 shows that any solution is unique. The remaining properties are proved in lemmas 10.30, 10.31, 10.22, 10.36, 10.6 and 10.32.
□

As mentioned above, this section is inspired by Cormen, Leiserson and Rivest's notion of the "forefather" function and its use in applying depth-first search to the computation of strongly connected components [CLR90, pp.488–494]. However, our presentation is more general than theirs; in particular, we do not assume that the choice function is injective.

The motivation for our more general presentation is primarily to kill two birds with one stone. As do Cormen, Leiserson and Rivest, we apply the results of this section to computing strongly connected components: see section 13. This is one of the "birds". The second "bird" is represented by the case that the choice function is a constant function (for example, $f.a = 0$, for all nodes a). In this case, the choice of node a in the algorithm of fig. 10.1 reduces to the one condition $a{\circ}seen = \perp\!\!\perp$ (in words, a has not yet been seen) and the function f plays no role whatsoever. Despite this high level of nondeterminism, the specification of a delegate (see section 10.2.1) allows many solutions that are not computed by the algorithm. (For example, the identity function satisfies the specification.) The analysis of section 10.2.2 is therefore about the properties of a function that records the history of repeated searches of a graph until all nodes have been seen: the delegate function computed by repeated graph search records for each node b, the node a from which the search that sees b was initiated.

This analysis reveals many properties of graph searching that other accounts may suggest are peculiar to depth-first search. Most notable is the property that strongly connected nodes are assigned the same delegate. As shown in lemma 10.31, this is a necessary property when the choice function is injective; otherwise, it is not a necessary property but it is a property of the delegate function computed by repeated graph search, whatever graph-searching algorithm is used. The second notable property of repeated graph search is that there is a path from each node to its delegate on which all nodes have the same delegate. This is closely related to the property that Cormen, Leiserson and Rivest call the "white-path theorem" [CLR90, pp.482], which we discuss shortly.

Our analysis shows that the property is a generic property of repeated graph search and not specific to depth-first search.

In order to discuss the so-called “white-path theorem”, it is necessary to give a preliminary explanation. Operational descriptions of graph-searching algorithms often use the colours white, grey and black to describe nodes. A white node is a node that has not been seen, a grey node is a node that has been seen but not all edges from the node have been “processed”, and a black node is a node that has been seen and all edges from the node have been “processed”. The property “white”, “grey” or “black” is, of course, time-dependent since initially all nodes are white and on termination all nodes are black.

Now lemmas 10.18 and 10.19 express subtley different versions of what is called the “white-path theorem”. Suppose a search from node a is initiated in the *outer* loop. The search finds nodes on paths starting from a. There are three formally different properties of the paths that are found:

(i) The final node on the path is white at the time the search from a is initiated.

(ii) All nodes on the path are white at the time the search from a is initiated.

(iii) All nodes on the path are white at the time the search from their predecessor on the path is initiated.

In general, if nodes are labelled arbitrarily white or non-white, the sets of paths described by (i), (ii) and (iii) are different. (They are ordered by the subset relation, with (i) being the largest and (iii) the smallest.) However, in a repeated graph search, the sets of paths satisfying (i) and (ii) are equal. This is the informal meaning of lemma 10.17. Moreover, the right side of the assignment to s in fig. 10.1 is the set of nodes reached by paths satisfying (i); lemma 10.18 states that, in a repeated graph search, the nodes that are added by a search initiated from node a are the nodes that can be reached by a path satisfying (ii).

We claim —without formal proof— that it is also the case that, in a repeated graph search, all three sets of paths are equal. That is, the set of paths described by (iii) is also equal to the set of paths described by (i). We don’t give a proof because it is not a fact that we exploit and, without introducing additional auxiliary variables, it is impossible to express formally. Informally, it is clear from the implementation shown in fig. 10.2, in particular the choice of nodes b and c. The introduction of timestamps does allow us to prove the claim formally for depth-first search. See section 14.2.

Cormen, Leiserson and Rivest’s [CLR90, pp.482] “white-path theorem” states that it is a property of depth-first search that paths found satisfy (ii). Characteristic of depth-first search is that the property is true for all nodes, and not just nodes from which a search is initiated in the outer loop. We discuss this in more detail later.

Finally, let us briefly remark on lemma 10.32. As we see later, not only can depth-first search be used to calculate the strongly connected components of a graph, in doing so it also computes a topological ordering of these components (more precisely a topological ordering of the converse of the homomorphic-image graph discussed in section 9.7). Lemma 10.32 is more general than this. It states that, if the choice function is injective, it is a topological ordering of the converse of the graph obtained by coalescing all the nodes with the same delegate and then omitting self-loops. In fact, this is also true of the delegate function computed as above. We leave its proof to the reader: remembering that during execution of the algorithm φ is partial with right domain $\varphi{>}$, identify and verify an invariant that states that f is a topological ordering on a subgraph of $\varphi \circ G^{\cup} \circ \varphi^{\cup} \cap \neg I$.

Chapter 11

Depth-First Search

In section 10.1, we described a generic graph-searching algorithm and concluded with the claim that depth-first search is an instance of the algorithm whereby unexplored edges are stored in a stack. The stack-based, iterative implementation was the basis of Tarjan's [Tar72] seminal paper on depth-first search. In this and later sections, we base the discussion on the (equivalent) recursive formulation of depth-first search commonly presented in textbooks (for example, [AHU82, pp.222–226]). The reason for this choice is that "timestamping" events during the search (see section 13.1) is easier to present. On the other hand, reasoning about the recursive algorithm poses new challenges. The challenges could be overcome by transforming the recursive implementation into the equivalent stack-based iterative algorithm but we choose to tackle them head on. This section introduces the basic graph-searching algorithm as a relatively straightforward illustration of how to reason about recursion. (Later sections are more complicated.)

Given a finite graph G, the following procedure initiates a depth-first search at selected nodes of G until all nodes of the graph have been "seen" (denoted by $seen$). Nodes are selected arbitrarily from the set of nodes that have not yet been seen (denoted by $\sim seen$).

```
$seen := \emptyset$;
while $\sim seen \neq \emptyset$ do
   begin
      choose node $a$ such that $a \in \sim seen$
   ;  $dfs(a)$
   end
```

(We use standard set-theory notation temporarily for reasons of familiarity. Shortly, we switch to the notation of the point-free calculus.)

The procedure $dfs(a)$ for executing a depth-first search of the nodes reachable from a is implemented as follows:

```
      $seen := seen \cup \{a\}$
;     while there is an edge $(a,b)$ such that $b \in \sim seen$ do
        begin
          choose node $b$ such that $(a,b) \in G \wedge b \in \sim seen$
        ; $dfs(b)$
        end
```

To see the connection with section 10.1 note that the criterion for choosing b is equivalent to the property

$$a \circ \top\!\!\top \circ b \subseteq seen \circ G \circ \sim seen \quad ,$$

the property that is key to an efficient implementation of graph searching. As a consequence, lemma 10.1 will also play an important part in reasoning about the recursive implementation.

There is a large element of non-determinism in the execution of depth-first search. Fig. 11.1 illustrates one particular execution sequence. The labels O1 thru O6 are the nodes that initiate a search in the outer loop; the numbers indicate the order in which they have been chosen. Searches from the nodes that do not have such a label are initiated in the inner loop. The edges (a,b) that are chosen in the inner loop are highlighted.

The pairs of numbers labelling each node are "timestamps". We discuss the implementation and properties of these timestamps in detail in section 13.1. For the moment, we use them to illustrate some remarks we make. The first component of such a pair gives the "time" at which the search from the node is started and the second component is the "time" at which the search is finished. For example, the search from the top-left node (the node labelled O2) is begun at "time" 19 and finished at "time" 20; the search from the node labelled O1 is begun at "time" 1 and finished at "time" 18. By chasing the timestamps, it is possible to see which choices were made in the particular execution shown in fig. 11.1.

Fig. 11.1 has been designed to illustrate a couple of points about depth-first search. First, most accounts of depth-first search emphasise the construction of a forest of so-called "spanning" trees. In fig. 11.1 only two trees are readily visible: the trees defined by the highlighted edges. Note that the tree with root O1 "spans" a non-trivial strongly connected component of the graph. That is, the component has more than one node and every node in the component is reachable by a path consisting of tree edges from O1; however, not all nodes in the tree are strongly connected. Similarly, the tree with

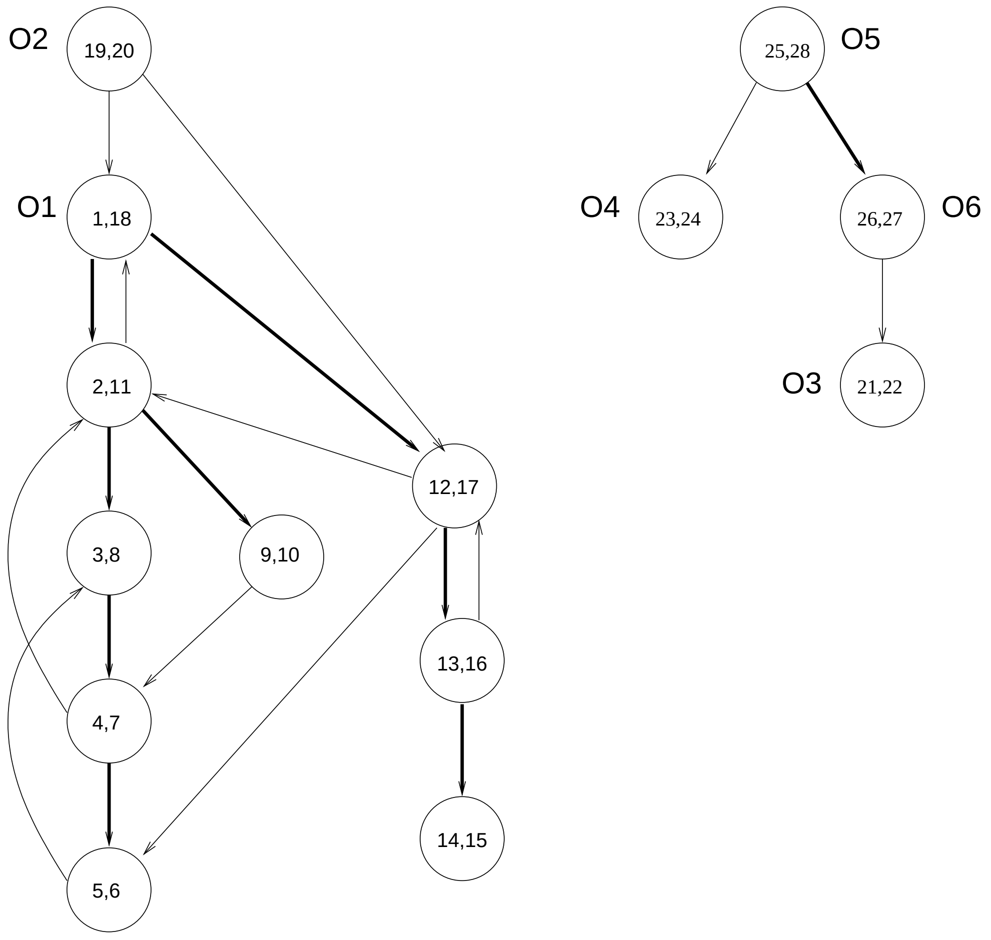

Figure 11.1: Timestamps

root O5 has two nodes, O5 and O6. Each of these nodes forms a strongly connected component but the two nodes are not strongly connected.

There are three additional trees: each of the nodes O2, O3 and O4 is the root of a tree with just one node. Thus, although the subgraph defined by the nodes O3, O4, O5 and O6 forms a “spanning” tree in the graph, it is not one of the forest of “spanning” trees constructed by this particular execution of depth-first search.

The second point to make about depth-first search is that a call of $\mathsf{dfs}(a)$ does not necessarily “see” all nodes reachable from node a. For example, the node with timestamp $12,17$ is reachable from the node with timestamp $2,11$ but, as indicated by the fact that $11<12$, the search from the node with timestamp $2,11$ is ended before the search from the node with timestamp $12,17$ begins.

In the next section, we formulate precisely what is meant by the informal statement that depth-first search is a graph-searching algorithm. Essentially, we show that (the recursive implementation of) depth-first search is an instance of repeated graph search (see section 10.2). But we do more than this. We formulate precisely the differences in properties of calls of dfs in the outer and inner loops as well as the properties that are common to both. An important step in the analysis is to show that the procedure dfs implements a *function* mapping a set of nodes to a set of nodes. This is done in section 11.3 following which properties of the inner and outer loops are formally verified in sections 11.4 and 11.5. Section 11.1 formulates these properties (without proof) whilst section 11.2 formulates precisely the (relational) semantics of the procedure dfs that we assume in the formal verifications.

The final section serves as an introduction to our later discussion of applying depth-first search to the calculation of strongly connected components. We show that, although a call of $dfs(b)$ in the inner loop does not "see" all the nodes that can be reached from b —including nodes that can be reached from b by paths along edges that have not already been "seen"— it is the case that all calls of dfs, whether from the outer or inner loops, "see" every strongly connected component of the input graph either in its entirety or not at all; in other words, calls of dfs never "see" a non-empty, proper subset of the nodes of a strongly connected component of the graph.

11.1 Properties of Depth-First Search

As illustrated by fig. 11.1, a call of the procedure dfs does not always "see" all the nodes that are reachable from a given node. This claim is, however, true of the searches that are initiated in the *outer* loop. Just as for the generic repeated-graph-search algorithm that we analysed in section 10.2, the function of a call of the procedure $dfs(a)$, in general, is to find all the nodes that are reachable from a along edges that have not already been "seen". We make this precise in this section.

There are several elements to this claim. One is that the procedure dfs is always guaranteed to terminate (provided the graph is finite). The second is an assertion about the relation between the variable $seen$ and the nodes reachable from a node in $seen$. We shall prove that the property

$$(11.1)\quad (seen\circ G^{*})_{>} \;=\; seen$$

is an invariant of the *outer* loop. In words, before and after each iteration of the outer loop, $seen$ is closed under reachability in the graph G. In general, we shall prove that $dfs(a)$ implements the function $D.a$ given by, for all coreflexives s and nodes a,

$$(11.2)\quad D.a.s \;=\; s\cup(a\circ(G\circ{\sim}s)^{*})_{>}\quad .$$

This fact is critical to reasoning about depth-first search because it means that a call of $dfs(a)$ is equivalent to the assignment statement

(11.3) $seen \;:=\; seen \cup (a\circ(G\circ{\sim}seen)^*)\!\!>$.

The difficulty posed by the recursion has thus been conquered: subsequent reasoning can use straightforward and well-known techniques for reasoning about iterative programs. (The difficulties of recursion will, however, reappear when we consider timestamps.)

We introduce the relations GE and GT on sets of nodes, defined by

$$s1[\![GE]\!]s0 \;\equiv\; s1\supseteq s0 \text{ , and}$$

$$s1[\![GT]\!]s0 \;\equiv\; s1\supseteq s0 \wedge s1\neq s0 \text{ .}$$

The name "GE" is just another name for the containment relation ("$\supseteq$") on sets of nodes, which is reflexive and transitive. That is,

(11.4) $I_{\mathsf{Node}} \subseteq GE \;\wedge\; GE\circ GE \subseteq GE$.

We introduce a new name because, otherwise, the overloading of notation in (11.4) and similar statements could be confusing. We continue to use the familiar mathematical symbol where no confusion can occur. See, for example, the use of the "$\subseteq$" symbol in (11.7) below. (Another way of resolving the problem is to adorn all occurrences of relations like the subset relation with their type, but that would introduce a lot of unnecessary noise in the formulae.)

Similarly, the name "GT" denotes "proper" containment. It is thus transitive (but not reflexive) and GE is the reflexive closure of GT. That is,

(11.5) $GT^* \;=\; I_{\mathsf{SetOfNode}} \cup GT \;=\; GE$.

Because $\mathsf{equiv}.G\subseteq G^*$, it is straightforward to show that (11.1) implies

(11.6) $(seen\circ\mathsf{equiv}.G)\!\!> \;=\; seen$.

Given that (11.1) is an invariant of the outer loop, we see from (11.6) that strongly connected components are added to $seen$ as a whole and not in parts by calls of dfs initiated in the outer loop. (A formal proof of this claim is given in corollary 11.25.) More significant, however, is what happens when dfs is called from the inner loop. We show that the strongly connected component p containing node a is added to $seen$ by a call of $dfs(a)$ if a is the first node in p to be added to $seen$. See theorem 11.24.

As remarked earlier, (11.1) is not an invariant of the inner loop. So we are obliged to seek a relation that is an invariant of both the inner and outer loops and whose invariance implies (11.1) in the outer loop. The appropriate relation is suggested by lemma 10.1.

The term $seen \circ G \circ \sim seen$ represents a subset of the set of edges of the graph G: the edges at the "frontier" of the search. To emphasise the importance of the frontier edges, we introduce the function Fr of type

$$(\mathsf{SetOfNode}\sim\mathsf{SetOfNode}) \leftarrow \mathsf{SetOfNode}$$

defined by

$$(11.7)\quad Fr.s \;=\; s \circ G \circ \sim s \;\;.$$

We show that the subset relation on frontier edges is an invariant of the procedure dfs. To be precise, we show that the relation $Fr^{\cup} \circ (\subseteq) \circ Fr$ of type $\mathsf{SetOfNode}\sim\mathsf{SetOfNode}$ defined by

$$(11.8)\quad s1[\![Fr^{\cup} \circ (\subseteq) \circ Fr]\!]s0 \;\;\equiv\;\; s1 \circ G \circ \sim s1 \subseteq s0 \circ G \circ \sim s0$$

is an invariant relation of the procedure dfs.

Similarly, as remarked earlier, calls of $dfs(b)$ in the inner loop do not "see" all the nodes that can be reached from node b along edges that have not already been "seen". However, this is the case for calls of $dfs(a)$. This is formalised by introducing two relations of type $\mathsf{SetOfNode}\sim\mathsf{SetOfNode}$. The relation New where

$$s1[\![New]\!]s0 \;\;\equiv\;\; s1 \;=\; s0 \cup (s1 \circ \sim s0 \circ (G \circ \sim s0)^*)\!>$$

is an invariant of the outer loop whereas the weaker relation NR, where

$$s1[\![NR]\!]s0 \;\;\equiv\;\; s1 \circ \sim s0 \;\subseteq\; (\sim s0 \circ (G \circ \sim s0)^*)\!> \;\;,$$

is an invariant of the inner loop.

The invariants of the outer and inner loops are documented in figs. 11.2 and 11.3. (The meaning of $D.a$ being an invariant value of the inner loop will be discussed in detail later.) The remainder of this section is about formally verifying the assertions made in these figures.

We invite the reader to compare fig. 11.2 with fig. 10.1. When doing so, a warning is in order: the repeated search shown in fig. 10.1 is about searching $G^{\cup}$, not G as in fig. 11.2. So comparisons may be confusing. (The reason for this difference is that the calculation of strongly connected components has two phases. In the first phase, a depth-first search of the graph G is used to construct a function f; in the second phase, the function f is used by the delegate algorithm in a search of $G^{\cup}$; the output function φ assigns to each strongly connected component of the graph a representative element of the component.)

Apart from this difference, the two algorithms still look very different. However, supposing the choice function f used in fig. 10.1 is a constant function, all mention of

```
seen := $\perp\!\!\perp$ ;
{ Invariant Relation:  $New :: (SetOfNode{\sim}SetOfNode)$
      where $s1[\![New]\!]s0 \;\equiv\; s1 \;=\; s0 \cup (s1\circ{\sim}s0\circ(G\circ{\sim}s0)^*)_>$
  Invariant Property: $Fr.seen = \perp\!\!\perp \;\wedge\; (seen\circ G^*)_> \;=\; seen$
      where $Fr.s \;=\; s\circ G\circ{\sim}s$
  Invariant Property: $\langle\forall p : scc.p : p\circ seen = \perp\!\!\perp \;\vee\; p\circ seen = p\rangle$
      where $scc.p$ means $p$ is a strongly connected component of $G$ }
while $seen \neq I_{Node}$ do
  begin
      choose node $a$ such that $a\circ seen = \perp\!\!\perp$
  ; {  $a\circ{\sim}seen = a$  }
      /*  dfs(a) implements the function $D.a$  */
      /*  where $D.a.s \;=\; s\cup(a\circ(G\circ{\sim}s)^*)_>$ .  */
      /*  So it is equal to the assignment $seen := D.a.seen$ .  */
      dfs(a)
  end
```

Figure 11.2: Invariants of the Outer Loop

it can be elided. Also, ignore assignments to φ . Then, in fig. 11.2, the assignment to $seen$ is

$$seen := seen \cup (a\circ(G\circ{\sim}seen)^*)_>$$

whereas in fig. 10.1, it is

$$seen := seen \cup {\sim}seen\circ(G^*\circ a)_< \;\; .$$

Lemma 10.18 states that the latter assignment is equivalent to the assignment

$$seen := seen \cup (({\sim}seen\circ G)^*\circ a)_<$$

which, using distributivity properties of converse, is the same as

$$seen := seen \cup (a\circ(G^{\cup}\circ{\sim}seen)^*)_> \;\; .$$

```
{ $a\circ\sim seen = a$ }
{ Invariant Relation:
        $(Fr^{\cup}\circ(\subseteq)\circ Fr\ \cap\ NR) :: (SetOfNode\sim SetOfNode)$
    where $Fr.s\ =\ s\circ G\circ\sim s$
     and $s1[\![NR]\!]s0\ \equiv\ s1\supseteq s0\ \wedge\ s1\circ\sim s0\ \subseteq\ (\sim s0\circ(G\circ\sim s0)^{*})_{>}$ .
  Invariant Value: $D.a$
    where $D.a.s\ =\ s\cup(a\circ(G\circ\sim s)^{*})_{>}$ .
  Invariant Property: $\langle\forall p : scc.p : p\circ seen = \perp\!\!\perp \vee p\circ seen = p\rangle$
    where $scc.p$ means $p$ is a strongly connected component of $G$ }
seen := seen $\cup$ a
{ $a\circ seen = a$ }
; { Invariant Property:
      $\langle\forall p : scc.p\wedge p\neq(a\circ equiv.G)_{>} : p\circ seen = \perp\!\!\perp \vee p\circ seen = p\rangle$ }
while $a\circ G\circ\sim seen \neq \perp\!\!\perp$ do
  begin
    choose node $b$ such that $a\circ\top\!\!\top\circ b\ \subseteq\ a\circ G\circ\sim seen$
  ; { $b\circ\sim seen = b$ }
    dfs(b)
  end
{ Input/Output Relation: $D.a\cap Fr^{\cup}\circ(\subseteq)\circ Fr\cap NR$ }
```

Figure 11.3: Invariants of the Procedure dfs

So, in the case that the choice function f is a constant function, the two algorithms are the same except for the replacement of G by $G^{\cup}$. By showing that $dfs(a)$ implements the function $D.a$ we effectively show that depth-first search is an instance of the generic repeated graph search algorithm presented in section 10.2. Consequently, we may instantiate properties of repeated graph search proved in section 10.2.

11.2 Semantics of the Basic Procedure

In order to reason formally about the search procedure, we have to formalise its semantics. Chapter 6 explained the semantics of the basic programming constructs (assignment statements, sequential composition, etc.) but stopped short of explaining the semantics of recursion.

Let us write $\mathsf{DFS}.a$ for the semantics of $\mathtt{dfs}(a)$. This is a combination of the semantics of the component statements in its implementation (as discussed in section 6) and the "equation" between the text "$\mathtt{dfs}(a)$" and its implementation.

Let $pre.a$ denote the coreflexive representing the assertion $a\circ{\sim}seen = a$ and let $S.a$ denote $[\![seen := seen \cup a]\!]$.

The semantics of the inner **while** statement is clearly dependent on the parameter a; it also depends on $\mathsf{DFS}.b$ (the meaning of the call of $\mathtt{dfs}(b)$ in its body). In order to formulate the semantics more precisely, we abstract from DFS and let $W.d.a$ denote the semantics of the **while** loop when the semantics of $\mathtt{dfs}(b)$ is generalised to $d.b$ for some function d of type

$$(\mathsf{SetOfNode}{\sim}\mathsf{SetOfNode}) \leftarrow \mathsf{Node} \quad .$$

Then we define the semantics of $\mathtt{dfs}$ to be a least fixed point:

$$(11.9) \quad \mathsf{DFS} \;=\; \langle \mu d :: \langle a :: W.d.a\circ S.a\rangle\rangle \quad .$$

In words, DFS is the least fixed point of the function that maps a function d of appropriate type into a function that maps node a into (the relation) $W.d.a\circ S.a$. Let $\dot{\subseteq}$ denote the pointwise ordering on functions from nodes to relations. That is, for all nodes a and all functions f and g mapping nodes to relations

$$f\,\dot{\subseteq}\,g \;\equiv\; \langle \forall a :: f.a \subseteq g.a\rangle \quad .$$

Similarly, we extend composition of relations to functions from nodes to relations. Specifically, if functions f and g map nodes to relations, we define $f\dot{\circ}g$ by

$$f\dot{\circ}g \;=\; \langle a :: f.a\circ g.a\rangle \quad .$$

Using this notation, we can rewrite definition (11.9) as:

$$(11.10)\; \mathsf{DFS} \;=\; \langle \mu d \;::\; W.d\,\dot{\circ}\,S\rangle \quad .$$

Recall that $S.a$ is the meaning of the assignment statement that adds a to $seen$. (Another notation for it would be $(\cup a)$.) The function W gives the meaning of the inner **while** statement after abstracting from the call of $\mathtt{dfs}$. That is,

$$(11.11)\; W.d.a \;=\; t.a\circ(\langle \cup b \;::\; d.b\circ C.b.a\rangle \circ {\sim}t.a)^{*}$$

where, for all a and b, $C.b.a$ is the criterion for choosing b given a in the current state and $t.a$ is the condition for terminating the **while** statement.

An important first step in our verification of our "obvious" property of depth-first search is to incorporate the precondition on calls of dfs into the fixed-point definition of DFS. Recall that $pre.a$ denotes the coreflexive representing the assertion $a \circ {\sim}seen = a$. Then we have:

Lemma 11.12

$$DFS \dot{\circ} pre \;=\; \langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle$$

Proof We use fixed-point fusion. See theorem 2.43. Recall that composition of relations is the lower adjoint in a Galois connection —specifically (4.6)— it is easily proved that the lifted composition ($\dot{\circ} pre$) is also the lower adjoint in a Galois connection. Thus we can calculate as follows:

$$DFS \dot{\circ} pre \;=\; \langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle$$

$= \quad \{ \quad$ definition (11.10) of DFS $\quad \}$

$$\langle \mu d :: W.d \dot{\circ} S \rangle \dot{\circ} pre \;=\; \langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle$$

$\Leftarrow \quad \{ \quad$ fixed-point fusion: theorem 2.43 $\quad \}$

$$\langle \forall d :: W.d \dot{\circ} S \dot{\circ} pre = W.(d \dot{\circ} pre) \dot{\circ} S \dot{\circ} pre \rangle$$

$\Leftarrow \quad \{ \quad$ Leibniz $\quad \}$

$$\langle \forall d :: W.d = W.(d \dot{\circ} pre) \rangle \quad .$$

Continuing with the left side of the equality, we have, for all d and all a,

$$W.d.a$$

$= \quad \{ \quad$ (11.11) $\quad \}$

$$t.a \circ (\langle \cup b :: d.b \circ C.b.a \rangle \circ {\sim}t.a)^*$$

$= \quad \{ \quad$ $C.b.a$ is a coreflexive representing the state

$\qquad\qquad a \circ \top\!\!\top \circ b \;\subseteq\; seen \circ G \circ {\sim}seen$

$\qquad\qquad$ thus $b \subseteq {\sim}seen$, i.e. $pre.b \circ C.b.a = C.b.a$ $\quad \}$

$$t.a \circ (\langle \cup b :: d.b \circ pre.b \circ C.b.a \rangle \circ {\sim}t.a)^*$$

$= \quad \{ \quad$ definition of lifted composition $\quad \}$

$$t.a \circ (\langle \cup b :: (d \dot{\circ} pre).b \circ C.b.a \rangle \circ {\sim}t.a)^*$$

$= \quad \{ \quad$ definition (11.11) of W $\quad \}$

$$W.(d \dot{\circ} pre).a \quad .$$

Combining the two calculations completes the proof.
□

11.3 The Function of a Depth-First Search

Our goal in this section is to prove that, for each node a, the procedure $dfs(a)$ implements the function $D.a$ of type $\mathsf{SetOfNode} \leftarrow \mathsf{SetOfNode}$ defined by equation (11.2).

Theorem 11.13 is the theorem that we described earlier as being crucial to understanding depth-first search. Lemma 11.12 gives a relational semantics to depth-first search but theorem 11.13 shows that it is, in fact, a function from sets of nodes to sets of nodes. Thus, in spite of the unlimited nondeterminism in the choice of nodes in the inner loop, the outcome is always the same.

The proof of theorem 11.13 is unusual because we are obliged to switch from the point-free formulation of the relational semantics to pointwise reasoning about sets of nodes. Since point-free reasoning is less well-known, we begin the proof with the equivalent pointwise rendition of the argument used which we hope will make it more accessible.

Theorem 11.13 The procedure $dfs(a)$ implements the function $D.a$, where

$$D.a.s \;=\; s \cup (a \circ (G \circ \sim s)^{*})> \quad .$$

That is, with precondition $pre.a$ defined to be the coreflexive representing the set of states s such that

$$s \circ a = \perp\!\!\perp \quad ,$$

then

$$DFS.a \circ pre.a \;=\; D.a \circ pre.a \quad .$$

Proof Recalling lemma 11.12, which gives the semantics $DFS.a$ of the procedure $dfs(a)$, our task is to prove that, for all a,

$$\langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle \;=\; D \dot{\circ} pre \quad .$$

The occurrence of a least fixed point on the left side of the equation immediately suggests the use of fixed-point induction. Now, $D.a$ is a total function, and to prove that a relation R is equal to a function f restricted to some right domain p, it suffices to prove that the right domain of R is p and R is a subset of f. (We leave the straightforward verification of this claim to the reader.) That is, we have to prove that

$$\langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle \;\dot{\subseteq}\; D$$

and

$$\langle\forall a :: (\langle\mu d :: W.d \dot{\circ} S \dot{\circ} pre\rangle.a)_{>} \;=\; pre.a\rangle \ .$$

The proof of the second property is much more straightforward than it might look at first sight. It is in fact a use of fixed-point fusion: the "apply to a" function ("$(.a)$") and the right domain operator are both lower adjoints in Galois connections of appropriate type, and $W.d.a$ and $S.a$ are both total functions. It follows that $((W.d \dot{\circ} S \dot{\circ} pre).a)_{>}$ equals $pre.a$, for all d, and hence its least fixed point is also $pre.a$, as required.

The proof of the inequation is more demanding. The basis of the proof is summarised in the annotations added to $dfs(a)$ in the text below.

```
    { $a \circ {\sim}seen = a \wedge seen = s0$ }
    $seen := seen \cup a$
  ; { **Invariant Property**: $a \circ seen = a$
      **Invariant Value**: $D.a$ }
    **while** $a \circ G \circ {\sim}seen \neq \perp\!\!\perp$ **do**
      **begin**
        choose node $b$ such that $a \circ \top\!\!\top \circ b \subseteq a \circ G \circ {\sim}seen$
      ; { $b \circ {\sim}seen = b$ }
        $dfs(b)$
      **end**
    { $a \circ seen = a \wedge D.a.seen = D.a.s0 \wedge a \circ G \circ {\sim}seen = \perp\!\!\perp$ }
    { $seen = D.a.s0$ }
```

Note the precondition $seen = s0$ and the postcondition $seen = D.a.s0$. The introduction of the auxiliary variable $s0$ in this precondition-postcondition pair is a familiar pointwise mechanism for expressing the relation between the input value, $s0$, of $seen$ and its output value.

The claim is that $a \circ seen = a$ is an invariant property and $D.a$ is an invariant value of the **while** statement. In point-free terms, their combination is an intersection of relations. The property $a \circ seen = a$ is represented by the coreflexive relation $q.a$ where, for all $s0$ and s,

(11.14) $s' [\![q.a]\!] s \;\equiv\; a \circ s = a \wedge s' = s$.

