Calculational combinatorics

Roland Backhouse and Arjan J. Mooij

School of Computer Science and Information Technology,
The University of Nottingham, United Kingdom.
{Roland.Backhouse@nottingham.ac.uk, Arjan.Mooij@cs.nott.ac.uk}

Abstract We present a new generalised distributivity rule for manipulating quantified expressions. We use the rule to calculate the solution to a well-known light-bulb problem, which is traditionally solved using induction. Our rule appears to have more general applicability to combinatorial problems.

Keywords: combinatorics, distributivity, problem solving, quantifier manipulation

1 Introduction

The following problem was communicated to us by Benjamin Kelly[1]:

> "There are 100 light bulbs and 100 people, both numbered from 1 to 100. Initially, all the light bulbs are off. Person number k toggles all the light bulbs that are divisible by k. For example, person 2 toggles bulbs $2, 4, 6, \ldots, 100$. After all 100 people have finished toggling the light bulbs, which light bulbs are on?"

The traditional approach to solving this problem is to experiment with the first so-many light bulbs until a pattern emerges, and then verify the pattern using induction. However, guess-and-verify is not very effective, as has also been noted in [GKP94]:

> "Induction has its place, (...) but it's still not really what we're seeking. (...) Flashes of inspiration should not be necessary. We should be able to do sums even on our less creative days."

The light-bulb problem is easily formalised —see Section 2— which formalisation suggests that it should be solvable by straightforward calculation along the lines of those in [BM06]. Disappointingly, we got stuck at a very early stage —see Section 3— ; the published rules for manipulating quantifiers [GKP94,GS00,Bac03] turn out to be inadequate for this problem. The difficulty was soon rectified by a generalisation of the distributivity rule given in [Bac03], which we develop in Section 4.

Our new distributivity rule enables the direct application of the calculational approach to some combinatorial problems, as we illustrate in Section 5. It also enables us to calculate the solution to the light-bulb problem — see Section 6. Section 7 draws some conclusions and suggests directions for further work.

[1] Kelly saw the problem on a colleague's website; his colleague was given the problem by Jan van de Snepscheut.

2 Modelling the problem

A straightforward modelling of whether light bulb n is finally on is

$$\mathsf{odd}.\langle \Sigma k : k\backslash n : 1 \rangle \quad .$$

In general, $\langle \Sigma x : s.x : 1 \rangle$ denotes the number of values x for which condition $s.x$ holds, and this particular instance denotes the number of times that light bulb n is toggled. As the light bulbs are initially off, the ones that are finally on are those that have been toggled an odd number of times.

Aside We use the uniform "Eindhoven" quantifier notation [Dij75,Dij76,Dij00] throughout. The notation extends the binary operator, $\oplus$ say, of an abelian monoid to an arbitrary finite bag of values, the bag being defined by a function (the term) acting on a set (the range). The general form of a quantified expression is

$$\langle \bigoplus v \in \mathsf{type} : \mathsf{range} : \mathsf{term} \rangle \quad .$$

where $\bigoplus$ is the quantifier, v is the dummy or bound variable and type is its type, range defines a subset of the type of the dummy over which the dummy ranges, and term defines a function on the range. The value of the quantification is the result of applying the operator $\oplus$ to all the values generated by evaluating the term at all instances of the dummy in the range. The type of the dummy is often cumbersome to repeat. For this reason, the type is omitted and a convention on the naming of dummies is adopted. In addition, the range is sometimes omitted if it is equivalent to a true range. Hints in our calculation refer to the formulation of the quantifier manipulation rules as presented in [Bac03]. **End of Aside**

