Automated Theorem Proving

Georg Struth

University of Sheffield

Motivation

everybody loves my baby
but my baby ain't love nobody but me

(Doris Day)

Overview

main goal: we will learn

- how ATP systems work (in theory)
- where ATP systems can be useful (in practice)

main topics: we will discuss

- solving equations: term rewriting and Knuth-Bendix completion
- saturation-based ATP
- conjecture and refutation games in mathematics
- logical modelling and problem solving with ATP systems and SAT solvers

glimpses into: universal algebra, order theory/combinatorics, termination, computational algebra, semantics, . . .

Term Rewriting

example: (grecian urn) An urn holds 150 black beans and 75 white beans. You successively remove two beans. A black bean is put back if both beans have the same colour. A white bean is put back if their colour is different.
Is the colour of the last bean fixed? Which is it?

$$BB \to B \qquad WW \to B \qquad WB \to W \qquad BW \to W$$
$$BW \to WB \qquad WB \to BW$$

questions:

- are these "good" rules?
- does system terminate?
- is there determinism?

Term Rewriting

example: (chameleon island) The chameleons on this island are either red, yellow or green. When two chameleons of different colour meet, they change to the third colour. Assume that 15 red, 14 yellow and 13 green chameleons live on the island. Is there a stable (monochromatic) state?

$$RY \to GG \qquad YR \to GG \qquad GY \to RR$$
$$YG \to RR \qquad RG \to YY \qquad GR \to YY$$

questions:

- does system terminate?
- how can rewriting solve the puzzle?

Term Rewriting

example: Consider the following rules for monoids

$$(xy)z \to x(yz) \qquad 1x \to x \qquad x1 \to x$$

questions:

- does this yield normal forms?
- can we decide whether two monoid terms are equivalent?

Term Rewriting

examples: consider the following rules for the stack

$$\mathsf{top}(\mathsf{push}(x,y)) \rightarrow x \qquad \mathsf{pop}(\mathsf{push}(x,y)) \rightarrow y$$

$$\mathsf{empty?}(\perp) \rightarrow \mathsf{T} \qquad \mathsf{empty?}(\mathsf{push}(x,y)) \rightarrow \mathsf{F}$$

question: what about the rule

$$\mathsf{push}(\mathsf{top}(x), \mathsf{pop}(x)) \rightarrow x$$

which applies if $\mathsf{empty?}x = \mathsf{F}$?

Terms and Term Algebras

terms: $T_\Sigma(X)$ denotes set of terms over signature Σ and variables from X

$$t ::= x \mid f(t_1, \dots t_n)$$

constants are functions of arity 0

ground term: term without variables

remark: terms correspond to labelled trees

Terms and Term Algebras

example: Boolean algebra

- signature $\{+, \cdot, ^{-}, 0, 1\}$
- $+$, $\cdot$ have arity 2; $^{-}$ has arity 1; 0,1 have arity 0
- terms

$$+(x, y) \;\approx\; x + y \qquad\qquad \cdot\,(x, +(y, z)) \;\approx\; x \cdot (y + z)$$

intuition: terms make the sides of equations

$$(x + y) + z = x + (y + z) \qquad x + y = y + x \qquad x = \overline{\overline{x} + \overline{y}} + \overline{\overline{x} + y}$$

$$x \cdot y = \overline{\overline{x} + \overline{y}}$$

Terms and Term Algebras

substitution:

- partial map $\sigma : X \to T_\Sigma(X)$ (with finite domain)
- all occurrences of variables in $\mathsf{dom}(\sigma)$ are replaced by some term
- "homomorphic" extension to terms, equations, formulas,. . .

example: for $f(x, y) = x + y$ and $\sigma : x \mapsto x \cdot z, y \mapsto x + y$,

$$f(x, y)\sigma = f(x \cdot z, x + y) = (x \cdot z) + (x + y)$$

remark: substitution is different from replacement:
replacing term s in term $r(\dots s \dots)$ by term t yields $r(\dots t \dots)$

Terms and Term Algebras

Σ-algebra: structure $(A, (f_A : A^n \to A)_{f \in \Sigma})$

interpretation (meaning) of terms

- assignment $\alpha : X \to A$ gives meaning to variables
- homomorphism $I_\alpha : T_\Sigma(X) \to A$
  - $I_\alpha(x) = \alpha(x)$ for all variables
  - $I_\alpha(c) = c_A$ for all constants
  - $I_\alpha(f(t_1, \ldots, t_n)) = f_A(I_\alpha(t_1), \ldots, I_\alpha(t_n))$

equations: $A \models s = t \;\Leftrightarrow\; I_\alpha(s) = I_\alpha(t)$ for all α.

Terms and Term Algebras

examples:

- BA terms can be interpreted in BA $\{0, 1\}$ via truth tables; row gives I_α
- operations on finite sets can be given as Cayley tables

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

($\mathbb{N}$ mod 4)

Deduction and Reduction

equtional reasoning: does E imply $s = t$?

- Proofs:
  1. use rules of equational logic
     (reflexivity, symmetry, transitivity,congruence,substitution,Leibniz,. . .)
  2. use rewriting (orient equations, look for canonical forms)
- Refutations: Find model A with $A \models E$ and $A \models s \neq t$

example: equations for Boolean algebra

- imply $x \cdot y = y \cdot x$ (prove it)
- but not $x + y = x$ (find counterexample)

question: does $f f f\ x = f\ x$ imply $f f\ x = f\ x$?

Rewriting

question: how can we effectively reduce to canonical form?

