Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.W. Klop

Term rewriting systems: a tutorial

Computer Science/Department of Software Technology Note CS-N8701 May

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

69D11, 69F41

Term Rewriting Systems: a tutorial

J.W. Klop
Centre for Mathematics and Computer Science,
Kruislaan 413, 1098 SJ Amsterdam;
Department of Mathematics and Computer Science,
Free University, de Boelelaan 1081, 1081 HV Amsterdam.

ABSTRACT. Term Rewriting Systems play an important role in various areas, such as abstract data type specifications, implementations of functional languages and automated deduction. In this tutorial we introduce some of the basic concepts and facts for TRSs. No attempt is made to present a comprehensive survey: e.g. the tutorial does not contain material about conditional TRSs or equational TRSs. The spirit of the material presented here is syntactic rather than semantic. An emphasis is put on Abstract Reduction Systems, of which not only TRSs are instances, but also Semi-Thue Systems, tree replacement systems, graph rewrite systems. As an example of an important termination proof technique we present the recursive path orderings in a new presentation.

1980 Mathematics Subject Classification (1985): 03B40, 68Q99.
1982 CR Categories: D.1.1, F.4.1.
Key words and phrases: Abstract Reduction Systems, Term Rewriting Systems, Combinatory Logic, reduction strategies, regular Term Rewriting Systems.

Note: Research partially supported by ESPRIT project 432: Meteor.

This report will be published in the Bulletin of the European Association for Theoretical Computer Science, Nr.32, 1987.

Note CS-N8701
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Introduction

Term Rewriting Systems form an important computational paradigm and applications can be found in several fields. Here one can mention many key words and phrases; we will restrict ourselves and mention only: proving properties of abstract data type specifications, implementing functional languages, computing by means of equations, mechanizing deduction systems.

The origin of the study of TRSs dates from half a century ago, when Combinatory Logic and Lambda Calculus were developed and deeply studied, in order to formalize the concept of computable functions. The first system will also appear below. In more recent days both systems made their entrance in Computer Science: Lambda Calculus gave the inspiration for LISP and caused a major advance in denotational semantics of programming languages, while Combinatory Logic turned out to have its use for implementations of functional languages. This tutorial does not dwell on these applications. Rather, we aim to present some of the basic concepts and facts about TRSs which are obvious prerequisites for understanding applications as mentioned.

The topics treated below comprise only an initial part of the theory of Term Rewriting Systems. In the first section we have collected many properties valid already for Abstract Reduction Systems; this makes these properties also applicable to other rewrite systems, such as string rewrite systems, tree replacement systems etc. The second section discusses some basic issues in general TRSs; in the third section a powerful termination proof technique (recursive path orderings) is explained in a new presentation. Next a short exposition of the Knuth-Bendix completion technique is given. The last part is about regular (i.e. left-linear and non-ambiguous) TRSs and reduction strategies for such TRSs, such as leftmost-outermost reduction, parallel-outermost reduction etc.

Contents

Introduction
1. Abstract Reduction Systems
2. Term Rewriting Systems: basic notions
3. A termination proof technique
4. Knuth-Bendix completion
5. Regular Term Rewriting Systems
6. Reduction strategies for regular Term Rewriting Systems
References

1. Abstract Reduction Systems

Many of the basic definitions for and properties of TRSs (Term Rewriting Systems) can be stated more abstractly, viz. for sets equipped with one or more binary relations. As it is instructive to see which definitions and properties depend on the term structure and which are more basic, we start with a (relatively extensive) section about Abstract Reduction Systems. Moreover, the concepts and properties of Abstract Reduction Systems also apply to other rewrite systems than TRSs, such as string rewrite systems (Thue systems), tree rewrite systems, graph grammars. In fact the same abstract approach was taken in an earlier tutorial in this Bulletin (JANTZEN [86]), where several of

the concepts below are defined for 'transformation systems'. (There is a certain amount of duplication between the present tutorial and the one just mentioned, as regards some of the basic definitions.) First we present a sequence of simple definitions.