The division ordering on the positive integers, denoted by $\backslash$, can be defined in several ways. One definition exploits existential quantification, but in our calculation this quickly leads to a dead end. An alternative is to exploit the Unique Prime Factorisation theorem, leading to the universal quantification in the following definition:

$$\langle \forall k,n :: \quad k\backslash n \quad \equiv \quad \langle \forall p :: \mathsf{exp}.k.p \leq \mathsf{exp}.n.p \rangle \, \rangle$$

where p ranges over the prime numbers. The so-called *exponent* function exp is such that $\mathsf{exp}.n$, for any positive integer n, maps prime numbers to natural numbers. It is defined as follows:

$$\langle \forall n,f :: \quad \mathsf{exp}.n = f \quad \equiv \quad n = \langle \Pi p :: p^{f.p} \rangle \, \rangle$$

Note that this definition states formally that exp is a bijection from the positive integers to the type of f, which is the set of all functions from prime numbers to the natural numbers that are eventually always zero. Its inverse is the function that maps f to $\langle \Pi p :: p^{f.p} \rangle$.

3 Towards a calculational solution

Based on the modelling from Section 2, we begin calculating the solution of the light-bulb problem:

$$
\begin{array}{ll}
 & \mathsf{odd}.(\Sigma k:\ k\backslash n:\ 1) \\
= & \{\ \text{distribution of odd over } \Sigma/\not\equiv\ \} \\
 & (\not\equiv k:\ k\backslash n:\ \mathsf{odd}.1) \\
= & \{\ \text{use odd.1; trading}\ \} \\
 & (\not\equiv k::\ k\backslash n) \\
= & \{\ \text{definition of the division ordering}\ \} \\
 & (\not\equiv k::\ (\forall p::\ \mathsf{exp}.k.p \le \mathsf{exp}.n.p)\ ) \\
= & \{\ \text{range translation } f := \mathsf{exp}.k\ \} \\
 & (\not\equiv f::\ (\forall p::\ f.p \le \mathsf{exp}.n.p)\ )\quad .
\end{array}
$$

The first two steps simplify the formula — at this stage, there is little choice of what to do. The distribution rule used in the first step may be unfamiliar. It exploits the fact that odd distributes through addition turning it into boolean inequality (which is more commonly known as exclusive-or). See [Bac03] for further details.

The introduction of boolean inequality motivates the choice of definition of the divides relation in the third step. The simpler choice would be to replace $k\backslash n$ by an existential quantification. However, the crucial consideration is that conjunction ($\wedge$) distributes over boolean inequality ($\not\equiv$), whereas disjunction does not.

The range translation in the last step is motivated by the fact that the only interest in the positive number k, is in $\mathsf{exp}.k.p$ for any p. Its use is valid because of the aforementioned fact that exp is a bijection from the type of k to the type of f.

This is the point at which our calculation gets stuck. The difficulty is that, although $\wedge$ distributes over $\not\equiv$, none of the published rules of quantifier manipulation caters for nested quantifications of this particular shape. This is the topic of the next section, following which we continue the calculation in Section 6.

4 Distributivity properties

The key to our difficulty is a further generalisation of the distributivity rule documented in [Bac03]. A binary operator $\otimes$ is said to *distribute* over the binary operator $\oplus$ if

$$0_\otimes = 1_\oplus \quad \text{and} \quad (\forall x,y,z::\quad x\otimes(y\oplus z)\ =\ (x\otimes y)\ \oplus\ (x\otimes z)\ )\quad ,$$

where $0_\otimes$ denotes the zero element of $\otimes$, and $1_\oplus$ denotes the unit element of $\oplus$. If this is the case, the operator $\otimes$ distributes over finite quantifications $\bigoplus$ as well[2]. That is,

$$(\forall x,t :: \quad x \otimes (\bigoplus i :: \ t.i) \quad = \quad (\bigoplus i :: \ x \otimes t.i) \ ) \quad .$$

Some well-known instances of $\otimes$ and $\oplus$ include $\times$ and $+$, $\wedge$ and $\vee$, and —significantly— $\wedge$ and $\not\equiv$.