- reduction sequences must terminate
- reduction must be deterministic
  (diverging reductions must eventually converge)

examples:

- the monoid rules generate canonical forms (why?)
- the adjusted grecian urn rules are terminating (why?)
- the chameleon island rules are not terminating (why?)

Abstract Reduction

abstract reduction system: structure $(A, (R_i)_{i \in I})$
with set A and binary relations R_i

here: one single relation $\longrightarrow$ with

- $\longleftarrow$ converse of $\longrightarrow$
- $\longrightarrow \circ \longrightarrow$ relative product
- $\longleftrightarrow = \longrightarrow \cup \longleftarrow$
- $\longrightarrow^+$ transitive closure of $\longrightarrow$
- $\longrightarrow^*$ reflexive transitive closure of $\longrightarrow$

remarks:

- $\longrightarrow^+$ is transitive
- $\longrightarrow^*$ is preorder

Abstract Reduction

terminology:

- $a \in A$ reducible if $a \in \mathsf{dom}(\longrightarrow)$
- $a \in A$ normal form if $a \in \overline{\mathsf{dom}(\longrightarrow)}$
- b nf of a if $a \longrightarrow^* b$ and b nf
- $\longrightarrow^* \circ \longleftarrow^*$ is called rewrite proof

properties:

- Church-Rosser $\longleftrightarrow^* \subseteq \longrightarrow^* \circ \longleftarrow^*$
- confluence $\longleftarrow^* \circ \longrightarrow^* \subseteq \longrightarrow^* \circ \longleftarrow^*$
- local confluence $\longleftarrow \circ \longrightarrow \subseteq \longrightarrow^* \circ \longleftarrow^*$
- wellfounded no infinite $\longrightarrow$ sequences
- convergence is confluence and wf

Abstract Reduction

theorems: (canonical forms)

- Church-Rosser equivalent to confluence
- confluence equivalent to local confluence and wf

intuition: local confluence yields local criterion for CR

termination proofs: let $(A, <_A)$ and $(B, \leq_B)$ be posets with $\leq_B$ wf
then $\leq_A$ wf if there is monotonic $f : A \to B$

intuition: reduce termination analysis to "well known" order like $\mathbb{N}$

proofs: as exercises

Term Rewriting

term rewrite system: set R of rewrite rules $l \longrightarrow r$ for $l, r \in T_\Sigma(X)$

one-step rewrite: $t(\dots l\sigma \dots) \longrightarrow t(\dots r\sigma \dots)$ for $l \longrightarrow r \in R$ and σ substitution
(if l matches subterm of t then subterm is replaced by $r\sigma$)

rewrite relation: smallest $\longrightarrow_R$ containing R and closed
under contexts (monotonic) and substitutions (fully invariant)

example: $1 \cdot (x \cdot (y \cdot z)) \longrightarrow x \cdot (y \cdot z)$ is one-step rewrite with
monoid rule $1 \cdot x \longrightarrow x$ and substitution $\sigma : x \mapsto x \cdot (y \cdot z)$

Term Rewriting

fact: convergent TRSs can decide equational theories

theorem: (Birkhoff) $E \models \forall \vec{x}.s = t \;\Leftrightarrow\; s \leftrightarrow_E^* t \;\Leftrightarrow\; \mathsf{cf}(s) = \mathsf{cf}(t)$
(canonical forms generate free algebra $T_\Sigma(X)/E$)

corollary: theories of finite convergent sets of equations are decidable

question: how can we turn E into convergent TRS?

Local Confluence in TRS

observation:

- local confluence depends on overlap of rewrite rules in terms
- if $l_1 \to r_1$ rewrites a "skeleton subterm" l'_2 of $l_2 \to r_2$ in some t then $l_1\sigma_1$ and $l_2\sigma_2$ must be subterms of t and $l_1\sigma_1 = l'_2\sigma_2$
- if variables in l_1 and l'_2 are disjoint, then $l_1(\sigma_1 \cup \sigma_2) = l'_2(\sigma_1 \cup \sigma_2)$
- $\sigma_1 \cup \sigma_2$ can be decomposed into σ which "makes l_1 and l'_2 equal" and σ' which further instantiates the result

unifier of s and t: a subsitution σ such that $s\sigma = t\sigma$

facts:

- if terms are unifiable, they have most general unifiers
- mgus are unique and can be determined by efficient algorithms

Unification

naive algorithm: (exponential in size of terms)

$$E, s = s \Rightarrow E$$

$$E, f(s_1, \dots, s_n) = f(t_1, \dots, t_n) \Rightarrow E, s_1 = t_1, \dots, s_n = t_n$$

$$E, f(\dots) = g(\dots) \Rightarrow \bot$$

$$E, t = x \Rightarrow E, x = t \quad \text{if } t \notin X$$

$$E, x = t \Rightarrow \bot \quad \text{if } x \neq t \text{ and } x \text{ occurs in } t$$

$$E, x = t \Rightarrow E[t/x], x = t \quad \text{if } x \text{ doesn't occur in } t$$

Unification

example:

$$f(g(x,b),f(x,z)) = f(y,f(g(a,b),c))$$

$$\Downarrow$$

$$\ldots$$

$$\Downarrow$$

$$y = g(g(a,b),b),\ x = g(a,b),\ z = c$$

Critical Pairs

task: establish local confluence in TRS

question: how can rewrite rules overlap in terms?