The invariance of the value $D.a$ is expressed by the relation $(D.a)^{\cup} \circ D.a$ and their conjunction is the relation

$$q.a \circ \top\!\!\top \cap (D.a)^{\cup} \circ D.a \ .$$

(Equivalently, this is $q.a \circ (D.a)^{\cup} \circ D.a$. However, expressing it as an intersection allows a simple decomposition of the proof obligations.) The annotation asserts that both relations are truthified by the initial assignment $S.a$ and maintained by the **while** statement $W.D.a$. On termination, the combination of the invariant and termination condition imply that the final value of *seen* is the result of applying the function $D.a$ to its initial value. Formally, we have:

$$
\begin{array}{ll}
 & \langle \mu d :: W.d \dot{\circ} S \dot{\circ} pre \rangle \;\dot{\subseteq}\; D \\
\Leftarrow & \{ \quad \text{fixed-point induction} \quad \} \\
 & W.D \dot{\circ} S \dot{\circ} pre \;\dot{\subseteq}\; D \\
= & \{ \quad \text{definition of pointwise operators} \quad \} \\
 & \langle \forall a :: W.D.a \circ S.a \circ pre.a \subseteq D.a \rangle \quad .
\end{array}
$$

Now, for all a,

$$
\begin{array}{ll}
 & W.D.a \circ S.a \circ pre.a \subseteq D.a \\
\Leftarrow & \{ \quad \text{prelude to introducing invariant} \\
 & \quad\;\; D.a \text{ is a function, so } D.a \circ (D.a)^{\cup} \subseteq I \quad \} \\
 & W.D.a \circ S.a \circ pre.a \subseteq D.a \circ (D.a)^{\cup} \circ D.a \\
\Leftarrow & \{ \quad \text{introduce invariant } q.a \circ \top\!\!\top \cap (D.a)^{\cup} \circ D.a \\
 & \quad\;\; \text{monotonicity of composition} \quad \} \\
 & \quad S.a \circ pre.a \subseteq q.a \circ \top\!\!\top \\
 & \wedge \quad S.a \circ pre.a \subseteq (D.a)^{\cup} \circ D.a \\
 & \wedge \quad W.D.a \circ q.a \subseteq D.a \\
\Leftarrow & \{ \quad W.D.a = t.a \circ (\langle \cup b :: D.b \circ C.b.a \rangle \circ {\sim}t.a)^{*} \\
 & \quad\;\; \text{monotonicity of composition} \quad \} \\
 & \quad S.a \circ pre.a \subseteq q.a \circ \top\!\!\top \\
 & \wedge \quad S.a \circ pre.a \subseteq (D.a)^{\cup} \circ D.a \\
 & \wedge \quad (\langle \cup b :: D.b \circ C.b.a \rangle \circ {\sim}t.a)^{*} \circ q.a \subseteq q.a \circ \top\!\!\top \\
 & \wedge \quad (\langle \cup b :: D.b \circ C.b.a \rangle \circ {\sim}t.a)^{*} \subseteq (D.a)^{\cup} \circ D.a \\
 & \wedge \quad t.a \circ q.a \circ (D.a)^{\cup} \circ D.a \subseteq D.a \quad .
\end{array}
$$

Noting that

$$(\langle\cup b :: D.b\circ C.b.a\rangle \circ \sim t.a)^* \subseteq (D.a)^\cup \circ D.a$$

$\Leftarrow$ { fixed-point induction }

$$I \subseteq (D.a)^\cup \circ D.a$$

$$\wedge \quad (D.a)^\cup \circ D.a \circ (\langle\cup b :: D.b\circ C.b.a\rangle \circ \sim t.a) \subseteq (D.a)^\cup \circ D.a$$

$\Leftarrow$ { first conjunct: $D.a$ is a total function;
second conjunct: monotonicity, distributivity }

$$\langle\forall b :: D.a \circ D.b \circ C.b.a \circ \sim t.a \subseteq D.a\rangle \ ,$$

we have derived from the formal semantics of $DFS.a$ five verification conditions. Two establish $q.a$ as a postcondition, one of $S.a$:

$$\langle\forall a :: S.a\circ pre.a \subseteq q.a\circ\top\!\top\rangle$$

and one of the **while** statement:

$$\langle\forall a :: (\langle\cup b :: D.b\circ C.b.a\rangle \circ \sim t.a)^*\circ q.a \subseteq q.a\circ\top\!\top\rangle \ .$$

Three verification conditions are properties of D : the verification condition for the initial assignment:

$$\langle\forall a :: S.a\circ pre.a \subseteq (D.a)^\cup\circ D.a\rangle \ ,$$

the verification condition for the body of the loop:

$$\langle\forall a,b :: D.a \circ D.b \circ C.b.a \circ \sim t.a \subseteq D.a\rangle \ ,$$

and the verification condition for termination of the loop:

$$\langle\forall a :: t.a\circ q.a\circ (D.a)^\cup\circ D.a \subseteq D.a\rangle \ .$$

Recalling (11.14) —the definition of $q.a$— it is obvious that the first property is valid. The second property is less obvious but involves a straightforward application of the fusion theorem and the property that $s\subseteq D.b.s$ for all s; we omit the details. We complete the proof by translating the final three point-free properties of relations into pointwise boolean conditions relating successive states of the program variable *seen*. This is done in lemmas 11.15, 11.16 and 11.17 below.
□

Lemma 11.15 Suppose a is a node and s is a coreflexive representing a set of nodes. Then

$$s\circ a = \perp\!\!\perp \ \Rightarrow\ D.a.s \ =\ D.a.(s\cup a) \ .$$

Proof

$$
\begin{array}{ll}
& D.a.s \;=\; D.a.(s{\cup}a) \\
= & \quad \{ \quad \text{anti-symmetry} \quad \} \\
& D.a.s \subseteq D.a.(s{\cup}a) \wedge D.a.(s{\cup}a) \subseteq D.a.s \\
= & \quad \{ \quad \text{definition of } D \text{ (see (11.2)) and distributivity} \quad \} \\
& \quad s \subseteq D.a.(s{\cup}a) \\
& \wedge \quad (a\circ(G\circ{\sim}s)^{*})_{>} \subseteq D.a.(s{\cup}a) \\
& \wedge \quad s\cup a \subseteq D.a.s \\
& \wedge \quad (a\circ(G\circ{\sim}(s\cup a))^{*})_{>} \subseteq D.a.s \quad .
\end{array}
$$

With the exception of the second, it is easily checked that each of these conjuncts is true. The second conjunct is where the condition $s{\circ}a = \perp\!\!\perp$ is needed:

$$
\begin{array}{ll}
& (a\circ(G\circ{\sim}s)^{*})_{>} \subseteq D.a.(s{\cup}a) \\
\Leftarrow & \quad \{ \quad \text{definition of } D\text{, distributivity} \quad \} \\
& (a\circ(G\circ{\sim}s)^{*})_{>} \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
\Leftarrow & \quad \{ \quad \text{fixed-point fusion} \quad \} \\
& \quad a \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
& \wedge \quad ((a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>}\circ G\circ{\sim}s)_{>} \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
= & \quad \{ \quad \text{first conjunct is true since } I\subseteq(G\circ{\sim}s\circ{\sim}a)^{*}; \\
& \qquad\quad \text{domains} \quad \} \\
& (a\circ(G\circ{\sim}s\circ{\sim}a)^{*}\circ G\circ{\sim}s)_{>} \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
= & \quad \{ \quad \text{assumption: } s{\circ}a = \perp\!\!\perp\text{, hence } {\sim}s \;=\; {\sim}s\circ{\sim}a \,\cup\, a \\
& \qquad\quad \text{distributivity} \quad \} \\
& \quad (a\circ(G\circ{\sim}s\circ{\sim}a)^{*}\circ G\circ a)_{>} \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
& \wedge \quad (a\circ(G\circ{\sim}s\circ{\sim}a)^{*}\circ G\circ{\sim}s\circ{\sim}a)_{>} \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
= & \quad \{ \quad \text{first conjunct: } a \text{ is a node, so } (a{\circ}R{\circ}a)_{>}\subseteq a\text{, for all } R, \\
& \qquad\quad a \subseteq (a\circ(G\circ{\sim}s\circ{\sim}a)^{*})_{>} \\
& \qquad\quad \text{second conjunct: definition of } {}^{*} \quad \} \\
& \text{true} \quad .
\end{array}
$$

□

Lemma 11.16 Suppose a and b are nodes and s is a coreflexive. Then

$$s{\circ}b = \perp\!\!\perp \wedge a{\circ}G{\circ}b = a{\circ}\top\!\!\top{\circ}b \Rightarrow D.a.(D.b.s) \subseteq D.a.s \quad .$$

Proof We have:

$$\begin{array}{ll} & D.a.(D.b.s) \subseteq D.a.s \\ = & \{ \quad \text{definition of } D \text{ and set union} \quad \} \\ & D.b.s \subseteq D.a.s \;\wedge\; (a\circ(G\circ{\sim}(D.b.s))^{*})_{>} \subseteq D.a.s \quad . \end{array}$$

We consider the conjuncts in order. First,

$$\begin{array}{ll} & D.b.s \\ = & \{ \quad \text{definition} \quad \} \\ & s \cup (b\circ(G\circ{\sim}s)^{*})_{>} \\ = & \{ \quad \text{assumption: } a{\circ}G{\circ}b = a{\circ}\top\!\!\top{\circ}b \text{, so } (a{\circ}G{\circ}b)_{>} = b \text{, domains} \quad \} \\ & s \cup (a\circ G\circ b\circ(G\circ{\sim}s)^{*})_{>} \\ \subseteq & \{ \quad \text{assumption: } s{\circ}b = \perp\!\!\perp \text{, so } b \subseteq {\sim}s \quad \} \\ & s \cup (a\circ G\circ{\sim}s\circ(G\circ{\sim}s)^{*})_{>} \\ \subseteq & \{ \quad [\; R\circ R^{*} \subseteq R^{*} \;] \text{ with } R := G\circ{\sim}s \quad \} \\ & s \cup (a\circ(G\circ{\sim}s)^{*})_{>} \\ = & \{ \quad \text{definition} \quad \} \\ & D.a.s \quad . \end{array}$$

Second,

$$\begin{array}{ll} & (a\circ(G\circ{\sim}(D.b.s))^{*})_{>} \subseteq D.a.s \\ \Leftarrow & \{ \quad \text{definition, monotonicity} \quad \} \\ & {\sim}(D.b.s) \subseteq {\sim}s \\ = & \{ \quad \text{anti-monotonicity of complementation} \quad \} \\ & s \subseteq D.b.s \\ = & \{ \quad \text{definition, set union} \quad \} \\ & \mathsf{true} \quad . \end{array}$$

□

Lemma 11.17 Suppose a is a node and s and $s0$ are coreflexives representing sets of nodes. Then

$$a\circ G\circ{\sim}s = \perp\!\!\perp \;\wedge\; a{\circ}s = a \;\wedge\; D.a.s = D.a.s0 \quad\Rightarrow\quad s = D.a.s0 \ \ .$$

Proof

$$\begin{array}{ll}
 & D.a.s0 \\
= & \{ \quad \text{assumption: } D.a.s = D.a.s0 \quad \} \\
 & D.a.s \\
= & \{ \quad \text{definition} \quad \} \\
 & s \cup (a\circ(G\circ{\sim}s)^*)_> \\
= & \{ \quad a\circ(G\circ{\sim}s)^* \;=\; a \,\cup\, a\circ G\circ{\sim}s\circ(G\circ{\sim}s)^* \\
 & \quad\ \ \text{assumption: } a\circ G\circ{\sim}s = \perp\!\!\perp \quad \} \\
 & s \cup a_> \\
= & \{ \quad a \text{ is a node, so } a_> = a \\
 & \quad\ \ \text{assumption: } a{\circ}s = a \quad \} \\
 & s \ \ .
\end{array}$$

□

11.4 Properties of the Inner Loop

A consequence of theorem 11.13 is that GE is an invariant of the inner loop and $dfs(a)$ satisfies the relation GT. Although requiring formal proof, and used extensively below, these are obvious properties. This section is about less obvious properties.

From the definition of $D.a.s$, it is clear that no new frontier edges are added by a call of $dfs(a)$. This is made precise in lemma 11.18. An important corollary is the claim that (11.1) is an invariant of the outer loop.

Lemma 11.18 A depth-first search reduces the set of frontier edges. That is, for all a, s and G,

$$D.a.s\circ G\circ{\sim}(D.a.s) \;\subseteq\; s\circ G\circ{\sim}s \ \ .$$

Proof

$$D.a.s\circ G\circ{\sim}(D.a.s) \;\subseteq\; s\circ G\circ{\sim}s$$

$=$ { definition, complements and set union }

$$s\circ G\circ{\sim}(D.a.s) \;\subseteq\; s\circ G\circ{\sim}s$$

$$\wedge\quad (a\circ(G\circ{\sim}s)^{*})_{>}\circ G\circ{\sim}s\circ(a\circ(G\circ{\sim}s)^{*})_{>\bullet} \;\subseteq\; s\circ G\circ{\sim}s$$

$\Leftarrow$ { $s\subseteq D.a.s$, so ${\sim}(D.a.s)\subseteq{\sim}s$, monotonicity;

$\perp\!\!\perp \subseteq s\circ G\circ{\sim}s$, transitivity }

$$(a\circ(G\circ{\sim}s)^{*})_{>}\circ G\circ{\sim}s\circ(a\circ(G\circ{\sim}s)^{*})_{>\bullet} \;\subseteq\; \perp\!\!\perp$$

$\Leftarrow$ { domains }

$$a\circ(G\circ{\sim}s)^{*}\circ G\circ{\sim}s\circ(a\circ(G\circ{\sim}s)^{*})_{>\bullet} \;\subseteq\; \perp\!\!\perp$$

$=$ { $(G\circ{\sim}s)^{*}\circ G\circ{\sim}s \;\subseteq\; (G\circ{\sim}s)^{*}$

definition of complemented right domain }

true .

□

It is useful to also observe that the nodes “newly” reached by a call of the procedure are connected by paths between previously unreached nodes. That is, we introduce the relation NR of type SetOfNode∼SetOfNode and defined by

(11.19) $s1[\![NR]\!]s0 \;\equiv\; s1\supseteq s0 \;\wedge\; s1\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$

and show that it is an invariant relation of the procedure dfs .

Lemma 11.20 The relation NR is an invariant of the inner loop.

Proof The verification condition is: for all $s0$, $s1$ and $s2$,

$$s2\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$$

$$\Leftarrow\quad s1\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>} \;\wedge\; s2=D.b.s1 \;\wedge\; b\circ s1=\perp\!\!\perp \;\wedge\; s0\subseteq s1$$

This we prove as follows.

$$s2\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$$

$=$ { assumption: $s2=D.b.s1$ }

$$(s1\cup(b\circ(G\circ{\sim}s1)^{*})_{>})\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$$

$\Leftarrow$ { distributivity,

assumption: $s1\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$ }

$$(b\circ(G\circ{\sim}s1)^{*})_{>}\circ{\sim}s0 \;\subseteq\; ({\sim}s0\circ(G\circ{\sim}s0)^{*})_{>}$$

$\Leftarrow$ { assumption: $b\circ s1=\perp\!\!\perp$, i.e. $b\subseteq{\sim}s1$, monotonicity }

$$(\sim s1\circ(G\circ\sim s1)^*)_> \circ \sim s0 \;\subseteq\; (\sim s0\circ(G\circ\sim s0)^*)_>$$

$= \quad \{ \quad s0\subseteq s1\text{ , i.e. } \sim s1\subseteq\sim s0 \text{ and monotonicity} \quad \}$

$$\sim s0\subseteq I$$

$= \quad \{ \quad \sim s0 \text{ is a coreflexive} \quad \}$

true .

□

11.5 Properties of the Outer Loop

Theorem 11.13 has important consequences for reasoning about depth-first search. Effectively, it says that the call of the procedure $dfs(a)$ is equal to the assignment

$$seen \;:=\; seen\cup(a\circ(G\circ\sim seen)^*)_> \;\; .$$

This section is about its consequences for the outer loop. We show, for example, that (11.1) is an invariant of the outer loop. (Recall that (11.1) is not an invariant of the inner loop.)

Corollary 11.21 The property (11.1) is an invariant of the outer loop.

Proof Clearly the property

(11.22) $seen\circ G\circ\sim seen = \bot\!\!\bot$

is truthified by the initial assignment $seen := \bot\!\!\bot$. Since $\bot\!\!\bot$ is the least element in the subset ordering of relations, and each call of $dfs(a)$ has the effect of assigning $D.a.seen$ to $seen$, it follows from lemma 11.18 that (11.22) is an invariant of the outer loop. Consequently, by lemma 10.1 (with $R,p := G,seen$), property (11.1) is also an invariant of the outer loop.

□

The relation NR only establishes an upper bound on newly reached nodes. In the outer loop, the relation can be sharpened from a containment to an equality. This is achieved by exploiting the property (11.1).

Lemma 11.23

$$\langle\forall s1,s0$$
$$: \quad s1\,[\![GE\cap NR]\!]\,s0 \;\wedge\; (s1\circ G^*)_> = s1$$
$$: \quad s1 \;=\; s0\cup(s1\circ\sim s0\circ(G\circ\sim s0)^*)_>$$
$$\rangle$$

In words, each iteration of the outer loop "closes" $seen$ under reachability by edges connecting previously unseen nodes.

Proof

$$\begin{array}{ll}
 & s1 \;=\; s0 \cup (s1 \circ {\sim}s0 \circ (G \circ {\sim}s0)^{*})_{>} \\
= & \{ \quad \text{antisymmetry of the subset relation} \\
 & \quad\;\; \text{assumption: } s1 \,[\![NR]\!]\, s0 \text{ and shunting (2.27)} \quad \} \\
 & s1 \supseteq s0 \cup (s1 \circ {\sim}s0 \circ (G \circ {\sim}s0)^{*})_{>} \\
= & \{ \quad \text{assumption: } s1 \,[\![GE]\!]\, s0 \text{ (i.e. } s1 \supseteq s0 \text{)} \quad \} \\
 & s1 \supseteq (s1 \circ {\sim}s0 \circ (G \circ {\sim}s0)^{*})_{>} \\
= & \{ \quad \text{assumption: } (s1 \circ G^{*})_{>} = s1 \quad \} \\
 & (s1 \circ G^{*})_{>} \supseteq (s1 \circ {\sim}s0 \circ (G \circ {\sim}s0)^{*})_{>} \\
= & \{ \quad {\sim}s0 \subseteq I \text{, monotonicity} \quad \} \\
 & \mathsf{true} \;\; .
\end{array}$$

□

11.6 Strongly Connected Components

We conclude this section with the theorem on strongly connected components announced earlier: each strongly connected component is added in its entirety within a single call of dfs.

Theorem 11.24 Suppose a and b are nodes and p is the coreflexive representing the strongly connected component containing b. That is, suppose $p = (b \circ \mathsf{equiv}.G)_{>}$. Then, if $dfs(a)$ is executed with precondition $p \circ seen = \perp\!\!\perp$, and it terminates with postcondition $b \circ seen = b$, it will terminate with postcondition $p \circ seen = p$.

Proof Suppose $p = (b \circ \mathsf{equiv}.G)_{>}$ and suppose $s1$ and $s0$ satisfy the relation $GT \cap NR$; furthermore, suppose $p \circ s0 = \perp\!\!\perp$ (the assumed precondition for executing $dfs(a)$) and $b \circ s1 = b$ (the assumed postcondition of $dfs(a)$). We prove that $p \circ s1 = p$.

$$\begin{array}{ll}
 & p \circ s1 = p \\
= & \{ \quad \text{coreflexives} \quad \} \\
 & p \subseteq s1 \\
= & \{ \quad \text{assume: } s1 \,[\![GT \cap NR]\!]\, s0 \text{, lemma 11.23} \quad \}
\end{array}$$

$$p \subseteq s0 \cup (s1 \circ {\sim}s0 \circ (G \circ {\sim}s0)^*)_>$$
$\Leftarrow$ { assumption: $p{\circ}s0 = \bot\!\!\bot$, i.e. $p \subseteq {\sim}s0$); assumption: $b{\circ}s1 = b$ }
$$p \subseteq (b \circ p \circ (G{\circ}p)^*)_>$$
$=$ { absolute connectivity: lemma 9.19 }
$$p \subseteq (b \circ p \circ G^* \circ p)_>$$
$\Leftarrow$ { $b{\circ}p = b$; $\mathrm{equiv}.G \subseteq G^*$ }
$$p \subseteq (b \circ \mathrm{equiv}.G \circ p)_>$$
$\Leftarrow$ { $p \;=\; (b \circ \mathrm{equiv}.G)_>$, domains }
$$\mathrm{true} \quad .$$
□

Corollary 11.25 An invariant property of the inner loop is

(11.26) $\langle \forall p \;:\; scc.p \;\wedge\; p \neq (a \circ \mathrm{equiv}.G)_> \;:\; p{\circ}seen = \bot\!\!\bot \;\vee\; p{\circ}seen = p \rangle$

where $scc.p$ is the property that p is a coreflexive representing a strongly connected component of the graph G . An invariant property of the outer loop is

(11.27) $\langle \forall p \;:\; scc.p \;:\; p{\circ}seen = \bot\!\!\bot \;\vee\; p{\circ}seen = p \rangle$.

Proof The property (11.27) is clearly truthified by the initialisation $seen := \bot\!\!\bot$ in the outer loop. Then, assuming that (11.27) is a precondition of a call of $dfs(a)$, (11.26) remains true after the initialisation

$$seen \;:=\; seen \cup a \quad .$$

(It is at this point that it becomes clear why the case

$$p \;=\; (a \circ \mathrm{equiv}.G)_>$$

is excluded from the universal quantification.) The invariance of the property (11.26) under execution of $dfs(b)$ in the inner loop is then immediate from theorem 11.24.

In the outer loop, (11.27) is a consequence of (11.6). Specifically, suppose p is a coreflexive representing a strongly connected component of G . Suppose also that $p{\circ}seen \neq \bot\!\!\bot$. Then, with dummy a ranging over nodes of G , we have:

$$p{\circ}seen \neq \bot\!\!\bot$$
$=$ { theorem 9.24 }

$\langle\exists a : p = (a\circ \mathsf{equiv}.G)_< : \mathit{seen}\circ a = a\rangle$

$\Rightarrow \quad \{ \quad$ Leibniz $\quad \}$

$\langle\exists a : p = (a\circ \mathsf{equiv}.G)_< : (\mathit{seen}\circ a\circ \mathsf{equiv}.G)_> \;=\; (a\circ \mathsf{equiv}.G)_>\rangle$

$= \quad \{ \quad$ seen and a are coreflexives, so $\mathit{seen}\circ a = a\circ \mathit{seen}$ $\quad \}$

$\langle\exists a : p = (a\circ \mathsf{equiv}.G)_< : (a\circ \mathit{seen}\circ \mathsf{equiv}.G)_> \;=\; p\rangle$

$= \quad \{ \quad$ (11.6) and domains $\quad \}$

$\langle\exists a : p = (a\circ \mathsf{equiv}.G)_< : (a\circ \mathit{seen}\circ \mathsf{equiv}.G\circ \mathit{seen})_> \;=\; p\rangle$

$\Rightarrow \quad \{ \quad$ domains $[\; \mathit{seen} \supseteq (R\circ \mathit{seen})_> \;]$ with $R := a\circ \mathit{seen}\circ \mathsf{equiv}.G$ $\quad \}$

$\langle\exists a : p = (a\circ \mathsf{equiv}.G)_< : \mathit{seen} \supseteq p\rangle$

$= \quad \{ \quad$ coreflexives and theorem 9.24 $\quad \}$

$p\circ \mathit{seen} = p \quad .$

□

Corollary 11.25 anticipates the use of depth-first search in constructing strongly connected components. As representative element of each strongly connected component, one chooses the first node in the component that is "seen" by a depth-first search. Tarjan [Tar72], Sharir [Sha81] and Aho, Hopcroft and Ullman [AHU82] call the representative of a strongly connected component the "root" of the component, whilst Cormen, Leiserson and Rivest [CLR90, p.482] call it the "forefather" of the component. (In a later edition, Cormen, Leiserson, Rivest and Stein [CLRS09, p.619] have elided the explicit discussion of the "forefather" of the compenent; implicitly, they also call the representative the "root".) The problem is to identify which nodes are "roots". This problem is solved in section 13. The solution involves "timestamping" searches with both start and finish "times". The addition of finish "times" means that we have to extend the semantics of depth-first search to include the effect of adding an assignment statement at the end of each call of the procedure $\mathtt{dfs}$. This is the topic of the next section.

Fig. 11.3 documents the properties we have established of the procedure $\mathtt{dfs}(a)$.

Chapter 12

An Induction Theorem for Depth-First Search

The primary purpose of this section is to formulate a general rule for reasoning about different implementations of depth-first search. See theorem 12.5. Several choices are made in formulating the rule. In order to motivate the choices, we show how to sharpen the reachability property of depth-first search; we also establish a property that is the basis of a crucial classification of the edges following a depth-first search of a graph. Specifically, in section 12.3, we show that whenever a call of $\mathsf{dfs}(a)$ is executed, for some node a, the node a is reachable from all nodes from which the search has started but not finished; moreover, there are no edges in the graph from a node from which the search has finished to a node from which the search has not started.

Both these properties involve augmenting the implementation with a variable that records the set of nodes from which the search has finished. The form that the revised implementation takes anticipates the implementation of timestamps in section 13.

The generic implementation that we consider is a composition of three statements: an initial assignment, which we call S, a **while** statement, which we call W, and a final assignment statement, which we call F. (See fig. 12.4 for an example.) Accordingly, we need to revise the definition (11.9). The appropriate definition is as follows.

$$(12.1)\quad \mathsf{DFS} \;=\; \langle \mu d :: F \,\dot{\circ}\, W.d \,\dot{\circ}\, S \rangle$$

where

$$(12.2)\quad W.d.a \;=\; t.a \circ ((\langle \cup b :: d.b \circ C.b.a \rangle \circ \sim t.a)^{*} \;\; .$$

(The definition of W has not changed but is repeated here for convenience.)

Fig. 12.1 may help the reader to better understand the development. It shows the generic form of the procedure $\mathsf{dfs}(a)$ and the relevant documentation. The assertion $p.a$ is a precondition on the execution of the procedure $\mathsf{dfs}(a)$; note that $p.b$ is a precondition on the execution of $dfs(b)$. The assertion $q.a$ is a so-called *intermediate assertion.* Use of the induction theorem requires some creativity in the formulation of both p and q. The combination of the precondition p and the invariant relation R is the specification of the procedure; theorem 12.5 gives sufficient conditions that guarantee when an implementation meets the specification.

```
    {  $p.a$  }
    {  **Invariant Relation:**  $R$  }
    $S.a$
    {  $q.a$  }
;   {  **Invariant Relation:**  $R^*$
       **Invariant Property:**  $q.a$  }
    **while** $\sim t.a$ **do**
       **begin**
          choose node $b$ such that $C.b.a$
       ;  {  $C.b.a \circ \sim t.a \circ q.a$  }
          {  $p.b$  }
          $dfs(b)$
       **end**
;   {  $t.a \circ q.a$  }
    $F.a$
```

Figure 12.1: Documenting Depth-First Search Induction

12.1 Formal Statement and Proof

In the following, we assume that $p.a$, $q.a$, $\sim t.a$ and $C.b.a$ are coreflexives and R, $S.a$ and $F.a$ are homogeneous relations on the state space. In all the implementations we consider, $S.a$ and $F.a$ are assignment statements; that is, $S.a$ and $F.a$ are total endofunctions on the state space.

Previously, lemma 11.12 was used to establish an induction rule for $DFS \mathbin{\dot\circ} pre$ (where pre was the "obvious" precondition). Theorem 12.5 below replaces lemma 11.12. Unfortunately, the fusion theorem does not appear to be strong enough and we have been obliged to find a more specific proof technique based on the following two lemmas.

Lemma 12.3 For all relations T and coreflexives q,

$$T^* \circ q \;=\; (T{\circ}q)^* \circ q \;\Leftarrow\; T{\circ}q = q{\circ}T{\circ}q \;\;.$$

Proof

$$\begin{array}{ll}
 & T^* \circ q \;=\; (T{\circ}q)^* \circ q \\
= & \{ \quad q \subseteq I\text{, monotonicity and anti-symmetry} \quad \} \\
 & T^* \circ q \;\subseteq\; (T{\circ}q)^* \circ q \\
\Leftarrow & \{ \quad T^* \circ q \text{ is a least fixed point, induction} \quad \} \\
 & q \cup T \circ (T{\circ}q)^* \circ q \;\subseteq\; (T{\circ}q)^* \circ q \\
\Leftarrow & \{ \quad (T{\circ}q)^* \;=\; I \cup T \circ q \circ (T{\circ}q)^* \\
 & \quad\;\; \text{distributivity and monotonicity} \quad \} \\
 & (T{\circ}q)^* \circ q \;\subseteq\; q \circ (T{\circ}q)^* \circ q \\
= & \{ \quad q{\circ}q = q \text{ (applied twice) and mirror rule} \quad \} \\
 & (T{\circ}q)^* \circ q \;\subseteq\; (q{\circ}T{\circ}q)^* \circ q \\
\Leftarrow & \{ \quad \text{Leibniz} \quad \} \\
 & T{\circ}q \;=\; q{\circ}T{\circ}q \;\;.
\end{array}$$

□

Lemma 12.4 For all relations R and coreflexives p and q,

$$R/p \circ q \;=\; R{\circ}q \;\Leftarrow\; q = p{\circ}q$$

Proof

$$\begin{array}{ll}
 & R/p \circ q \\
\subseteq & \{ \quad \text{assume: } q = p{\circ}q\text{, cancellation of factors} \quad \} \\
 & R{\circ}q \\
\subseteq & \{ \quad p \subseteq I\text{, (anti-)monotonicity of factors} \quad \} \\
 & R/p \circ q \;\;.
\end{array}$$

The lemma follows by anti-symmetry.

□

We are now in a position to formulate a theorem for reasoning about the generic form of depth-first search expressed by (12.1).

Theorem 12.5 (Depth-First Search Induction) Suppose R is a relation on the state space of depth-first search, and $p.a$ and $q.a$ are coreflexives representing subsets of the state space. Then

$$\langle\forall a :: DFS.a\circ p.a \subseteq R\rangle \Leftarrow (12.6)\wedge(12.7)\wedge(12.8)\wedge(12.9)$$

where the premises (12.6), (12.7), (12.8) and (12.9) are defined as follows.

(12.6) $S\dot{\circ}p = q\dot{\circ}S\dot{\circ}p$.

This specifies the intermediate assertion q: when S is executed with precondition p, q is a valid postcondition. Equivalently, $q.a$ is at least the left domain of $S.a\circ p.a$.

(12.7) $\langle\forall a,b :: C.b.a\circ{\sim}t.a\circ q.a \subseteq p.b\rangle$.

This is the property that if the (inner) loop body is executed with precondition $q.a$ then the call of $dfs(b)$ will be executed with precondition $p.b$.

(12.8) $K.R\dot{\circ}q = q\dot{\circ}K.R\dot{\circ}q$.

(Recall that K denotes the constant combinator.) This asserts that property q is "maintained by" relation R.

(12.9) $F\dot{\circ}t\dot{\circ}K.R^*\dot{\circ}S\dot{\circ}p \;\dot{\subseteq}\; K.R$.

This asserts that executing F after S with precondition p maintains the relation R.

Proof We begin by proving that, assuming (12.6), (12.7) and (12.8),

(12.10) $DFS\dot{\circ}p \;\dot{\subseteq}\; K.R \quad\Leftarrow\quad F\dot{\circ}W.(K.R)\dot{\circ}S\dot{\circ}p \;\dot{\subseteq}\; K.R$.