Given that operator $\otimes$ distributes over operator $\oplus$, we have developed the following new rule for distributing a finite $\bigotimes$ quantification over a $\bigoplus$ quantification, for any t:

$$(1) \quad (\bigotimes i :: \ (\bigoplus j :: \ t.i.j) \ ) \quad = \quad (\bigoplus f :: \ (\bigotimes i :: \ t.i.(f.i)) \ )$$

The types of the dummy f is a function which maps a value of the type of variable i to a value of the type of variable j.

The rule is easily established by induction on the size of the range of dummy i. The base case exploits that the type of f reduces to a singleton, and the inductive case is justified by exploiting distributivity of $\otimes$ over $\bigoplus$ (twice). A detailed proof is provided in Appendix A.

Our new distribution rule can be described as formalising the axiom of choice on finite domains. This is the rule

$$(\forall i :: \ (\exists j :: \ t.i.j) \ ) \quad = \quad (\exists f :: \ (\forall i :: \ t.i.(f.i)) \ )$$

obtained by instantiating $\otimes$ to $\wedge$ and $\oplus$ to $\vee$. That conjunction distributes over disjunction in the way defined above is the combination of the properties

$$0_\wedge = 1_\vee = \mathsf{false} \quad \text{and} \quad (\forall x,y,z :: \quad x \wedge (y \vee z) \quad = \quad (x \wedge y) \ \vee \ (x \wedge z) \ ) \quad .$$

The function f, that is introduced in a left-to-right application of our distribution rule, names (or “Skolemizes” in the jargon of foundational mathematics) the dummy j that is chosen for a particular value of the dummy i. A possibly less well-known example of “Skolemization” is the rule

$$(\exists i :: \ (\forall j :: \ t.i.j) \ ) \quad = \quad (\forall f :: \ (\exists i :: \ t.i.(f.i)) \ ) \quad .$$

It is obtained by instantiating $\otimes$ to $\vee$ and $\oplus$ to $\wedge$. That disjunction distributes over conjunction is the combination of the properties

$$0_\vee = 1_\wedge = \mathsf{true} \quad \text{and} \quad (\forall x,y,z :: \quad x \vee (y \wedge z) \quad = \quad (x \vee y) \ \wedge \ (x \vee z) \ ) \quad .$$

Since the range of the quantification over variable i cannot depend on j or f, any proper range can be modelled using the type of variable i. This does not hold for the range

[2] Whenever we introduce a quantification, we assume, of course, that the binary operator in question is associative and symmetric

of variable j, which could depend on variable i. To make this explicit, we aim to generalise the rule further by introducing a condition $s.i.j$ for the range of the quantification over j. Let us calculate the corresponding effect on the right-hand side of the rule:

$$
\begin{array}{ll}
 & (\otimes i :: \ (\oplus j : \ s.i.j : \ t.i.j) \ ) \\
= & \{ \text{ trading, in order to eliminate the explicit range } \} \\
 & (\otimes i :: \ (\oplus j :: \ \mathsf{if} \ \ s.i.j \ \ \mathsf{then} \ \ t.i.j \ \ \mathsf{else} \ \ 1_{\oplus} \ \ \mathsf{fi}) \ ) \\
= & \{ \text{ (3), distribution of } \otimes \text{ over } \oplus \ \} \\
 & (\oplus f :: \ (\otimes i :: \ \mathsf{if} \ \ s.i.(f.i) \ \ \mathsf{then} \ \ t.i.(f.i) \ \ \mathsf{else} \ \ 1_{\oplus} \ \ \mathsf{fi}) \ ) \\
= & \{ \text{ use } 0_{\otimes} = 1_{\oplus}, \text{ since } \otimes \text{ distributes over } \oplus \ \} \\
 & (\oplus f :: \ \mathsf{if} \ \ (\forall i :: \ s.i.(f.i)) \ \ \mathsf{then} \ \ (\otimes i :: \ t.i.(f.i)) \ \ \mathsf{else} \ \ 1_{\oplus} \ \ \mathsf{fi} \ ) \\
= & \{ \text{ trading, in order to introduce an explicit range } \} \\
 & (\oplus f : \ (\forall i :: \ s.i.(f.i)) : \ (\otimes i :: \ t.i.(f.i)) \ ) \quad .
\end{array}
$$