- disjoint redexes (automatically confluent)
- variable overlap (automatically confluent)
- skeleton overlap (not necessarily confluent)

. . . see diagrams

conclusion: skeleton overlaps lead to equations that may not have rewrite proofs

Critical Pairs

critical pairs: $l_1\sigma(\dots r_2\sigma\dots) = r_1\sigma$ where

- $l_1 \to r_1$ and $l_2 \to r_2$ rewrite rules
- σ mgu of l_2 and subterm l_1' of l_1
- $l_1' \notin X$

example: $x + (-x) \to 0$ and $x + ((-x) + y) \to y$ have cp $x + 0 = -(-x)$

theorem: A TRS is locally confluent iff all critical pairs have rewrite proofs

remark: confluence decidable for finite wf TRS
(only finitely many cps must be inspected)

Wellfoundedness/Termination

fact: proving termination of TRSs requires complex constructions

lexicographic combination: for posets $(A_1, <_1)$ and $(A_2, <_2)$ define $<$ of type $A_1 \times A_2$ by

$$(a_1, a_2) > (b_1, b_2) \;\Leftrightarrow\; a_1 >_1 b_1, \text{ or } a_1 = b_1 \text{ and } a_2 > b_2$$

then $(A_1 \times A_2, <)$ is a poset and $<$ is wf iff $<_1$ and $<_2$ are

proof: exercise (wellfoundedness)

Wellfoundedness/Termination

multiset over set A: map $m : A \to \mathbb{N}$

remark: consider only finite multisets

multiset extension: for poset $(A, <)$ define $<$ of type $(A \to \mathbb{N}) \times (A \to \mathbb{N})$ by

$$m_1 > m_2 \;\Leftrightarrow\; m_1 \neq m_2 \text{ and}$$
$$\forall a \in A.(m_2(a) > m_1(a) \Rightarrow \exists b \in A.(b > a \text{ and } m_1(b) > m_2(b)))$$

this is a partial order; it is wellfounded if the underlying order is

proof: exercise (wellfoundedness)

Reduction Orderings

idea: for finite TRS, inspect only finitely many rules for termination

reduction ordering: wellfounded partial ordering on terms
such that all operations and substitutions are order preserving

fact: TRS terminates iff $\rightarrow$ is contained in some reduction ordering

nontermination: rewrite rules of form

- $x \rightarrow t$
- $l(x_1, \ldots, x_n) \rightarrow r(x_1, \ldots, x_n, y)$ (why?)

in practice: reduction orderings should have computable approximations
(halting problem)

interpretation: reduction orderings are wf iff all ground instantiations are wf

Reduction Orderings

polynomial orderings:

- associate function terms with polynomial weight functions with integer coeficients
- checking ordering constraints can be undecidable (Hilbert's 10th problem)
- restrictions must be imposed

Reduction Orderings

simplification orderings: monotonic ordering on terms that contain the (strict) subterm ordering

theorem: simplification orderings over finite signatures are wf

proof: by Kruskal's theorem

example: $ff\ x \to fgf\ x$ terminates and induces reduction ordering $>$

1. assume $>$ is simplification ordering
2. $f\ x$ is subterm of $gf\ x$, hence $gf\ x > f\ x$
3. then $fgf\ x > ff\ x$ by monotonicity
4. so $ff\ x > ff\ x$, a contradiction
5. conclusion: wf not always captured by simplification ordering

Simplification Orderings

lexicographic path ordering: for precedence $\succ$ on Σ define relation $>$ on $T_\Sigma(X)$

- $s > x$ if x proper subterm of s, or
- $s = f(s_1, \ldots s_m) > g(t_1, \ldots, t_n) = t$ and
  - $s_i > t$ for some i or
  - $f \succ g$ and $s > t_i$ for all i or
  - $f = g$, $s > t_i$ for all i and $(s_1, \ldots, s_m) > (t_1, \ldots, t_m)$ lexicographically

fact: lpo is simplification ordering, it is total if the precedence is

variations:

- multiset path ordering: compare subterms as multisets
- recursive path ordering: function symbols have either lex or mul status
- Knuth-Bendix ordering: hybrid of weights and precedences

Knuth-Bendix Completion

idea: take set of equations and reduction ordering

- orient equations into decreasing rewrite rules
- inspect all critial pairs and add resulting equations
- delete trivial equations
- if all equations can be oriented, KB-closure contains convergent TRS

extension: delete redundant expressions, e.g.
if $r \rightarrow s, s \rightarrow t \in R$, then adding $r \rightarrow t$ to R makes $r \rightarrow s$ redundant

therefore:

- KB-completion combines deduction and reduction
- this is essentially basis construction

Knuth-Bendix Completion

rule based algorithm: let $<$ be reduction ordering

- delete $E, R, t = t \Rightarrow E, R$
- orient: $E, R, s = t \Rightarrow E, R, s \rightarrow t \quad$ if $s > t$
- deduce: $E, R \Rightarrow E, R, s = t \quad$ if $s = t$ is cp from R
- simplify: $E, R, r = s \Rightarrow E, R, r = t \quad$ if $s \rightarrow_R t$
- compose: $E, R, r \rightarrow s \Rightarrow E, R, r \rightarrow t \quad$ if $s \rightarrow_R t$
- collapse: $E, R, r \rightarrow s \Rightarrow E, R, s = t \quad$ if $r \rightarrow_R t$ rewrites strict subterm

remark: permutations in $s = t$ are implicit

strategy: $(((simplify + delete)^*; (orient; (compose + collapse)^*))^*; deduce)^*$