We give a pointwise calculation (primarily because giving a point-free calculation involves introducing additional notation that is used just once). Apart from the highlighted step, the calculation below is straightforward.

$$
\begin{array}{ll}
 & DFS\dot{\circ}p \;\dot{\subseteq}\; K.R \\
= & \{ \quad \text{definitions of pointwise operators} \quad \} \\
 & \langle\forall a :: DFS.a\circ p.a \subseteq R\rangle \\
= & \{ \quad \text{factors} \quad \} \\
 & \langle\forall a :: DFS.a \subseteq R/p.a\rangle \\
= & \{ \quad \text{definition of pointwise ordering} \quad \} \\
 & DFS \;\dot{\subseteq}\; \langle a :: R/p.a\rangle
\end{array}
$$

$\Leftarrow \qquad \{ \qquad (12.1)$ and fixed point induction $\}$

$\quad F \mathbin{\dot{\circ}} W.\langle a :: R/p.a\rangle \mathbin{\dot{\circ}} S \quad \dot{\subseteq} \quad \langle a :: R/p.a\rangle$

$= \qquad \{ \qquad$ definition of pointwise orderings, definition (12.2) of $W \quad \}$

$\quad \langle \forall a :: F.a \circ t.a \circ (\langle \cup b :: R/p.b \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \subseteq R/p.a\rangle$

$= \qquad \{ \qquad$ factors $\}$

$\quad \langle \forall a :: F.a \circ t.a \circ (\langle \cup b :: R/p.b \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \circ p.a \subseteq R\rangle$

$= \qquad \{ \qquad$ **see below** $\}$

$\quad \langle \forall a :: F.a \circ t.a \circ (\langle \cup b :: R \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \circ p.a \subseteq R\rangle$

$= \qquad \{ \qquad$ definition of $W \quad \}$

$\quad \langle \forall a :: F.a \circ W.(K.R).a \circ S.a \circ p.a \ \subseteq\ R\rangle$

$= \qquad \{ \qquad$ definitions of pointwise operators $\}$

$\quad F \mathbin{\dot{\circ}} W.(K.R) \mathbin{\dot{\circ}} S \mathbin{\dot{\circ}} p \ \dot{\subseteq}\ K.R \ .$

The highlighted step above was to replace the term $R/p.b$ by R. This apparently innocuous step is the most difficult step of all. In anticipation of later steps, we introduce the coreflexive $q.a$ into the calculation as follows:

$\quad R/p.b \circ C.b.a \circ {\sim}t.a \circ q.a$

$= \qquad \{ \qquad$ assumption (12.7): $C.b.a \circ {\sim}t.a \circ q.a \subseteq p.b$

$\qquad\qquad\quad$ lemma 12.4 with $p,q \ := \ p.b \ , \ C.b.a \circ {\sim}t.a \circ q.a \quad \}$

$\quad R \circ C.b.a \circ {\sim}t.a \circ q.a$

$= \qquad \{ \qquad$ coreflexives commute $\}$

$\quad R \circ q.a \circ C.b.a \circ {\sim}t.a$

$= \qquad \{ \qquad$ assumption (12.8): $R \circ q.a \ = \ q.a \circ R \circ q.a \quad \}$

$\quad q.a \circ R \circ q.a \circ C.b.a \circ {\sim}t.a$

$= \qquad \{ \qquad$ reverse first two steps $\}$

$\quad q.a \circ R/p.b \circ C.b.a \circ {\sim}t.a \circ q.a \ .$

In summary, assuming (12.7) and (12.8),

(12.11) $R/p.b \circ C.b.a \circ {\sim}t.a \circ q.a \ = \ q.a \circ R/p.b \circ C.b.a \circ {\sim}t.a \circ q.a \ .$

Now, for all a,

$(\langle\cup b :: R/p.b \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \circ p.a$

$=\qquad\{\qquad$ assumption (12.6): $S.a \circ p.a = q.a \circ S.a \circ p.a \quad\}$

$(\langle\cup b :: R/p.b \circ C.b.a\rangle \circ {\sim}t.a)^* \circ q.a \circ S.a \circ p.a$

$=\qquad\{\qquad$ distributivity $\quad\}$

$\langle\cup b :: R/p.b \circ C.b.a \circ {\sim}t.a\rangle^* \circ q.a \circ S.a \circ p.a$

$=\qquad\{\qquad$ lemma 12.3 with $T := \langle\cup b :: R/p.b \circ C.b.a \circ {\sim}t.a\rangle$ and $q := q.a$

(applicable because of (12.11)), distributivity $\quad\}$

$\langle\cup b :: R/p.b \circ C.b.a \circ {\sim}t.a \circ q.a\rangle^* \circ q.a \circ S.a \circ p.a$

$=\qquad\{\qquad$ assumption (12.7): $C.b.a \circ {\sim}t.a \circ q.a \subseteq p.b$, lemma 12.4 $\quad\}$

$\langle\cup b :: R \circ C.b.a \circ {\sim}t.a \circ q.a\rangle^* \circ q.a \circ S.a \circ p.a$

$=\qquad\{\qquad$ assumption (12.8): $R \circ q.a = q.a \circ R \circ q.a$, distributivity

lemma 12.3 with $T := \langle\cup b :: R \circ C.b.a \circ {\sim}t.a\rangle$ and $q := q.a \quad\}$

$\langle\cup b :: R \circ C.b.a \circ {\sim}t.a\rangle^* \circ q.a \circ S.a \circ p.a$

$=\qquad\{\qquad$ assumption (12.6): $S.a \circ p.a = q.a \circ S.a \circ p.a \quad\}$

$(\langle\cup b :: R \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \circ p.a$.

This verifies the postponed step (the step marked "see below") in the initial calculation and concludes the proof of (12.10).

We now apply (12.10). Assume (12.6), (12.7) and (12.8). Then

$DFS \,\dot{\circ}\, p \;\dot{\subseteq}\; K.R$

$\Leftarrow\qquad\{\qquad$ (12.10) and assumptions $\quad\}$

$F \,\dot{\circ}\, W.(K.R) \,\dot{\circ}\, S \,\dot{\circ}\, p \;\dot{\subseteq}\; K.R$

$=\qquad\{\qquad$ definition of pointwise operators $\quad\}$

$\langle\forall a :: F.a \circ t.a \circ (\langle\cup b :: R \circ C.b.a\rangle \circ {\sim}t.a)^* \circ S.a \circ p.a \subseteq R\rangle$

$\Leftarrow\qquad\{\qquad$ $C.b.a$ and ${\sim}t.a$ are coreflexives, monotonicity $\quad\}$

$\langle\forall a :: F.a \circ t.a \circ \langle\cup b :: R\rangle^* \circ S.a \circ p.a \subseteq R\rangle$

$\Leftarrow\qquad\{\qquad$ $\langle\cup b :: R\rangle \subseteq R \quad\}$

$\langle\forall a :: F.a \circ t.a \circ R^* \circ S.a \circ p.a \subseteq R\rangle$

$=\qquad\{\qquad$ definition of pointwise operators $\quad\}$

$F \,\dot{\circ}\, t \,\dot{\circ}\, K.R^* \,\dot{\circ}\, S \,\dot{\circ}\, p \;\dot{\subseteq}\; K.R$.

□

12.2 Verification Conditions

In order to apply theorem 12.5, four *verification conditions* must be met: one for the initial assignment S (that it truthifies q), one for the precondition for execution of $\mathsf{dfs}(b)$ (that it is truthified by the condition for choosing b), one for the intermediate assertion q (that it is invariant under R), and finally one for the combination of the initial assignment S and the final assignment F (that they maintain the invariant relation R, assuming precondition p and condition for terminating the loop t).

Fig. 12.2 summarises theorem 12.5. The symbol "σ" denotes the current state; the notation $p(\sigma,a)$ is used rather than the point-free $p.a$ to signify the dependence of precondition p on the state, and the fact that $p(\sigma,a)$ is a syntactic expression. Two ghost variables σ_0 and σ_1 help to relate the state at (respectively) the start of execution of the procedure and at the start of execution of the **while** statement to its value at later points during the execution.

The initial assignment statement gives rise to a verification condition. Applying the assignment axiom, this is

(12.12) $\langle\forall\,\sigma,a :: q(S(\sigma,a)\,,a) \Leftarrow p(\sigma,a)\rangle$.

The two instances of consecutive assertions each give rise to a verification condition:

(12.13) $\langle\forall\,\sigma,a \;::\; p(\sigma,b) \;\Leftarrow\; C(\sigma,b,a)\wedge{\sim}t(\sigma,a)\wedge q(\sigma,a)\rangle$, and

(12.14) $\langle\forall\,\sigma,\sigma_1,a \;::\; q(\sigma,a) \;\Leftarrow\; t(\sigma,a)\wedge q(\sigma_1,a)\wedge \sigma[\![R^*]\!]\,\sigma_1\rangle$.

The final assignment statement also gives rise to a verification condition.

(12.15) $\langle\forall\,\sigma,\sigma_0,a \;::\; F(\sigma,a)[\![R]\!]\sigma_0 \;\Leftarrow\; t(\sigma,a)\wedge \sigma[\![R^*]\!]\,S(\sigma_0,a)\wedge p(\sigma_0,a)\rangle$.

The sequence

$$\begin{array}{l}\{\;\; p(\sigma,b)\wedge\sigma[\![R^*]\!]\,\sigma_1 \;\;\}\\ \mathsf{dfs}(b)\\ \{\;\; \sigma[\![R^*]\!]\,\sigma_1 \;\;\}\end{array}$$

in the loop body is the *induction hypothesis*: fixed-point induction enables the assumption that this is valid.

Sometimes (12.9) is used in combination with (12.6) and (12.8) in order to strengthen the precondition on F in (12.15) from $t(\sigma,a)$ to $t(\sigma,a)\wedge q(\sigma,a)$ as shown in fig. 12.2. Formally, this is based on the theorem that, in the context of (12.6) and (12.8),

$$F \mathbin{\dot\circ} t \mathbin{\dot\circ} K.R^* \mathbin{\dot\circ} S \mathbin{\dot\circ} p \;=\; F \mathbin{\dot\circ} t \mathbin{\dot\circ} q \mathbin{\dot\circ} K.R^* \mathbin{\dot\circ} q \mathbin{\dot\circ} S \mathbin{\dot\circ} p \;\;,$$

```
    { $p(\sigma,a) \wedge \sigma = \sigma_0$ }
    { **Invariant Relation** $R$ }
    $\sigma := S(\sigma,a)$
    { $q(\sigma,a) \wedge \sigma = \sigma_1$ }
;   { **Invariant Relation** $R^*$ }
    **while** $\sim t(\sigma,a)$ **do**
       **begin**
          choose node $b$ such that $C(\sigma,b,a)$
       ; { $C(\sigma,b,a) \wedge \sim t(\sigma,a) \wedge q(\sigma,a)$ }
          { $p(\sigma,b) \wedge \sigma [\![R^*]\!] \sigma_1$ }
          $dfs(b)$
          { $\sigma [\![R^*]\!] \sigma_1$ }
       **end**
    { $t(\sigma,a) \wedge q(\sigma_1,a) \wedge \sigma [\![R^*]\!] \sigma_1$ }
;   { $t(\sigma,a) \wedge q(\sigma,a)$ }
    $\sigma := F(\sigma,a)$
    { $\sigma [\![R]\!] \sigma_0$ }
```

Figure 12.2: Summary of the Induction Theorem

and hence property (12.9) is equivalent to

(12.16) $F \dot{\circ} t \dot{\circ} q \dot{\circ} K.R^* \dot{\circ} q \dot{\circ} S \dot{\circ} p \;\; \dot{\subseteq} \;\; K.R$.

For completeness, we give the proof:

$$F \dot{\circ} t \dot{\circ} K.R^* \dot{\circ} S \dot{\circ} p \;=\; F \dot{\circ} t \dot{\circ} q \dot{\circ} K.R^* \dot{\circ} q \dot{\circ} S \dot{\circ} p$$

$\Leftarrow$ { assumption: (12.6) and $q = q \dot{\circ} q$ }

$$K.R^* \dot{\circ} q \;=\; q \dot{\circ} K.R^* \dot{\circ} q$$

$=$ { $q \subseteq I$, domains }

$$(K.R^* \dot{\circ} q)_< \;\subseteq\; q$$

$\Leftarrow$ { fixed-point fusion and distributivity }

$$q \,\dot{\cup}\, (K.R\dot{\circ}q)_{<} \;\subseteq\; q$$

$= \qquad \{ \quad$ distributivity, assumption: (12.8) and domains $\quad\}$

$\quad\mathsf{true}$.

Clearly the verification of (12.9) (or its equivalent (12.16)) is the most complicated task because it involves considering the combined effect of the initial assignment S and the final assignment F. Sometimes this is unavoidable but often it can be decomposed and/or simplified using properties of the relation R.

Firstly, R is typically transitive so that R^* equals $I{\cup}R$. In some cases, R is both transitive and reflexive so that R^* equals R. This allows a simplification of (12.9), albeit with the additional obligation to show that R is indeed transitive (and possibly reflexive), whereby R^* is simplified to $I{\cup}R$ (or just R if R is reflexive).

Secondly, R is typically the intersection of several relations. By definition, $R{\cap}T$ is transitive if

$$(R{\cap}T){\circ}(R{\cap}T) \;\subseteq\; R \;\wedge\; (R{\cap}T){\circ}(R{\cap}T) \;\subseteq\; T \;\; .$$

This proof obligation is often simpler than it looks because one or both of R and T is transitive. Some relations, such as set inclusion, are obviously transitive and reflexive. What we have called “property invariants” and “value invariants” (see section 6.8.4) are also transitive and reflexive.

Assuming that $R{\cap}T$ is transitive, our proof obligation takes the form

$$F \dot{\circ} t \dot{\circ} K.(I{\cup}(R{\cap}T)) \dot{\circ} S \dot{\circ} p \;\dot{\subseteq}\; K.(R{\cap}T) \;\; .$$

Letting $\alpha.R$ be $F \dot{\circ} t \dot{\circ} K.(I{\cup}R) \dot{\circ} S \dot{\circ} p$, this follows from the conjunction of

$$\alpha.R \,\dot{\cap}\, \alpha.T \;\dot{\subseteq}\; K.R$$

and

$$\alpha.R \,\dot{\cap}\, \alpha.T \;\dot{\subseteq}\; K.T \;\; .$$

Sometimes it suffices to show that

$$\alpha.R \;\dot{\subseteq}\; K.R \;\;\wedge\;\; \alpha.T \;\dot{\subseteq}\; K.T$$

(if R and T are entirely independent) or

$$\alpha.R \;\dot{\subseteq}\; K.R \;\;\wedge\;\; \alpha.R \,\dot{\cap}\, \alpha.T \;\dot{\subseteq}\; K.T$$

(R can be validated independently of T but the validity of T depends on the validity of R.) This helps to eliminate unnecessary detail.

In some cases, a substantial simplification of (12.9) is possible. Specifically,

$$(12.16) \quad \Leftarrow \quad F{\circ}t{\circ}q \ \dot{\subseteq} \ K.R \ \wedge \ q{\circ}S{\circ}p \ \dot{\subseteq} \ K.R$$

if R is reflexive and transitive. It follows that —sometimes— the verification condition (12.9) can be replaced by the two conditions

(12.17) $q{\circ}S{\circ}p \ \dot{\subseteq} \ K.R$

and

(12.18) $F{\circ}t{\circ}q \ \dot{\subseteq} \ K.R$.

In words, for some reflexive and transitive relations R, it is the case that R is independently an invariant of F and an invariant of S; that is, it is not necessary to consider the combined effect of F and S to establish the invariance of R. Exploiting this simplification is a reason for distinguishing between invariant relations, invariant properties and invariant values: the technique is typically applied to invariant properties and values but not to other relations. The downside is that more care needs to be taken in formulating the intermediate assertion q since its role becomes more pronounced.

12.3 "Grey" Paths and Impossible Edges

In this section, we return to the implementation of depth-first search documented in figs. 11.2 and 11.3, adding a new variable that records the set of nodes from which a depth-first search has finished. This addition anticipates the computation of timestamps in section 13. The induction theorem, theorem 12.5, is used to establish a number of properties that are crucial to calculating strongly connected components. (See section 13.)

We call the new variable fnd (short for "finished") and begin by adding its initialisation to the outer loop. See fig. 12.3.

The invariant relation New has been replaced by the relation GE^2: the state space is now a cartesian product of two sets of nodes, and GE^2 is likewise the cartesian product of two instances of GE, which is clearly a subset of the relation New. (The extra detail supplied by New was used to prove theorem 11.24; we don't need the information here.) Thus, the claim is that both $seen$ and fnd are increasing. The frontier function Fr has been redefined so that it depends only on the state of the $seen$ nodes. A new invariant property has been added as well. Finally, the precondition on the call of $dfs(a)$ has also been augmented with an additional conjunct.

The invariant property has, in total, four conjuncts, divided into two pairs of two, but there is some (obvious) redundancy in the conjuncts. They have been stated in this

```
$seen,fnd := \perp\!\!\perp,\perp\!\!\perp$ ;
{ Invariant Relation: $GE^2 \;\cap\; Fr^{\cup}\circ(\subseteq)\circ Fr$
      where   $(s',f')\,[\![GE^2]\!]\,(s,f) \;\equiv\; s'\supseteq s \wedge f'\supseteq f$
      and     $Fr(s,f) \;=\; s\circ(\subseteq)\circ{\sim}s$
  Invariant Property:
        $fnd\subseteq seen \;\;\wedge\;\; fnd\circ G\circ{\sim}seen = \perp\!\!\perp$
  Invariant Property:
        $seen\circ{\sim}fnd = \perp\!\!\perp \;\;\wedge\;\; fnd\circ G^*\circ{\sim}seen = \perp\!\!\perp$   }
while $\sim seen \neq \perp\!\!\perp$ do
  begin
    choose node $a$ such that $a\circ{\sim}seen = a$
  ; {   $a\circ{\sim}seen = a \;\;\wedge\;\; seen\circ{\sim}fnd\circ\top\!\!\top\circ a \subseteq (seen\circ{\sim}fnd\circ G)^*$   }
    $dfs(a)$
  end
```

Figure 12.3: Grey Paths and Impossible Edges. Outer Loop

way in order to clarify differences between the invariants of the outer and the inner loops. Specifically, the first pair is also an invariant of the inner loop, but the second is not.

It is easy to check that all are truthified by the initial assignment to $seen$ and fnd. Thus, to say they are "invariant" means that their true value is unchanged before each iteration of the loop body. The meaning of the first, $fnd\subseteq seen$, is obvious. In the third conjunct, the term $seen\circ{\sim}fnd$ represents the set of nodes from which a search has started but has not finished. The assertion

(12.19) $seen\circ{\sim}fnd \;=\; \perp\!\!\perp$

states that this set is empty at every iteration of the outer loop; this is not necessarily the case when executing the inner loop. The combination of $fnd\subseteq seen$ and (12.19) implies $fnd = seen$. The conjunct

(12.20) $fnd\circ G\circ{\sim}seen \;=\; \perp\!\!\perp$

asserts that there are no edges in G from nodes from which the search has finished to nodes that have not been seen. (These are the "impossible edges" referred to in the title of this section.) As we shall see, this property is crucial to the use of depth-first search in determining the strongly connected components of a graph. The final conjunct

(12.21) $fnd \circ G^* \circ {\sim}seen \;=\; \perp\!\!\perp$

asserts that there are no unseen nodes that are reachable from the set of finished nodes. Like (12.19), this is a property of the outer loop but not the inner loop.

Many descriptions of depth-first search use colours to distinguish nodes and edges of the graph at different stages of the search. The nodes represented by ${\sim}seen$ are called *white* nodes, those represented by $seen \circ {\sim}fnd$ are called *grey* nodes and, finally, those represented by fnd are called *black* nodes. The property $fnd \subseteq seen$, which we shall establish to be an invariant at all stages, implies that all nodes are either white, grey or black (as is easily shown). The property (12.20) asserts that there are no *edges* from a black node to a white node. The invariant (12.21) asserts that there are no *paths* from a black node to a white node (in the outer loop).

The property (12.21) clearly subsumes (12.20). Indeed, since $fnd = seen$ in the outer loop, (12.20) is equivalent to the property

$$seen \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$$

which we established in corollary 11.21. The importance of (12.20) is that it is also an invariant of the inner loop, whereas (12.19) and (12.21) may be $false$ when executing the inner loop. This is one way in which this section sharpens the results of section 11.1. The second way is the additional precondition on the call of $dfs(a)$:

(12.22) $seen \circ {\sim}fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim}fnd \circ G)^*$.

This assertion is trivially $true$ in the outer loop given that $fnd = seen$. However, the boolean (12.19) is not an invariant of the inner loop, which means that (12.22) is non-trivial.

The next step is to add code to the procedure dfs that updates fnd whenever a call is completed, and to add the appropriate documentation. This is shown in fig. 12.4.

First note that an assignment to fnd has been added following the **while** statement; as in the outer loop, the invariant has been weakened to the relation GT on successive values of $seen$ and then extended to GT^2 in order to include fnd; the invariant GE of the inner loop is similarly extended. Now note that

(12.23) $fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$

is asserted to be an invariant property of both the procedure dfs and the inner loop; the assertion about

(12.24) $seen \circ {\sim}fnd$

```
    {  **Invariant Relation**:   $GT^2 \;\cap\; Fr^{\cup} \circ (\subseteq) \circ Fr$
       **Invariant Property**:   $fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$
       **Invariant Value**:   $seen \circ {\sim}fnd$  }
    {  $a \circ seen = \perp\!\!\perp \;\wedge\; seen \circ {\sim}fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim}fnd \circ G)^*$  }
    $seen := seen \cup a$
    {  **Invariant Relation**:   $GE^2 \;\cap\; Fr^{\cup} \circ (\subseteq) \circ Fr$
       **Invariant Property**:
                $fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$
           $\wedge$   $a \circ seen \circ {\sim}fnd = a$
           $\wedge$   $seen \circ {\sim}fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim}fnd \circ G)^*$
       **Invariant Value**:   $seen \circ {\sim}fnd$  }
;   **while** $a \circ G \circ {\sim}seen \neq \perp\!\!\perp$ **do**
       **begin**
         choose node $b$ such that $a \circ \top\!\!\top \circ b \;\subseteq\; a \circ G \circ {\sim}seen$
       ; {      $fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$
            $\wedge$  $b \circ seen = \perp\!\!\perp \;\wedge\; seen \circ {\sim}fnd \circ \top\!\!\top \circ b \subseteq (seen \circ {\sim}fnd \circ G)^*$  }
         $dfs(b)$
       **end**
    {      $fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp$
       $\wedge$   $a \circ seen \circ {\sim}fnd = a$
       $\wedge$   $a \circ G \circ {\sim}seen = \perp\!\!\perp$  }
;   $fnd := fnd \cup a$
```

Figure 12.4: Grey Paths and Impossible Edges. The Procedure $dfs(a)$.

is that it is an invariant value. Since the distinction between invariant "relation", "property" and "value" is uncommon, the reader may wish to take the opportunity to review the discussion in sections 6.8.4 and 12.2. When applying theorem 12.5, the relation R is instantiated to

$$GT^2 \;\cap\; Fr^{\cup} \circ (\subseteq) \circ Fr \;\cap\; H1^{\cup} \circ (\Leftarrow) \circ H1 \;\cap\; Grey^{\cup} \circ Grey$$

where $H1(s,f)$ is the boolean

$$f \subseteq s \;\wedge\; f \circ G \circ {\sim} s \;=\; \perp\!\!\perp$$

and $Grey(s,f)$ is the set of nodes represented by the coreflexive

$$s \circ {\sim} f \;\;.$$

(That is, $Grey(s,f)$ is the set of nodes that are "grey" at a given time.) See section 12.2 for discussion of how we break down the verification condition (12.9).

Assertion (12.23) is a boolean expression (i.e. has value true or false) but we want to show that it is an invariant "property", i.e. the fact that it has the value true is unchanging. In such cases, additional arguments must be given to establish that the value is truthified appropriately. This means that the precondition of the call of $dfs(a)$ is the conjunction of (12.23) and the condition that immediately precedes the assignment to $seen$:

$$\begin{array}{ll} & fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim} seen \;=\; \perp\!\!\perp \\ \wedge & a \circ seen = \perp\!\!\perp \;\wedge\; seen \circ {\sim} fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim} fnd \circ G)^* \;\;. \end{array}$$

In order to facilitate the application of theorem 12.5, we denote the coreflexive corresponding to this precondition by $p.a$. For the same reason, we denote the assertion that immediately follows the assignment to $seen$, viz.

$$\begin{array}{ll} & fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim} seen \;=\; \perp\!\!\perp \\ \wedge & a \circ seen \circ {\sim} fnd = a \;\wedge\; seen \circ {\sim} fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim} fnd \circ G)^* \;\;, \end{array}$$

by $q.a$.

The assertion $p.b$ (that is, the assertion denoted $p.a$ above but with a replaced by b) prefixes the call of $dfs(b)$ in the inner loop. The first two conjuncts, which are independent of a or b, are also listed as invariant properties of the procedure dfs. They are truthified by the initial assignment to $seen$ and fnd in the outer loop. We shall show that their true value remains unchanged at every point in the execution of the algorithm. The validity of the third conjunct is easily verified. The fourth conjunct

(12.25) $seen \circ {\sim} fnd \circ \top\!\!\top \circ a \;\subseteq\; (seen \circ {\sim} fnd \circ G)^*$

asserts that there is a path from all “grey” nodes to the node from which the search is about to start; moreover, each edge on such a path is from a “grey” node. (Recall that “grey” nodes are the nodes represented by $seen \circ {\sim}fnd$.) In the outer loop, this condition is clearly implied by the invariant (12.19). The proof that this is also the case in the inner loop is provided here for two reasons: it helps to explain the requirements used in the formulation of the basic induction theorem (theorem 12.5) for reasoning about depth-first search, and it is needed in chapter 13.

The assignment to fnd does not have an explicit postcondition. It is implicit in the invariants of the procedure dfs: the postcondition is that (12.23) and (12.24) are unchanged from their initial values. Similarly, the call $dfs(b)$ is not documented by a postcondition; when reasoning about it, we exploit the “induction hypothesis” that the relations GT^2, $Fr^{\cup} \circ (\subseteq) \circ Fr$, (12.23) and (12.24) are invariants of dfs. That assertion (12.25) is also an invariant is an immediate consequence of the fact that (12.24) is an invariant value.

Let us use theorem 12.5 to prove that each of the claimed invariants in fig. 12.4 is indeed invariant.

12.3.1 Truthifying the Intermediate Assertion

The coreflexives $p.a$ and $q.a$ were defined earlier. Thus property (12.6) is equivalent to

$$\begin{array}{ll} \{ & fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp \\ \;\;\wedge & a \circ seen = \perp\!\!\perp \;\wedge\; seen \circ {\sim}fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim}fnd \circ G)^* \quad \} \\ \multicolumn{2}{l}{seen := seen \cup a} \\ \{ & fnd \subseteq seen \;\wedge\; fnd \circ G \circ {\sim}seen \;=\; \perp\!\!\perp \\ \;\;\wedge & a \circ seen \circ {\sim}fnd = a \;\wedge\; seen \circ {\sim}fnd \circ \top\!\!\top \circ a \subseteq (seen \circ {\sim}fnd \circ G)^* \quad \} \end{array}$$

whose validity can be verified using the assignment axiom.

12.3.2 The Precondition in the Inner Loop

Property (12.7) involves establishing that four conjuncts follow from the conjunction of $q.a$, the condition for executing the loop body and the criterion for choosing b. Detailed inspection of what is required reveals that two of the conjuncts are immediate. The remaining two follow from the verification condition:

$$\begin{array}{l} \langle \forall\, a,b,s,f \\ \qquad : \quad a \circ \top\!\!\top \circ b \subseteq s \circ {\sim}f \circ G \circ {\sim}s \;\wedge\; s \circ {\sim}f \circ \top\!\!\top \circ a \subseteq (s \circ {\sim}f \circ G)^* \end{array}$$

$$: \quad b\circ s = \perp\!\!\perp \;\wedge\; s\circ{\sim}f\circ\top\!\!\top\circ b \subseteq (s\circ{\sim}f\circ G)^*$$

$$\rangle \;\; .$$

The only non-trivial part of verifying this theorem is covered by the following simple calculation.

$$
\begin{array}{ll}
 & a\circ\top\!\!\top\circ b \;\subseteq\; s\circ{\sim}f\circ G\circ{\sim}s \;\;\wedge\;\; s\circ{\sim}f\circ\top\!\!\top\circ a \subseteq (s\circ{\sim}f\circ G)^* \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & s\circ{\sim}f\circ\top\!\!\top\circ a\circ a\circ\top\!\!\top\circ b \;\subseteq\; (s\circ{\sim}f\circ G)^*\circ s\circ{\sim}f\circ G\circ{\sim}s \\
\Rightarrow & \{ \quad a \neq \perp\!\!\perp\text{, so } \top\!\!\top\circ a\circ a\circ\top\!\!\top = \top\!\!\top\text{ ; } {\sim}s \text{ is a coreflexive} \quad \} \\
 & s\circ{\sim}f\circ\top\!\!\top\circ b \;\subseteq\; (s\circ{\sim}f\circ G)^*\circ s\circ{\sim}f\circ G \\
\Rightarrow & \{ \quad [\, R^*\circ R \subseteq R^* \,]\text{, transitivity} \quad \} \\
 & s\circ{\sim}f\circ\top\!\!\top\circ b \;\subseteq\; (s\circ{\sim}f\circ G)^* \;\; .
\end{array}
$$

12.3.3 Maintaining the Intermediate Assertion

Checking (12.8) is trivial. We have to show that the property $q.a$ is maintained by the stated invariants. But $q.a$ is the composition of the coreflexive corresponding to (12.23), which is an invariant property, and

$$a\circ seen\circ{\sim}fnd = a \;\wedge\; seen\circ{\sim}fnd\circ\top\!\!\top\circ a \subseteq (seen\circ{\sim}fnd\circ G)^* \;\; ,$$

which is obviously invariant because the value of $seen\circ{\sim}fnd$ is invariant. That is, $q.a$ is maintained, as required. (We still have to check that the value of $seen\circ{\sim}fnd$ is indeed invariant but this is independent of property (12.8). We also still have to check that (12.23) is an invariant property.)

12.3.4 Invariant Relations

We now consider each of the claimed invariant relations, values and properties in turn. We begin with the invariant relation

$$GT^2 \;\cap\; Fr^{\cup}\circ(\subseteq)\circ Fr$$

because the fact that it is invariant is needed when verifying the remaining invariants.

Apart from the replacement of NR by GT^2 and the redefinition of the frontier function Fr (necessitated by the addition of the assignment to fnd) that this relation is invariant was discussed in section 11.1. We leave the reader the straightforward task of checking the validity of the replacements. (The main task is to use theorem 12.5 to formally check that the assignment to fnd causes its value to strictly increase. Just as for $seen$ this is straightforward.)

12.3.5 Invariant Value

We now show that $\mathit{seen}\circ{\sim}\mathsf{fnd}$ is a *value* invariant. This means literally that the value of $\mathit{seen}\circ{\sim}\mathsf{fnd}$ remains unchanged by a call of dfs. To show that this is the case, we define the function Grey by $\mathsf{Grey}(s,f) = s\circ{\sim}f$ and we instantiate R in (12.9) with $\mathsf{Grey}^{\cup}\circ\mathsf{Grey}$. (The relation $\mathsf{Grey}^{\cup}\circ\mathsf{Grey}$ asserts an equality between two evaluations of the function Grey. That is,

$$(s',f')[\![\mathsf{Grey}^{\cup}\circ\mathsf{Grey}]\!](s,f) \quad\equiv\quad s'\circ{\sim}f' = s\circ{\sim}f \;\;.)$$

We have

$$\begin{array}{ll}
 & \langle\forall a :: \mathsf{Grey}^{\cup}\circ\mathsf{Grey}\circ S.a\circ p.a \;\subseteq\; (F.a)^{\cup}\circ\mathsf{Grey}^{\cup}\circ\mathsf{Grey}\rangle \\
\Leftarrow & \{\quad \text{definitions of } p\,,\ F\,,\ \mathsf{Grey} \text{ and } S \quad\} \\
 & \langle\forall\, a,s',f',s,f \;:\; s'\circ{\sim}f' = (s\cup a)\circ{\sim}f \,\wedge\, s\circ a = \perp\!\!\perp \;:\; s'\circ{\sim}(f'\cup a) = s\circ{\sim}f\rangle \;\;.
\end{array}$$

Now,

$$\begin{array}{ll}
 & s'\circ{\sim}(f'\cup a) \\
= & \{\quad \text{distributivity} \quad\} \\
 & s'\circ{\sim}f'\circ{\sim}a \\
= & \{\quad \text{assume: } s'\circ{\sim}f' = (s\cup a)\circ{\sim}f \quad\} \\
 & (s\cup a)\circ{\sim}f\circ{\sim}a \\
= & \{\quad \text{distributivity and commutativity of coreflexives} \quad\} \\
 & a\circ{\sim}a\circ{\sim}f \,\cup\, s\circ{\sim}f\circ{\sim}a \\
= & \{\quad a\circ{\sim}a = \perp\!\!\perp \quad\} \\
 & s\circ{\sim}f\circ{\sim}a \\
= & \{\quad \text{assume: } a\circ s = \perp\!\!\perp\text{, equivalently } {\sim}a\circ s = s \quad\} \\
 & s\circ{\sim}f \;\;.
\end{array}$$

This completes the verification.

12.3.6 Invariant Properties

We now check that (12.23) is an invariant property.

Visual inspection of the code in fig. 12.4 suggests that verification of the conjunct $\mathsf{fnd}\subseteq\mathit{seen}$ is straightforward. Indeed, this is the case. Rather than establish (12.9), we verify (12.17) and (12.18). That is, we establish that the property is maintained independently by the assignments S and F.

