Term Rewriting Systems

Terese

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011–4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2003

Printed in the United Kingdom at the University Press, Cambridge

Typeset by the author in Computer Modern using $\LaTeX\,2_\varepsilon$

A catalogue record for this book is available from the British Library

Library of Congress Cataloguing in Publication data

ISBN 0 521 39115 6 hardback

Contents

0

Introduction

We start the book by presenting some basic examples of rewriting, in order to set the stage.

At school, many of us have been drilled to simplify arithmetical expressions, for instance:

$$(3+5)\cdot(1+2) \to 8\cdot(1+2) \to 8\cdot 3 \to 24$$

This simplification process has several remarkable properties.

First, one can perceive a direction, at least in the drill exercises, from complicated expressions to simpler ones. For this reason we use $\to$ rather than $=$, even though the expressions are equal in the sense that they all denote the same number (24). The relation $\to$ is called a *reduction relation*.

Second, in most drill exercises the simplification process yields a result in the form of an expression that cannot be simplified any further, which we call a *normal form*. In the above example the result 24 is such a normal form.

Third, the simplification process is non-deterministic, often different simplifications are possible. It is clearly desirable that all simplifications lead to the same result. Indeed the above outcome 24 can be obtained in different ways, e.g. also by

$$(3+5)\cdot(1+2) \to (3+5)\cdot 3 \to 3\cdot 3+5\cdot 3 \to 9+5\cdot 3 \to 9+15 \to 24$$

The property that simplifications can have at most one final result is called *uniqueness of normal form*.

The process of simplifying arithmetical expressions built up from numbers with operations like $+$ and $\cdot$ can be analysed (and taught) as performing *elementary steps* in *contexts*. The elementary steps are based on the tables of addition and multiplication. The contexts are arithmetical expressions with a hole, denoted by $\Box$, indicating the place where the elementary step is to take place. In the first example above, the elementary step $3+5 \to 8$ is performed in the context $\Box\cdot(1+2)$. This means that $\Box$ is $3+5$ before, and 8 after, the elementary step, yielding the simplification $(3+5)\cdot(1+2) \to 8\cdot(1+2)$. (Brackets are not part of the term, but are auxiliary symbols preventing false readings of the term.) The next elementary step is $1+2 \to 3$, performed in the context $8\cdot\Box$, yieding the simplification $8\cdot(1+2) \to 8\cdot 3$. Finally, the elementary step $8\cdot 3 \to 24$ is performed in the context $\Box$, as this elementary step involves the whole term.

Among our early experiences are, besides arithmetical simplification, also drill exercises that do not yield a final result, such as counting:

$$1 \to 2 \to 3 \to \cdots$$

Another example is the cyclic elapsing of the hours on a clock:

$$1 \to 2 \to \cdots \to 11 \to 12 \to 1 \to \cdots$$

The absence of a normal form is called *non-termination*. In many applications termination is a desirable property. The following example shows that termination can be a non-trivial question.

Define a reduction relation on positive integer numbers

$$n \to n'$$

by putting $n' = n/2$ if $n > 1$ is even, and $n' = 3n + 1$ if $n > 1$ is odd. Thus 1 is the only normal form. We have reductions such as

$$\begin{aligned} &7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \\ &\to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1 \end{aligned}$$

Note that we have both decrease (in the case $n/2$) and increase (in the case $3n + 1$). It is an open question[1] whether or not every positive integer can be reduced to 1 in this way.

This book is about the theory of stepwise, or discrete, transformations of objects, as opposed to continuous transformations of objects. Many computations, constructions, processes, translations, mappings and so on, can be modelled as stepwise transformations of objects. Clearly, this yields a large spectrum of applications, depending on what are the objects of interest and what transformations, or *rewriting*, one wishes to do. One has string rewriting, term rewriting, graph rewriting, to name some of the principal subjects that will be treated. In turn, these main subjects lead to several specialized theories, such as conditional term rewriting, infinitary term rewriting, term graph rewriting, and many more. In all these different branches of rewriting the basic concepts are the same, and known as *termination* (guaranteeing the existence of normal forms) and *confluence* (securing the uniqueness of normal forms), and many variations of them.

In order to appreciate the variety of applications and to further introduce the main concepts, let us view a few examples from different fields. We will return to some of these examples in future chapters.

[1]Collatz's problem, also known as the Syracuse problem.