Knuth-Bendix Completion

properties: the following facts can be shown

- soundness: completion doesn't change equational theory
- correctness: if process is fair (all cps eventually computed) and all equations can be oriented, then limit yields convergent TR; "KB-basis"

main construction: use complex wf order on proofs to show that all completion steps decrease proofs, hence induce rewrite proofs

observation: completion need not succeed

- it can fail to orient persistent equations
- it can loop forever

fact: if completion succeeds, it yields canonical TRS (convergent and interreduced)

Knuth-Bendix Completion

observation:

- KB-completion always succeeds on ground TRSs (congruence closure)
- KB-completion wouldn't fail when $<$ is total
- but rules $xy = yx$ can never be oriented

unfailing completion: only rewrite with equations when this causes decrease

- let $l_1 \to r_1$ and $l_2 \to r_2$
- let l_1' be "skeleton" subterm of l_1
- let σ be mgu of l_1' and l_2
- let μ be substitution with $l_1\sigma\mu \not\leq r_1\sigma\mu$ and $l_1\sigma\mu \not\leq l_1\sigma(\ldots r_2\sigma \ldots)\mu$

then $l_1\sigma(\ldots r_2\sigma \ldots) = r_1\sigma$ is ordered cp for deduction

Knuth-Bendix Completion

remarks:

- unfailing completion is a complete ATP procedure for pure equations
- this has been implemented in the Waldmeister tool

Knuth-Bendix Completion

example: groups

- input: appropriate ordering and equations

$$1 \cdot x = x \qquad x^{-1} \cdot x = 1 \qquad (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

- output: canonical TRS

$$1^{-1} \to 1 \qquad x \cdot 1 \to x \qquad 1 \cdot x \to x \qquad (x^{-1})^{-1} \to x$$

$$x^{-1} \cdot x \to 1 \qquad x \cdot x^{-1} \to x \qquad x^{-1} \cdot (x \cdot y) \to y$$

$$x \cdot (x^{-1} \cdot y) \to y \qquad (x \cdot y)^{-1} \to y^{-1} \cdot x^{-1} \qquad (x \cdot y) \cdot z \to x \cdot (y \cdot z)$$

Knuth-Bendix Completion

example: groups (cont.)

proof of $(x^{-1} \cdot (x \cdot y))^{-1} = (x^{-1} \cdot y)^{-1} \cdot x^{-1}$

$$\begin{aligned}
(x^{-1} \cdot (x \cdot y))^{-1} &\to_R (y^{-1} \cdot (x^{-1})^{-1}) \cdot x^{-1} \\
&\to_R y^{-1} \cdot ((x^{-1})^{-1} \cdot x^{-1}) \\
&\to_R y^{-1} \cdot 1 \\
&\leftarrow_R (x^{-1} \cdot y)^{-1} \cdot x^{-1}
\end{aligned}$$

Propositional Resolution

literals are either

- propositional variables P (positive literals) or
- negated propositional variables $\neg P$ (negative literals)

clauses are disjunctions (multisets) of literals

clause sets are conjunctions of clauses

property: every propositional formula is equivalent to a clause set
(linear structure preserving algorithm)

Propositional Resolution

orders Let S be clause set

- consider total wf order $<$ on variables
- extend lexicographically to pairs (P, π) on literals where π is 0 for positive literals and 1 for negative ones
- compare clauses with the multiset extension of that order

consequence: S totally ordered by wf order $<$

Propositional Resolution

building models: partial model H is set of positive literals

- inspect clauses in increasing order
- if clause is false and maximal literal P, throw P in H
- if clause is true, or false and maximal literal negative, do nothing

question: does this yield model of S?

first reason for failure: clause set $\{\Gamma \vee P \vee P\}$ has no model if P maximal

remedy: merge these literals (ordered factoring)

$$\frac{\Gamma \vee P \vee P}{\Gamma \vee P} \qquad \text{if } P \text{ maximal}$$

Propositional Resolution

second reason for failure: literals ordered according to indices

clauses	partial models
P_1	$\{P_1\}$
$P_0 \vee \neg P_1$	$\{P_1\}$
$P_3 \vee P_4$	$\{P_1, P_4\}$

$\{P_1, P_4\} \not\models P_0 \vee \neg P_1$, but $\{P_0, P_1, P_4\} \models P_0 \vee \neg P_1$

remedy: add clause P_0 to set (it is entailed)

more generally: (ordered resolution)

$$\frac{\Gamma \vee P \qquad \Delta \vee \neg P}{\Gamma \vee \Delta} \qquad \text{if } (\neg)P \text{ maximal}$$

Propositional Resolution

resolution closure: (saturation) $R(S)$

theorem: If $R(S)$ doesn't contain the empty clause then the construction yields model for S

proof: by wf induction

1. failing construction has minimal counterexample C
2. either positive maximal literal occurs more then once, then factoring yields smaller counterexample
3. or maximal literal is negative, then resolution yields smaller counterexample
4. both cases yield contradiction

corollary: $R(S)$ contains empty clause iff R inconsistent

Propositional Resolution

resolution proofs: (refutational completeness) the empty clause can be derived from all finite inconsistent clause sets

proof: by closure construction, the empty clause is derived after finitely many steps

theorem: (compactness) S is unsatisfiable iff some finite subset is

proof: use the hypotheses from refutation

theorem: resolution decides propositional logic

proof: the maximal clause C in S is the maximal clause in $R(S)$, and there are only finitely many smaller clauses that S

Propositional Resolution

alternative completeness proof:

- write rules as

$$\frac{\Gamma \to P \vee \Delta \qquad \Gamma' \wedge P \to \Delta'}{\Gamma \wedge \Gamma' \to \Delta \vee \Delta'} \qquad\qquad \frac{\Gamma \to P \vee P \vee \Delta}{\Gamma \to P \vee \Delta}$$

- read them as inequalities between nf terms in bounded distributive lattice
- understand resolution as cp computation for inequalities
- use wf proof order argument to prove existence of proof $1 \to 0$

A Resolution Proof

```
1 -A | B.  [assumption].
2 -B | C.  [assumption].
3 A | -C.  [assumption].
4 A | B | C.  [assumption].
5 -A | -B | -C.  [assumption].
6 A | B.  [resolve(4,c,3,b),merge(c)].
7 A | C.  [resolve(6,b,2,a)].
8 A.  [resolve(7,b,3,b),merge(b)].
9 -B | -C.  [back_unit_del(5),unit_del(a,8)].
10 B.  [back_unit_del(1),unit_del(a,8)].
11 -C.  [back_unit_del(9),unit_del(a,10)].
12 $F.  [back_unit_del(2),unit_del(a,10),unit_del(b,11)].
```

First-Order Resolution

idea:

- transform formulas in prenex form
  (quantfier prefix follows by quantifier free formula)
- Skolemise existential quantifiers $\forall\vec{x}\exists y.\phi \;\Rightarrow\; \forall\vec{x}.\phi[f(\vec{x})/y]$
- drop universal quantifier
- transform in CNF

fact: Skolemisation preserves satisfiability

example: $\forall x.R(x,x) \wedge (\exists y.P(y) \vee \forall x.\exists y.R(x,y) \vee \forall z.Q(z))$ becomes
$\forall x.R(x,x) \wedge (P(a) \vee \forall x.R(x,f(x)) \vee \forall z.Q(z))$

First-Order Resolution

motivation:

- the premises $P(f(x,a)$ and $\neg P(f(y,z) \vee \neg P(f(z,y))$ imply $\neg P(f(a,x)$
- this conclution is most general with respect to instantiation
- it can be obtained from the mgu of $f(x,a)$ and $f(z,y)$ etc

first-order resolution:

- don't instantiate, unify (less junk in resolution closure)
- unification istead of identification

$$\frac{\Gamma \vee P \qquad \Delta \vee \neg P'}{(\Gamma \vee \Delta)\sigma} \qquad \frac{\Gamma \vee P \vee P'}{(\Gamma \vee P)\sigma} \qquad \sigma = mgu(P, P')$$

Lifting

question: are all ground inferences instances of non-ground ones?

theorem: (lifting lemma)

- let $\mathsf{res}(C_1, C_2)$ denote the resolvent of C_1 and C_2
- let C_1 and C_2 have no variables in common
- let σ be substitution

then $\mathsf{res}(C_1\sigma, C_2\sigma) = \mathsf{res}(C_1, C_2)\rho$ for some substitution ρ

remark: similar property for factoring

consequences: (refutational completeness)

- if clause set is closed then set of all ground instances is closed
- resolution derives the empty clause from all inconsistent inputs

Redundancy

question:

- KB-completion allows the deletion of redundant equations
- is this possible for resolution?

idea: basis construction

- compute resolution closure
- then delete all clauses that are entailed by other clauses
- but model construction "forgets" what happened in the past
- clauses entailed by smaller clauses need not be inspected
- they can never contribute to model or become counterexamples
- can deletion of redundant clauses be stratified?
- can that be formalised?

Redundancy

idea: approximate notion of redundancy with respect to clause ordering

definition:

- clause C is redundant with respect to clause set Γ
  if for some finite $\Gamma' \subseteq \Gamma$

$$\Gamma' \models C \quad \text{and} \quad C > \Gamma'$$

- resolution inference is redundant if its conclusion is entailed by one of the premises and smaller clauses (more or less)

fact: it can be shown that resolution is refutationally complete up to redundancy

intuition: construction of ordered resolution bases

Redundancy

examples:

- tautologies are redundant (they are entailed by the empty set of clauses)
- clause C' is subsumed by clause C if

$$C\sigma \subseteq C'$$

clauses that are subsumed are redundant

A Simple Resolution Prover

rule-based procedure: N "new resolvents", P "processed clauses", O "old clauses"

- tautology deletion if C tautology

$$N, C; P; O \Rightarrow N; P; O$$

- forward subsumption if clause in $P; O$ subsumes C

$$N, C; P; O \Rightarrow N; P; O$$

- backward subsumption if clause in N properly subsumes C

$$N; P, C; O \Rightarrow N; P; O \qquad N; P; , O, C \Rightarrow N; P; O$$

A Simple Resolution Prover

- forward reduction if ex. $D \vee L'$ in $P; O$ such that $\overline{L} = L'\sigma$ and $C\sigma \subseteq D$

$$N, C \vee L; P; O \Rightarrow N, C; P; O$$

- backward reduction if ex. $D \vee L'$ in N such that $\overline{L} = L'\sigma$ and $C\sigma \subseteq D$

$$N; P, C \vee L; O \Rightarrow N; P, C; O \qquad N; P; O, C \vee L \Rightarrow N; P; O, C$$

- clause processing

$$N, C; P; O \Rightarrow N; P, C; O$$

- inference computation N is closure of O, C

$$\emptyset; P, C; O \Rightarrow N; P; O, C$$

ATP in First-Order Logic with Equations

naive approach:

- equality is a prediate; axiomatise it
- . . . not very efficient

but KB-completion is very similar to ordered resolution
deduction and reduction techniques are combined

idea:

- integrate KB-completion/unfailing completion into ordered resolution
- this yields superposition calculus

Superposition Calculus

assumption: consider equality as only predicate (predicates as Boolean functions)

inference rules: (ground case)

- equality resolution

$$\frac{\Gamma \vee t \neq t}{\Gamma}$$

- positive and negative superposition

$$\frac{\Gamma \vee l = r \qquad \Delta \vee s(\dots l \dots) = t}{\Gamma \vee \Delta \vee s(\dots r \dots) = t} \qquad \frac{\Gamma \vee l = r \qquad \Delta \vee s(\dots l \dots) \neq t}{\Gamma \vee \Delta \vee s(\dots r \dots) \neq t}$$

- equality factoring

$$\frac{\Gamma \vee s = t \vee s = t'}{\Gamma \vee t \neq t' \vee s = t'}$$

Superposition Calculus

operational meaning of rules:

- red terms must be “maximal” in respective equations and clauses
- equality resolution is resolution with “forgotten” reflexivity axiom
- superpositions are resolution with “forgotten” transitivity axiom
- equality factoring is resolution and factoring step with “forgotten” transitivity

consequence: equality axioms replaced by focussed inference rules

property: equality factoring not needed for Horn clauses

model construction: adaptation of resolution case, integrating critical pair criteria

Model Construction

idea:

- force canonical TRS in resolution model construction
- this effectively constructs a congruence with respect to input equations
- the model constructed is the resulting quotient algebra

building models: partial model is set of rewrite rules

- inspect equational clauses in increasing order
- if clause is false, maximal equation $s = t$ ($s > t$), and s in nf,
  then throw $s = t$ into model
- otherwise do nothing

Model Construction

ordering: make negative identities larger than positive ones

- associate $s = t$ with multiset $\{s, t\}$
- associate $s \neq t$ with multiset $\{s, s, t, t\}$

consequence: each stage yields convergent TRS for clauses

- termination holds since all equations are oriented and $>$ wf
- (local) confluence holds since only reduced lhs are forced into model

Model Construction

refutational completeness: (Horn clauses) if $R(S)$ doesn't contain the empty clause then construction yields model for S

proof: by wf induction

1. failing construction has minimal counterexample C
2. $C = \Gamma \vee s = s$ impossible since C must be false
3. $C = \Gamma \vee s = t$, hence s must be reducible by rule $l \to r$ generated by clause $\Delta \vee l = r$ and positive superposition yields smaller counterexample $\Gamma \vee \Delta \vee s(\dots r \dots) = t$
4. $C = \Gamma \vee s \neq s$, then equality resolution yields smaller counterexample Γ
5. $C = \Gamma \vee s \neq t$, then exists rewrite proof for $s = t$, hence s reducible by rule $l \to r$ generated by $\Delta \vee l = r$ and negative superposition yields smaller counterexample $\Gamma \vee \Delta \vee s(\dots r \dots) \neq t$

Example

let $f \succ a \succ b \succ c \succ d$

Horn clauses	partial models
$c = d$ $f(d) \neq d \vee a = b$ $f(c) = d$	$\{c \rightarrow d\}$
$c = d$ $f(d) \neq d \vee a = b$ $f(c) = d$ $f(d) = d$	$\{c \rightarrow d, f(d) \rightarrow d\}$
$c = d$ $f(d) \neq d \vee a = b$ $f(c) = d$ $f(d) = d$ $d \neq d \vee a = b$	$\{c \rightarrow d, f(d) \rightarrow d, a \rightarrow b\}$

Model Construction

non-Horn case: $C = \Gamma \vee s = t \vee s = t'$ false, $t > t'$ and $t = t'$ has rewrite proof, then equality factoring yields smaller counterexample $\Gamma \vee t \neq t' \vee s = t'$

non-ground case: (lifting)

- do construction at level of ground instances
- for skeleton overlaps use superposition etc
- for variable overlaps, maximal term can be instantiated with rhs of reducing rule to obtain smaller counterexample

Redundancy

forward redundancy: simplify new clauses immediately after generation
(by subsumption, rewriting, . . .)

backward redundancy: simplify existing clauses by rewrite rules
that have been generated at later stage

Redundancy

example: consider lpo with precedence $f \succ a \succ b$ and equations

$$f(a, x) = x$$
$$f(x, a) = f(x, b)$$

Redundancy

example:

$$f(a,x) = x$$
$$f(x,a) = f(x,b)$$
$$f(a,b) = a$$

is obtained by superposition

Redundancy

example:

$$
\begin{aligned}
f(a,x) &= x \\
f(x,a) &= f(x,b) \\
f(a,b) &= a \\
b &= a
\end{aligned}
$$

then follows by rewriting the third equation by the first one. . .

Redundancy

example:

$$f(a, x) = x$$
$$f(x, a) = f(x, b)$$

$$a = b$$

. . . and the third equation can be deleted (forward redundancy)

Redundancy

example:

$$f(a, x) = x$$
$$f(x, a) = f(x, b)$$
$$a = b$$
$$f(x, b) = f(x, b)$$

then follows by rewriting the second equation by the third one. . .

Redundancy

example:

$$f(a,x) = x$$

$$a = b$$

. . . and the second and fourth identity can be deleted

Redundancy

example:

$$f(a, x) = x$$

$$a = b$$

$$f(b, x) = x$$

finally, the first equation can be rewritten by the second one. . .

Redundancy

example:

$$f(a, x) = x$$
$$a = b$$
$$f(b, x) = x$$

finally, the first equation can be rewritten by the second one. . .