The verification of (12.17) corresponds to verifying the validity of

$$\{\ \ fnd \subseteq seen\ \ \}$$
$$seen := seen \cup a$$
$$\{\ \ fnd \subseteq seen\ \ \}$$

and the verification of (12.18) corresponds to verifying the validity of

$$\{\ \ fnd \subseteq seen \wedge a \circ seen = a\ \ \}$$
$$fnd := fnd \cup a$$
$$\{\ \ fnd \subseteq seen\ \ \}\ \ .$$

Both are easy applications of the assignment axiom.

Note that the verification of (12.18) is slightly more complex than that of (12.17) because of the additional conjunct in the precondition. That the additional conjunct is needed demonstrates why it is necessary to prove that the relation GT^2 is an invariant of calls of dfs: the property $a \circ seen = a$ is truthified by the initial assignment to $seen$ but we need to be sure that it is not falsified by subsequent calls of dfs.

We now check the conjunct

$$fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp \ \ .$$

First, it is an invariant of the initial assignment to $seen$:

$$
\begin{array}{ll}
 & (fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp)[seen := seen \cup a] \\
= & \quad \{ \quad \text{substitution and distributivity} \quad \} \\
 & fnd \circ G \circ {\sim} seen \circ {\sim} a \ = \ \perp\!\!\perp \\
\Leftarrow & \quad \{ \quad {\sim} a \subseteq I\,,\ \perp\!\!\perp \text{ is least} \quad \} \\
 & fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp
\end{array}
$$

Second, it is an invariant of the assignment to fnd:

$$
\begin{array}{ll}
 & (fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp)[fnd := fnd \cup a] \\
= & \quad \{ \quad \text{substitution and distributivity} \quad \} \\
 & fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp \ \wedge \ a \circ G \circ {\sim} seen \ = \ \perp\!\!\perp \\
= & \quad \{ \quad \text{assume } t.a\text{, i.e. } a \circ G \circ {\sim} seen = \perp\!\!\perp \\
 & \qquad\quad \text{and } q.a\text{, in particular } fnd \circ G \circ {\sim} seen \ = \ \perp\!\!\perp \quad \} \\
 & \mathsf{true}\ \ .
\end{array}
$$

12.3.7 Invariants of the Outer Loop

Our last task is to verify the assertions in the outer loop (fig. 12.3).

That the relation $GE^2 \cap Fr^{\cup} \circ (\subseteq) \circ Fr$ is an invariant of the **while** statement is immediate from the fact that it is an invariant of calls of dfs.

For the invariant property, recall that it suffices to establish the three conjuncts

$$fnd \subseteq seen \;\wedge\; seen \circ {\sim}fnd = \perp\!\!\perp \;\wedge\; fnd \circ G^* \circ {\sim}seen = \perp\!\!\perp \;\; .$$

The initialisation of $seen$ and fnd to $\perp\!\!\perp$ clearly truthifies each of the conjuncts.

Since $fnd \subseteq seen$ is an invariant of the procedure dfs, as is the value $seen \circ {\sim}fnd$, it follows that the first two conjuncts are invariants of the **while** statement so long as we can prove that, together with the condition for choosing a, they guarantee the precondition for executing $dfs(a)$. But this is trivially true.

That the third conjunct is also an invariant of the **while** statement now follows from the property (11.1) proved earlier. Specifically,

$$\begin{array}{ll}
 & fnd \circ G^* \circ {\sim}seen \\
= & \{ \quad fnd \subseteq seen \;\wedge\; seen \circ {\sim}fnd = \perp\!\!\perp \\
 & \quad\;\; \text{hence } fnd = seen \quad \} \\
 & seen \circ G^* \circ {\sim}seen \\
= & \{ \quad \text{domains and (11.1)} \quad \} \\
 & seen \circ G^* \circ seen \circ {\sim}seen \\
= & \{ \quad seen \circ {\sim}seen = \perp\!\!\perp \quad \} \\
 & \perp\!\!\perp \;\; .
\end{array}$$

This completes the verification.

Chapter 13

Calculating Strongly Connected Components

In this chapter, we establish the correctness of an algorithm to calculate the strongly connected components of a finite graph. The algorithm is based on the one described in [AHU82, pp.222–226].

Algorithms for calculating strong components are well cited. Invariably they are based on depth-first search. The algorithm presented here searches the graph in two consecutive phases. In the first phase depth-first search is used but in the second phase any search algorithm can be used. By presenting a formal proof of correctness, we hope to clarify the key properties.

We assume that G is the edge relation of a finite (directed) graph. The algorithm calculates a (total) function φ that assigns to each node a of the graph a *representative* of the strongly connected component at a. Formally, φ has the properties

$$\varphi \circ \varphi^{\cup} \subseteq I_{\mathsf{Node}} \quad \wedge \quad \varphi^{\cup} \circ \varphi = \mathsf{equiv}.G \quad .$$

The first phase calculates a function f that is then used in the second phase as the choice function in the delegate algorithm discussed in section 10.2. The function f records the order in which the depth-first searches in the first phase finish: the representative of a strongly connected component p is the node in p from which the depth-first search in the first phase finishes last.

13.1 Timestamps

Timestamps comprise two functions s (for “start”) and f (for “finish”) that record the order in which searches are started and finished. The specification is thus a relation of type

$$\langle \Pi\mathsf{Node} \ : \ \mathsf{finite.Node} \ : \ (\mathbb{N} \leftarrow \mathsf{Node}) \times (\mathbb{N} \leftarrow \mathsf{Node}) \sim (\mathsf{Node}{\sim}\mathsf{Node}) \rangle$$

such that, for a given graph G, the constructed values s and f, both of which have type $\mathbb{N}\leftarrow\mathsf{Node}$, are total, injective functions. To ensure totality, the first phase takes the following form:

```
$f{,}s \;:=\; \bot\!\!\bot{,}\bot\!\!\bot$ ;
while $s{>\!\!\bullet} \neq \bot\!\!\bot$ do
   begin
     choose node $a$ such that $a\circ s{>} = \bot\!\!\bot$
   ;  $dfs(a)$
   end
```

We refer to this part of the implementation as the *outer loop*. Compared to the implementation in fig. 12.3, the variables $seen$ and fnd have been removed: the nodes that have been "seen" (i.e. from which a search has been started) are the nodes represented by $s{>}$ —the nodes that have a start time—, and the nodes have not been "seen" —previously $\sim\!seen$— is represented by $s{>\!\!\bullet}$. In practice, a boolean array $seen$ indexed by nodes would be added to the implementation with the invariant property that $seen$ and $s{>}$ represent the same set, namely the set of nodes for which the function s is defined. It helps to keep the account shorter if we don't do so. Similarly, an auxiliary variable fnd might be added to the implementation with the invariant property that fnd equals $f{>}$ (and $\sim\!fnd$ equals $f{>\!\!\bullet}$) but we don't do so for reasons of economy. However, we do translate all the properties established in previous sections of $seen$ and fnd into properties of $s{>}$ and $f{>}$.

The implementation of $dfs(a)$ is as follows:

```
     $s \;:=\; s \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a$
;    while $a\circ G\circ s{>\!\!\bullet} \neq \bot\!\!\bot$ do
        begin
          choose node $b$ such that $a\circ\top\!\!\top\circ b \subseteq a\circ G\circ s{>\!\!\bullet}$
        ;  $dfs(b)$
        end
;    $f \;:=\; f \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a$
```

The **while** statement is bracketed by two statements that set the value of the functions s and f at a. The notation $MAX.s$ denotes the maximum value of the function s, and similarly for $MAX.f$. If $s=\bot\!\!\bot$, the value of $MAX.s$ is defined to be 0. The value

$MAX.s\uparrow MAX.f$ counts the number of times that an assignment to s or f has been made; in this way, the assignment

$$s ~:=~ s \cup \overline{(MAX.s\uparrow MAX.f)+1}\circ\top\!\!\top\circ a$$

records the "time" that the search from node a is initiated. The assignment to f is similarly interpreted.

The coreflexive $s{>}$ represents the set of nodes from which a search has been started, i.e. the nodes that have been "seen". As already mentioned, a depth-first search from a is only initiated when $a\circ s{>} = \perp\!\!\perp$. This guarantees that the property that s is functional is invariant; the fact that $(MAX.s\uparrow MAX.f)+1$ is distinct from any existing value of s guarantees that the property that s is injective is also invariant. Similarly, the coreflexive $f{>}$ represents the set of nodes from which a search has finished —previously represented by the variable fnd (see fig. 12.4)— and the assignment to f guarantees that its functionality and injectivity remain invariant.

Apart from the replacement of $\sim\!seen$ by $s{>}\!\bullet$, the **while** statement itself is unchanged from section 11.

The reader may wish to look at fig. 11.1 again as an example of the calculation of timestamps. Recall that the first element of the pair of numbers labelling a node is the timestamp of the start of the search from that node and the second element is the timestamp of the finish of the search from that node. Recall also that the labels O1 thru O6 indicate the nodes from which a search has been started in the outer loop.

For the purpose of calculating strongly connected components, the start timestamp s is not needed in its entirety: computations in the first phase make use of just $s{>}$ and only the function f is used in the second phase. Thus a practical implementation would introduce a boolean variable $seen$, as outlined earlier, and document s as an auxiliary variable.

13.1.1 Specification

The timestamps s and f have a number of properties that are crucial to the successful use of f in the second phase to compute representatives of the strongly connected components. Fig. 13.1 documents the outer loop with the relevant properties[1].

For the moment, note particularly the postcondition. The first conjunct states formally that s and f are total, injective functions (the property mentioned earlier). The second and third conjuncts relate s and f to the given graph G; take care to note that G is on the left of an inclusion in the second conjunct and on the right in the third conjunct. The final conjunct is a characteristic property of depth-first search.

[1]Stronger properties are discussed in chapter 14.

```
{   G is a finite graph   }
f,s := $\perp\!\!\perp , \perp\!\!\perp$ ;
{  Invariant Relation: $\mathsf{ME}^2 \cap \mathsf{Fr}^{\cup} \circ (\supseteq) \circ \mathsf{Fr} \cap \mathsf{sInc}$
       where   $(s',f')\ \mathsf{ME}^2\ (s,f) \;\equiv\; s'{>} \supseteq s{>} \wedge f'{>} \supseteq f{>}$
       and     $\mathsf{Fr}(s,f) \;=\; s{>} \circ G \circ s{>\!\bullet}$
       and     $(s',f')\ \mathsf{sInc}\ (s,f) \;\equiv\; s{>\!\bullet} \circ s'^{\cup} \circ \leq \circ s = \perp\!\!\perp$
   Invariant Property: (13.1) ∧ (13.2) ∧ (13.3) ∧ (13.5) ∧ (13.4)   }
while $s{>} \neq I_{\mathsf{Node}}$ do
   begin
      choose node $a$ such that $a \circ s{>} = \perp\!\!\perp$
   ;  dfs($a$)
   end
{       $f{>} = f^{\cup} \circ f = s{>} = s^{\cup} \circ s = I_{\mathsf{Node}} \;\wedge\; s \circ s^{\cup} \subseteq I_{\mathbb{N}} \;\wedge\; f \circ f^{\cup} \subseteq I_{\mathbb{N}}$
   ∧   $G^{\cup} \subseteq s^{\cup} \circ \leq \circ f$
   ∧   $s^{\cup} \circ \leq \circ s \cap f^{\cup} \circ \geq \circ f \subseteq G^*$
   ∧   $s^{\cup} \circ \leq \circ s \cap f^{\cup} \circ < \circ f \;=\; f^{\cup} \circ < \circ s$   }
```

Figure 13.1: Timestamps: Outer Loop

In this section, we prepare the ground for applying the induction theorem, theorem 12.5, to establish these properties of the implementation of timestamps. That is, we formulate an invariant relation R, precondition p and intermediate assertion q that precisely capture the execution of depth-first search.

In section 11, we proved formally that the set of "seen" nodes is strictly increased by calls of the procedure dfs. This and other elements of the invariants studied there need to be adapted, replacing the variable *seen* by $s{>}$ and the variable fnd by $f{>}$. Fig. 13.1 documents the fact that the relation $\mathsf{ME}^2 \cap \mathsf{Fr}^{\cup} \circ (\supseteq) \circ \mathsf{Fr}$ is an invariant, where the definitions of relations ME and Fr are suitably modified versions of the relations GE and the frontier function Fr of fig. 12.3. These are supplemented in fig. 13.1 by the relation sInc where, for all s', f', s and f

$$(s',f')\ \mathsf{sInc}\ (s,f) \;\equiv\; s{>\!\bullet} \circ s' \circ \leq \circ s = \perp\!\!\perp \;\; .$$

This relation expresses the property that, at each iteration of the outer loop body, no newly started node has a starting timestamp that is at most the starting timestamp of a node from which the search has already been started. (The equivalent contrapositive of this is that nodes from which a search is newly started by an execution of the body of the outer loop have a timestamp that is strictly greater than the timestamp of nodes from which a search has already been started.)

The invariant properties of the outer loop are formulated below. Accompanying each is a verbal explanation.

At the start of each iteration of the outer loop body, f and s are injective functions with equal right domains (but they are not total until termination of the loop): that is

(13.1) $f{>} = f^{\cup}\circ f = s{>} = s^{\cup}\circ s \;\wedge\; s\circ s^{\cup} \subseteq I_{\mathbb{N}} \;\wedge\; f\circ f^{\cup} \subseteq I_{\mathbb{N}}$.

For each node in the right domain of f, the start timestamp is less than the finish timestamp:

(13.2) $f{>} \;\subseteq\; s^{\cup}\circ{<}\circ f$.

There are no edges from nodes from which the search has finished to nodes from which the search has either not started or started at a later time:

(13.3) $f{>}\circ G\circ s{>}{\bullet} \;=\; \perp\!\!\perp \;=\; f^{\cup}\circ{<}\circ s \cap G$.

For all nodes a and b, if the search from a starts before the start of the search from b, and the search from a finishes after the search from b finishes there is a path from a to b:

(13.4) $s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f \;\subseteq\; G^{*}$.

(Property (13.4) asserts an inclusion only. The inclusion can, in fact, be strengthened to an equality by appropriately modifying the right side. See chapter 14 for discussion on this and the “white-path theorem”. Property (13.4) is sufficient for our current goals.)

For all nodes a and b, if the search from a starts before the start of the search from b and finishes before the finish of the search from b, the search from a finishes before the search from b starts.

(13.5) $s^{\cup}\circ{<}\circ s \;\cap\; f^{\cup}\circ{<}\circ f \;=\; f^{\cup}\circ{<}\circ s$.

It is straightforward to check that the stated postcondition is implied by the conjunction of the termination condition and the above invariant properties — with the exception of the second conjunct:

(13.6) $G^{\cup} \;\subseteq\; s^{\cup}\circ{\leq}\circ f$.

This is a consequence of the invariant (13.3), as the following calculation shows[2].

$$
\begin{array}{ll}
 & \bot\!\!\bot \;=\; f^{\cup}\circ<\circ s\cap G \\
= & \{ \quad \text{shunting rule (2.27)} \quad \} \\
 & f^{\cup}\circ<\circ s \;\subseteq\; \neg G \\
= & \{ \quad \text{middle-exchange rule (4.18)} \quad \} \\
 & f\circ G\circ s^{\cup} \;\subseteq\; \neg(<) \\
= & \{ \quad \text{not less-than relation on numbers is at-least} \quad \} \\
 & f\circ G\circ s^{\cup} \;\subseteq\; (\geq) \\
= & \{ \quad \text{on termination, } f \text{ and } s \text{ are total functions} \quad \} \\
 & G \;\subseteq\; f^{\cup}\circ\geq\circ s \\
= & \{ \quad \text{converse} \quad \} \\
 & G^{\cup} \;\subseteq\; s^{\cup}\circ\leq\circ f \quad .
\end{array}
$$

In order to check that the additional properties (13.1), (13.2) (13.3), (13.4) and (13.5) are indeed maintained invariant by the body of the outer loop, we need to provide details of the implementation of $dfs(a)$, which we now do.

As demonstrated in section 11.1, properties that hold in the outer loop may not hold in the inner loop. Such properties must be weakened in the inner loop, but properties added to guarantee the stronger properties in the outer loop.

For convenience, we summarise the properties in the following definition.

Definition 13.7 (Specification of Timestamped Depth-First Search) Suppose G is a relation of type $\mathsf{Node}{\sim}\mathsf{Node}$ where Node is a finite set. Suppose also that s, f, s' and f' are all functions of type $\mathbb{N}\leftarrow\mathsf{Node}$. The invariant property of a depth-first search from an arbitrary node, which we abbreviate to $Inv(s,f)$, is the conjunction of the following properties:

(13.8) $f{>} = f^{\cup}\circ f \;\subseteq\; s{>} = s^{\cup}\circ s \;\;\wedge\;\; s\circ s^{\cup}\subseteq I_{\mathbb{N}} \;\;\wedge\;\; f\circ f^{\cup}\subseteq I_{\mathbb{N}}$,

(13.9) $f{>} \;\subseteq\; s^{\cup}\circ<\circ f$,

[2]Effectively, the calculation shows that —on termination of the outer loop— the negation of $f^{\cup}\circ<\circ s$ is $f^{\cup}\circ\geq\circ s$. When reasoning pointwise, it is tempting to dismiss this as an obvious property of the less-than ordering on numbers. However, the proven equality is only valid on termination since, during execution, s and f are partial functions and the negation of $f^{\cup}\circ<\circ s$ relates certain nodes on which f and/or s are undefined. It is incorrect to assert that the property $G^{\cup}\subseteq s^{\cup}\circ\leq\circ f$ is an invariant of depth-first search.

(13.10) $f{\scriptstyle >}\circ G\circ s{\scriptstyle >\bullet} \;=\; \perp\!\!\perp \;=\; f^{\cup}\circ<\circ s\cap G$,

(13.11) $f{\scriptstyle >\bullet}\circ s^{\cup}\circ\leq\circ s \;\subseteq\; G^{*}$,

(13.12) $s^{\cup}\circ\leq\circ s \;\cap\; f^{\cup}\circ\geq\circ f \;\subseteq\; G^{*}$,

(13.13) $s^{\cup}\circ\leq\circ s \;\cap\; f^{\cup}\circ<\circ f \;=\; f^{\cup}\circ<\circ s\circ f{\scriptstyle >}$.

Formally, $\mathit{Inv}(s,f)$ is defined to be

$$(13.8)\wedge(13.9)\wedge(13.10)\wedge(13.11)\wedge(13.12)\wedge(13.13)\quad .$$

Also, $P(a,s,f)$ is defined to be

$$a\circ s{\scriptstyle >\bullet} \;=\; a \;\wedge\; s{\scriptstyle >}\circ f{\scriptstyle >\bullet}\circ\top\!\!\top\circ a \;\subseteq\; (s{\scriptstyle >}\circ f{\scriptstyle >\bullet}\circ G)^{*}$$

and $Q(a,s,f)$ is defined to be

$$\begin{array}{ll} & a\circ s{\scriptstyle >}\circ f{\scriptstyle >\bullet} \;=\; a \\ \wedge & f{\scriptstyle >}\circ s^{\cup}\circ<\circ s\circ a \;=\; f^{\cup}\circ<\circ s\circ a \\ \wedge & a\circ s^{\cup}\circ<\circ s\circ f{\scriptstyle >\bullet} \;=\; \perp\!\!\perp \\ \wedge & f{\scriptstyle >\bullet}\circ s{\scriptstyle >}\circ\top\!\!\top\circ a \;\subseteq\; (s{\scriptstyle >}\circ f{\scriptstyle >\bullet}\circ G)^{*}\quad . \end{array}$$

Finally, Invrel is defined by

$$\mathit{Invrel} \;=\; MT^{2} \cap Fr^{\cup}\circ(\subseteq)\circ Fr \cap \mathit{Grey}^{\cup}\circ\mathit{Grey} \cap \mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv} \cap s\mathit{Inc}$$

where, for all s , f , s' and f',

$$(s',f')\;[\![MT^{2}]\!]\;(s,f) \;\equiv\; s'{\scriptstyle >}\supseteq s{\scriptstyle >} \wedge f'{\scriptstyle >}\supseteq f{\scriptstyle >} \wedge s'{\scriptstyle >}\neq s{\scriptstyle >} \wedge f'{\scriptstyle >}\neq f{\scriptstyle >}\quad ,$$

$$Fr(s,f) \;=\; s{\scriptstyle >}\circ G\circ s{\scriptstyle >\bullet}\quad ,$$

$$\mathit{Grey}(s,f) = s{\scriptstyle >}\circ f{\scriptstyle >\bullet}\quad \text{, and}$$

$$(s',f')\;[\![s\mathit{Inc}]\!]\;(s,f) \;\equiv\; s{\scriptstyle >\bullet}\circ s'^{\cup}\circ\leq\circ s = \perp\!\!\perp\quad .$$

□

The reader is invited to compare invariant properties (13.8) thru (13.13) with invariant properties (13.1) thru (13.5) of the outer loop. Properties (13.12) and (13.4) are identical; the remainder are almost identical with some small differences.

Property (13.8) is weaker than (13.1): the equality $f{\scriptstyle>} = s{\scriptstyle>}$ has been weakened to $f{\scriptstyle>} \subseteq s{\scriptstyle>}$. This is the same weakening made in section 11.1 where the equality $fnd = seen$ was weakened to $fnd \subseteq seen$.

The property (13.11) is missing from the invariant properties of the outer loop. It asserts that there is a path from node a to node b if a is grey and the search from a started before the search from b. In the outer loop $s{\scriptstyle>} \circ f{\scriptstyle>\!\bullet} = \perp\!\!\perp$ —it is the invariant $seen \circ {\sim}fnd = \perp\!\!\perp$ discussed in section 11.1— so (13.11) is easily shown to be true.

Finally, (13.13) differs from (13.5) in that it includes an additional domain restriction "$f{\scriptstyle>}$". In the outer loop, the domain restriction is superfluous because $s{\scriptstyle>}$ and $f{\scriptstyle>}$ are equal.

The property $P(a,s,f)$ should be compared with the properties used to instantiate $p.a$ when reasoning about the implementation of depth-first search shown in fig. 12.3. They are identical but for the replacement of $seen$ by $s{\scriptstyle>}$, ${\sim}seen$ by $s{\scriptstyle>\!\bullet}$ and ${\sim}fnd$ by $f{\scriptstyle>\!\bullet}$. We exploit this fact later.

The relations MT^2, $Fr^{\cup} \circ (\subseteq) \circ Fr$ and $Grey^{\cup} \circ Grey$ have been considered in depth in section 11.1 —albeit before the replacement of the variable $seen$ by $s{\scriptstyle>}$ and fnd by $f{\scriptstyle>}$— . Consequently, we mention their verification only briefly below. (The term $Grey^{\cup} \circ Grey$ expresses the property that the value of $s{\scriptstyle>} \circ f{\scriptstyle>\!\bullet}$ is an invariant value; this is equivalent to the invariance of the value of $seen \circ {\sim}fnd$ discussed in section 12.3.)

The term $Inv^{\cup} \circ (\Leftarrow) \circ Inv$ states that Inv is an invariant property. The invariant Inv is different from the invariant property in section 12.3 because it expresses properties of the orderings on start and finish times. Nevertheless, we use the same techniques to verify its validity.

Fig. 13.2 summarises the implementation of $dfs(a)$ with assertions bracketing each statement.

Conditional correctness is established using the induction theorem, theorem 12.5. The term $p.a$ in theorem 12.5 is instantiated to the coreflexive corresponding to the property $P(a,s,f) \wedge Inv(s,f)$; similarly, the term $q.a$ is the coreflexive corresponding to the property $Q(a,s,f) \wedge Inv(s,f)$. The relation R is instantiated to $Invrel$ (see definition 13.7). The task is thus to verify (12.6), (12.7), (12.8) and (12.9) with these instantiations.

Because of the number of clauses that have to be established, a large number of calculations have to be carried out. We begin with (12.9). Although it is typically the hardest to verify, the groundwork that we have done in section 11 means that it is relatively easy to verify.

```
    { **Invariant Relation**: $MT^2 \cap Fr^{\cup} \circ (\subseteq) \circ Fr \cap sInc$
      **Invariant Property**: $Inv(s,f)$
      **Invariant Value**: $s{>} \circ f{>}\bullet$
      **Invariant Value**: $s^{\cup} \circ \leq \circ s \circ f{>}\bullet$  }
    {  $P(a,s,f) \wedge Inv(s,f)$  }
    $s := s \cup \overline{(MAX.s \uparrow MAX.f)+1} \circ \top\!\!\top \circ a$
    { **Inner Loop Invariant**:  $Q(a,s,f) \wedge Inv(s,f)$  }
;   **while** $a \circ G \circ s{>}\bullet \neq \perp\!\!\perp$ **do**
      **begin**
        choose node $b$ such that $a \circ \top\!\!\top \circ b \subseteq a \circ G \circ s{>}\bullet$
      ; {  $P(b,s,f) \wedge Inv(s,f)$  }
        $dfs(b)$
        {  $Q(a,s,f) \wedge Inv(s,f)$  }
      **end**
    {  $a \circ G \circ s{>}\bullet = \perp\!\!\perp \wedge Q(a,s,f) \wedge Inv(s,f)$  }
;   $f := f \cup \overline{(MAX.s \uparrow MAX.f)+1} \circ \top\!\!\top \circ a$
```

Figure 13.2: Timestamps: The Procedure $dfs(a)$

13.1.2 The Relation Invrel

Comparing the relation Invrel with the invariant relation discussed in section 12.3, the changes that have been made are the addition of the the relation $s\mathit{Inc}$, and the changes to the invariant properties captured by Inv. In this section, we verify (12.9) for $s\mathit{Inc}$.

(The verification of (12.9) for the relation $\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv}$ is the subject of later sections: because it is reflexive and transitive,

$$
\begin{array}{ll}
 & \mathsf{F}\mathbin{\dot\circ}\mathsf{t}\mathbin{\dot\circ}\mathsf{K}.(\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv})^{*}\mathbin{\dot\circ}\mathsf{S}\mathbin{\dot\circ}\mathsf{p} \;\;\dot{\subseteq}\;\; \mathsf{K}.(\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv}) \\
\Leftarrow & \mathsf{F}\mathbin{\dot\circ}\mathsf{t}\mathbin{\dot\circ}\mathsf{q} \;\;\dot{\subseteq}\;\; \mathsf{K}.(\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv}) \;\;\wedge\;\; \mathsf{S}\mathbin{\dot\circ}\mathsf{p} \;\;\dot{\subseteq}\;\; \mathsf{K}.(\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv}) \;\; .
\end{array}
$$

That is, we show that $\mathit{Inv}(\mathsf{s},\mathsf{f})$ is an invariant property of the final assignment to f and of the initial assignment to s, assuming the preconditions $\mathsf{t}\mathbin{\dot\circ}\mathsf{q}$ and p.)

We first show that (in combination with other relations) $s\mathit{Inc}$ is reflexive and transitive. This means that the verification of (12.9) for this new relation can be decomposed into verifying that the initial assignment to s maintains the relation, and that the final assignment to f also maintains the relation. The latter is trivially true (because $s\mathit{Inc}$ is independent of f) so only the simple proof of the former is required.

We have already seen that the relations MT^2, $\mathsf{Fr}^{\cup}\circ(\subseteq)\circ\mathsf{Fr}$ and $\mathit{Inv}^{\cup}\circ(\Leftarrow)\circ\mathit{Inv}$ are transitive. The remaining relation, $s\mathit{Inc}$, is *not* transitive. However, in combination with the other relations, it is:

Lemma 13.14 When restricted to states (s,f) such that s is functional, the relation

$$s\mathit{Inc}\cap\mathsf{MT}^2$$

is transitive. With the same restriction, the relation

$$s\mathit{Inc}\cap(\mathsf{I}\cup\mathsf{MT}^2)$$

is reflexive and transitive. (The relation $(\mathsf{I}\cup\mathsf{MT}^2)$ is ME^2 where ME is the superset relation on coreflexives.)

Proof The relation $s\mathit{Inc}$ is reflexive since

$$
\begin{array}{ll}
 & \mathsf{I}\subseteq s\mathit{Inc} \\
= & \{ \qquad \text{type of } s\mathit{Inc} \qquad \} \\
 & \langle\forall\,\mathsf{s},\mathsf{f} :: \mathsf{s}{\succ\!\bullet}\circ\mathsf{s}^{\cup}\circ\leq\circ\,\mathsf{s} = \perp\!\!\!\perp\rangle \\
= & \{ \qquad \mathsf{s}^{\cup} = \mathsf{s}{\scriptstyle>}\circ\mathsf{s}^{\cup} \qquad \} \\
 & \langle\forall\,\mathsf{s},\mathsf{f} :: \mathsf{s}{\succ\!\bullet}\circ\mathsf{s}{\scriptstyle>}\circ\mathsf{s}^{\cup}\circ\leq\circ\,\mathsf{s} = \perp\!\!\!\perp\rangle \\
= & \{ \qquad \mathsf{s}{\succ\!\bullet}\circ\mathsf{s}{\scriptstyle>} = \perp\!\!\!\perp\,,\;\; \perp\!\!\!\perp \text{ is zero of composition} \qquad \} \\
 & \mathsf{true} \;\; .
\end{array}
$$

The relation $I \cup MT^2$ is also reflexive. It follows that the intersection of the two relations is reflexive.

That $s\mathcal{I}nc \cap MT^2$ is transitive follows from the following calculation.

$$\begin{array}{cl}
 & s0{\scriptstyle >\!\!\bullet} \circ s2^{\cup} \circ \leq \circ s0 \\
= & \{ \quad I = s1{\scriptstyle >} \cup s1{\scriptstyle >\!\!\bullet} \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1{\scriptstyle >} \circ s2^{\cup} \circ \leq \circ s0 \;\cup\; s0{\scriptstyle >\!\!\bullet} \circ s1{\scriptstyle >\!\!\bullet} \circ s2^{\cup} \circ \leq \circ s0 \\
\subseteq & \{ \quad \text{assume: } s0 \subseteq s1 \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1{\scriptstyle >} \circ s2^{\cup} \circ \leq \circ s0 \;\cup\; s0{\scriptstyle >\!\!\bullet} \circ s1{\scriptstyle >\!\!\bullet} \circ s2^{\cup} \circ \leq \circ s1 \\
= & \{ \quad \text{assume: } s1{\scriptstyle >\!\!\bullet} \circ s2^{\cup} \circ \leq \circ s1 = \bot\!\!\bot \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1{\scriptstyle >} \circ s2^{\cup} \circ \leq \circ s0 \\
\subseteq & \{ \quad \text{domains (specifically } [\; R{\scriptstyle >} = I \cap R^{\cup} \circ R \;] \text{ with } R{:=}s1\text{)} \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1^{\cup} \circ s1 \circ s2^{\cup} \circ \leq \circ s0 \\
\subseteq & \{ \quad \text{assume: } s1 \subseteq s2 \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1^{\cup} \circ s2 \circ s2^{\cup} \circ \leq \circ s0 \\
\subseteq & \{ \quad \text{(13.8) (specifically, } s2 \text{ is functional) and monotonicity} \quad \} \\
 & s0{\scriptstyle >\!\!\bullet} \circ s1^{\cup} \circ \leq \circ s0 \\
= & \{ \quad \text{assume: } s0{\scriptstyle >\!\!\bullet} \circ s1^{\cup} \circ \leq \circ s0 = \bot\!\!\bot \quad \} \\
 & \bot\!\!\bot \;\; .
\end{array}$$

Summarising, we have shown that, for all $s0$, $s1$, $s2$, $f0$, $f1$ and $f2$

$$\begin{array}{cl}
 & (s2,f2)\,[\![s\mathcal{I}nc]\!]\,(s0,f0) \\
\Leftarrow & (s2,f2)\,[\![s\mathcal{I}nc]\!]\,(s1,f1) \;\wedge\; (s1,f1)\,[\![s\mathcal{I}nc]\!]\,(s0,f0) \\
\wedge & s0 \subseteq s1 \subseteq s2 \;\wedge\; s2{\scriptstyle >} = s2^{\cup} \circ s2 \quad .
\end{array}$$

Since MT^2 is transitive, it follows that $s\mathcal{I}nc \cap MT^2$ is transitive under the stated restriction. (The final conjunct, $s2{\scriptstyle >} = s2^{\cup} \circ s2$, is the reason for introducing the type restriction.)
□

Lemma 13.14 requires that the variable s in the definition of $s\mathcal{I}nc$ is functional. This is a consequence of the property $\mathcal{I}nv(s,f)$ which we show to be an invariant property in the subsections that follow this one.

Corollary 13.15 The relation $\mathcal{I}nvrel$ is transitive.