0.1. An example from functional programming

Rewriting techniques can be used to specify operations on abstract data types. The first introductory example, taken from Arts [1997], describes in a concise way an algorithm for dividing natural numbers, giving for all pairs (m, n) with $n \geq 1$ and $m = n{\cdot}q \geq 0$ the correct result q. The abstract data type in question is that of the natural numbers built up with $\mathbf{0} : \mathbb{N}$ and $\mathbf{S} : \mathbb{N}{\rightarrow}\mathbb{N}$. We write 0 for $\mathbf{0}$, 1 for $\mathbf{S}(\mathbf{0})$, 2 for $\mathbf{S}(\mathbf{S}(\mathbf{0}))$, and so on. There are four rewrite rules:

$$\begin{array}{rcl} \mathit{minus}(x, \mathbf{0}) & \to & x \\ \mathit{minus}(\mathbf{S}(x), \mathbf{S}(y)) & \to & \mathit{minus}(x, y) \\ \mathit{quot}(\mathbf{0}, \mathbf{S}(y)) & \to & 0 \\ \mathit{quot}(\mathbf{S}(x), \mathbf{S}(y)) & \to & \mathbf{S}(\mathit{quot}(\mathit{minus}(x, y), \mathbf{S}(y))) \end{array}$$

As an example we exhibit the following reduction sequence.

$$\begin{array}{rcl} \mathit{quot}(4, 2) & \to & \mathbf{S}(\mathit{quot}(\mathit{minus}(3, 1), 2)) \\ & \to & \mathbf{S}(\mathit{quot}(\mathit{minus}(2, 0), 2)) \\ & \to & \mathbf{S}(\mathit{quot}(2, 2)) \\ & \to & \mathbf{S}(\mathbf{S}(\mathit{quot}(\mathit{minus}(1, 1), 2))) \\ & \to & \mathbf{S}(\mathbf{S}(\mathit{quot}(\mathit{minus}(0, 0), 2))) \\ & \to & \mathbf{S}(\mathbf{S}(\mathit{quot}(0, 2))) \\ & \to & \mathbf{S}(\mathbf{S}(\mathbf{0})) = 2 \end{array}$$

Normal forms are $0, 1, 2, \ldots$, but also $\mathit{minus}(0, 1)$ and $\mathit{quot}(2, 0)$. For the correctness of such algorithms with respect to the operations on the abstract data type, it is obviously desirable that the algorithms have unique results. This is guaranteed if every reduction sequence eventually terminates, and moreover the resulting normal form is independent of the particular choice of reduction sequence. Later on in this book we will develop methods to facilitate the proofs of such properties.

0.2. An example from topology: Reidemeister moves

In this example the objects are knots. For our purposes it suffices to state that a knot is a piece of (flexible) wire whose ends coincide. We assume knots to be two-dimensional, lying on a flat surface such as a kitchen table, with crossings only in one single point. In case of a crossing, it is always clear which of the two parts of the wire involved is the upper and which is the lower (in the pictures, the lower wire is interrupted just before and after the crossing).

Knots are considered to be equivalent when they can be transformed into one another in a continuous way, that is, without breaking the wire. As a

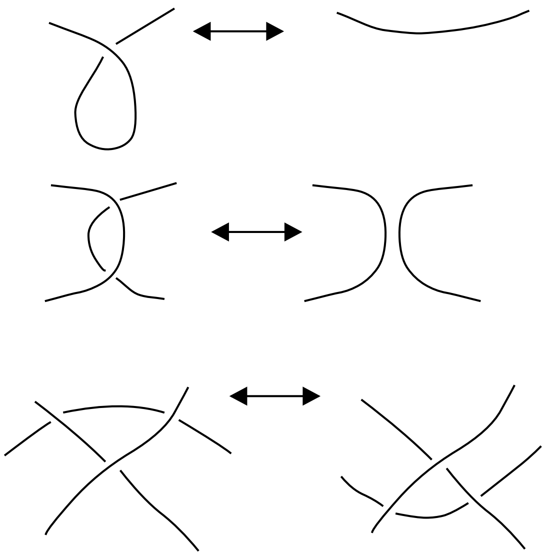

Figure 1: Reidemeister moves

consequence, knots could be taken as equivalence classes, but here we are interested in the equivalence relation itself. There are some well-known elementary transformations, known as the 'Reidemeister moves', such that two knots are equivalent if and only if they can be transformed into one another using only Reidemeister moves.

Figure 1 depicts the Reidemeister moves, Figure 2 gives a transformation between a certain knot and the trivial knot, also called the 'un-knot'. Although the un-knot is the simplest knot, it should not be considered as a normal form: the Reidemeister moves can be used in both directions. There is no use for the concept of normal form in this example. Not every knot can be transformed into the un-knot; there are in fact infinitely many different knots.