Thus, under the same premises as before, we obtain the following calculational rule. For any s and t:

$$
(2) \quad (\bigotimes i :: \ (\bigoplus j : \ s.i.j : \ t.i.j) \ ) \quad = \quad (\bigoplus f : \ (\forall i :: \ s.i.(f.i)) : \ (\bigotimes i :: \ t.i.(f.i)) \ ) \quad .
$$

5 Intermezzo: relation to combinatorics

To provide a simple illustration of the effectiveness of our new distributivity rule, we consider the following combinatorial problem:

"How many ways are there to colour n objects using m colours?"

To model this problem, we represent each object i by a natural number between 1 and n, each colour j by a natural number between 1 and m, and each colouring c by a function from objects to colours. Thus the problem can be formulated as computing the value of

$$
(\Sigma c : \ (\forall i : \ 1 \leq i \leq n : \ 1 \leq c.i \leq m) : \ 1) \quad .
$$

In contrast to the traditional approach to this problem, we simply calculate the solution using our new rule:

$$
\begin{array}{ll}
 & (\Sigma c :\ (\forall i :\ 1 \le i \le n :\ 1 \le c.i \le m) :\ 1) \\
= & \quad \{ \text{ introduce a product } \} \\
 & (\Sigma c :\ (\forall i :\ 1 \le i \le n :\ 1 \le c.i \le m) :\ (\Pi i :\ 1 \le i \le n :\ 1)\ ) \\
= & \quad \{ \text{ (4), distribution of } \Pi \text{ over } \Sigma \ \} \\
 & (\Pi i :\ 1 \le i \le n :\ (\Sigma j :\ 1 \le j \le m :\ 1)\ ) \\
= & \quad \{ \text{ eliminate the summation } \} \\
 & (\Pi i :\ 1 \le i \le n :\ m\ ) \\
= & \quad \{ \text{ eliminate the product } \} \\
 & m^n \quad .
\end{array}
$$

The standard way of solving this problem —which we stress is a *counting* problem— is to immediately formulate the *product*, with only a verbal justification (if any) for the replacement. This, in our view, is another example of the proverbial rabbit-in-a-hat with which traditional mathematical practice abounds and which puts many students off.

The formulation of the quantifier rule has the advantage of capturing the use of induction in one general rule, which has applicability in a wide range of circumstances (including to operators other than addition and multiplication). Another simple variation on this combinatorial problem is to restrict the available number of colours for each object i to i. The solution can be computed along the same lines, using that $(\Pi i :\ 1 \le i \le n :\ i)$ denotes the factorial of n.

6 Completing a calculational solution

Armed with our new distributivity rule, we continue the calculation in Section 3 using the following maxim [GKP94]:

"Once you, the reader, have learned the material (...), all you will need is a cool head, a large sheet of paper, and fairly decent handwriting (...)."

So, let us calculate:

$$
\begin{array}{ll}
 & (\not\equiv f ::\ (\forall p ::\ f.p \le exp.n.p)\ ) \\
= & \quad \{ \text{ (3), distribution of } \forall \text{ over } \not\equiv \ \} \\
 & (\forall p ::\ (\not\equiv e ::\ e \le exp.n.p)\ ) \\
= & \quad \{ \text{ trading; use odd.1 } \} \\
 & (\forall p ::\ (\not\equiv e :\ e \le exp.n.p :\ odd.1)\ ) \\
= & \quad \{ \text{ distribution of odd over } \Sigma/\not\equiv \ \} \\
 & (\forall p ::\ odd.(\Sigma e :\ e \le exp.n.p :\ 1)\ ) \\
= & \quad \{ \text{ eliminate the summation } \} \\
 & (\forall p ::\ odd.(exp.n.p + 1)\ ) \\
= & \quad \{ \text{ relation between odd and even } \} \\
 & (\forall p ::\ even.(exp.n.p)\ )
\end{array}
$$

Observing both the introduction and the elimination of the $\not\equiv$ quantifier, one might wonder whether we could not just stick to the Σ quantifier in Section 3. This turns out to be possible, and it even slightly reduces the length of the calculation by using rule (4) which embeds the two trading steps. However, the choice of the particular definition of the division ordering becomes a bit harder to motivate, and before applying rule (4) a Π quantifier has to be introduced, as we did in the example in Section 5.