Proof The intersection of transitive relations is transitive. (This is well-known. Its easy (point-free) proof is left as an exercise for the reader.) Thus, combining lemma 13.14 with the known transitivity of MT^2, $Fr^{\cup}\circ(\subseteq)\circ Fr$, $Grey^{\cup}\circ Grey$ and $Inv^{\cup}\circ(\Leftarrow)\circ Inv$, we conclude that $Invrel$ is transitive.
□

Lemma 13.16 The property (12.9) is valid with R instantiated to $sInc\cap ME^2$. To be precise,

$$F \,\dot{\circ}\, K.Invrel \,\dot{\circ}\, S \,\dot{\circ}\, p \;\;\dot{\subseteq}\;\; K.sInc$$

where F and S are the timestamp assignments to s and f, respectively, and $p.a$ is the coreflexive corresponding to the assertion $P(a,s,f)\wedge Inv(s,f)$.

Proof We begin by showing that

$$K.Invrel \,\dot{\circ}\, S \,\dot{\circ}\, p \;\;\dot{\subseteq}\;\; K.sInc \;\; .$$

With m denoting $\overline{(MAX.s0\uparrow MAX.f0)+1}$, we have:

$$
\begin{array}{ll}
 & (s1,f1)[\![Invrel\circ S.a\circ p.a]\!](s0,f0) \\
\Rightarrow & \{ \quad \text{definitions of } F,\ S \text{ and } p \quad \} \\
 & \quad (s1,f1)[\![Invrel]\!](s0\cup m\circ\top\!\!\top\circ a\ ,\ f0) \\
\wedge & \quad s0{\succ}\!\bullet\circ{\sim}a\circ s1^{\cup}\circ{\leq}\circ(s0\cup m\circ\top\!\!\top\circ a) = \bot\!\!\bot \\
\wedge & \quad s0\circ a = \bot\!\!\bot \;\;\wedge\;\; Inv(s0,f0) \\
\Rightarrow & \{ \quad \text{distributivity and properties of } \bot\!\!\bot \quad \} \\
 & (s1,f1)[\![Invrel]\!](s0\cup m\circ\top\!\!\top\circ a\ ,\ f0) \\
\wedge & s0{\succ}\!\bullet\circ{\sim}a\circ s1^{\cup}\circ{\leq}\circ s0 \;=\; \bot\!\!\bot \\
\wedge & s0\circ a = \bot\!\!\bot \;\;\wedge\;\; Inv(s0,f0) \;\; .
\end{array}
$$

Also,

$$(s1,f1)[\![K.sInc]\!](s0,f0) \;\;\equiv\;\; s0{\succ}\!\bullet\circ s1^{\cup}\circ{\leq}\circ s0 = \bot\!\!\bot \;\; .$$

Our goal is thus to prove that, for all a, m, $s0$, $s1$, $f0$ and $f1$,

$$
\begin{array}{ll}
 & (s1,f1)[\![Invrel]\!](s0\cup m\circ\top\!\!\top\circ a\ ,\ f0) \\
\wedge & s0{\succ}\!\bullet\circ{\sim}a\circ s1^{\cup}\circ{\leq}\circ s0 \;=\; \bot\!\!\bot \\
\wedge & s0\circ a = \bot\!\!\bot \;\;\wedge\;\; Inv(s0,f0) \\
\Rightarrow & s0{\succ}\!\bullet\circ s1^{\cup}\circ{\leq}\circ s0 = \bot\!\!\bot \;\; .
\end{array}
$$

Unusually, we begin with the simpler side (because it is not immediately clear which components of $Invrel$ are required).

$$
\begin{array}{lll}
 & s0{\succ\!\!\bullet}\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\Leftarrow & \{ \quad \text{case analysis on } \sim\!a\cup a \quad \} & \\
 & \quad s0{\succ\!\!\bullet}\circ\sim\!a\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad s0{\succ\!\!\bullet}\circ a\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\Leftarrow & \{ \quad \text{introduce assumption: } s0{\circ}a=\perp\!\!\perp\,; & \\
 & \qquad \text{equivalently, } s0{\succ\!\!\bullet}\circ a \;=\; a \quad \} & \\
 & \quad s0{\succ\!\!\bullet}\circ\sim\!a\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad a\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad s0{\circ}a=\perp\!\!\perp & \\
\Leftarrow & \{ \quad \text{1st conjunct: monotonicity;} & \\
 & \qquad \text{2nd conjunct: introduce assumption } s1{\circ}a \;=\; m{\circ}\top\!\!\!\top{\circ}a \quad \} & \\
 & \quad s0{\succ\!\!\bullet}\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad a\circ\top\!\!\!\top\circ m\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad s0{\circ}a=\perp\!\!\perp \;\wedge\; s1{\circ}a \;=\; m{\circ}\top\!\!\!\top{\circ}a & \\
= & \{ \quad \text{by definition of } m\,,\; m\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp \quad \} & \\
 & \quad s0{\succ\!\!\bullet}\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad s0{\circ}a=\perp\!\!\perp \;\wedge\; s1{\circ}a \;=\; m{\circ}\top\!\!\!\top{\circ}a & \\
\Leftarrow & \{ \quad \text{lemma 5.49 with } f,h := m{\circ}\top\!\!\!\top{\circ}a\,,\,s1{\circ}a & \\
 & \qquad (\,m{\circ}\top\!\!\!\top{\circ}a \text{ is functional with right domain } a\,, & \\
 & \qquad \text{so too is } s1{\circ}a \text{ is if } s1 \text{ is functional)} \quad \} & \\
 & \quad s0{\succ\!\!\bullet}\circ s1^{\cup}\circ\underline{\leq}\circ s0 \;=\; \perp\!\!\perp & \\
\wedge & \quad s0{\circ}a=\perp\!\!\perp & \\
\wedge & \quad s1 \supseteq m{\circ}\top\!\!\!\top{\circ}a \;\wedge\; s1\circ s1^{\cup} = s1{<} \quad . &
\end{array}
$$

Comparing the goal with what has just been established, we have to prove that

$$
\begin{array}{ll}
 & (s1,f1)[\![Invrel]\!](s0\cup m\circ\top\!\!\!\top\circ a\;,\; f0) \;\wedge\; Inv(s0,f0) \\
\Rightarrow & s1\supseteq m{\circ}\top\!\!\!\top{\circ}a \;\wedge\; s1\circ s1^{\cup} = s1{<} \quad .
\end{array}
$$

The first conjunct follows from the MT^2 component of $Invrel$, and the second and third conjuncts follow from the $Inv^{\cup}\circ(\Leftarrow)\circ Inv$ component of $Invrel$, in particular (13.8).

That we can now deduce that

$$F \dot{\circ} K.Invrel \dot{\circ} S \dot{\circ} p \ \dot{\subseteq}\ K.sInc$$

is a straightforward consequence of the property

$$(s1,f1)[\![F]\!](s0,f0) \ \Rightarrow\ s1 = s0 \quad ,$$

for all $s1$, $f1$, $s0$ and $f0$, and $sInc(s,f)$ is independent of f. (The details are left to the reader.)
□

The following theorem summarises the results of this section.

Theorem 13.17 The verification condition (12.9) is valid if it is valid for the invariant properties expressed by Inv. That is,

$$(12.9)[R \ := \ MT^2 \ \cap\ Fr^{\cup}\circ(\subseteq)\circ Fr \ \cap\ Grey^{\cup}\circ Grey \ \cap\ sInc]$$

and it remains to prove

$$(12.9)[R \ := \ Inv^{\cup}\circ(\Leftarrow)\circ Inv] \ \ .$$

Proof The relations MT^2, $Fr^{\cup}\circ(\subseteq)\circ Fr$ and $Grey^{\cup}\circ Grey$ involve only the right domains of s and f and, consequently, that (12.9) is valid for them was established in section 12.3 (with $seen$ and fnd taking the place of $s{>}$ and $f{>}$, respectively). Lemma 13.16 establishes (12.9) for the relation $sInc$. (Recall the discussion in section 12.2 of how the proof obligations are broken down.)
□

13.1.3 Assigning Start Times

The specification of the assignment to s is that, assuming precondition P, it truthifies the property Q whilst maintaining the invariant property Inv. That is, we must verify that

$$\{\ \ P(a,s,f) \wedge Inv(s,f)\ \ \}$$
$$s := s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a$$
$$\{\ \ Q(a,s,f) \ \wedge\ Inv(s,f)\ \ \}$$

for all a, s and f.

Because of the number of conjuncts in the postcondition (ten in total!), the calculation is inevitably long.

We begin with a lemma on the effect of the assignment on subterms of Q and Inv.

Lemma 13.18

$$(s^{\cup}\circ\leq\circ s)[s := s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\top\circ a] \;=\; s^{\cup}\circ\leq\circ s \;\cup\; (s{>}\cup a)\circ\top\!\top\circ a \quad ,$$

$$(s^{\cup}\circ<\circ s)[s := s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\top\circ a] \;=\; s^{\cup}\circ<\circ s \;\cup\; s{>}\circ\top\!\top\circ a \quad \text{, and}$$

$$(s{>})[s := s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\top\circ a] \;=\; s{>}\cup a \quad .$$

Proof In the calculations below, we use m to denote $\overline{(MAX.s\uparrow MAX.f){+}1}$; that is, m is a coreflexive representing a natural number that is strictly greater than $MAX.s\uparrow MAX.f$. It is thus a proper atom and

$$\begin{array}{ll} & m\circ\leq\circ s \;=\; \perp\!\!\perp \;=\; m\circ<\circ s \\ \wedge & s^{\cup}\circ\leq\circ m \;=\; s{>}\circ\top\!\top\circ m \;=\; s^{\cup}\circ<\circ m \\ \wedge & m\circ\leq\circ m \;=\; m \quad . \end{array}$$

We have:

$$\begin{array}{ll} & (s^{\cup}\circ\leq\circ s)[s := s\cup m\circ\top\!\top\circ a] \\ = & \{\quad \text{definition of substitution and distributivity,} \\ & \qquad a \text{ and } m \text{ are coreflexives, so } a=a^{\cup} \text{ and } m=m^{\cup} \quad \} \\ & \quad s^{\cup}\circ\leq\circ s \;\cup\; a\circ\top\!\top\circ m\circ\leq\circ s \\ & \cup\quad s^{\cup}\circ\leq\circ m\circ\top\!\top\circ a \;\cup\; a\circ\top\!\top\circ m\circ\leq\circ m\circ\top\!\top\circ a \\ = & \{\quad \text{defining properties of } m \text{ (see above)} \quad \} \\ & s^{\cup}\circ\leq\circ s \;\cup\; s{>}\circ\top\!\top\circ m\circ\top\!\top\circ a \;\cup\; a\circ\top\!\top\circ m\circ\top\!\top\circ a \\ = & \{\quad m\neq\perp\!\!\perp\text{, cone rule and distributivity} \quad \} \\ & s^{\cup}\circ\leq\circ s \;\cup\; (s{>}\cup a)\circ\top\!\top\circ a \quad . \end{array}$$

Similarly, we have:.

$$\begin{array}{ll}
& (s^{\cup}\circ{<}\circ s)[s \;:=\; s \cup m\circ\top\!\!\top\circ a] \\
= & \{ \quad \text{definition of substitution and distributivity,} \\
& \quad\;\; a \text{ and } m \text{ are coreflexives, so } a = a^{\cup} \text{ and } m = m^{\cup} \quad \} \\
& \quad s^{\cup}\circ{<}\circ s \;\cup\; a\circ\top\!\!\top\circ m\circ{<}\circ s \\
& \cup \quad s^{\cup}\circ{<}\circ m\circ\top\!\!\top\circ a \;\cup\; a\circ\top\!\!\top\circ m\circ{<}\circ m\circ\top\!\!\top\circ a \\
= & \{ \quad \text{defining properties of } m \text{ (see above)} \quad \} \\
& s^{\cup}\circ{<}\circ s \;\cup\; s{\scriptstyle >}\circ\top\!\!\top\circ m\circ\top\!\!\top\circ a \\
= & \{ \quad m \neq \bot\!\!\bot\text{, cone rule and distributivity} \quad \} \\
& s^{\cup}\circ{<}\circ s \;\cup\; s{\scriptstyle >}\circ\top\!\!\top\circ a \quad .
\end{array}$$

(It is also possible to derive the second assertion from the first using the obvious relation between less-than and at-most. The copy-and-paste we have just used is quicker.) Finally,

$$\begin{array}{ll}
& (s{\scriptstyle >})[s \;:=\; s \cup m\circ\top\!\!\top\circ a] \\
= & \{ \quad \text{definition of substitution and distributivity} \quad \} \\
& s{\scriptstyle >} \cup (m\circ\top\!\!\top\circ a){\scriptstyle >} \\
= & \{ \quad \text{domains} \quad \} \\
& s{\scriptstyle >} \cup (\top\!\!\top\circ m\circ\top\!\!\top\circ a){\scriptstyle >} \\
= & \{ \quad m \neq \bot\!\!\bot\text{, cone rule, domains and } a \text{ is coreflexive} \quad \} \\
& s{\scriptstyle >} \cup a \quad .
\end{array}$$

□

Lemma 13.19 The property $Q(a{,}s{,}f)$ is truthified by the assignment to s. That is,

$$(Q(a{,}s{,}f))[s \;:=\; s \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a] \;\Leftarrow\; P(a{,}s{,}f)\wedge Inv(s{,}f) \quad .$$

Proof There are four conjuncts in the definition of $Q(a{,}s{,}f)$:

$$\begin{array}{ll}
& a\circ s{\scriptstyle >}\circ f{\scriptstyle >\!\bullet} \;=\; a \\
\wedge & f{\scriptstyle >}\circ s^{\cup}\circ{<}\circ s\circ a \;=\; f^{\cup}\circ{<}\circ s\circ a \\
\wedge & a\circ s^{\cup}\circ{<}\circ s\circ f{\scriptstyle >\!\bullet} \;=\; \bot\!\!\bot \\
\wedge & f{\scriptstyle \bullet\!>}\circ s{\scriptstyle >}\circ\top\!\!\top\circ a \;\subseteq\; (s{\scriptstyle >}\circ f{\scriptstyle >\!\bullet}\circ G)^{*} \quad .
\end{array}$$

(Because they depend only on the domains $s{>}$ and $f{>\!\!\bullet}$, the first and last conjuncts have effectively been verified already in section 12.3. Nevertheless, we repeat the proofs here to show the additional elements in the calculation.)

The validity of the postcondition $a \circ s{>} \circ f{>\!\!\bullet} = a$ is straightforward:

$$
\begin{array}{ll}
 & (a \circ s{>} \circ f{>\!\!\bullet})[s \;:=\; s \cup m \circ \top\!\!\top \circ a] \\
= & \{ \quad \text{definition of substitution, distributivity} \quad \} \\
 & a \circ s{>} \circ f{>\!\!\bullet} \;\cup\; a \circ (m \circ \top\!\!\top \circ a){>} \\
= & \{ \quad \text{domains} \quad \} \\
 & a \circ s{>} \circ f{>\!\!\bullet} \;\cup\; a \circ a \\
= & \{ \quad \text{assumption: } P(a,s,f)\text{, hence } a \circ s{>} = \bot\!\!\bot\,; \\
 & \quad\;\; a \text{ is a coreflexive, so } a \circ a = a \quad \} \\
 & a \;\;.
\end{array}
$$

That is (for arbitrary m),

$$
(a \circ s{>} \circ f{>\!\!\bullet} = a)[s \;:=\; s \cup m \circ \top\!\!\top \circ a] \;\Leftarrow\; a \circ s{>\!\!\bullet} \circ f{>\!\!\bullet} = a
$$

as required.

That the second is truthified is also obvious[3]. With m denoting $\overline{(MAX.s \uparrow MAX.f) + 1}$, we have:

$$
\begin{array}{ll}
 & (f{>} \circ s^{\cup} \circ {<} \circ s \circ a \;=\; f^{\cup} \circ {<} \circ s \circ a)[s \;:=\; s \cup m \circ \top\!\!\top \circ a] \\
= & \{ \quad \text{substitution and lemma 13.18,} \\
 & \quad\;\; \text{assumption: } P(a,s,f)\text{, hence } a \circ s{>} = \bot\!\!\bot \quad \} \\
 & f{>} \circ s{>} \circ \top\!\!\top \circ a \;=\; f^{\cup} \circ {<} \circ m \circ \top\!\!\top \circ a \\
= & \{ \quad \text{assumption: } Inv(s,f)\text{, in particular } f{>} \subseteq s{>}\;; \\
 & \quad\;\; \text{by definition of } m,\; f^{\cup} \circ {<} \circ m = f{>} \circ \top\!\!\top \circ m \quad \} \\
 & f{>} \circ \top\!\!\top \circ a \;=\; f{>} \circ \top\!\!\top \circ m \circ \top\!\!\top \circ a \\
= & \{ \quad \top\!\!\top \circ m \circ \top\!\!\top = \top\!\!\top \quad \} \\
 & \mathsf{true} \;\;.
\end{array}
$$

That the third is truthified is slightly less obvious:

[3] This is perhaps not obvious in the point-free form. This is one case where the pointwise formulation is clearer.

$(a \circ s^{\cup} \circ < \circ s \circ f{>\!\bullet})[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a]$

$=$ { substitution and lemma 13.18 }

$a \circ (s^{\cup} \circ < \circ s \;\cup\; s{>} \circ \top\!\top \circ a) \circ f{>\!\bullet}$

$=$ { distributivity and domains }

$a \circ s{>} \circ (s^{\cup} \circ < \circ s \;\cup\; s{>} \circ \top\!\top \circ a) \circ f{>\!\bullet}$

$=$ { assumption: $P(a,s,f)$, hence $a \circ s{>} = \bot\!\bot$ }

$\bot\!\bot$.

The validity of the fourth is established as follows.

$(f{>\!\bullet} \circ s{>} \circ \top\!\top \circ a \;\subseteq\; (s{>} \circ f{>\!\bullet} \circ G)^*)[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a]$

$=$ { substitution and lemma 13.18 }

$f{>\!\bullet} \circ (s{>} \cup a) \circ \top\!\top \circ a \;\subseteq\; ((s{>} \cup a) \circ f{>\!\bullet} \circ G)^*$

$\Leftarrow$ { distributivity, suprema }

$f{>\!\bullet} \circ s{>} \circ \top\!\top \circ a \;\subseteq\; (s{>} \circ f{>\!\bullet} \circ G)^* \;\wedge\; f{>\!\bullet} \circ a \circ \top\!\top \circ a \;\subseteq\; (a \circ f{>\!\bullet} \circ G)^*$

$\Leftarrow$ { $a \circ \top\!\top \circ a = a$, $a \subseteq I$, $f{>\!\bullet} \subseteq I$, $I \subseteq G^*$, and monotonicity }

$f{>\!\bullet} \circ s{>} \circ \top\!\top \circ a \;\subseteq\; (s{>} \circ f{>\!\bullet} \circ G)^*$

$\Leftarrow$ { assumption: $P(a,s,f)$,

in particular $f{>\!\bullet} \circ s{>} \circ \top\!\top \circ a \;\subseteq\; (s{>} \circ f{>\!\bullet} \circ G)^*$ }

true .

□

We now consider in turn each of the conjuncts of $Inv(s,f)$ and show that they are invariants of the assignment to s .

Lemma 13.20 The property (13.8) is an invariant of the assignment to s .

Proof We leave this to the reader. The calculation is very similar to the one in lemma 10.21.

□

Lemma 13.21 The property (13.9) is an invariant of the assignment to s .

Proof This a trivial consequence of the theorem $s \subseteq s \cup a$ for all s and a .

□

Lemma 13.22 The property (13.10) is an invariant of the assignment to s . Specifically,

$$(13.10)\,[s\ :=\ s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\ \Leftarrow\ (13.10)\wedge P(a,s,f)\ \ .$$

Proof Obviously

$$(f_{>}\circ G\circ s_{\succ\bullet}\ =\ \bot\!\!\bot)[s\ :=\ s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\ \ \Leftarrow\ \ f_{>}\circ G\circ s_{\succ\bullet}\ =\ \bot\!\!\bot\ \ .$$

(Formally, monotonicity is the key: the assignment increases s and decreases $s_{\succ\bullet}$.) For the second conjunct, we have

$$
\begin{array}{ll}
 & (\bot\!\!\bot\ =\ f^{\cup}\circ{<}\circ s\cap G)[s\ :=\ s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\\
\Leftarrow & \{\quad \text{definition of substitution and (13.10)}\quad\}\\
 & \bot\!\!\bot\ =\ f^{\cup}\circ{<}\circ\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a\cap G\\
= & \{\quad \text{property of } MAX\text{:}\ \ (f^{\cup}\circ{<}\circ\overline{(MAX.s\uparrow MAX.f){+}1})_{<}\ =\ f_{>}\,,\\
 & \qquad \text{domains}\quad\}\\
 & \bot\!\!\bot\ =\ f_{>}\circ G\circ a\\
= & \{\quad \text{assumptions: } P(a,s,f)\text{, so } a\subseteq s_{\succ\bullet}\text{, and (13.10)}\quad\}\\
 & \mathsf{true}\ \ .
\end{array}
$$

□

Lemma 13.23 The property (13.11) is an invariant of the assignment to s. Specifically,

$$(13.11)\,[s\ :=\ s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\ \Leftarrow\ (13.11)\ \wedge\ P(a,s,f)\ \ .$$

Proof

$$
\begin{array}{ll}
 & (f_{\succ\bullet}\circ s^{\cup}\circ{\le}\circ s)[s\ :=\ s\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\\
= & \{\quad \text{definition of substitution and lemma 13.18}\quad\}\\
 & f_{\succ\bullet}\circ(s^{\cup}\circ{\le}\circ s\ \cup\ (s_{>}\cup a)\circ\top\!\!\top\circ a)\\
= & \{\quad \text{distributivity, } a\circ\top\!\!\top\circ a = a\quad\}\\
 & f_{\succ\bullet}\circ s^{\cup}\circ{\le}\circ s\ \cup\ f_{\succ\bullet}\circ s_{>}\circ\top\!\!\top\circ a\ \cup\ f_{\succ\bullet}\circ a\\
\subseteq & \{\quad \text{assumption: } P(a,s,f)\\
 & \qquad \text{in particular, } f_{\succ\bullet}\circ s_{>}\circ\top\!\!\top\circ a\ \subseteq\ G^{*}\\
 & \qquad \text{and } f_{\succ\bullet}\circ a = a\quad\}\\
 & f_{\succ\bullet}\circ s^{\cup}\circ{\le}\circ s\ \cup\ f_{\succ\bullet}\circ s_{>}\circ G^{*}\circ a\ \cup\ a\\
\subseteq & \{\quad \text{assumption: (13.11),}
\end{array}
$$

$$
\begin{array}{ll}
 & \qquad a\,,\ f{>}\!\bullet \text{ and } s{>} \text{ are all coreflexives} \quad \} \\
 & G^* \cup G^* \cup I \\
= & \quad \{ \qquad \text{definition of } G^*\text{, idempotency of union,} \\
 & \qquad\quad \text{substitution} \quad \} \\
 & (G^*)[s \ := \ s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \ .
\end{array}
$$

□

Lemma 13.24 Property (13.12) is an invariant of the assignment to s. Specifically,

$$(13.12)\,[s \ := \ s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \quad \Leftarrow \quad (13.12) \ \wedge \ a \circ s{>}\!\bullet \circ f{>}\!\bullet \ = \ a \ .$$

Proof It suffices to prove that the left side of (13.12) is invariant under the assignment.

$$
\begin{array}{ll}
 & (s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f)[s \ := \ s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \\
= & s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f
\end{array}
$$

assuming that $a \circ s{>}\!\bullet \circ f{>}\!\bullet \ = \ a$.

$$
\begin{array}{ll}
 & (s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f)[s \ := \ s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \\
= & \quad \{ \qquad \text{definition of substitutivity and lemma 13.18} \quad \} \\
 & (s^{\cup} \circ \leq \circ s \ \cup \ (s^{\cup} \cup a) \circ \top\!\!\top \circ a) \ \cap \ f^{\cup} \circ \geq \circ f \\
= & \quad \{ \qquad \text{distributivity} \quad \} \\
 & (s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f) \ \cup \ ((s^{\cup} \cup a) \circ \top\!\!\top \circ a \ \cap \ f^{\cup} \circ \geq \circ f) \\
= & \quad \{ \qquad \text{domains} \\
 & \qquad\quad (\text{specifically } R \cap S \ = \ R \circ S{>} \cap S \text{ and } (f^{\cup} \circ \geq \circ f){>} \subseteq f{>}) \quad \} \\
 & (s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f) \ \cup \ ((s^{\cup} \cup a) \circ \top\!\!\top \circ a \circ f{>} \ \cap \ f^{\cup} \circ \geq \circ f) \\
= & \quad \{ \qquad \text{assumption: } a \circ s{>}\!\bullet \circ f{>}\!\bullet \ = \ a \ \text{, hence} \quad a \circ f{>} = \bot\!\!\bot \quad \} \\
 & s^{\cup} \circ \leq \circ s \ \cap \ f^{\cup} \circ \geq \circ f \ .
\end{array}
$$

The lemma now follows from the definition of substitution.

□

Lemma 13.25 Property (13.13) is an invariant of the assignment to s. Specifically,

$$(13.13)[s \;:=\; s \cup \overline{(MAX.s\!\uparrow\!MAX.f){+}1} \circ \top\!\!\top \circ a] \quad \Leftarrow \quad (13.13) \;\wedge\; a \circ s_{>\bullet} \circ f_{>\bullet} \;=\; a \;\;.$$

Proof For brevity, we use m to denote $\overline{(MAX.s\!\uparrow\!MAX.f){+}1}$.

$(13.13)[s \;:=\; s \cup \overline{(MAX.s\!\uparrow\!MAX.f){+}1} \circ \top\!\!\top \circ a]$

$=$ { definition of substitutivity and lemma 13.18 }

$(s^{\cup} \circ {\leq} \circ s \;\cup\; (s^{\cup} \cup a) \circ \top\!\!\top \circ a) \;\cap\; f^{\cup} \circ {<} \circ f \;\;=\;\; f^{\cup} \circ {<} \circ (s \;\cup\; m \circ \top\!\!\top \circ a) \circ f_{>}$

$\Leftarrow$ { distributivity and assumption: (13.13) }

$(s^{\cup} \cup a) \circ \top\!\!\top \circ a \;\cap\; f^{\cup} \circ {<} \circ f \;\;=\;\; f^{\cup} \circ {<} \circ m \circ \top\!\!\top \circ a \circ f_{>}$

$=$ { domains }

$(s^{\cup} \cup a) \circ f^{\cup} \circ {<} \circ f \circ a \;\;=\;\; f^{\cup} \circ {<} \circ m \circ \top\!\!\top \circ a \circ f_{>}$

$=$ { assumption: $a \circ s_{>\bullet} \circ f_{>\bullet} \;=\; a$, hence $a \circ f_{>} = \perp\!\!\perp$ }

$\perp\!\!\perp = \perp\!\!\perp$

$=$ { reflexivity }

true .

□

In summary:

Lemma 13.26 The claimed postcondition in the program segment

{ $P(a,s,f) \;\wedge\; Inv(s,f)$ }

$s \;:=\; s \cup \overline{(MAX.s\!\uparrow\!MAX.f){+}1} \circ \top\!\!\top \circ a$

{ $Q(a,s,f) \;\wedge\; Inv(s,f)$ }

is indeed valid.

□

13.1.4 The Precondition

Next we consider the precondition P of a call of dfs. That is, we establish condition 12.7 of the depth-first search induction theorem, theorem 12.5.

Recall our earlier remark that the property P is effectively identical to the the property used to instantiate $p.a$ when reasoning about the implementation of depth-first search shown in fig. 12.3: one is obtained from the other by replacing $seen$ by $s_{>}$, $\sim\!seen$ by $s_{>\bullet}$ and $\sim\!fnd$ by $f_{>\bullet}$. The reasoning used in section 12.3 (and more specifically section 12.3.2) is therefore applicable here with only minor changes.

The procedure dfs is called from two places: the outer and the inner loops. Also, $P(a,s,f)$ is defined to be

$$a \circ s_{>\bullet} \;=\; a \;\wedge\; s_{>} \circ f_{>\bullet} \circ \top\!\!\top \circ a \;\subseteq\; (s_{>} \circ f_{>\bullet} \circ G)^{*} \quad .$$

The validity of $P(a,s,f)$ when $\mathsf{dfs}(a)$ is called in the outer loop follows from properties established in section 12.3. In the outer loop, $f_{>} = s_{>}$ is an invariant —it is the property $fnd = seen$ established in 12.3— and the choice of a is $a \circ s_{>} \;=\; \perp\!\!\perp$. Thus $a \circ s_{>\bullet} \;=\; a$ follows immediately. The second conjunct follows because the invariant $f_{>} = s_{>}$ implies that $f_{>\bullet} \circ s_{>} = \perp\!\!\perp$.

When $\mathsf{dfs}(b)$ is called in the inner loop, the validity of the two conjuncts was established in section 12.3.2. The calculation given there can be repeated here by replacing $seen$ by $s_{>}$, $\sim\!seen$ by $s_{>\bullet}$ and $\sim\!fnd$ by $f_{>\bullet}$.

This completes the proof that the property P holds when dfs is called from either the outer or the inner loop.

13.1.5 Maintaining the Invariant of the Inner Loop

Recall that our task is to apply theorem 12.5 with the term $q.a$ instantiated to the coreflexive corresponding to the property labelled "Inner Loop Invariant" in fig. 13.2. That is, for all s', f', s and f,

$$(s',f')[\![q.a]\!](s,f) \quad\equiv\quad s'=s \;\wedge\; f'=f \;\wedge\; Q(a,s,f) \;\wedge\; Inv(s,f) \quad .$$

The relation R is instantiated to $Invrel$ (see definition 13.7). In this section, we consider the task of verifying (12.8) with these instantiations. Specifically, we show that, for all a, $s0$, $f0$, $s1$ and $f1$,

$$\begin{array}{ll} & Q(a,s1,f1) \;\wedge\; Inv(s1,f1) \\ \Leftarrow & Q(a,s0,f0) \;\wedge\; Inv(s0,f0) \wedge (s1,f1)\;[\![Invrel]\!]\;(s0,f0) \quad . \end{array}$$

In words, $q.a$ is maintained by $Invrel$.

That the property Inv is maintained by $Invrel$ is immediate: one of the terms in the definition of $Invrel$ is $Inv^{\cup} \circ (\Leftarrow) \circ Inv$. The task is thus to show that the coreflexive corresponding to the boolean function $\langle s,f :: Q(a,s,f)\rangle$ is maintained by $Invrel$, with the additional assumption that Inv is both a valid precondition and postcondition. That is, we prove that

$$\begin{array}{ll} & Q(a,s1,f1) \\ \Leftarrow & Inv(s1,f1) \;\wedge\; Q(a,s0,f0) \;\wedge\; Inv(s0,f0) \;\wedge\; (s1,f1)\;[\![Invrel]\!]\;(s0,f0) \quad . \end{array}$$

The predicate Q captures properties of a that cannot be strengthened further. For example, the conjunct

(13.27) $a \circ s^{\cup} \circ {<} \circ s \circ f{>}\!\bullet \;=\; \perp\!\!\perp$

cannot be strengthened to

$$f{>}\!\bullet \circ s^{\cup} \circ {<} \circ s \circ f{>}\!\bullet \;=\; \perp\!\!\perp \quad .$$

Property (13.27) asserts that, at the beginning of each iteration of the inner loop, node a is the last node from which a search has started but not finished. It is weaker than the property

$$a \circ s^{\cup} \circ {<} \circ s \;=\; \perp\!\!\perp \quad ,$$

which we showed to be established by the assignment to s (see lemma 13.26). The introduction of the term "$f{>}\!\bullet$" is necessary because, during execution of $dfs(a)$, searches from other nodes are started and finished. Because the assertion is weaker, it is clearly true initially. That it is maintained by subsequent executions of $dfs(b)$ in the inner loop is an immediate consequence of Invrel, in particular the properties that $s{>} \circ f{>}\!\bullet$ is invariant and s is increasing. This is proven in lemma 13.31. Similarly, it is shown that the other conjuncts of Q are maintained by the inner loop in lemmas 13.28, 13.30 and 13.32.

The first lemma is easy:

Lemma 13.28 For all a, $s0$, $f0$, $s1$ and $f1$,

$$\begin{array}{ll} & a \circ s1{>} \circ f1{>}\!\bullet \;=\; a \\ \Leftarrow & a \circ s0{>} \circ f0{>}\!\bullet \;=\; a \;\wedge\; (s1,f1)\; [\![\mathit{Invrel}]\!]\; (s0,f0) \quad . \end{array}$$

Proof This is immediate from the conjunct $s1{>} \circ f1{>}\!\bullet = s0{>} \circ f0{>}\!\bullet$ in the definition of Invrel. (See definition 13.7.)
□

Some of the remaining lemmas are not so easy. The following lemma is used repeatedly in the calculations.