1.1. DEFINITION. (i) An *Abstract Reduction System* (ARS) is a structure $\mathcal{A} = \langle A, (\rightarrow_\alpha)_{\alpha\in I}\rangle$ consisting of a set A and a sequence of binary relations $\rightarrow_\alpha$ on A, also called reduction or rewrite relations. Sometimes we will refer to $\rightarrow_\alpha$ as α. (An ARS with just one reduction relation is called 'replacement system' in STAPLES [75], and a 'transformation system' in JANTZEN [86].) If for a,b $\in$ A we have $(a,b) \in \rightarrow_\alpha$, we write $a \rightarrow_\alpha b$ and call b a one-step (α-)reduct of a.

(ii) The transitive reflexive closure of $\rightarrow_\alpha$ is written as $\twoheadrightarrow_\alpha$. (More customary is the notation $\rightarrow_\alpha{}^*$, but we prefer the double arrow notation as we find it more convenient in diagrams.)
So $a \twoheadrightarrow_\alpha b$ if there is a possibly empty, finite sequence of 'reduction steps' $a \equiv a_0 \rightarrow_\alpha a_1 \rightarrow_\alpha \ldots \rightarrow_\alpha a_n \equiv b$. Here $\equiv$ denotes identity of elements of A. The element b is called an (α-)reduct of a. The equivalence relation generated by $\rightarrow_\alpha$ is $=_\alpha$, also called the *convertibility* relation generated by $\rightarrow_\alpha$. The reflexive closure of $\rightarrow_\alpha$ is $\rightarrow_\alpha{}^{\equiv}$. The converse relation of $\rightarrow_\alpha$ is $\leftarrow_\alpha$ or $\rightarrow_\alpha{}^{-1}$. The union $\rightarrow_\alpha \cup \rightarrow_\beta$ is denoted by $\rightarrow_{\alpha\beta}$. The composition $\rightarrow_\alpha \circ \rightarrow_\beta$ is defined by: $a \rightarrow_\alpha \circ \rightarrow_\beta b$ if $a \rightarrow_\alpha c \rightarrow_\beta b$ for some $c \in A$.

(iii) If α,β are reduction relations on A, we say that α *commutes weakly* with β if the following diagram (see Figure 1a) holds, i.e. if $\forall a,b,c\in A\ \exists d\in A\ (c \leftarrow_\beta a \rightarrow_\alpha b \Rightarrow c \rightarrow\!\!\!\!\rightarrow_\alpha d \leftarrow\!\!\!\!\leftarrow_\beta b)$, or in a shorter notation: $\leftarrow_\beta \circ \rightarrow_\alpha \subseteq \rightarrow\!\!\!\!\rightarrow_\alpha \circ \leftarrow\!\!\!\!\leftarrow_\beta$. (This terminology differs from that of BACHMAIR & DERSHOWITZ [86], where α commutes with β if $\alpha^{-1}\circ\beta \subseteq \beta^{-1}\circ\alpha$.)

Further, α *commutes* with β if $\twoheadrightarrow_\alpha$ and $\twoheadrightarrow_\beta$ commute weakly.

(iv) The reduction relation $\rightarrow$ is called *weakly confluent* or *weakly Church-Rosser* (WCR) if it is self-commuting (see Figure 1b), i.e. if $\forall a,b,c\in A\ \exists d\in A\ (c \leftarrow a \rightarrow b \Rightarrow c \rightarrow\!\!\!\!\rightarrow d \leftarrow\!\!\!\!\leftarrow b)$.
(The property WCR is also often called 'local confluence', e.g. in JANTZEN [86].)

(v) $\rightarrow$ is *subcommutative* (notation $WCR^{\leq 1}$) if the diagram in Figure 1c holds, i.e. if $\forall a,b,c\in A\ \exists d\in A\ (c \leftarrow a \rightarrow b \Rightarrow c \rightarrow^{\equiv} d \leftarrow^{\equiv} b)$.

(vi) $\rightarrow$ is *confluent* or is Church-Rosser, has the Church-Rosser property (CR) if it is self-commuting (see Figure 1d), i.e. $\forall a,b,c\in A\ \exists d\in A\ (c \leftarrow\!\!\!\!\leftarrow a \rightarrow\!\!\!\!\rightarrow b \Rightarrow c \rightarrow\!\!\!\!\rightarrow d \leftarrow\!\!\!\!\leftarrow b)$.

In the sequel we will use the terms 'confluent' and 'Church-Rosser' or 'CR' without preference. Likewise for weakly confluent and WCR, etc. The following proposition follows immediately from the definitions. Note especially the equivalence of (i) and (vi); sometimes (vi) is called 'Church-Rosser' and the situation as in Definition 1.1.(vi) 'confluent'.