Redundancy

example:

$$a = b$$
$$f(b, x) = x$$

. . . and then deleted

Redundancy

```
assign(order,lpo).

function_order([b,a,f]).  % f>a>b

formulas(sos).

f(a,x)=x.
f(x,a)=f(x,b).

end_of_list.
```

Redundancy

```
given #1 (I,wt=5): 1 f(a,x) = x.  [assumption].

given #2 (I,wt=7): 2 f(x,a) = f(x,b).  [assumption].

given #3 (A,wt=3): 3 a = b.  [para(2(a,1),1(a,1)),rewrite([1(3)]),flip(a)].

given #4 (T,wt=5): 5 f(b,x) = x.  [back_rewrite(1),rewrite([3(1)])].

         ...

SEARCH FAILED
```

Redundancy

redundancy: same concepts as for ordered resolution

closure computation: only irredundant inferences

model construction: clause sets have models if they are closed
(up to redundant inferences) and don't contain the empty clause

proof: as previously, but contradictions arising from inferences being redundant
example: positive superposition

$$\frac{\Gamma \vee l = r \qquad \Delta \vee s(\ldots l \ldots) = t}{\Gamma \vee \Delta \vee s(\ldots r \ldots) = t}$$

right premise has not been forced into model;
it is redundant by this inference (entailed by smaller premise and conclusion)

Redundancy

example: demodulation

$$P(f(a))$$

$$f(a) = a$$

Redundancy

example: demodulation

$$P(f(a))$$
$$f(a) = a$$
$$P(a)$$

by rewriting “Leibniz principle”

Redundancy

example: demodulation

$$f(a) = a$$
$$P(a)$$

first literal has been deleted since it is now redundant

Example

precedence: $P \succ Q \succ f \succ a$

clause set: initial clauses

$$Q(a)$$

$$Q(a) \Rightarrow f(a) = a$$

$$\neg P(a)$$

$$P(f(a))$$

Example

precedence: $P \succ Q \succ f \succ a$

clause set: fifth clause by resolution from first and second one

$$Q(a)$$

$$Q(a) \Rightarrow f(a) = a$$

$$\neg P(a)$$

$$P(f(a))$$

$$f(a) = a$$

Example

precedence: $P \succ Q \succ f \succ a$

clause set: fourth clause rewritten by last one

$$Q(a)$$

$$Q(a) \Rightarrow f(a) = a$$

$$\neg P(a)$$

$$P(a)$$

$$f(a) = a$$

Example

precedence: $P \succ Q \succ f \succ a$

clause set: empty clause by resolution from third and fourth one

$$Q(a)$$

$$Q(a) \Rightarrow f(a) = a$$

$$\neg P(a)$$

$$P(a)$$

$$f(a) = a$$

$$\perp$$

Example

```
assign(order,lpo).

predicate_order([Q,P]). % P>Q
function_order([a,f]).  % f>a

formulas(sos).

Q(a).
Q(a)->f(a)=a.
-P(a).
P(f(a)).

end_of_list.
```

Example

```
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 8.
% Level of proof is 4.
% Maximum clause weight is 6.
% Given clauses 2.

1 Q(a) -> f(a) = a # label(non_clause).  [assumption].
2 Q(a).  [assumption].
3 -Q(a) | f(a) = a.  [clausify(1)].
4 -P(a).  [assumption].
5 P(f(a)).  [assumption].
6 f(a) = a.  [hyper(3,a,2,a)].
7 P(a).  [back_rewrite(5),rewrite([6(2)])].
8 $F.  [resolve(7,a,4,a)].
```

Verification Examples in Kleene Algebras

observation: ATP systems have rather been neglected in formal methods

idea: combine computational algebras with ATPs and counter example generators

results: experiments with various ATPs (Prover9, SPASS, Waldmeister,. . .)

- ~ 500 theorems automatically proved
- successful case studies in program refinement, termination analysis,. . .

benefits:

- special-purpose calculi made redundant
- generic flexible library of lemmas
- new style of verification

Semirings and Kleene Algebras

semiring: $(S, +, \cdot, 0, 1)$ (as in exercises)

$$x + (y + z) = (x + y) + z \qquad x + y = y + x \qquad x + 0 = x$$

$$x(yz) = (xy)z \qquad x1 = x \qquad 1x = x$$

$$x(y + z) = xy + xz \qquad (x + y)z = xz + yz$$

$$x0 = 0 \qquad 0x = 0$$

interpretation: S represents actions of some discrete dynamical system

- $+$ models nondeterministic choice
- $\cdot$ models sequential composition
- 0 models abortive action
- 1 models ineffective action

Semirings and Kleene Algebras

Kleene algebras: idempotent semiring with star satisfying

- unfold axiom $1 + xx^* \leq x^*$
- induction axiom $y + xz \leq z \Rightarrow x^*y \leq z$
- and their opposites $1 + x^*x \leq x^*$ and $y + zx \leq z \Rightarrow yx^* \leq z$

omega algebras: KAs with omega satisfying

- unfold axiom $xx^\omega = x^\omega$
- coinduction axiom $y \leq xy + z \Rightarrow y \leq x^\omega + x^*z$

Automating Bachmair and Dershowitz's Termination Theorem

theorem: [BachmairDershowitz86] *termination of the union of two rewrite systems can be separated into termination of the individual systems if one rewrite system quasicommutes over the other*

formalisation: omega algebra

encoding:

- quasicommutation $yx \leq x(x+y)^*$
- separation of termination $(x+y)^\omega = 0 \Leftrightarrow x^\omega + y^\omega = 0$

statement: termination of x and y can be separated if x quasicommutes over y

remark: posed as challenge by Ernie Cohen in 2001

Automating Bachmair and Dershowitz's Termination Theorem

results: ATP finds an extremely short proof in < 5min

$$
\begin{aligned}
(x+y)^\omega &= y^\omega + y^* x (x+y)^\omega && \text{(sum unfold)} \\
&\le y^\omega + x(x+y)^*(x+y)^\omega && \text{(strong quasicommmutation)} \\
&= y^\omega + x(x+y)^\omega && \text{(since } z^\omega = z^* z^\omega\text{)} \\
&\le x^\omega + x^* y^\omega && \text{(coinduction)} \\
&= 0 && \text{(assumption } x^\omega = y^\omega = 0\text{)}
\end{aligned}
$$

Automating Bachmair and Dershowitz's Termination Theorem

surprise: proof reveals new refinement law

$$yx \leq x(x+y)^* \Rightarrow (x+y)^\omega = x^\omega + x^* y^\omega$$

for separating infinite loops

remarks:

- reasoning essentially coinductive
- theorem holds in large class of models
- translation safe since relations form omega algebra

Automating Back's Atomicity Refinement Law

demonic refinement algebra: [von Wright04] Kleene algebra

- with axiom $x0 = 0$ dropped
- extended by strong iteration $^\infty$ encompassing finite and infinite iteration

remark: abstracted from refinement calculus [BackvonWright]

atomicity refinement law for action systems

- complex theorem first published by Back in 1989
- long proof in set theory analysing infinite sequences
- proof by hand in demonic refinement algebra still covers 2 pages
- automated analysis reveals some glitches and yields generalisation

first task: build up library of verified basic refinement laws for proof

Automating Back's Atomicity Refinement Law

theorem: if (i) $s \leq sq$ (ii) $a \leq qa$ (iii) $qb = 0$ (iv) $rb \leq br$
(v) $(a + r + b)l \leq l(a + r + b)$ (vi) $rq \leq qr$ (vii) $ql \leq lq$
(viii) $r^* = r^\infty$ (ix) $q \leq 1$

then

$$s(a + r + b + l)^\infty q \leq s(ab^\infty q + r + l)^\infty$$

two-step proof with "hypothesis learning"

1. assumptions imply $s(a + r + b + l)^\infty q \leq sl^\infty qr^\infty q(ab^\infty qr^\infty)^\infty$
   wait $60s$ for 75-step proof with Prover9
2. $q \leq 1$ implies $sl^\infty qr^\infty q(ab^\infty qr^\infty)^\infty \leq s(ab^\infty q + r + l)^\infty$
   wait $< 1s$ for 30-step proof

remark: full proof succeeds for $l = 0$ ($1013s$ for 46-step proof)

Automating Back's Atomicity Refinement Law

equational proof can be reconstructed

$$
\begin{aligned}
s(a+b+r+l)q &= sl^{\infty}(a+b+r)^{\infty}q \\
&= sl^{\infty}(b+r)^{\infty}(a(b+r)^{\infty})^{\infty}q \\
&= sl^{\infty}b^{\infty}r^{\infty}(ab^{\infty}r^{\infty})^{\infty}q \\
&\leq sl^{\infty}b^{\infty}r^{\infty}(qab^{\infty}r^{\infty})^{\infty}q \\
&= sl^{\infty}b^{\infty}r^{\infty}q(ab^{\infty}r^{\infty}q)^{\infty} \\
&\leq sql^{\infty}b^{\infty}r^{\infty}q(ab^{\infty}r^{\infty}q)^{\infty} \\
&\leq sl^{\infty}qb^{\infty}r^{\infty}q(ab^{\infty}r^{\infty}q)^{\infty} \\
&\leq sl^{\infty}qr^{\infty}q(ab^{\infty}r^{\infty}q)^{\infty} \\
&= sl^{\infty}qr^{\infty}q(ab^{\infty}r^{*}q)^{\infty} \\
&\leq sl^{\infty}qr^{\infty}q(ab^{\infty}qr^{*})^{\infty} \\
&= sl^{\infty}qr^{\infty}q(ab^{\infty}qr^{\infty})^{\infty}.
\end{aligned}
$$

Further Results

refinement calculus: library of automated refinement laws

modal extensions of KA:

- partial automated of Hoare logic (SMT solvers needed for full automation)
- automation of modal correspondence results
- automation of propositional dynamic logics and temporal logics

relation algebras: automation of basic calculus

A Personal Conclusion

suggestion: ATP systems + computational algebras motivate

verification challenge

- off-the-shelf ATP with domain-specific algebras
- promising alternative to conventional approaches (model checking, HOL)
- light-weight formal methods with heavy-weight automation

future work:

- hypothesis learning
- proof presentation
- integration of decision procedures
- integration of ordered reasoning

Literature

- A. Robinson and A. Voronkov: Handbook of Automated Reasoning
- F. Baader and T. Nipkow: Term Rewriting and All That
- "Terese" Term Rewriting Systems
- T. Hillenbrand: Waldmeister `www.waldmeister.org`
- W. McCune: Prover9 and Mace4 `www.cs.unm.edu/~mccune/mace4`
- G. Sutcliffe and C. Suttner: The TPTP Problem Library `www.cs.miami.edu/~tptp/`
- P.Höfner and G. Struth: Proof Library `www.dcs.shef.ac.uk/~georg/ka/`