Lemma 13.29 For all a, $s0$, $f0$, $s1$ and $f1$,

$$\begin{array}{ll} & s1 \;=\; s0 \cup s1 \circ s0{>}\!\bullet \;\wedge\; f1 \;=\; f0 \cup f1 \circ f0{>}\!\bullet \;\wedge\; s1 \circ a \;=\; s0 \circ a \;\wedge\; s1 \circ s0{>} = s0 \\ \Leftarrow & a \circ s0{>} = a \;\wedge\; (s1,f1)\; [\![\mathit{Invrel}]\!]\; (s0,f0) \quad . \end{array}$$

Proof For the first conjunct, we exploit the key is that $s1 \supseteq s0$ and both $s0$ and $s1$ are injective and functional.

$s1 \;=\; s0 \cup s1 \circ s0{>\bullet}$

$=\qquad \{\qquad$ anti-symmetry $\quad\}$

$s1 \;\subseteq\; s0 \cup s1 \circ s0{>\bullet} \;\;\wedge\;\; s1 \;\supseteq\; s0 \cup s1 \circ s0{>\bullet}$

$=\qquad \{\qquad$ assumption: $(s1, f1)\; [\![\mathit{Invrel}]\!]\; (s0, f0)$,

in particular $s1 \supseteq s0$;

also $I \supseteq s0{>\bullet}$ $\quad\}$

$s1 \;\subseteq\; s0 \cup s1 \circ s0{>\bullet}$

$\Leftarrow\qquad \{\qquad I = s0{>} \cup s0{>\bullet}$, distributivity $\quad\}$

$s1 \circ s0{>} \;\subseteq\; s0$

$=\qquad \{\qquad$ assumption: $(s1, f1)\; [\![\mathit{Invrel}]\!]\; (s0, f0)$,

in particular $s0{>} = s0^{\cup} \circ s0$ $\quad\}$

$s1 \circ s0^{\cup} \circ s0 \subseteq s0$

$\Leftarrow\qquad \{\qquad$ monotonicity $\quad\}$

$s1 \circ s0^{\cup} \subseteq I$

$\Leftarrow\qquad \{\qquad$ assumption: $(s1, f1)\; [\![\mathit{Invrel}]\!]\; (s0, f0)$,

in particular $s1{>} = s1^{\cup} \circ s1$,

$s1{>} \subseteq I$ $\quad\}$

$s0^{\cup} \subseteq s1^{\cup}$

$=\qquad \{\qquad$ assumption: $(s1, f1)\; [\![\mathit{Invrel}]\!]\; (s0, f0)$,

in particular $s0 \subseteq s1$; monotonicity $\quad\}$

true .

The second conjunct is proved in the same way. The final two conjuncts are straightforward.

$s1 \circ a$

$=\qquad \{\qquad s1 \;=\; s0 \cup s1 \circ s0{>\bullet}$, distributivity $\quad\}$

$s0 \circ a \cup s1 \circ s0{>\bullet} \circ a$

$=\qquad \{\qquad$ assumption: $a \circ s0{>} = a$, so $s0{>\bullet} \circ a = \perp\!\!\perp$ $\quad\}$

$s0 \circ a$.

Similarly,

$$
\begin{array}{ll}
 & s1 \circ s0{>} \\
= & \{ \quad s1 \;=\; s0 \cup s1 \circ s0{>\!\bullet} \text{ , distributivity } \quad \} \\
 & s0 \circ s0{>} \cup s1 \circ s0{>\!\bullet} \circ s0{>} \\
= & \{ \quad s0 \circ s0{>} = s0 \, , \;\; s0{>\!\bullet} \circ s0{>} = \perp\!\!\perp \quad \} \\
 & s0 \;\; .
\end{array}
$$

□

We now proceed to establish the maintenance properties as explained earlier.

Lemma 13.30 For all a, $s0$, $f0$, $s1$ and $f1$,

$$
\begin{array}{ll}
 & f1{>} \circ s1^{\cup} \circ {<} \circ s1 \circ a \;=\; f1^{\cup} \circ {<} \circ s1 \circ a \\
\Leftarrow & \quad a \circ s0{>} \circ f0{>\!\bullet} \;=\; a \;\wedge\; f0{>} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; f0^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad Inv(s1,f1) \;\wedge\; Inv(s0,f0) \;\wedge\; (s1,f1) \; [\![Invrel]\!] \; (s0,f0) \quad .
\end{array}
$$

Proof

$$
\begin{array}{ll}
 & f1{>} \circ s1^{\cup} \circ {<} \circ s1 \circ a \;=\; f1^{\cup} \circ {<} \circ s1 \circ a \\
= & \{ \quad s1 \circ a \;=\; s0 \circ a \;\; \text{(see lemma 13.29)} \quad \} \\
 & f1{>} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; f1^{\cup} \circ {<} \circ s0 \circ a \\
\Leftarrow & \{ \quad I = s0{>} \cup s0{>\!\bullet} \;\text{ and distributivity} \quad \} \\
 & \quad f1{>} \circ s0{>} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad f1{>} \circ s0{>\!\bullet} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
= & \{ \quad s1 \circ s0{>} = s0 \;\; \text{(see lemma 13.29)} \quad \} \\
 & \quad f1{>} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad f1{>} \circ s0{>\!\bullet} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
= & \{ \quad f1 \circ f0{>} = f0 \;\; \text{(see lemma 13.29)} \quad \} \\
 & \quad f0{>} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; f0^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad f1{>} \circ f0{>\!\bullet} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>} \circ f0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad f1{>} \circ s0{>\!\bullet} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
= & \{ \quad \text{assumption: } \; f0{>} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; f0^{\cup} \circ {<} \circ s0 \circ a \quad \} \\
 & \quad f1{>} \circ f0{>\!\bullet} \circ s0^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>} \circ f0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
 & \wedge \quad f1{>} \circ s0{>\!\bullet} \circ s1^{\cup} \circ {<} \circ s0 \circ a \;=\; s0{>\!\bullet} \circ f1^{\cup} \circ {<} \circ s0 \circ a \\
= & \{ \quad \text{assumption: } \; (s1,f1) \; [\![Invrel]\!] \; (s0,f0) \;\; ;
\end{array}
$$

in particular $(s1,f1)\ [\![sInc]\!]\ (s0,f0)$, i.e. $s0{>\!\bullet}\circ s1^{\cup}\circ{\leq}\circ s0 = \perp\!\!\perp$ $\}$

$$f1{>}\circ f0{>\!\bullet}\circ s0^{\cup}\circ{<}\circ s0\circ a \;=\; s0{>}\circ f0{>\!\bullet}\circ f1^{\cup}\circ{<}\circ s0\circ a$$

$$\wedge\quad \perp\!\!\perp \;=\; s0{>\!\bullet}\circ f1^{\cup}\circ{<}\circ s0\circ a$$

$\Leftarrow$ $\{$ assumption: $(s1,f1)\ [\![Invrel]\!]\ (s0,f0)$,

in particular $s0{>}\circ f0{>\!\bullet} = s1{>}\circ f1{>\!\bullet}$,

$f1{>}\circ f1{>\!\bullet} = \perp\!\!\perp$ and $\perp\!\!\perp$ is zero of composition $\}$

$$\perp\!\!\perp = \perp\!\!\perp \quad\wedge\quad \perp\!\!\perp \;=\; s0{>\!\bullet}\circ f1^{\cup}\circ{<}\circ s0$$

$\Leftarrow$ $\{$ assumption: $Inv(s1,f1)$; in particular $f1{>}\ \subseteq\ s1^{\cup}\circ{<}\circ f1$ $\}$

$$s0{>\!\bullet}\circ s1^{\cup}\circ{<}\circ f1\circ f1^{\cup}\circ{<}\circ s0 \subseteq \perp\!\!\perp$$

$\Leftarrow$ $\{$ $f1\circ f1^{\cup}\subseteq I$, $<$ is transitive $\}$

$$s0{>\!\bullet}\circ s1^{\cup}\circ{<}\circ s0 \subseteq \perp\!\!\perp$$

$=$ $\{$ assumption: $(s1,f1)\ [\![Invrel]\!]\ (s0,f0)$;

in particular $(s1,f1)\ [\![sInc]\!]\ (s0,f0)$, i.e. $s0{>\!\bullet}\circ s1^{\cup}\circ{\leq}\circ s0 = \perp\!\!\perp$ $\}$

true .

□

Lemma 13.31 For all a , $s0$, $f0$, $s1$ and $f1$,

$$a\circ s1^{\cup}\circ{<}\circ s1\circ f1{>\!\bullet} \;=\; \perp\!\!\perp$$

$$\Leftarrow\ a\circ s0{>} = a \wedge (s1,f1)\ [\![Invrel]\!]\ (s0,f0) \wedge a\circ s0^{\cup}\circ{<}\circ s0\circ f0{>\!\bullet} \;=\; \perp\!\!\perp\ .$$

Proof

$$a\circ s1^{\cup}\circ{<}\circ s1\circ f1{>\!\bullet}$$

$=$ $\{$ by lemma 13.29 and converse, $a\circ s1^{\cup} = a\circ s0^{\cup}$,

also $s1 = s1\circ s1{>}$ $\}$

$$a\circ s0^{\cup}\circ{<}\circ s1\circ s1{>}\circ f1{>\!\bullet}$$

$=$ $\{$ assumption: $(s1,f1)\ [\![Invrel]\!]\ (s0,f0)$,

in particular $s1{>}\circ f1{>\!\bullet} = s0{>}\circ f0{>\!\bullet}$ $\}$

$$a\circ s0^{\cup}\circ{<}\circ s1\circ s0{>}\circ f0{>\!\bullet}$$

$=$ $\{$ $s1\circ s0{>} = s0$ (see lemma 13.29) $\}$

$$a\circ s0^{\cup}\circ{<}\circ s0\circ f0{>\!\bullet}$$

$$= \quad \{ \quad \text{assumption: } a \circ s0^{\cup} \circ < \circ s0 \circ f0{\succ\!\bullet} \;=\; \perp\!\!\perp \quad \}$$

$$\perp\!\!\perp \;.$$

□

Lemma 13.32 For all a, $s0$, $f0$, $s1$ and $f1$,

$$\begin{array}{ll} & f1{\succ\!\bullet} \circ s1{>} \circ \top\!\!\top \circ a \;\subseteq\; G^* \\ \Leftarrow & f0{\succ\!\bullet} \circ s0{>} \circ \top\!\!\top \circ a \;\subseteq\; G^* \;\wedge\; (s1,f1)\;[\![\mathit{Invrel}]\!]\;(s0,f0) \;. \end{array}$$

Proof This is immediate from the conjunct $s1{>} \circ f1{\succ\!\bullet} = s0{>} \circ f0{\succ\!\bullet}$ in the definition of Invrel. (See definition 13.7.)
□

13.1.6 Postcondition of Inner Loop

Now we consider the stated postcondition of the inner loop. Comparing the conjuncts with those of the loop invariant, one conjunct has been added, viz.

$$a \circ G \circ s{\succ\!\bullet} \;=\; \perp\!\!\perp \;.$$

This is the condition for the termination of the loop. We conclude, therefore, that the postcondition of the inner loop is valid.

For later reference, we state this as a lemma.

Lemma 13.33 On termination of the inner loop, the assertion

$$\begin{array}{ll} & a \circ G \circ s{\succ\!\bullet} \;=\; \perp\!\!\perp \\ \wedge & a \circ s{>} \circ f{\succ\!\bullet} = a \;\wedge\; a \circ s^{\cup} \circ < \circ s \circ f{\succ\!\bullet} \;=\; \perp\!\!\perp \\ \wedge & s{>} \circ f{\succ\!\bullet} \circ \top\!\!\top \circ a \;\subseteq\; G^* \;\wedge\; \mathit{Inv}(s,f) \end{array}$$

is valid.

Proof As remarked above, the first conjunct is the condition for terminating the inner loop. See subsection 13.1.5 for the validity of the remaining conjuncts.
□

13.1.7 Assigning Finish Times

Now we turn to the final assignment to f. The task is to show that the property Inv is maintained by the assignment. Again we begin by considering the effect of the assignment on various subterms.

Lemma 13.34

$$(f^{\cup}\circ{\geq}\circ f)[f \;:=\; f \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ \top\!\!\top\circ a] \;=\; f^{\cup}\circ{\geq}\circ f \cup a \cup a\circ\top\!\!\top\circ f{>} \quad ,$$

$$(f{>})[f \;:=\; f \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ \top\!\!\top\circ a] \;=\; f{>} \cup a \quad \text{, and}$$

$$(f{>\!\!\bullet})[f \;:=\; f \cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ \top\!\!\top\circ a] \;=\; f{>\!\!\bullet}\circ{\sim}a \quad .$$

Proof For brevity, we let m denote $\overline{(MAX.s\uparrow MAX.f){+}1}$.

$$
\begin{array}{ll}
 & (f^{\cup}\circ{\geq}\circ f)[f \;:=\; f \cup m\circ\top\!\!\top\circ a] \\
= & \{ \quad \text{definition of substitution and distributivity,} \\
 & \quad\;\; a \text{ and } m \text{ are coreflexives, so } a = a^{\cup} \text{ and } m = m^{\cup} \quad \} \\
 & \quad f^{\cup}\circ{\geq}\circ f \;\cup\; a\circ\top\!\!\top\circ m\circ{\geq}\circ f \\
 & \cup \quad f^{\cup}\circ{\geq}\circ m\circ\top\!\!\top\circ a \;\cup\; a\circ\top\!\!\top\circ m\circ{\geq}\circ m\circ\top\!\!\top\circ a \\
= & \{ \quad \text{by definition of } m,\; m\circ{\geq}\circ f{<} = m\circ\top\!\!\top\circ f{<} \text{ and } f{<}\circ{\geq}\circ m = \bot\!\!\bot \;; \\
 & \quad \text{also } m\circ{\geq}\circ m \;=\; m\circ\top\!\!\top\circ m \quad \} \\
 & f^{\cup}\circ{\geq}\circ f \;\cup\; a\circ\top\!\!\top\circ m\circ\top\!\!\top\circ f \;\cup\; a\circ\top\!\!\top\circ m\circ\top\!\!\top\circ m\circ\top\!\!\top\circ a \\
= & \{ \quad m \neq \bot\!\!\bot \text{, cone rule and distributivity} \quad \} \\
 & f^{\cup}\circ{\geq}\circ f \cup a\circ\top\!\!\top\circ(f\cup a) \\
= & \{ \quad \text{distributivity, } a = a\circ\top\!\!\top\circ a\,,\; \top\!\!\top\circ f = \top\!\!\top\circ f{>} \quad \} \\
 & f^{\cup}\circ{\geq}\circ f \cup a \cup a\circ\top\!\!\top\circ f{>} \quad .
\end{array}
$$

Also,

$$
\begin{array}{ll}
 & (f{>})[f \;:=\; f \cup m\circ\top\!\!\top\circ a] \\
= & \{ \quad \text{definition of substitution} \quad \} \\
 & (f \cup m\circ\top\!\!\top\circ a){>} \\
= & \{ \quad \text{distributivity} \quad \} \\
 & f{>} \cup (m\circ\top\!\!\top\circ a){>} \\
= & \{ \quad \text{domains} \quad \}
\end{array}
$$

$$
\begin{array}{ll}
 & f_{>} \cup (\top\!\!\top \circ m \circ \top\!\!\top \circ a)_{>} \\
= & \{ \quad \text{cone rule } (\, m \neq \bot\!\!\bot \,) \text{ and } (\top\!\!\top \circ a)_{>} = a \quad \} \\
 & f_{>} \cup a \;\; .
\end{array}
$$

The final equality follows straightforwardly from $f_{\bullet} = \sim(f_{>})$ and the properties of complements.
□

Lemma 13.35 The property (13.8) is an invariant of the assignment to f.

Proof As for lemma 13.20, we leave this straightforward calculation to the reader.
□

Lemma 13.36 The property (13.9) is an invariant of the assignments to f.

Proof The invariance of the first conjunct is straightforward.

$$
\begin{array}{ll}
 & (f_{>} \subseteq s^{\cup} \circ < \circ f)[f \;:=\; f \,\cup\, \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \\
= & \{ \quad \text{substitution and lemma 13.34} \quad \} \\
 & f_{>} \cup a \;\subseteq\; s^{\cup} \circ < \circ (f \,\cup\, \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a) \\
\Leftarrow & \{ \quad \text{assumption: } f_{>} \subseteq s^{\cup} \circ < \circ f \quad \} \\
 & a \;\subseteq\; s^{\cup} \circ < \circ \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a \\
\Leftarrow & \{ \quad a \subseteq I \text{ and monotonicity} \quad \} \\
 & a \;\subseteq\; a \circ s^{\cup} \circ < \circ \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a \\
\Leftarrow & \{ \quad a = a \circ \top\!\!\top \circ a \text{ and } a = a \circ s_{>} \text{ (so } a = a \circ s_{>} \circ \top\!\!\top \circ a \text{)} \quad \} \\
 & s_{>} \circ \top\!\!\top \;\subseteq\; s^{\cup} \circ < \circ \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \\
= & \{ \quad \text{by definition of } MAX, \\
 & \qquad s_{>} \circ \top\!\!\top \;\subseteq\; s^{\cup} \circ < \circ \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \quad \} \\
 & \text{true} \;\; .
\end{array}
$$

The properties of MAX exploited in the last step are well known; we omit a formal proof of their validity.
□

Lemma 13.37 The property (13.10) is an invariant of the assignment to f.

Proof First,

$$(f{>}\circ G\circ s{>\!\bullet})[f\ :=\ f\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]$$
$$=\qquad\{\qquad\text{substitution and lemma 13.34}\qquad\}$$
$$(f{>}\cup a)\circ G\circ s{>\!\bullet}$$
$$=\qquad\{\qquad\text{assume: } f{>}\circ G\circ s{>\!\bullet}=\perp\!\!\perp\text{, distributivity}\qquad\}$$
$$a\circ G\circ s{>\!\bullet}$$
$$=\qquad\{\qquad\text{termination of inner loop: lemma 13.33}\qquad\}$$
$$\perp\!\!\perp\ .$$

Second,

$$(f^{\cup}\circ{<}\circ s\cap G)[f\ :=\ f\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]$$
$$=\qquad\{\qquad\text{substitution, distributivity and assumption: } f^{\cup}\circ{<}\circ s\cap G\ =\ \perp\!\!\perp\qquad\}$$
$$(\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a)^{\cup}\circ{<}\circ s\cap G$$
$$=\qquad\{\qquad\text{converse}\qquad\}$$
$$a\circ\top\!\!\top\circ\overline{(MAX.s\uparrow MAX.f){+}1}\circ{<}\circ s\cap G$$
$$=\qquad\{\qquad\overline{(MAX.s\uparrow MAX.f){+}1}\circ{<}\circ s=\perp\!\!\perp\qquad\}$$
$$\perp\!\!\perp\ .$$

□

Lemma 13.38 The property (13.11) is an invariant of the assignment to f.

Proof This is obvious:

$$(f{>\!\bullet}\circ s^{\cup}\circ{\leq}\circ s)[f\ :=\ f\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]$$
$$=\qquad\{\qquad\text{substitution and lemma 13.34}\qquad\}$$
$$\sim\!a\circ f{>\!\bullet}\circ s^{\cup}\circ{\leq}\circ s$$
$$\subseteq\qquad\{\qquad\sim\!a\subseteq I\text{ and assumption: (13.11)}\qquad\}$$
$$G^{*}$$
$$=\qquad\{\qquad\text{subsitution}\qquad\}$$
$$(G^{*})[f\ :=\ f\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]\ .$$

□

Lemma 13.39 The property (13.12) is an invariant of the assignment to f. Specifically,

$$(13.12)\,[f \;:=\; f \cup \overline{(MAX.s{\uparrow}MAX.f){+}1} \circ \top\!\!\top \circ a] \;\;\Leftarrow\;\; (13.11) \;\wedge\; (13.12) \;\wedge\; a \circ f{>}\!\bullet = a \;\;.$$

Proof

$(\, s^{\cup} \circ {\leq} \circ s \;\cap\; f^{\cup} \circ {\geq} \circ f \,)\,[f \;:=\; f \cup \overline{(MAX.s{\uparrow}MAX.f){+}1} \circ \top\!\!\top \circ a]$

$=$ { definition of substitution and lemma 13.34 }

$s^{\cup} \circ {\leq} \circ s \;\cap\; (f^{\cup} \circ {\geq} \circ f \,\cup\, a \cup a \circ \top\!\!\top \circ f{>})$

$=$ { distributivity, $s^{\cup} \circ {\leq} \circ s \cap a \;=\; a$ }

$(s^{\cup} \circ {\leq} \circ s \;\cap\; f^{\cup} \circ {\geq} \circ f) \,\cup\, a \,\cup\, (s^{\cup} \circ {\leq} \circ s \;\cap\; a \circ \top\!\!\top \circ f{>})$

$\subseteq$ { assume: (13.12) }

$G^{*} \;\cup\; a \;\cup\; (s^{\cup} \circ {\leq} \circ s \;\cap\; a \circ \top\!\!\top \circ f{>})$

$=$ { $a \subseteq I \subseteq G^{*}$, domains }

$G^{*} \;\cup\; a \circ s^{\cup} \circ {\leq} \circ s \circ f{>}$

$\subseteq$ { assumption: $a \circ f{>}\!\bullet = a$, i.e. $a \subseteq f{>}\!\bullet$ }

$G^{*} \;\cup\; f{>}\!\bullet \circ s^{\cup} \circ {\leq} \circ s \circ f{>}$

$\subseteq$ { (13.11) }

$G^{*} \;\cup\; G^{*} \circ f{>}$

$=$ { $f{>}\!\bullet \subseteq I$, idempotency and substitution }

$(G^{*})[f \;:=\; f \cup \overline{(MAX.s{\uparrow}MAX.f){+}1} \circ \top\!\!\top \circ a] \;\;.$

□

Lemma 13.40 The property (13.13) is an invariant of the assignment to f . Specifically,

$$(13.13)\,[f \;:=\; f \cup \overline{(MAX.s{\uparrow}MAX.f){+}1} \circ \top\!\!\top \circ a] \;\Leftarrow\; (13.13) \;\;\wedge\;\; Q(a,s,f) \;\;.$$

(See the proof below for the specific conjunct of $Q(a,s,f)$ that is needed.)

Proof

$(\, s^{\cup} \circ {\leq} \circ s \cap f^{\cup} \circ {<} \circ f \,)\,[f \;:=\; f \cup \overline{(MAX.s{\uparrow}MAX.f){+}1} \circ \top\!\!\top \circ a]$

$=$ { definition of substitution and lemma 13.34 }

$s^{\cup} \circ {\leq} \circ s \cap (f^{\cup} \circ {<} \circ f \;\cup\; f{>} \circ \top\!\!\top \circ a)$

$=$ { distributivity and domains }

$(s^{\cup} \circ {\leq} \circ s \cap f^{\cup} \circ {<} \circ f) \;\cup\; f{>} \circ s^{\cup} \circ {\leq} \circ s \circ a$

$= \quad \{ \quad$ assumption: (13.13) $\quad \}$

$\quad f^{\cup} \circ {<} \circ s \circ f{>} \ \cup \ f{>} \circ s^{\cup} \circ {\leq} \circ s \circ a$

$= \quad \{ \quad s$ is injective, assumption: $f{>} \circ a = \perp\!\!\perp \quad \}$

$\quad f^{\cup} \circ {<} \circ s \circ f{>} \ \cup \ f{>} \circ s^{\cup} \circ {<} \circ s \circ a$

Also, letting m denote $\overline{(MAX.s \uparrow MAX.f)+1}$,

$\quad (f^{\cup} \circ {<} \circ s \circ f{>})[f := f \cup \overline{(MAX.s \uparrow MAX.f)+1} \circ \top\!\!\top \circ a]$

$= \quad \{ \quad$ definition of substitution and lemma 13.34 $\quad \}$

$\quad (f \cup m \circ \top\!\!\top \circ a)^{\cup} \circ {<} \circ s \circ (f{>} \cup a)$

$= \quad \{ \quad$ distributivity $\quad \}$

$\quad f^{\cup} \circ {<} \circ s \circ (f{>} \cup a) \ \cup \ a \circ \top\!\!\top \circ m \circ {<} \circ s \circ (f{>} \cup a)$

$= \quad \{ \quad$ by definition of m, $m \circ {<} \circ s = \perp\!\!\perp \quad \}$

$\quad f^{\cup} \circ {<} \circ s \circ (f{>} \cup a)$

$= \quad \{ \quad$ distributivity $\quad \}$

$\quad f^{\cup} \circ {<} \circ s \circ f{>} \ \cup \ f^{\cup} \circ {<} \circ s \circ a \quad .$

So,

$\quad (13.13)[f := f \cup \overline{(MAX.s \uparrow MAX.f)+1} \circ \top\!\!\top \circ a]$

$\Leftarrow \quad \{ \quad$ above and assumption: (13.13) $\quad \}$

$\quad f{>} \circ s^{\cup} \circ {<} \circ s \circ a = f^{\cup} \circ {<} \circ s \circ a$

$\Leftarrow \quad \{ \quad$ definition $\quad \}$

$\quad Q(a,s,f) \quad .$

□

This completes the proof that timestamped depth-first search meets the specification given in definition 13.7 using theorem 12.5. The verification of (12.6) was completed in subsection 13.1.3, that of (12.7) in subsection 13.1.4, that of (12.8) in subsection 13.1.5 and, finally, the verification of (12.9) was completed in subsection 13.1.2 and (for the relation $Inv^{\cup} \circ (\Leftarrow) \circ Inv$) in subsections 13.1.3 and 13.1.7.

13.2 Calculating a Representative

The conclusion of this section on calculating strongly connected components is quite short. It suffices to observe that the delegate function on G according to the timestamp f is a representative function for the strongly connected components of G.

Suppose φ is the delegate function on G according to the timestamp f. From theorem 10.37, we know that

$$\mathsf{equiv}.G \subseteq \varphi^{\cup}\circ\varphi \quad .$$

It remains to show that

$$\varphi^{\cup}\circ\varphi \subseteq \mathsf{equiv}.G \quad .$$

We do this by showing that $\varphi\subseteq\mathsf{equiv}.G$. That is, we show that the delegate of a node according to f is strongly connected to the node. The key is to use induction, the main difficulty being to identify a suitable induction hypothesis. This is done in the following lemma. Its proof combines two properties of delegates: (i) for each node, there is a path to its delegate on which all nodes have the same delegate and (ii) the delegate has the largest f-value.

Lemma 13.41

$$\varphi \;\subseteq\; \langle \mu X :: f^{\cup}\circ\geq\circ f \cap (I \cup X\circ G^{\cup})\rangle \quad .$$

Proof

$$\begin{array}{ll}
 & \varphi \\
= & \{\quad \text{lemma 10.36} \quad\} \\
 & \langle \mu X :: \varphi\cap(I \cup X\circ G^{\cup})\rangle \\
\subseteq & \{\quad \text{theorem 10.37 (specifically, } \varphi \subseteq f^{\cup}\circ\geq\circ f\text{)} \\
 & \quad\;\; \text{and monotonicity} \quad\} \\
 & \langle \mu X :: f^{\cup}\circ\geq\circ f \cap (I \cup X\circ G^{\cup})\rangle \quad .
\end{array}$$

□

Lemma 13.41 enables us to use fixed-point induction to establish a key lemma:

Lemma 13.42

$$\varphi \;\subseteq\; s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f \quad .$$

Proof

$$\begin{array}{ll}
 & \varphi \subseteq s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f \\
\Leftarrow & \{\quad \text{lemma 13.41} \quad\} \\
 & \langle \mu X :: f^{\cup}\circ\geq\circ f \cap (I \cup X\circ G^{\cup})\rangle \;\subseteq\; s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f \\
\Leftarrow & \{\quad \text{fixed-point induction} \quad\}
\end{array}$$

$$f^{\cup}\circ\geq\circ f \cap (I \cup (s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f)\circ G^{\cup}) \subseteq s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f$$

$\Leftarrow$ { $[\,R\cup S = R\cup(\neg R\cap S)\,]$ with $R{,}S := I\,,\, s^{\cup}\circ\leq\circ s\circ G^{\cup}$
and distributivity }

$$f^{\cup}\circ\geq\circ f \cap I \subseteq s^{\cup}\circ\leq\circ s$$

$$\wedge \quad f^{\cup}\circ\geq\circ f \cap \neg I \cap (s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f)\circ G^{\cup} \subseteq s^{\cup}\circ\leq\circ s$$

$=$ { $\leq$ is reflexive and s is total, so $I \subseteq s^{\cup}\circ\leq\circ s$
f is injective, so $f^{\cup}\circ\geq\circ f \cap \neg I = f^{\cup}\circ>\circ f$ }

$$f^{\cup}\circ>\circ f \cap (s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f)\circ G^{\cup} \subseteq s^{\cup}\circ\leq\circ s \ .$$

We continue with the left-hand side of the inclusion.

$$f^{\cup}\circ>\circ f \cap (s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f)\circ G^{\cup}$$

$\subseteq$ { assumption (13.6): $G^{\cup} \subseteq s^{\cup}\circ\leq\circ f$ }

$$f^{\cup}\circ>\circ f \cap (s^{\cup}\circ\leq\circ s \cap f^{\cup}\circ\geq\circ f)\circ(s^{\cup}\circ\leq\circ f)$$

$\subseteq$ { $[\,R\cap S\subseteq R\,]$ with $R{,}S := s^{\cup}\circ\leq\circ s\,,\, f^{\cup}\circ\geq\circ f$
and monotonicity }

$$f^{\cup}\circ>\circ f \cap s^{\cup}\circ\leq\circ s\circ s^{\cup}\circ\leq\circ f$$

$\subseteq$ { s is functional, so $s\circ s^{\cup} \subseteq I$, $\leq$ is transitive }

$$f^{\cup}\circ>\circ f \cap s^{\cup}\circ\leq\circ f$$

$=$ { assumption : (13.5), i.e. (taking converse and complements)
$s^{\cup}\circ\leq\circ f = s^{\cup}\circ\leq\circ s \cup f^{\cup}\circ\leq\circ f$ }

$$f^{\cup}\circ>\circ f \cap (s^{\cup}\circ\leq\circ s \cup f^{\cup}\circ\leq\circ f)$$

$=$ { $f^{\cup}\circ>\circ f \cap f^{\cup}\circ\leq\circ f = \perp\!\!\perp$ }

$$f^{\cup}\circ>\circ f \cap s^{\cup}\circ\leq\circ s$$

$\subseteq$ { monotonicity }

$$s^{\cup}\circ\leq\circ s \ .$$

Combining the two calculations, the proof is complete.
□

Now we can proceed to show that every node is strongly connected to its delegate.

Lemma 13.43 Suppose φ is the delegate function on G according to the timestamp f. Then

$$\varphi \subseteq \mathsf{equiv}.G \ .$$

Proof

$$
\begin{array}{ll}
& \varphi \subseteq \mathsf{equiv.G} \\
= & \{ \quad \text{definition of } \mathsf{equiv.G}\text{, distributivity} \quad \} \\
& \varphi \subseteq \mathsf{G}^* \wedge \varphi \subseteq (\mathsf{G}^*)^\cup \\
= & \{ \quad \text{by definition of delegate (see theorem 10.37), } \varphi \subseteq (\mathsf{G}^*)^\cup \quad \} \\
& \varphi \subseteq \mathsf{G}^* \\
\Leftarrow & \{ \quad \text{(13.4) is a postcondition of repeated depth-first search} \quad \} \\
& \varphi \ \subseteq\ \mathsf{s}^\cup \circ \leq \circ \mathsf{s} \ \cap\ \mathsf{f}^\cup \circ \geq \circ \mathsf{f} \\
\Leftarrow & \{ \quad \text{lemma 13.42} \quad \} \\
& \mathsf{true} \ .
\end{array}
$$

□

Theorem 13.44 The delegate function on G according to the timestamp f is a representative function for strongly connected components of G. That is, if φ denotes the delegate function,

$$\varphi^\cup \circ \varphi \ = \ \mathsf{equiv.G} \ .$$

Proof

$$
\begin{array}{ll}
& \varphi^\cup \circ \varphi \ = \ \mathsf{equiv.G} \\
= & \{ \quad \text{anti-symmetry} \quad \} \\
& \mathsf{equiv.G} \subseteq \varphi^\cup \circ \varphi \ \wedge \ \varphi^\cup \circ \varphi \subseteq \mathsf{equiv.G} \\
\Leftarrow & \{ \quad \text{theorem 10.37, lemma 13.43} \quad \} \\
& \mathsf{true} \ \wedge \ (\mathsf{equiv.G})^\cup \circ \mathsf{equiv.G} \subseteq \mathsf{equiv.G} \\
= & \{ \quad (\,\mathsf{equiv.G}\,) \text{ is symmetric and transitive} \quad \} \\
& \mathsf{true} \ .
\end{array}
$$

□

Chapter 14

A Short Comparison

Our analysis of timestamps and their use in constructing strongly connected components has been influenced by Tarjan's [Tar72] and Cormen, Leiserson and Rivest's [CLR90] thorough but informal proofs. This chapter explains the connection. Section 14.1 explains how the different types of edge identified by Tarjan are expressed using timestamps. Sections 14.2 and 14.3 delve further into Cormen, Leiserson and Rivest's [CLR90] so-called "white path theorem". Finally, section 14.4 formulates and proves a lemma said by Lengauer and Tarjan [LT79] to be crucial to a "dominators" algorithm.