Experienced mathematicians with a good knowledge of prime factorisation might already be able to interpret the last formula in terms of the original problem. Otherwise, the only thing we can do is to apply the definition of even, and bravely continue our calculation:

$$
\begin{array}{ll}
& (\forall p :: \ \text{even}.(\text{exp}.n.p)\ ) \\
= & \{ \text{ definition of even } \} \\
& (\forall p :: \ (\exists m :: \ \text{exp}.n.p = m \times 2)\ ) \\
= & \{ \text{ (3), distribution of } \forall \text{ over } \exists \} \\
& (\exists f :: \ (\forall p :: \ \text{exp}.n.p = f.p \times 2)\ ) \\
= & \{ \text{ definition of exp, prime factorisation } \} \\
& (\exists f :: \ n = (\Pi p :: \ p^{(f.p \times 2)})\ ) \\
= & \{ \text{ use } x^{(y \times z)} = (x^y)^z\text{; distribution of square over } \Pi \} \\
& (\exists f :: \ n = (\Pi p :: \ p^{f.p})^2\ ) \\
= & \{ \text{ range translation: } k := (\Pi p :: \ p^{f.p}) \} \\
& (\exists k :: \ n = k^2\ ) \quad .
\end{array}
$$

(Observe the use of (3) again.) The final solution can easily be interpreted in terms of the original problem. The light bulbs that are finally on are the ones that are numbered by a square.

7 Conclusions and further work

This work was driven by the desire to solve a light-bulb problem just by calculation. Although this was our source of inspiration, the main research results are methodological. We have developed a general rule for the distribution of quantifiers over quantifiers. From a problem-solving perspective, this rule offers an important calculational tool that deals with patterns that could not be addressed before, viz., any quantification whose range is a universal quantification, and quantifications whose term is another quantification.

We have also demonstrated that the distributivity rule captures a fundamental aspect of combinatorial problems. Namely, that such problems are formulated in terms of functions and sums (i.e., for counting the number of possibilities), whilst their solutions are obtained by translating to products.

Although we have as yet considered only a small number of examples, we expect that many other problems will yield to a similar analysis. A particular direction for further work is to investigate how, in our distributivity rule, the range of the quantification over the function space can be generalised. In this way, we expect to be able to improve problem-solving skills in this particular area, which in turn should lead to new insights into many combinatorial problems.

Acknowledgement

We are very grateful to Diethard Michaelis for his suggestions for improvement, in particular with regard to the proof of (3) given in the appendix.

References

[Bac03] R.C. Backhouse. *Program Construction: Calculating Implementations from Specifications.* John Wiley and Sons, Inc., March 2003.

[BM06] R.C. Backhouse and D. Michaelis. Exercises in quantifier manipulation. In *Proceedings of the 8th international conference on Mathematics of Program Construction*, volume 4014 of *LNCS*, pages 69–81. Springer-Verlag, 2006.

[Dij75] E.W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, pages 453–457, 1975.

[Dij76] E.W. Dijkstra. *A Discipline of Programming.* Prentice Hall, 1976.

[Dij00] E.W. Dijkstra. The notational conventions I adopted, and why. EWD 1300, July 2000.

[GKP94] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, second edition, 1994.

[GS00] D. Gries and F.B. Schneider. *A Logical Approach to Discrete Math.* Springer-Verlag, September 2000.

A Proof of the distributivity rule

Although we have been very explicit regarding the main calculation, in Section 4 we have only sketched a proof of rule (3). The reason for doing so, is that a proper proof demands some theory about function spaces, which we only need for this proof.