1.2. PROPOSITION. *The following are equivalent:*

(i) $\rightarrow$ *is confluent*

(ii) $\twoheadrightarrow$ *is weakly confluent*

(iii) $\twoheadrightarrow$ *is self-commuting*

(iv) $\twoheadrightarrow$ *is subcommutative*

(v) *the diagram in Figure 1e holds, i.e.*

$$\forall a,b,c \in A\ \exists d \in A\ (c \leftarrow a \twoheadrightarrow b \Rightarrow c \twoheadrightarrow d \twoheadleftarrow b)$$

(vi) $\forall a,b \in A\ \exists c \in A\ (a = b \Rightarrow a \twoheadrightarrow c \twoheadleftarrow b)$

(*Here '=' is the convertibility relation generated by* $\rightarrow$. *See diagram in Figure 1f.*) □

1.3. DEFINITION. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS.

(i) We say that $a \in A$ is a *normal form* if there is no $b \in A$ such that $a \rightarrow b$. Further, $b \in A$ *has a normal form* if $b \twoheadrightarrow a$ for some normal form $a \in A$.

(ii) The reduction relation $\rightarrow$ is *weakly normalizing* (WN) if every $a \in A$ has a normal form. In this case we also say that $\mathcal{A}$ is WN.

(iii) $\mathcal{A}$ (or $\rightarrow$) is *strongly normalizing* (SN) if every reduction sequence $a_0 \rightarrow a_1 \rightarrow \ldots$ eventually must terminate. (Other terminology: $\rightarrow$ is terminating, or noetherian.) If the converse reduction relation $\leftarrow$ is SN, we say that $\mathcal{A}$ (or $\rightarrow$) is SN^{-1}.

(iv) $\mathcal{A}$ (or $\rightarrow$) has the *unique normal form property* (UN) if
$\forall a,b,c \in A\ (a \twoheadrightarrow b \ \&\ a \twoheadrightarrow c \ \&\ b,c \text{ are normal forms} \Rightarrow b \equiv c)$.

(v) $\mathcal{A}$ (or $\rightarrow$) has the *normal form property* (NF) if
$\forall a,b \in A\ (a \text{ is normal form} \ \&\ a = b \Rightarrow b \twoheadrightarrow a)$.

(vi) $\mathcal{A}$ (or $\rightarrow$) is *inductive* (Ind) if for every reduction sequence (possibly infinite) $a_0 \rightarrow a_1 \rightarrow \ldots$ there is an $a \in A$ such that $a_n \twoheadrightarrow a$ for all n.

(vii) $\mathcal{A}$ (or $\rightarrow$) is *increasing* (Inc) if there is a map $|\ |: A \rightarrow \mathbb{N}$ such that
$\forall a,b \in A\ (a \rightarrow b \Rightarrow |a| < |b|)$. Here $\mathbb{N}$ is the set of natural numbers with the usual ordering $<$.

(viii) $\mathcal{A}$ (or $\rightarrow$) is *finitely branching* (FB) if for all $a \in A$ the set of one step reducts of a, $\{b \in A \mid a \rightarrow b\}$, is finite. If the converse reduction relation $\leftarrow$ is FB, we say that $\mathcal{A}$ (or $\rightarrow$) is FB^{-1}. (In HUET [78], FB is called 'locally finite'.)

An ARS which is confluent and terminating (CR & SN) is also called *complete* (other terminology: 'canonical' or 'uniquely terminating'). If the ARS has unique normal forms and is weakly normalizing (UN & WN, or what is as we shall see equivalent, CR & WN) it will be called *semi-complete*. The former is standard terminology (especially in the field of 'Knuth-Bendix completions', see below); the latter is not.

Before exhibiting several facts about all these notions, let us first introduce some more concepts.

1.4. DEFINITION. Let $\mathcal{A} = \langle A, \rightarrow_\alpha \rangle$ and $\mathcal{B} = \langle B, \rightarrow_\beta \rangle$ be two ARSs. Then $\mathcal{A}$ is a *sub-ARS* of $\mathcal{B}$, notation $\mathcal{A} \subseteq \mathcal{B}$, if:

(i) $A \subseteq B$

(ii) α is the restriction of β to A, i.e. $\forall a,a' \in A\ (a \to_\beta a' \Leftrightarrow a \to_\alpha a')$

(iii) A is closed under β, i.e. $\forall a \in A\ (a \to_\beta b \Rightarrow b \in A)$.