0.3. An example from logic: tautology checking

This example exploits the well-known equivalence between boolean algebras and boolean rings (rings with $x^2 = x$). In the setting of boolean algebras $\Rightarrow, \vee, \neg$ are the usual propositional connectives; $+$ is exclusive disjunction and $\cdot$ is conjunction. In the setting of rings $+$ is the ring addition, $\cdot$ the ring multiplication. Furthermore, 0 stands for the boolean false as well as for the additive unit, and 1 stands for the boolean true as well as for the multiplicative unit. The operators $+$ and $\cdot$ are supposed to be commutative and associative.

Now we have the remarkable fact that every propositional formula is a tautology if and only if it can be rewritten to 1 using the rewrite rules be-

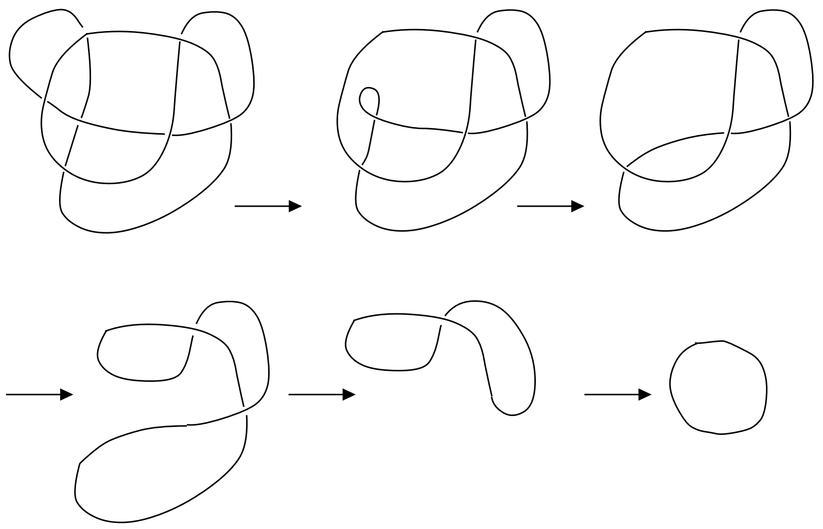

Figure 2: Un-knotting

low, *modulo associativity and commutativity.* This means that rewrite steps may be preceded by rearranging the order and the bracket structure of sums and products. (In fact this constitutes an example of rewriting modulo an equivalence relation, made precise in Section 14.3.)

The above equivalence and the rewrite rules already appear in Herbrand's PhD thesis of 1929. The symbol $\rightarrow$ is used for a rewrite step, and should not be confused with $\Rightarrow$ which stands for implication.

$$\begin{array}{lcl} x \Rightarrow y & \rightarrow & x \cdot y + x + 1 \\ x \vee y & \rightarrow & x \cdot y + x + y \\ \neg x & \rightarrow & x + 1 \\ x + 0 & \rightarrow & x \\ x + x & \rightarrow & 0 \\ x \cdot 0 & \rightarrow & 0 \\ x \cdot 1 & \rightarrow & x \\ x \cdot x & \rightarrow & x \\ x \cdot (y + z) & \rightarrow & x \cdot y + x \cdot z \end{array}$$

As an example we exhibit the following reduction of the tautology $p \Rightarrow (q \Rightarrow p)$ to 1, where the most relevant associative/commutative steps are also stipulated. As usual in algebra, we omit the multiplication sign $\cdot$.

$$\begin{array}{rcl} p \Rightarrow (q \Rightarrow p) & \rightarrow & p(q \Rightarrow p) + p + 1 \\ & \rightarrow & p(qp + q + 1) + p + 1 = p((qp + q) + 1) + p + 1 \\ & \rightarrow & p(qp + q) + p1 + p + 1 \\ & \rightarrow & pqp + pq + p1 + p + 1 \end{array}$$

$$\begin{aligned}
&\to \quad pqp + pq + p + p + 1 = pqp + pq + (p + p) + 1 \\
&\to \quad pqp + pq + 0 + 1 = pqp + (pq + 0) + 1 \\
&\to \quad pqp + pq + 1 = (pp)q + pq + 1 \\
&\to \quad pq + pq + 1 = (pq + pq) + 1 \\
&\to \quad 0 + 1 = 1 + 0 \\
&\to \quad 1
\end{aligned}$$

It is not our goal to give a correctness proof of this algorithm for tautology checking. Note, once more, that rewriting is non-deterministic: we could have started in the example above with the rightmost implication in $p \Rightarrow (q \Rightarrow p)$. It is necessary for the correctness of the algorithm that any reduction sequence starting with $p \Rightarrow (q \Rightarrow p)$ yields the outcome 1. This will be guaranteed by a notion called *confluence*, formulated for arbitrary abstract reduction systems in the chapter that follows now.