The following are well-known properties of finite function spaces:

$$m \leftarrow \mathbb{0} \;\cong\; \mathbb{1}$$

$$(m \leftarrow n) \times m \;\cong\; m \leftarrow (n + \mathbb{1})$$

where $\mathbb{0}$ denotes the empty space, and $\mathbb{1}$ denotes the singleton space. The sole element in $\mathbb{1}$ is denoted by $*$.

Using these properties, we can provide a detailed proof of rule (3) by structural induction on the range of dummy i. For the base case, i.e., the empty space $\mathbb{0}$, we calculate from the right-hand side:

$$
\begin{array}{ll}
 & (\bigoplus f :\ f \in (m \leftarrow \mathbb{0}) :\ (\bigotimes i :\ i \in \mathbb{0} :\ t.i.(f.i))\ ) \\
= & \{\ m \leftarrow \mathbb{0} \cong \mathbb{1};\quad \text{range translation and one-point rule}\ \} \\
 & (\bigotimes i :\ i \in \mathbb{0} :\ t.i.*) \\
= & \{\ \text{empty range}\ \} \\
 & 1_{\otimes} \\
= & \{\ \text{empty range}\ \} \\
 & (\bigotimes i :\ i \in \mathbb{0} :\ (\bigoplus j :\ j \in m :\ t.i.j)\ ) \quad .
\end{array}
$$

For the inductive case, i.e. the non-empty space $n+\mathbb{1}$, we calculate from the left-hand side:

$$
\begin{array}{ll}
 & (\bigotimes i :\ i \in n+\mathbb{1} :\ (\bigoplus j :\ j \in m :\ t.i.j)\ ) \\
= & \{\ \text{split off } i \in \mathbb{1}\ \} \\
 & (\bigotimes i :\ i \in n :\ (\bigoplus j :\ j \in m :\ t.i.j)\ ) \ \otimes\ (\bigoplus j :\ j \in m :\ t.*.j) \\
= & \{\ \text{induction hypothesis for the case } n\ \} \\
 & (\bigoplus f :\ f \in m \leftarrow n :\ (\bigotimes i :\ i \in n :\ t.i.(f.i))\ ) \ \otimes\ (\bigoplus j :\ j \in m :\ t.*.j) \\
= & \{\ \text{distribution of } \otimes \text{ over } \bigoplus\ \} \\
 & (\bigoplus f :\ f \in m \leftarrow n :\ (\bigotimes i :\ i \in n :\ t.i.(f.i)) \otimes (\bigoplus j :\ j \in m :\ t.*.j)\ ) \\
= & \{\ \text{distribution of } \otimes \text{ over } \bigoplus\ \} \\
 & (\bigoplus f :\ f \in m \leftarrow n :\ (\bigoplus j :\ j \in m :\ (\bigotimes i :\ i \in n :\ t.i.(f.i)) \otimes t.*.j)\ ) \\
= & \{\ \text{nesting}\ \} \\
 & (\bigoplus f,j :\ f \in m \leftarrow n \ \wedge\ j \in m :\ (\bigotimes i :\ i \in n :\ t.i.(f.i)) \otimes t.*.j\ ) \\
= & \{\ (m \leftarrow n) \times m \cong m \leftarrow (n+\mathbb{1});\quad \text{range translation}\ \} \\
 & (\bigoplus f :\ f \in m \leftarrow (n+\mathbb{1}) :\ (\bigotimes i :\ i \in n :\ t.i.(f.i)) \otimes t.*.(f.*)\ ) \\
= & \{\ \text{split off } i \in \mathbb{1}\ \} \\
 & (\bigoplus f :\ f \in m \leftarrow (n+\mathbb{1}) :\ (\bigotimes i :\ i \in n+\mathbb{1} :\ t.i.(f.i))\ ) \quad .
\end{array}
$$

This concludes our proof of rule (3).