The ARS $\mathcal{B}$ is also called an *extension* of $\mathcal{A}$.

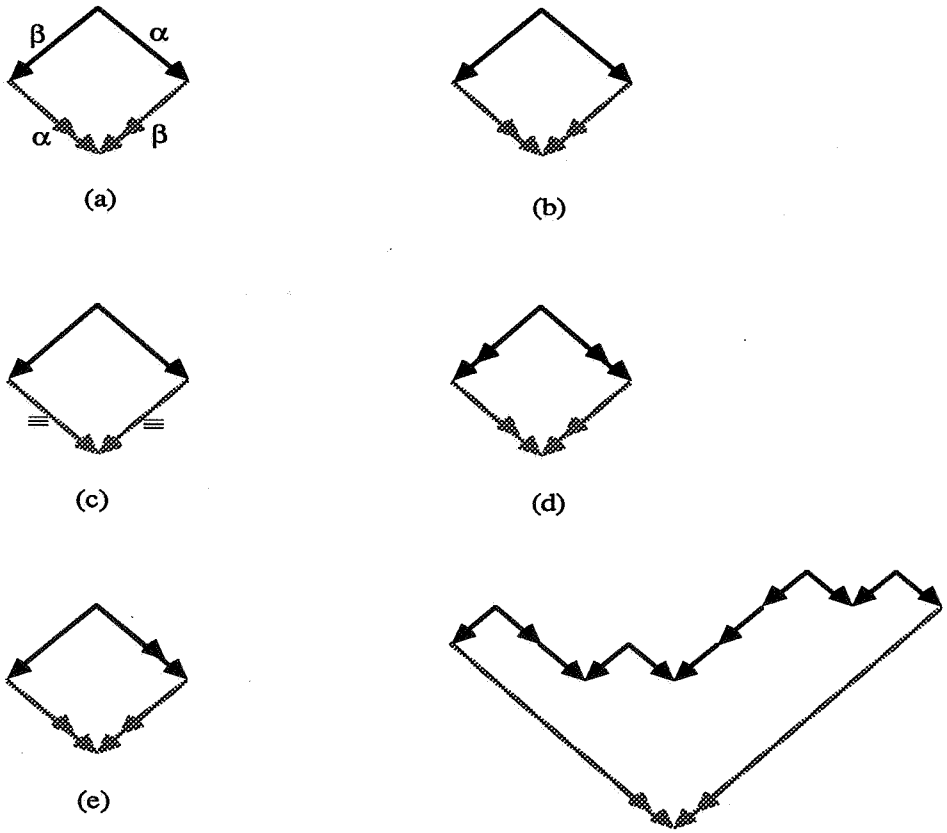


Figure 1

Note that all properties introduced so far (CR, WCR, WCR$^{\leq 1}$, WN, SN, UN, NF, Ind, Inc, FB) are preserved downwards: e.g. if $\mathcal{A} \subseteq \mathcal{B}$ and $\mathcal{B}$ is CR, then also $\mathcal{A}$ is so.

Of particular interest is the sub-ARS determined by an element a in an ARS:

1.5. DEFINITION. Let $\mathcal{A} = \langle A, \to \rangle$ be an ARS, and $a \in A$. Then $\mathcal{G}(a)$, the *reduction graph* of a, is the smallest sub-ARS of $\mathcal{A}$ containing a. So $\mathcal{G}(a)$ has as elements all reducts of a (including a itself) and is structured by the relation $\to$ restricted to this set of reducts.

We will now collect in one theorem several implications between the various properties of ARSs. The first part (i) is actually the main motivation for the concept of confluence: it guarantees unique normal forms, which is of course a desirable state of affairs in (implementations of) algebraic data type specifications such as we will consider later. Apart from the fundamental implication CR $\Rightarrow$ UN, the most important fact is (ii), also known as Newman's Lemma.