14.1 Classifying Edges

Tarjan's account of depth-first search [Tar72] classifies edges of the given graph into four categories: *tree* edges, *ancestor* edges, *fronds* and *vines*.

In order to make the classification precise, we formulate how the edges may be identified during execution of depth-first search. In the first instance, we consider the implementation shown in fig. 12.4. Suppose Wt is a relation on nodes such that node a is related by Wt to b if b is in $\sim seen$ *at the time that* $dfs(a)$ *is called.* Similarly, suppose Gy relates node a to node b if b is in $seen \circ \sim fnd$ and suppose Bk relates node a to node b if b is in fnd at the time that $dfs(a)$ is called. "Wt" abbreviates "White", "Gy" abbreviates "Grey" and "Bk" abbreviates "Black": the colours used in (eg.) Cormen, Leiserson and Rivest's [CLR90] account of depth-first search. Note carefully that Wt, Gy and Bk are *relations.*

Because $fnd \subseteq seen$ is an invariant property, there are no other possibilities. That is,

$$Wt \cap Gy \;=\; Gy \cap Bk \;=\; Bk \cap Wt \;=\; \perp\!\!\perp \;\;\wedge\;\; Wt \cup Gy \cup Bk = \top\!\!\top \quad .$$

Now, the identity

$$G \;=\; (G \cap Wt) \cup (G \cap \neg Wt)$$

splits the edges into two types: the edges represented by the relation $G\cap Wt$ are *tree* or *ancestor* edges. These are the edges from a node a to nodes that have not been seen at the time that $dfs(a)$ is called. Tree edges were highlighted in fig. 11.1. Whether an edge becomes a tree edge or an ancestor edge may depend on the order in which edges are chosen in the inner loop: a tree edge is an edge that is indeed chosen.

Next, the identity

$$G\cap\neg Wt \quad = \quad (G\cap Gy)\cup(G\cap Bk)$$

splits the second type of edge into two types: the edges represented by the relation $G\cap Gy$ are called *fronds*. In the iterative stack-based implementation of depth-first search, these are edges from a to nodes that are on the stack at the time that $dfs(a)$ is called. An important property of depth-first search is that there is a path from node b to node a in the graph if the edge from node a to node b is a frond. This is the invariant property (12.25).

Finally, the edges represented by the relation $G\cap Bk$ are called *vines*.

Because of the temporal nature of the classification of nodes as white, grey or black —every node is initially white but eventually black— it is impossible to reflect the classification of edges in the postcondition of the implementation shown in fig. 12.4. When timestamps are added this is (partially) possible.

First, we can split the edges according to start times: the edges represented by the relation

$$G \;\cap\; s^{\cup}\circ{\leq}\circ s$$

are either tree or ancestor edges. It is not possible to use timestamps to distinguish between these types of edges[1]; the distinction reflects the non-determinism in the implementation and is, in fact, irrelevant. Next, we can split the remaining edges according to finish times: the edges represented by the relation

$$G \;\cap\; s^{\cup}\circ{>}\circ s \;\cap\; f^{\cup}\circ{<}\circ f$$

are fronds, and the edges represented by the relation

$$G \;\cap\; s^{\cup}\circ{>}\circ s \;\cap\; f^{\cup}\circ{>}\circ f$$

are vines. A crucial property of depth-first search is the property (13.5). Applying this property, the vines are represented by the relation

$$G \;\cap\; s^{\cup}\circ{>}\circ f \;\; .$$

[1]As for many informal statements, this is not completely correct: self-loops are ancestor edges.

In words, a vine is an edge from a node a to a node b such that the search from a started after the search from b finished.

Just as important as the above classification of edges is the property expressed by the postcondition

$$G \;\subseteq\; f^{\cup}\circ{\geq}\circ s \quad .$$

Whenever there is an edge from node a to node b, the search from a finishes after the search from b starts; conversely, there are no edges from a node a to a node b such that the search from a finishes before the search from b starts. This property was identified as a crucial characteristic property by Tarjan [Tar72]. See also [CLR90, exercise 23.3-4, p.484] (after correction to include self-loops as in [CLRS09, exercise 22.3-5, p.611]).

When illustrating depth-first search, the layout of nodes and edges is typically informed as much as possible by the practice of reading Latin script from left to right and top to bottom. So tree and ancestor edges are most often (but not always) depicted by arrows pointing downwards, and vines are depicted by arrows pointing from right to left (and possibly downwards), thus suggesting the order in which the nodes are processed during the search. This extends to the display of strongly connected components: the top-to-bottom, left-to-right layout suggests the order in which they are recognised. We have adopted this practice in fig. 11.1. See also [AHU82, fig. 6.37] and [CLR90, fig. 23.4].

14.2 The White-Path Theorem

Section 11.3, in particular theorem 11.13, identifies the function implemented by $dfs(a)$ as $D.a$, where

$$D.a.seen \;\;=\;\; seen \cup (a\circ(G\circ{\sim}seen)^{*})> \quad .$$

When $dfs(a)$ is called, the nodes represented by ${\sim}seen$ are "white" relative to node a and the relation $(a\circ(G\circ{\sim}seen)^{*})>$ represents nodes that can be reached from a by a so-called "white" path. Theorem 11.13 is therefore a formal statement of what Cormen, Leiserson and Rivest [CLR90] call the "white path theorem"[2]. Introducing timestamps gives a different way of formulating the theorem. Specifically, on termination of the outer loop, the functions s and f record the history of the search in the sense that the nodes that were "white" at the time the search from node a is initiated are represented by

$$(a\circ s^{\cup}\circ{\leq}\circ s)>$$

[2]More precisely, this is our interpretation of the "white path theorem" as stated by Cormen, Leiserson and Rivest. Because of their informal, operational account, their statements are open to different interpretations and we may have inadvertently chosen an interpretation that was not intended.

and the nodes that could be reached by a "white path" at that time are represented by

$$(a\circ(G\circ(a\circ s^{\cup}\circ\leq\circ s)_{>})^{*})_{>} \ .$$

The nodes newly "seen" by the call of $\mathrm{dfs}(a)$ are represented by

$$(a\circ(s^{\cup}\circ\leq\circ s \ \cap\ f^{\cup}\circ\geq\circ f))_{>} \ ,$$

so the "white path theorem" is the theorem that, for all nodes a,

$$(14.1)\quad (a\circ(s^{\cup}\circ\leq\circ s \ \cap\ f^{\cup}\circ\geq\circ f))_{>} \ = \ (a\circ(G\circ(a\circ s^{\cup}\circ\leq\circ s)_{>})^{*})_{>} \ .$$

We do not prove the correctness of this reformulation of theorem 11.13 because it is not needed to establish the correctness of the algorithm for calculating strongly connected components[3]. Because of the two occurrences of "a" in the right side of (14.1), it cannot be directly restated in point-free form. Theorem 14.4 reformulates (14.1) and, in so doing, adds greater insight into the claim.

First we need a lemma on domains and a lemma on fixed points.

Lemma 14.2 For all atomic coreflexives a and relations R and S,

$$a\circ(R\cap S) \ = \ a\circ R\cap a\circ S \ = \ a\circ R\circ(a\circ S)_{>} \ .$$

Proof The first equality is immediate from the fact that a is coreflexive. Specifically,

$$\begin{array}{ll}
 & a\circ(R\cap S) \ = \ a\circ R\cap a\circ S \\
= & \quad\{ \quad R\cap S\subseteq R \ \text{ and } \ R\cap S\subseteq S \quad \} \\
 & a\circ(R\cap S) \ \supseteq \ a\circ R\cap a\circ S \\
\Leftarrow & \quad\{ \quad \text{modularity rule: (4.8)} \quad \} \\
 & a\circ(R\cap S) \ \supseteq \ a\circ(R\cap a^{\cup}\circ a\circ S) \\
\Leftarrow & \quad\{ \quad \text{monotonicity} \quad \} \\
 & I \ \supseteq \ a^{\cup}\circ a \\
= & \quad\{ \quad a \ \text{is coreflexive} \quad \} \\
 & \text{true} \ .
\end{array}$$

The second equality is proved by mutual inclusion. First we note that

[3] It should be possible to modify the statement and proof of theorem 11.13 appropriately but we have not checked that this is the case at the time of writing. We do establish the correctness of a stronger "white-path theorem" in section 14.3.

$$
\begin{array}{ll}
 & a\circ\top\!\!\top\circ(a\circ S)_{>} \\
= & \{ \quad \text{domains} \quad \} \\
 & a\circ\top\!\!\top\circ a\circ S \\
= & \{ \quad a \text{ is an atomic coreflexive, so } a = a\circ\top\!\!\top\circ a \quad \} \\
 & a\circ S \quad .
\end{array}
$$

Now for the containment, we have:

$$
\begin{array}{ll}
 & a\circ R \cap a\circ S \ \supseteq\ a\circ R\circ(a\circ S)_{>} \\
= & \{ \quad \text{distributivity, } I \supseteq (a\circ S)_{>} \text{ and monotonicity} \quad \} \\
 & a\circ S \ \supseteq\ a\circ R\circ(a\circ S)_{>} \\
\Leftarrow & \{ \quad \top\!\!\top \supseteq R \text{ and monotonicity} \quad \} \\
 & a\circ S \ \supseteq\ a\circ\top\!\!\top\circ(a\circ S)_{>} \\
= & \{ \quad \text{see above} \quad \} \\
 & \text{true} \quad .
\end{array}
$$

Secondly, for the inclusion we have

$$
\begin{array}{ll}
 & a\circ R\circ(a\circ S)_{>} \\
= & \{ \quad \text{domains} \quad \} \\
 & a\circ R \cap \top\!\!\top\circ a\circ S \\
\supseteq & \{ \quad \top\!\!\top \supseteq I \quad \} \\
 & a\circ R \cap a\circ S \quad .
\end{array}
$$

□

Lemma 14.3 Suppose a is an atomic coreflexive and R and S are arbitrary relations. Then

$$
a\circ(R\circ(a\circ S)_{>})^{*} \ =\ a\circ\langle \mu X :: I\cup(X\circ R\cap S)\rangle \quad .
$$

Proof

$$
\begin{array}{ll}
 & a\circ(R\circ(a\circ S)_{>})^{*} \ =\ a\circ\langle \mu X :: I\cup(X\circ R\cap S)\rangle \\
= & \{ \quad [\ R\circ S^{*} = \langle \mu X :: R\cup X\circ S\rangle\ ] \text{ with } R,S := a\ ,\ R\circ(a\circ S)_{>} \quad \} \\
 & \langle \mu X :: a\cup X\circ R\circ(a\circ S)_{>}\rangle \ =\ a\circ\langle \mu X :: I\cup(X\circ R\cap S)\rangle \\
\Leftarrow & \{ \quad (a\circ) \text{ is a lower adjoint, theorem 2.43} \quad \}
\end{array}
$$

$\langle\forall X :: a \cup a{\circ}X{\circ}R{\circ}(a{\circ}S)> \;=\; a\circ(I\cup(X{\circ}R\cap S))\rangle$

$\Leftarrow \qquad \{ \qquad$ distributivity and Leibniz $\quad \}$

$\langle\forall X :: a{\circ}X{\circ}R{\circ}(a{\circ}S)> \;=\; a{\circ}X{\circ}R \cap a{\circ}S\rangle$

$\Leftarrow \qquad \{ \qquad$ lemma 14.2 with $R,S := X{\circ}R\;,\;S \quad \}$

true .

□

Theorem 14.4 Assuming the validity of (14.1),

$$s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f \;=\; \langle\mu X :: I \cup (X{\circ}G \cap s^{\cup}\circ{<}\circ s)\rangle \;.$$

Proof We have:

$s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f$

$= \qquad \{ \qquad$ saturation axiom $\quad \}$

$\langle\cup a :: a\circ(s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f)\rangle$

$= \qquad \{ \qquad$ assumption: (14.1) $\quad \}$

$\langle\cup a :: a \circ (a\circ(G\circ(a\circ s^{\cup}\circ{\leq}\circ s)>)^{*})>\rangle$

$= \qquad \{ \qquad$ lemma 14.3 with $R,S := G\;,\; s^{\cup}\circ{\leq}\circ s \quad \}$

$\langle\cup a :: a \circ \langle\mu X :: I \cup (X{\circ}G \cap s^{\cup}\circ{\leq}\circ s)\rangle>\rangle$

$= \qquad \{ \qquad$ absorption rule, $s^{\cup}\circ{\leq}\circ s \cap \neg I \;=\; s^{\cup}\circ{<}\circ s$

saturation axiom $\quad \}$

$\langle\mu X :: I \cup (X{\circ}G \cap s^{\cup}\circ{<}\circ s)\rangle$.

□

The expression $\langle\mu X :: I \cup (X{\circ}G \cap s^{\cup}\circ{<}\circ s)\rangle$ is the relation between nodes a and b such that there is a path in G from a to b on which every node is “white” at the time that the search from a is initiated. So theorem 14.4 is a formal statement of the “white-path theorem”. If we interpret the relation $s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f$ as the ancestor relation, the property is that ancestor equals white path. Pointwise, node a is an ancestor of node b in a depth-first search if and only if there is a path from a to b on which each node is white at the time that the search from a starts.

14.3 Ancestor Paths

In the previous section, we gave a precise formulation of the “white-path theorem”: the property that, in a depth-first search of a finite graph, there is a path from node a to a node b in a graph comprising nodes that are white at the time that the search from a starts exactly when node a is an ancestor of b in the search (i.e. the search from node a starts before and finishes after the start of the search from node b).

A stronger statement is that, for each node a, a call of $dfs(a)$ calculates all nodes that can be reached from a by a path consisting of edges connecting nodes with strictly increasing start times. This is a smaller set of paths than the "white paths". For example, in fig. 11.1, it does not include the path from the node with start time 2 via the node with start time 9 to the node with start time 4 (this being nevertheless a "white" path because the nodes with start times 9 and 4 are both white relative to the node with start time 2); it does include the path via the node with start time 3.

In this section, we prove that for all nodes a, a call of $dfs(a)$ calculates all nodes that can be reached from a by a path consisting of "tree/ancestor" edges.

The proof is in two steps. We begin by proving that, on termination of depth-first search,

(14.5) $s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f \;=\; (s^{\cup}\circ{<}\circ s \cap G)^{*}$.

This is stronger than theorem 14.4 because the right side of the equality describes paths formed of edges whereby each node is white with respect to its predecessor on the path (as opposed to white with respect to the initial node on the path). Its proof involves strengthening the assertions made about depth-first search; in this way it gives greater understanding of the algorithm. Then, we can infer the formally stronger property

(14.6) $s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f \;=\; (s^{\cup}\circ{<}\circ s \;\cap\; f^{\cup}\circ{>}\circ f \cap G)^{*}$

by a straightforward calculation. See theorem 14.21. In words, this is the property that a node a is an ancestor of node b in the search exactly when there is a path from a to b of which each edge is a tree or ancestor edge.

In order to prove (14.5), the precondition for executing a depth-first search, the invariant, and the intermediate assertion must all be strengthened. The precondition, $P(a,s,f)$, for executing $dfs(a)$ is strengthened with the conjunct

(14.7) $f_{>\bullet}\circ s_{>}\circ \top\!\!\top\circ a \;\subseteq\; (s^{\cup}\circ{<}\circ s \cap G)^{*}\circ G$.

(This strengthens the precondition $f_{>\bullet}\circ s_{>}\circ \top\!\!\top\circ a \subseteq G^{*}$.) The invariant is strengthened by adding three conjuncts:

(14.8) $f_{>}\circ(G \cap s^{\cup}\circ{<}\circ s)^{*}\circ f_{>} \;=\; s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f$

(a strengthening of $s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f \;\subseteq\; G^{*}$),

(14.9) $f_{>}\circ(G \cap s^{\cup}\circ{<}\circ s)\circ f_{>\bullet} \;=\; \bot\!\!\bot$

(a supplement to $f{\scriptstyle >} \circ G \circ s{\scriptstyle >\!\bullet} \;=\; \perp\!\!\perp$) and

(14.10) $f{\scriptstyle >\!\bullet} \circ (G \cap s^{\cup} \circ < \circ s)^* \circ f{\scriptstyle >} \;=\; f{\scriptstyle >\!\bullet} \circ s^{\cup} \circ \leq \circ s \circ f{\scriptstyle >}$.

Finally, the intermediate assertion, $Q(a,s,f)$, is strengthened by adding the conjunct:

(14.11) $f{\scriptstyle >\!\bullet} \circ s{\scriptstyle >} \circ \top\!\!\top \circ a \subseteq (s^{\cup} \circ < \circ s \cap G)^*$.

It is an immediate consequence of (14.8) that (14.5) holds on termination of the outer loop: on termination, $f{\scriptstyle >} = I$. Properties (14.9) and (14.10) are needed to establish (14.8): see the proof of lemma 14.14 below.

The verification of these properties proceeds as follows. It is obvious that the three invariants (14.8), (14.9) and (14.10) are truthified by the initialisation in the outer loop (because the initial values of $s{\scriptstyle >}$ and $f{\scriptstyle >}$ are both $\perp\!\!\perp$).

In the outer loop, $f{\scriptstyle >} = s{\scriptstyle >}$ is an invariant property; it follows that in the outer loop, the truth of (14.9) and (14.10) is guaranteed. The remaining invariant, (14.8) is obviously truthified by the initialisation of s and f; so we must show that it is maintained by calls of dfs .

The precondition (14.7) is clearly satisfied when $dfs(a)$ is called in the outer loop: the left side is $\perp\!\!\perp$ because $f{\scriptstyle >} = s{\scriptstyle >}$). It is also satisfied when $dfs(b)$ is called because of the combination of (14.10) and the condition for choosing b .

$$
\begin{array}{ll}
 & a \circ \top\!\!\top \circ b \subseteq s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ G \circ s{\scriptstyle >\!\bullet} \;\wedge\; s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ \top\!\!\top \circ a \subseteq (s^{\cup} \circ < \circ s \cap G)^* \\
\Rightarrow & \{ \quad \text{monotonicity} \quad \} \\
 & s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ \top\!\!\top \circ a \circ a \circ \top\!\!\top \circ b \subseteq (s^{\cup} \circ < \circ s \cap G)^* \circ s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ G \circ s{\scriptstyle >\!\bullet} \\
\Rightarrow & \{ \quad a \neq \perp\!\!\perp \text{, so } \top\!\!\top \circ a \circ a \circ \top\!\!\top = \top\!\!\top \text{; } s{\scriptstyle >\!\bullet} \text{ is a coreflexive} \quad \} \\
 & s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ \top\!\!\top \circ b \subseteq (s^{\cup} \circ < \circ s \cap G)^* \circ s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ G \circ s{\scriptstyle >\!\bullet} \\
\Rightarrow & \{ \quad [\, R^* \circ R \subseteq R^* \,] \text{, transitivity} \quad \} \\
 & s{\scriptstyle >} \circ f{\scriptstyle >\!\bullet} \circ \top\!\!\top \circ b \subseteq (s^{\cup} \circ < \circ s \cap G)^* \circ G \quad .
\end{array}
$$

(This is just a repeat of the calculation in section 12.3.1 but with the strengthened precondition.)

Verifying the strengthened intermediate assertion (in particular, the conjunct (14.11)) is a straightforward application of the assignment axiom. It is also necessary to show that (14.11) is maintained by subsequent searches. (Formally, we have to establish (12.8).) As before, the crucial fact is that $f{\scriptstyle >\!\bullet} \circ s{\scriptstyle >}$ is an invariant value. Thus the left side of (14.11) is invariant whilst the right side increases (because s increases).

This leaves the verification of each of the new invariants. This is done by showing that they are invariant properties of both the assignment to s and the assignment to f .

(Formally, we split the verification of (12.9) into the verification of (12.17) and (12.18), as we did with other invariant properties.)

Because the subterm $(G \cap s^{\cup} \circ {<} \circ s)^*$ occurs in two of the invariants, we separate it out:

Lemma 14.12

$$\begin{array}{ll} & ((G \cap s^{\cup} \circ {<} \circ s)^*)[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a] \\ = & (G \cap s^{\cup} \circ {<} \circ s)^* \circ (I \cup s_{>} \circ G \circ a) \;\; . \end{array}$$

Proof

$$\begin{array}{ll} & ((G \cap s^{\cup} \circ {<} \circ s)^*)[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a] \\ = & \{ \quad \text{lemma 13.18} \quad \} \\ & (G \cap (s^{\cup} \circ {<} \circ s \;\cup\; s_{>} \circ \top\!\top \circ a))^* \\ = & \{ \quad \text{distributivity and star decomposition} \quad \} \\ & (G \cap s^{\cup} \circ {<} \circ s)^* \circ (s_{>} \circ G \circ a \circ (G \cap s^{\cup} \circ {<} \circ s)^*)^* \\ = & \{ \quad R^* \;=\; I \cup R \circ R^* \text{ with } R := G \cap s^{\cup} \circ {<} \circ s\,, \\ & \quad\quad a \circ s_{>} = \perp\!\!\perp \text{ (so } a \circ s^{\cup} = \perp\!\!\perp \text{)} \quad \} \\ & (G \cap s^{\cup} \circ {<} \circ s)^* \circ (s_{>} \circ G \circ a)^* \\ = & \{ \quad R^* \;=\; I \cup R^* \circ R \text{ with } R := s_{>} \circ G \circ a \\ & \quad\quad \text{distributivity, mirror rule and } a \circ s_{>} = \perp\!\!\perp \quad \} \\ & (G \cap s^{\cup} \circ {<} \circ s)^* \circ (I \cup s_{>} \circ G \circ a) \;\; . \end{array}$$

□

Lemma 14.13 Property (14.8) is an invariant of the assignment to s.

Proof

$$\begin{array}{ll} & (f_{>} \circ (G \cap s^{\cup} \circ {<} \circ s)^* \circ f_{>})[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\top \circ a] \\ = & \{ \quad \text{substitution and lemma 14.12} \quad \} \\ & f_{>} \circ (G \cap s^{\cup} \circ {<} \circ s)^* \circ (I \cup s_{>} \circ G \circ a) \circ f_{>} \\ = & \{ \quad \text{distributivity and } a \circ f_{>} = \perp\!\!\perp \quad \} \\ & f_{>} \circ (G \cap s^{\cup} \circ {<} \circ s)^* \circ f_{>} \;\; . \end{array}$$

By lemma 13.24 —the value of $s^{\cup} \circ {\leq} \circ s \;\cap\; f^{\cup} \circ {\geq} \circ f$ is invariant under the assignment to s— , invariance under the assignment to s follows.

□

Lemma 14.14 Property (14.8) is an invariant of the assignment to f.

Proof We begin by showing that (14.9) is equivalent to

(14.15) $f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>\bullet} \;=\; \perp\!\!\perp$.

We have:

$$f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>\bullet} \;=\; \perp\!\!\perp$$

$$= \quad \{ \quad \text{(complemented) domains} \quad \}$$

$$(f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*})_{>} \;\subseteq\; f_{>}$$

$$\Leftarrow \quad \{ \quad \text{fusion theorem: theorem 2.43} \quad \}$$

$$(f_{>} \cup f_{>} \circ (G \cap s^{\cup} \circ < \circ s))_{>} \;\subseteq\; f_{>}$$

$$= \quad \{ \quad \text{distributivity} \quad \}$$

$$(f_{>} \circ (G \cap s^{\cup} \circ < \circ s))_{>} \;\subseteq\; f_{>}$$

$$= \quad \{ \quad \text{(complemented) domains} \quad \}$$

$$f_{>} \circ (G \cap s^{\cup} \circ < \circ s) \circ f_{>\bullet} \;=\; \perp\!\!\perp$$

$$\Leftarrow \quad \{ \quad R \subseteq R^{*} \text{ with } R := G \cap s^{\cup} \circ < \circ s \text{ and monotonicity} \quad \}$$

$$f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>\bullet} \;=\; \perp\!\!\perp \;.$$

Now we can proceed to establish the invariance of (14.9).

$$(f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>})[f \;:=\; f \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a]$$

$$= \quad \{ \quad \text{lemma 13.34} \quad \}$$

$$(f_{>} \cup a) \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ (f_{>} \cup a)$$

$$= \quad \{ \quad \text{distributivity} \quad \}$$

$$f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>} \;\cup\; a \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ a$$

$$\cup \quad a \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>} \;\cup\; f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ a$$

$$= \quad \{ \quad a \text{ is an atom;} \quad (14.10) \text{ and } a \circ f_{>\bullet} = a \text{; } (14.15) \text{ and } a \circ f_{>\bullet} = a \quad \}$$

$$f_{>} \circ (G \cap s^{\cup} \circ < \circ s)^{*} \circ f_{>} \;\cup\; a \;\cup\; a \circ s^{\cup} \circ \leq \circ s \circ f_{>} \;.$$

Also,

$$(s^{\cup} \circ \leq \circ s \;\cap\; f^{\cup} \circ \geq \circ f)[f \;:=\; f \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a]$$

$=$ { lemma 13.34 }

$$s^{\cup}\circ{\leq}\circ s \;\cap\; (f^{\cup}\circ{\geq}\circ f \;\cup\; a \cup a\circ\top\!\!\top\circ f_{>})$$

$=$ { distributivity }

$$(s^{\cup}\circ{\leq}\circ s \;\cap\; f^{\cup}\circ{\geq}\circ f) \;\cup\; a \;\cup\; a\circ s^{\cup}\circ{\leq}\circ s\circ f_{>}$$

$=$ { assumption: (14.8) }

$$f_{>}\circ(G\cap s^{\cup}\circ{<}\circ s)^{*}\circ f_{>} \;\cup\; a \;\cup\; a\circ s^{\cup}\circ{\leq}\circ s\circ f_{>} \quad .$$

The lemma follows by the definition of substitution.
□

Lemma 14.16 Property (14.9) is an invariant of the assignment to s .

Proof

$$(f_{>}\circ(G\cap s^{\cup}\circ{<}\circ s)\circ f_{>\bullet})[s \;:=\; s\cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]$$

$=$ { lemma 13.18 }

$$f_{>}\circ(G\cap(s^{\cup}\circ{<}\circ s\cup s_{>}\circ\top\!\!\top\circ a))\circ f_{>\bullet}$$

$=$ { distributivity and assumption: (14.9), $f_{>}\circ s_{>}=f_{>}$ }

$$f_{>}\circ G\circ a$$

$\subseteq$ { $a\circ s_{>\bullet}=a$ }

$$f_{>}\circ G\circ s_{>\bullet}$$

$=$ { invariant (13.10) }

$$\perp\!\!\perp \quad .$$

□

Lemma 14.17 Property (14.9) is an invariant of the assignment to f .

Proof

$$(f_{>}\circ(G\cap s^{\cup}\circ{<}\circ s)\circ f_{>\bullet})[f \;:=\; f\cup \overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a]$$

$=$ { lemma 13.34 }

$$(f_{>}\cup a)\circ(G\cap s^{\cup}\circ{<}\circ s)\circ f_{>\bullet}\circ{\sim}a$$

$=$ { distributivity and assumption: (14.9) }

$$a\circ(G\cap s^{\cup}\circ{<}\circ s)\circ f_{>\bullet}\circ{\sim}a$$

$\subseteq$ { lemma 13.19, in particular $a\circ s^{\cup}\circ{<}\circ s\circ f_{>\bullet}=\perp\!\!\perp$, and section 13.1.5 $\}$

$\bot\!\!\bot$.

□

Lemma 14.18 Property (14.10) is an invariant of the assignment to s .

Proof For the left side of (14.10), we have:

$$
\begin{array}{ll}
 & (f{>\!\bullet} \circ (G \cap s^{\cup} \circ {<} \circ s)^{*} \circ f{>})[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \\
= & \{ \quad \text{substitution and lemma 14.12} \quad \} \\
 & f{>\!\bullet} \circ (G \cap s^{\cup} \circ {<} \circ s)^{*} \circ (I \cup s{>} \circ G \circ a) \circ f{>} \\
= & \{ \quad a \circ f{>} = \bot\!\!\bot \quad \} \\
 & f{>\!\bullet} \circ (G \cap s^{\cup} \circ {<} \circ s)^{*} \circ f{>} \quad .
\end{array}
$$

Also, for the right side, we have:

$$
\begin{array}{ll}
 & (f{>\!\bullet} \circ s^{\cup} \circ {\leq} \circ s \circ f{>})[s \;:=\; s \cup \overline{(MAX.s \uparrow MAX.f){+}1} \circ \top\!\!\top \circ a] \\
= & \{ \quad \text{lemma 13.18} \quad \} \\
 & f{>\!\bullet} \circ (s^{\cup} \circ {\leq} \circ s \cup s{>} \circ \top\!\!\top \circ a) \circ f{>} \\
= & \{ \quad \text{distributivity and } a \circ f{>} = \bot\!\!\bot \quad \} \\
 & f{>\!\bullet} \circ s^{\cup} \circ {\leq} \circ s \circ f{>} \quad .
\end{array}
$$

Both sides are thus unchanged by the assignment and so their equality is invariant.
□

In order to establish that (14.10) is an invariant of the assignment to f, we need to reformulate the intermediate property (14.11) as an equality. This is done in the following lemma.

Lemma 14.19 Suppose $s{>} \circ a = a$. Then

$$
\begin{array}{ll}
 & f{>\!\bullet} \circ s{>} \circ \top\!\!\top \circ a \;\subseteq\; (s^{\cup} \circ {<} \circ s \cap G)^{*} \\
\Rightarrow & f{>\!\bullet} \circ s^{\cup} \circ {\leq} \circ s \circ a \;=\; f{>\!\bullet} \circ (s^{\cup} \circ {<} \circ s \cap G)^{*} \circ a \quad .
\end{array}
$$

(The antecedent of this implication is the property (14.11).)

Proof

$$
\begin{array}{ll}
 & f_{>\bullet}\circ(s^{\cup}\circ{<}\circ s\cap G)^{*}\circ a \\
= & \{\quad \text{assumption: } s_{>}\circ a = a\,,\ s_{>}\circ s^{\cup} = s^{\cup}\,,\ s\circ s_{>} = s\,, \\
 & \qquad \text{mirror rule and distributivity property of coreflexives}\quad\} \\
 & f_{>\bullet}\circ s_{>}\circ(s^{\cup}\circ{<}\circ s\cap G)^{*}\circ a \\
\subseteq & \{\quad s^{\cup}\circ{<}\circ s\cap G\ \subseteq\ s^{\cup}\circ{<}\circ s \text{ and monotonicity}\quad\} \\
 & f_{>\bullet}\circ s_{>}\circ(s^{\cup}\circ{<}\circ s)^{*}\circ a \\
= & \{\quad \text{the less-than relation is transitive and } s \text{ is functional} \\
 & \qquad \text{so}\quad (s^{\cup}\circ{<}\circ s)^{*} = s^{\cup}\circ{\leq}\circ s \\
 & \qquad \text{(simple formal proof left to reader)}\quad\} \\
 & f_{>\bullet}\circ s_{>}\circ s^{\cup}\circ{\leq}\circ s\circ a \\
\subseteq & \{\quad \text{assumption: } f_{>\bullet}\circ s_{>}\circ\top\!\!\top\circ a\ \subseteq\ (s^{\cup}\circ{<}\circ s\cap G)^{*} \\
 & \qquad f_{>\bullet}\circ s_{>} \text{ and } a \text{ are coreflexives, } p\circ p = p \text{ for all coreflexives } p\quad\} \\
 & f_{>\bullet}\circ s_{>}\circ(s^{\cup}\circ{<}\circ s\cap G)^{*}\circ a \\
= & \{\quad \text{first step reversed}\quad\} \\
 & f_{>\bullet}\circ(s^{\cup}\circ{<}\circ s\cap G)^{*}\circ a\quad .
\end{array}
$$

The lemma follows by the anti-symmetry of the subset relation combined with $s_{>}\circ s^{\cup} = s^{\cup}$.
□

Lemma 14.20 Property (14.10) is an invariant of the assignment to f.

Proof The key step is the use of lemma 14.19. It is applicable because $s_{>}\circ a = a$ and (14.11) are both valid when the assignment is executed (as shown earlier).

$$
\begin{array}{ll}
 & (f_{>\bullet}\circ(G\cap s^{\cup}\circ{<}\circ s)^{*}\circ f_{>})[f\ :=\ f\cup\overline{(MAX.s\uparrow MAX.f){+}1}\circ\top\!\!\top\circ a] \\
= & \{\quad \text{lemma 13.34}\quad\} \\
 & f_{>\bullet}\circ{\sim}a\circ(G\cap s^{\cup}\circ{<}\circ s)^{*}\circ(f_{>}\cup a) \\
= & \{\quad \text{coreflexives commute and distributivity}\quad\} \\
 & {\sim}a\circ f_{>\bullet}\circ(G\cap s^{\cup}\circ{<}\circ s)^{*}\circ f_{>}\ \cup\ {\sim}a\circ f_{>\bullet}\circ(G\cap s^{\cup}\circ{<}\circ s)^{*}\circ a \\
= & \{\quad \text{(left summand) assumption: (14.10)} \\
 & \qquad \text{(right summand) assumption: (14.11) and lemma 14.19}\quad\} \\
 & {\sim}a\circ f_{>\bullet}\circ s^{\cup}\circ{\leq}\circ s\circ f_{>}\ \cup\ {\sim}a\circ f_{>\bullet}\circ s^{\cup}\circ{\leq}\circ s\circ a
\end{array}
$$

$$
\begin{array}{ll}
= & \{ \quad \text{distributivity} \quad \} \\
 & \sim a \circ f{\succ}\!\bullet \circ s^{\cup} \circ \leq \circ s \circ (f{>} \cup a) \\
= & \{ \quad \text{coreflexives commute and lemma 13.34} \quad \} \\
 & (f{\succ}\!\bullet \circ s^{\cup} \circ \leq \circ s \circ f{>})[f \;:=\; f \cup \overline{(MAX.s \uparrow MAX.f)+1} \circ \top\!\!\top \circ a] \quad .
\end{array}
$$

□

We summarise the results of this section in the following theorem. We call it the "ancestor paths" theorem because it asserts that, for all nodes a and b, the search from node a starts before and finishes after node b —i.e. a is an "ancestor" of node b— equivales there is a path from a to b formed of edges each of which is from an "ancestor" to a "descendant".

Theorem 14.21 (Ancestor Paths) On termination of depth-first search,

$$
s^{\cup} \circ \leq \circ s \;\cap\; f^{\cup} \circ \geq \circ f \;=\; (s^{\cup} \circ < \circ s \cap G)^{*} \;=\; (s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ > \circ f \cap G)^{*} \quad .
$$

Proof That

$$
s^{\cup} \circ \leq \circ s \;\cap\; f^{\cup} \circ \geq \circ f \;=\; (s^{\cup} \circ < \circ s \cap G)^{*}
$$

on termination is immediate from the invariant property (14.8) and the termination property $f{>} = I$. Now

$$
\begin{array}{ll}
 & s^{\cup} \circ < \circ s \;\cap\; G \\
= & \{ \quad \top\!\!\top_{\mathbb{N}} \;=\; (<) \cup I_{\mathbb{N}} \cup (>) \text{, and } f^{\cup} \circ f = I \text{,} \\
 & \qquad \text{so } \top\!\!\top_{\mathsf{Node}} \;=\; f^{\cup} \circ < \circ f \cup I_{\mathsf{Node}} \cup f^{\cup} \circ > \circ f \quad ; \\
 & \qquad \text{distributivity and } s^{\cup} \circ < \circ s \;\cap\; I_{\mathsf{Node}} \;=\; \perp\!\!\perp \quad \} \\
 & (s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ < \circ f \;\cap\; G) \;\cup\; (s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ > \circ f \;\cap\; G) \\
= & \{ \quad \text{invariant: (13.5)} \quad \} \\
 & (f^{\cup} \circ < \circ s \cap G) \;\cup\; (s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ > \circ f \;\cap\; G) \\
= & \{ \quad \text{invariant: (13.3)} \quad \} \\
 & s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ > \circ f \;\cap\; G \quad .
\end{array}
$$

The theorem follows by Leibniz's rule.

□

We now return to theorem 14.4. Recall that theorem 14.4 assumes the validity of (14.1) which we have not proved. It is straightforward to show that theorem 14.4 is implied by theorem 14.21. For completeness,we give the proof below.

The following lemma, which is relatively obvious, forms the core of the argument.

Lemma 14.22 Suppose T is a reflexive and transitive relation. Then, for all relations R,

$$(R\cap T)^* \subseteq \langle \mu X :: I \cup (X\circ R\cap T)\rangle \subseteq T \ .$$

Proof We begin by proving that

$$\langle \mu X :: I \cup (X\circ R\cap T)\rangle \subseteq T \ .$$

We have

$\langle \mu X :: I \cup (X\circ R\cap T)\rangle \subseteq T$

$\Leftarrow$ { fixed-point induction }

$I \cup (T\circ R\cap T) \subseteq T$

$=$ { definition of supremum }

$I\subseteq T \ \wedge \ T\circ R\cap T \subseteq T$

$=$ { assumption: T is reflexive (i.e. $I\subseteq T$) property of infimum }

true .

Using the above, we can infer that

$$(R\cap T)^* \subseteq \langle \mu X :: I \cup (X\circ R\cap T)\rangle \ .$$

Introducing the abbreviation M for $\langle \mu X :: I \cup (X\circ R\cap T)\rangle$, we have:

$(R\cap T)^* \subseteq \langle \mu X :: I \cup (X\circ R\cap T)\rangle$

$=$ { definition of M }

$(R\cap T)^* \subseteq M$

$\Leftarrow$ { fixed-point induction }

$I \cup M\circ(R\cap T) \subseteq M$

$=$ { by (fixed-point) computation rule, $M = I \cup (M\circ R\cap T)$ }

$I \cup M\circ(R\cap T) \subseteq I \cup (M\circ R\cap T)$

$\Leftarrow$ { monotonicity of $(I\cup)$ }

$M\circ(R\cap T) \subseteq M\circ R\cap T$

$=$ { definition of infimum }

$M\circ(R\cap T) \subseteq M\circ R \ \wedge \ M\circ(R\cap T) \subseteq T$

$$\begin{array}{ll}
\Leftarrow & \{ \quad \text{monotonicity and assumption: } T \text{ is transitive} \quad \} \\
& M\circ(R\cap T) \subseteq T\circ T \\
\Leftarrow & \{ \quad \text{property of infimum, monotonicity and definition of } M \quad \} \\
& \langle \mu X :: I \cup (X\circ R \cap T) \rangle \;\subseteq\; T \\
= & \{ \quad \text{see above} \quad \} \\
& \mathsf{true} \;\; .
\end{array}$$

□

Theorem 14.23 On termination of depth-first search,

$$\begin{array}{ll}
& s^{\cup}\circ\leq\circ s \;\cap\; f^{\cup}\circ\geq\circ f \\
= & (s^{\cup}\circ<\circ s \cap G)^* \\
= & (s^{\cup}\circ<\circ s \;\cap\; f^{\cup}\circ>\circ f \cap G)^* \\
= & \langle \mu X :: I \cup (X\circ G \cap s^{\cup}\circ<\circ s) \rangle \;\; .
\end{array}$$

(As remarked earlier, $\langle \mu X :: I \cup (X\circ G \cap s^{\cup}\circ\leq\circ s) \rangle$ is the relation between nodes a and b expressing the existence of a path from a to b on which every node is "white" at the time that the call of $\mathtt{dfs}(a)$ is made. See the remarks preceding theorem 14.21 for the interpretation of the other terms in the continued equality.)

Proof The first two equalities are as in theorem 14.21. We now prove that the first and last terms are equal. The proof is by mutual inclusion.

Instantiating lemma 14.22 with $R{,}T := G\,,\, s^{\cup}\circ\leq\circ s$ (and noting that $s^{\cup}\circ\leq\circ s$ is reflexive and transitive because the at-most relation is reflexive and transitive and s is a total function)

$$(G \cap s^{\cup}\circ\leq\circ s)^* \;\subseteq\; \big\langle \mu X :: I \cup (X\circ G \cap s^{\cup}\circ\leq\circ s) \big\rangle \;\subseteq\; s^{\cup}\circ\leq\circ s \;\; .$$

Moreover, $(G \cap s^{\cup}\circ\leq\circ s)^* = (s^{\cup}\circ<\circ s \cap G)^*$. (This is because $R^* = (\neg I \cap R)^*$ for all R, and the less-than relation is the intersection of the not-equal and the at-most relation.) So

$$(s^{\cup}\circ<\circ s \cap G)^* \;\subseteq\; \big\langle \mu X :: I \cup (X\circ G \cap s^{\cup}\circ\leq\circ s) \big\rangle \;\subseteq\; s^{\cup}\circ\leq\circ s \;\; .$$

Comparing with theorem 14.21, it remains to prove that

$$\big\langle \mu X :: I \cup (X\circ G \cap s^{\cup}\circ\leq\circ s) \big\rangle \;\subseteq\; f^{\cup}\circ\geq\circ f \;\; .$$

Now,

$$\begin{array}{ll}
& \langle \mu X :: I \cup (X \circ G \cap s^{\cup} \circ \leq \circ s) \rangle \;\subseteq\; f^{\cup} \circ \geq \circ f \\
\Leftarrow & \{ \quad \text{fixed-point induction, } f^{\cup} \circ \geq \circ f \text{ is reflexive} \quad \} \\
& f^{\cup} \circ \geq \circ f \circ G \;\cap\; s^{\cup} \circ \leq \circ s \;\subseteq\; f^{\cup} \circ \geq \circ f \\
= & \{ \quad \text{shunting rule (2.27)} \quad \} \\
& f^{\cup} \circ \geq \circ f \circ G \;\cap\; s^{\cup} \circ \leq \circ s \cap f^{\cup} \circ < \circ f \;\subseteq\; \perp\!\!\perp \\
\Leftarrow & \{ \quad \text{on termination of depth-first search} \\
& \qquad s^{\cup} \circ \leq \circ s \;\cap\; f^{\cup} \circ < \circ f \;=\; f^{\cup} \circ < \circ s \\
& \qquad \text{and } G \;\subseteq\; f^{\cup} \circ \geq \circ s \quad \} \\
& f^{\cup} \circ \geq \circ f \circ f^{\cup} \circ \geq \circ s \;\cap\; f^{\cup} \circ < \circ s \;\subseteq\; \perp\!\!\perp \\
\Leftarrow & \{ \quad \geq \text{ is transitive, } f \text{ is a total function} \quad \} \\
& f^{\cup} \circ \geq \circ s \;\cap\; f^{\cup} \circ < \circ s \;\subseteq\; \perp\!\!\perp \\
= & \{ \quad f \text{ and } s \text{ are total functions, } (\geq) \cap (<) \;=\; \perp\!\!\perp \quad \} \\
& \text{true} \;\; .
\end{array}$$

□

A yet stronger theorem than theorem 14.21 is expressed by the invariant property

$$s^{\cup} \circ \leq \circ s \;\cap\; f^{\cup} \circ \geq \circ f \;\;=\;\; Tree^{*}$$

where $Tree$ is the subset of G capturing the “tree” edges: the edges $a \circ \top\!\!\top \circ b$ chosen by the selection criterion

$$a \circ \top\!\!\top \circ b \;\subseteq\; a \circ G \circ s{\succ}\!\bullet$$

in the procedure $dfs(a)$ (see fig. 12.4). Recall, however, that it is impossible to distinguish tree and ancestor edges using timestamps. Thus we are unable to exclude (non-tree) ancestor edges in the statement of theorem 14.21. Of course, it would be straightforward to augment the implementation to record tree edges and then prove the stronger theorem using the techniques we have presented. The distinction between “tree” and “ancestor” edges is, however, irrelevant —it reflects the nondeterminism in the choice of edges rather than being of intrinsic importance— and, in any case, the full strength of theorem 14.21 is not needed to establish the correctness of the algorithm for constructing strongly connected components: only the inclusion of the left side of the equality in the right side is needed, as we have shown in earlier sections.

14.4 Common Ancestors

Lengauer and Tarjan [LT79, Lemma 1] assert a property of depth-first search that is useful in calculating “dominators”. We do not discuss algorithms for computing dominators here; see [SMC12] for a point-free formulation of the fundamental properties of dominance. Here we restrict attention to formulating and validating Lengauer and Tarjan's assertion.

Slightly adapted to fit the terminology used here, the assertion is the following:

> If a and b are nodes of G such that $s.a \leq s.b$, then any path from a to b must contain a common ancestor of a and b in Tree.

Our formulation is as follows.

Theorem 14.24 If a and b are nodes of G such that $a{\circ}\top\!\!\top{\circ}b \subseteq s^{\cup}{\circ}{\leq}{\circ}s$, then any path from a to b must contain a node c such that $c\ \mathit{Anc}\ a$ and $c\ \mathit{Anc}\ b$ where the ancestor relation Anc is defined by

$$\mathit{Anc} \;=\; (s^{\cup}{\circ}{<}{\circ}s \;\cap\; f^{\cup}{\circ}{>}{\circ}f \cap G)^{*} \quad .$$

□

Our theorem is slightly weaker than Lengauer and Tarjan's assertion in that our “ancestor relation” is not the relation Tree^{*}. (The relation Tree is a subset of the relation $s^{\cup}{\circ}{<}{\circ}s \;\cap\; f^{\cup}{\circ}{>}{\circ}f \cap G$.) See the remarks following theorem 14.21 for an explanation of what is involved in strengthening the theorem.

The theorem clearly demands a constructive proof: an algorithm that computes for a given path a common ancestor of the two end-nodes of the path. The precondition of the algorithm is the postcondition of depth-first search. Theorem 14.23 plays a significant role: in the proof we use the equivalent definition

$$\mathit{Anc} \;=\; s^{\cup}{\circ}{\leq}{\circ}s \;\cap\; f^{\cup}{\circ}{\geq}{\circ}f \quad .$$

Unlike elsewhere in this document, our proof is not completely formal. We have not formalised the notion of a “path”, or being “on” a path. This means that some assertions are not formally justified. The techniques used in [SMC12] are applicable to filling this gap.

The algorithm is very simple: in the case that $a = b$, the “common ancestor” is chosen to be a and, in the case that $a \neq b$, the “common ancestor” c in the statement of the theorem is chosen to be the node on the given path that minimises the value of the function s. That is, for all nodes d on the path,

$$c{\circ}\top\!\!\top{\circ}d \;\subseteq\; s^{\cup}{\circ}{\leq}{\circ}s \quad .$$

(The pointwise equivalent is $s.c \leq s.d$.)

The case analysis on $a=b$ or $a \neq b$ turns out to be useful for the argument below: in the case that $a=b$ the given path may be non-empty, and it is simpler not to have to consider this possibility. Clearly the choice of the common ancestor in the case that $a=b$ is correct because the ancestor relation is reflexive. From now on, we assume that $a \neq b$.

The choice of any node c on the given path divides the path into a path from a to c and a path from c to b; the specific choice of c has the implication that

$$c{\circ}\top\!\top{\circ}b \;\; \subseteq \;\; \langle \mu X :: I \cup (X{\circ}G \cap s^{\cup}{\circ}{\leq}{\circ}s) \rangle \;\; .$$

In words, the part of the given path from c to b is such that every node d on it satisfies $s.c \leq s.d$. Equivalently —just use the absorption rule to replace at-most by less-than—

$$c{\circ}\top\!\top{\circ}b \;\; \subseteq \;\; \langle \mu X :: I \cup (X{\circ}G \cap s^{\cup}{\circ}{<}{\circ}s) \rangle \;\; .$$

Applying theorem 14.23, it follows that $c \; \mathcal{A}nc \; b$ as required.

It remains to show that $c \; \mathcal{A}nc \; a$. If $a=c$, this is trivially true. So assume that $a \neq c$. Now $c \; \mathcal{A}nc \; b$ (which we have just proved) is, by theorem 14.23, the property

$$c{\circ}\top\!\top{\circ}b \;\; \subseteq \;\; s^{\cup}{\circ}{\leq}{\circ}s \;\cap\; f^{\cup}{\circ}{\geq}{\circ}f \;\; .$$

We also have, by assumption,

$$a{\circ}\top\!\top{\circ}b \subseteq s^{\cup}{\circ}{<}{\circ}s$$

and, by the choice of c,

$$c{\circ}\top\!\top{\circ}a \subseteq s^{\cup}{\circ}{<}{\circ}s \;\; .$$

(Recall the assumptions that $a \neq b$ and $a \neq c$.) Now there are just two possibilities: either $b{\circ}\top\!\top{\circ}a \subseteq f^{\cup}{\circ}{>}{\circ}f$ or $a{\circ}\top\!\top{\circ}b \subseteq f^{\cup}{\circ}{>}{\circ}f$. In the first case,

$$\begin{array}{ll}
 & \mathsf{true} \\
= & \{ \quad \text{assumptions and choice of } c \quad \} \\
 & c{\circ}\top\!\top{\circ}a \subseteq s^{\cup}{\circ}{<}{\circ}s \;\wedge\; c{\circ}\top\!\top{\circ}b \subseteq f^{\cup}{\circ}{\geq}{\circ}f \;\wedge\; b{\circ}\top\!\top{\circ}a \subseteq f^{\cup}{\circ}{>}{\circ}f \\
\Rightarrow & \{ \quad \text{monotonicity and transitivity of } f^{\cup}{\circ}{\geq}{\circ}f \quad \} \\
 & c{\circ}\top\!\top{\circ}a \subseteq s^{\cup}{\circ}{<}{\circ}s \;\wedge\; c{\circ}\top\!\top{\circ}a \subseteq f^{\cup}{\circ}{>}{\circ}f \\
\Rightarrow & \{ \quad \text{definition of } \mathcal{A}nc \text{ and theorem 14.23} \quad \} \\
 & c \; \mathcal{A}nc \; a \;\; .
\end{array}$$

In the second case,

$$
\begin{array}{ll}
 & c \circ \top\!\top \circ a \subseteq f^{\cup} \circ < \circ f \\
= & \{ \quad \text{assumption: } c \circ \top\!\top \circ a \subseteq s^{\cup} \circ < \circ s \quad \} \\
 & c \circ \top\!\top \circ a \subseteq s^{\cup} \circ < \circ s \cap f^{\cup} \circ < \circ f \\
= & \{ \quad \text{on termination of depth-first search (see (13.5))} \\
 & \quad\; s^{\cup} \circ < \circ s \;\cap\; f^{\cup} \circ < \circ f \;=\; f^{\cup} \circ < \circ s \quad \} \\
 & c \circ \top\!\top \circ a \subseteq f^{\cup} \circ < \circ s \\
= & \{ \quad \text{assumption: } a \circ \top\!\top \circ b \subseteq s^{\cup} \circ < \circ s \text{, monotonicity and transitivity} \quad \} \\
 & c \circ \top\!\top \circ b \subseteq f^{\cup} \circ < \circ s \circ s^{\cup} \circ < \circ s \\
\Rightarrow & \{ \quad s \text{ is functional, } < \circ s \subseteq < \circ f \text{ and less-than is transitive} \quad \} \\
 & c \circ \top\!\top \circ b \subseteq f^{\cup} \circ < \circ f \\
= & \{ \quad c \; \mathcal{A}nc \; b \text{ (proved above), so } c \circ \top\!\top \circ b \subseteq f^{\cup} \circ \geq \circ f \quad \} \\
 & c \circ \top\!\top \circ b \subseteq f^{\cup} \circ < \circ f \cap f^{\cup} \circ \geq \circ f \\
= & \{ \quad f \text{ is functional, } < \cap \geq = \bot\!\bot \text{, } c \circ \top\!\top \circ b \neq \bot\!\bot \quad \} \\
 & \mathsf{false} \;\; .
\end{array}
$$

We conclude that

$$c \circ \top\!\top \circ a \subseteq f^{\cup} \circ \geq \circ f \;\; .$$

Combined with the choice of c, in particular $c \circ \top\!\top \circ a \subseteq s^{\cup} \circ \leq \circ s$, we have thus shown that $c \; \mathcal{A}nc \; a$.

(The property $< \circ s \subseteq < \circ f$ used in the implication step is the point-free formulation of the property that, for all nodes d, $s.d < f.d$. In words, the start time of each node is less than its finish time. More precisely, it is the point-free formulation of the property that, for all numbers m and all nodes d, $m < s.d \Rightarrow m < f.d$. We haven't actually proved this property! To do so it suffices to add the property

$$< \circ s \circ f_{>} \subseteq < \circ f$$

to the invariants of depth-first search. Its verification is straightforward.)

Note that the full extent of theorem 14.23 is used to establish theorem 14.24; the white-path theorem on its own is inadequate.

Part V

Concluding Remarks

The goal of this text has been to demonstrate the effectiveness of point-free relation algebra in reasoning about graph algorithms. We hope that our work may form the basis for an investigation into the effectiveness of contemporary machine-supported verification systems.

Some readers may conclude that our experiment has failed. In spite of our claim that point-free reasoning combines concision with precision, the length of this document may lead some to argue otherwise. Certainly, compared to informal proofs our calculations are substantially longer.

Aho, Hopcroft and Ullman [AHU82, pp. 219–226] present depth-first search, its application to computing a topological ordering of the nodes in an acyclic graph as well as to computing the strongly connected components of an arbitrary graph, all within less than ten pages. Their discussion of the correctness of the strongly-connected-components algorithm takes less than one page. Cormen, Leiserson and Rivest [CLR90, pp.465–497] cover the same ground in less than forty pages. Their account of the correctness of the algorithm for computing strongly connected components —which is much more thorough than that of Aho, Hopcroft and Ullman— amounts to five pages. The formal verification we have given is modelled on these two accounts but totals more than 100 pages. One may question whether this represents progress.

It has long been known that formal, axiomatic proofs are substantially longer than informal proofs in natural language. One reason is that formal proofs are necessarily more complete and are less prone to the sin of omission. More often than their formal counterparts, informal proofs tend to omit details that are considered "obvious" but nevertheless are essential to the argument. (An example is the property that the times-tamps in depth-first search are total, injective functions.) Informal proofs undergo what has been called a "social process" before they become accepted as legitimate: they rely on the agreement of sufficiently many experts that all steps are correct and have been adequately substantiated. (Undoubtedly, the graph algorithms discussed here have long ago passed this test and there is no question about their correctness.) Informal proofs achieve concision at the expense of precision.

We would argue that the formal proofs we have given *do* combine precision with concision. This combination is evident in the documentation that we provide. See, for example, fig. 13.1 in which properties of depth-first search are fully documented. An experienced, well-trained programmer will study the documentation in order to gain a full understanding of the implementation. Formal documentation of this nature can also be "executed" as a means of testing the implementation. Indeed, a well-trained programmer should be able to check for themself the veracity of the documentation, using it to design tests in cases of doubt.

Of course, mathematical formulae are less "readable" than natural language (at least to those for whom the natural language in use is the mother tongue) but natural language can be misleading: mathematical vernacular tends to be chosen so that it mimics everyday language but its familiarity can be deceptive[4]. Our point-free formulae will be even less readable to those unfamiliar with them but, we would argue, it is just a question of practice to gain the necessary reading and writing skills. Traditional pointwise formulae name variables that do not need to be named, and sometimes involve several layers of universal and existential quantifications.

For concrete instances of the extra precision —without loss of concision— we refer the reader to our discussion of the "white-path theorem" in sections 10.2.5 and 14.3. As we explained in section 10.2.5, the notion of a "white path" can have different definitions. The point-free calculus used here enables us to make the distinction concisely and precisely — as we did in section 14.3. The calculus also allows us to identify exactly which properties are necessary to establish the correctness of the algorithm for computing strongly connected components: theorem 14.21 on "ancestor paths" does add to a proper understanding of depth-first search but weaker properties suffice for understanding how it is exploited.

Textbook accounts of graph algorithms typically rely on an informal, operational understanding of program statements. We have presented a basic "Algol-like" language to which we have given a simple non-operational relational semantics. In this way, we have met the goal of clarifying the basis of our formal arguments. The (now well-known) relevance of regular algebra to reasoning about simple loops has been prominent throughout; chapter 12 goes much further in demonstrating the application of fixed-point calculus in reasoning about so-called "recursive" programs.

As mentioned in the introduction, our next step is to explore how good contemporary verification systems are in the task of verifying non-trivial graph algorithms. For example, to what extent is it possible to supply such a system with the formal documentation and then have it checked without human intervention? Many of the calculations included here are straightforward, leading to the hope that they might be automatically reconstructed. If so, then the seemingly overwhelming explosion in the length of documents like this one may not be so inevitable after all.

[4]For example, we choose to use the term "conditional correctness" rather "partial correctness" because being "partially" correct may also be interpreted as partially incorrect. We use the term "correctness" reluctantly because it suggests something absolute. We prefer to say that a program "meets its specification", thus allowing for the possibility that the specification is flawed.

Bibliography

[ABH$^+$92] C.J. Aarts, R.C. Backhouse, P. Hoogendijk, T.S. Voermans, and J. van der Woude. A relational theory of datatypes. Available via World-Wide Web at `http://www.cs.nott.ac.uk/~psarb2/papers`, September 1992.

[AC75] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[AHU82] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, 1982.

[Bac75] R.C. Backhouse. *Closure algorithms and the star-height problem of regular languages.* PhD thesis, University of London, 1975. Available at `https://spiral.imperial.ac.uk/bitstream/10044/1/22243/2/Backhouse-RC-1976-PhD-Thesis.pdf`.

[Bac00] Roland Backhouse. Fixed point calculus. Summer School and Workshop on Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, available at `http:/www.cs.nott.ac.uk/~psarb2/MPC/acmmpc.pdf`, April 2000.

[Bac02] Roland Backhouse. Galois connections and fixed point calculus. In Roland Backhouse, Roy Crole, and Jeremy Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, volume 2297 of *LNCS Tutorial*, chapter 4, pages 89–148. Springer, 2002. International Summer School and Workshop, Oxford, UK, April 2000, Revised Lectures (Abridged version of [Bac00]).

[Bac03] Roland Backhouse. *Program Construction. Calculating Implementations From Specifications.* John Wiley & Sons, Ltd., 2003.

[Bac06] Roland Backhouse. Regular algebra applied to language problems. *Journal of Logic and Algebraic Programming*, 66:71–111, 2006.

[Bac11] Roland Backhouse. *Algorithmic Problem Solving.* John Wiley & Sons, 2011.

[Bac16] Roland Backhouse. Factor theory and the unity of opposites. *J. Logical and Algebraic Methods in Programming*, 85(5):824–846, 2016.

[BC75] R.C. Backhouse and B.A. Carré. Regular algebra applied to path-finding problems. *Journal of the Institute of Mathematics and its Applications*, 15:161–186, 1975.

[BC82] R.C. Backhouse and B.A. Carré. A comparison of Gaussian and Gauss-Jordan elimination in regular algebra. *International Journal of Computer Mathematics*, 10:311–325, 1982.

[BDGv22] Roland Backhouse, Henk Doornbos, Roland Glück, and Jaap van der Woude. Components and acyclicity of graphs. an exercise in combining precision with concision. *Journal of Logical and Algebraic Methods in Programming*, 124:100730, 2022.

[BdM97] Richard S. Bird and Oege de Moor. *Algebra of Programming.* Prentice-Hall International, 1997.

[BL77] R.C. Backhouse and R.K. Lutz. Factor graphs, failure functions and bi-trees. In A. Salomaa and M. Steinby, editors, *Fourth Colloquium on Automata, Languages and Programming*, pages 61–75. Springer-Verlag, LNCS 52, July 1977.

[BN98] Lex Bijlsma and Rob Nederpelt. Dijkstra-Scholten predicate calculus: concepts and misconceptions. *Acta Informatica*, 35:1007–1036, 1998.

[Brz67] J.A. Brzozowski. Roots of star events. *Journal of the ACM*, 14(3):466–477, July 1967.

[BW93] R.C. Backhouse and J. van der Woude. Demonic operators and monotype factors. *Mathematical Structures in Computer Science*, 3(4):417–433, December 1993.

[Car71] B.A. Carré. A network routing algebra. *J.Inst.Maths.Applics.*, 7:273–294, 1971.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Electrical Engineering and Computer Science Series, MIT Press, 1990.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd edition*. MIT Electrical Engineering and Computer Science Series, MIT Press, 2009.

[Con71] J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, London, 1971.

[DB02] Henk Doornbos and Roland Backhouse. Algebra of program termination. In Roland Backhouse, Roy Crole, and Jeremy Gibbons, editors, *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction*, volume LNCS 2297 of *Lecture Notes in Computer Science, Tutorial Series*, pages 203–235. Springer, 2002.

[DBvdW97] Henk Doornbos, Roland Backhouse, and Jaap van der Woude. A calculational approach to mathematical induction. *Theoretical Computer Science*, 179(1–2):103–135, 1 June 1997.

[DM60] A. De Morgan. On the syllogism, no. iv, and on the logic of relations. *Transactions of the Cambridge Philosophical Society*, 1860. Reprinted in [DM66].

[DM66] Augustus De Morgan. *On the Syllogism and Other Logical Writings*. Yale University Press, New Haven, 1966. Edited, with an Introduction, by Peter Heath.

[Doo96] H. Doornbos. *Reductivity arguments and program construction*. PhD thesis, Department of Mathematics and Computer Science, June 1996.

[DS90] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and Program Semantics*. Texts and monographs in Computer Science. Springer-Verlag, 1990.

[Fv90] P.J. Freyd and A. Ščedrov. *Categories, Allegories*. North-Holland, 1990.

[Glü17] Roland Glück. Algebraic investigation of connected components. In P. Höfner, D. Pous, and G. Struth, editors, *Relational and Algebraic Methods in Computer Science – 16th International Conference, RAMiCS 2017*, volume 10226 of *Lecture Notes in Computer Science*, pages 109–126. Springer, May 15–18 2017.

[Gri81] D. Gries. *The Science of Programming*. Springer-Verlag, New York, 1981.

[Hoo97] Paul Hoogendijk. *A Generic Theory of Datatypes*. PhD thesis, Department of Mathematics and Computer Science, 1997.

[KMP77] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6:325–350, June 1977.

[LT79] Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Transactions on Programming Languages and Systems*, 1(1):121–141, 1979.

[Mad91] Roger D. Maddux. The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica*, 50(3–4):421–455, 1991.

[Mat95] Mathematics of Program Construction Group, Eindhoven University of Technology. Fixed-point calculus. *Information Processing Letters*, 53(3):131–136, February 1995.

[Pei70] C.S. Peirce. Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic. *Memoirs of the American Academy of Sciences*, 9:317–378, 1870. Reprinted in [Pei33].

[Pei33] C.S. Peirce. *Collected Papers*. Harvard University Press, 1933.

[Pra92] V.R. Pratt. Origins of the calculus of binary relations. In *Logic in Computer Science*, pages 248–254. IEEE Computer Society Press, 1992.

[Rig48] J. Riguet. Relations binaires, fermetures, correspondances de Galois. *Bulletin de la Société Mathématique de France*, 76:114–155, 1948.

[Sal69] A. Salomaa. *Theory of Automata*. Pergamon Press, Oxford, 1969.

[Sch95] E. Schröder. *Algebra der Logik*, volume 3. Teubner, Leipzig, 1895.

[Sha81] M. Sharir. A strong-connectivity algorithm and its application in data flow analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.

[SMC12] Ilya Sergey, Jan Midtgaard, and Dave Clarke. Calculating graph algorithms for dominance and shortest path. In Jeremy Gibbons and Pablo Nogueira, editors, *Mathematics of Program Construction, 11th International Conference, MPC2012*, volume LNCS 7342, pages 132–156. Springer, 2012.

[SS88] G. Schmidt and T. Ströhlein. *Relationen und Grafen.* Springer-Verlag, 1988.

[SS93] G. Schmidt and T. Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin Heidelberg, 1993.

[Tar41] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.

[Tar72] Robert Endre Tarjan. Depth first search and linear graph algorithms. *SIAM J. Computing*, pages 146–160, 1972.

[TG87] Alfred Tarski and Steven Givant. *A Formalization of Set Theory without Variables*, volume 41 of *Colloquium Publications.* American Mathematical Society, Providence, Rhode Island, 1987.

[Voe99] Ed (Theodorus Sebastiaan) Voermans. *Inductive Datatypes with Laws and Subtyping – A Relational Model.* PhD thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, 1999.

[Wei73] P. Weiner. Linear pattern matching algorithms. In *Conf. Record IEEE 14th Annual Symposium on Switching and Automata*, pages 1–11, 1973.

Index