1.6. THEOREM.

(i) CR $\Rightarrow$ NF $\Rightarrow$ UN

(ii) SN & WCR $\Rightarrow$ CR (Newman's Lemma)

(iii) UN & WN $\Rightarrow$ CR

(iv) UN & WN $\Rightarrow$ Ind

(v) Ind & Inc $\Rightarrow$ SN

(vi) WCR & WN & Inc $\Rightarrow$ SN

(vii) CR $\Leftrightarrow$ CP for countable ARSs. □

Most of the proofs of (i)-(vii) are easy. For Newman's Lemma a short proof is given in HUET [78]. Proposition (v) is from NEDERPELT [73]; (vi) is proved in KLOP [80]; for (vii) see Ex.1.7 (13). The propositions in the statement of the theorem (and some more - for these see Ex. 1.7 (1-17)) are displayed also in Figure 2; here it is important whether an implication arrow points to the conjunction sign &, or to one of the conjuncts. Likewise for the tail of an implication arrow. (E.g. UN & WN $\Rightarrow$ Ind, SN & WCR $\Rightarrow$ UN & WN, Inc $\Rightarrow$ SN^{-1}, FB^{-1} & SN^{-1} $\Rightarrow$ Inc, CR $\Rightarrow$ UN but not CR $\Rightarrow$ UN & WN.)

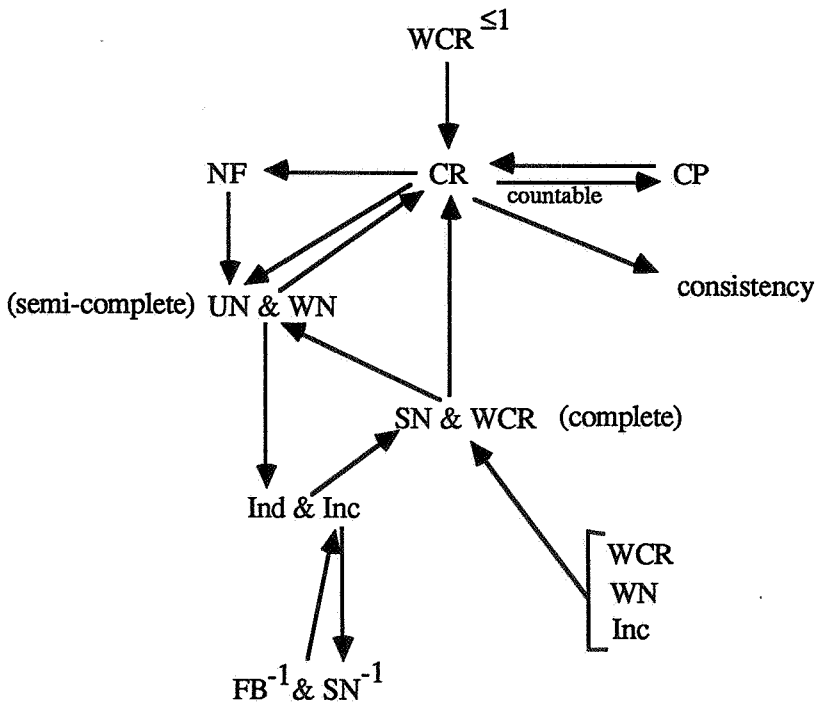


Figure 2

It does not seem possible to reverse any of the arrows in this diagram of implications. An instructive counterexample to WCR $\Rightarrow$ CR is the TRS in Figure 3 (given by R. Hindley, see also HUET [78]).

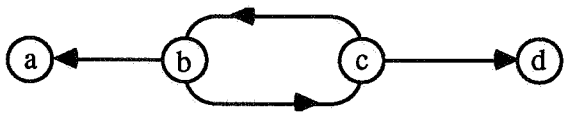


Figure 3

1.6.1. PUZZLE. Find a counterexample to the implication WCR & WN $\Rightarrow$ Ind. (A solution can be found at the end of this paper, before the References.)

There are several other facts about ARSs which often are very helpful e.g. in proving properties of algebraic data type specifications. We present them in the form of the following series Ex. 1.7 (1-17); here 'Ex.' stands for Exercise or Extension of the preceding theory. For an understanding of the sequel these additional facts are not necessary.

1.7. EX.

(1) (Rosen [73]) If $\langle A,\to_1,\to_2\rangle$ is an ARS such that $\twoheadrightarrow_1 = \twoheadrightarrow_2$ and $\to_1$ is subcommutative, then $\to_2$ is confluent.

(2) (Hindley [64) Let $\langle A, (\to_\alpha)_{\alpha\in I}\rangle$ be an ARS such that for all $\alpha,\beta \in I$, $\to_\alpha$ commutes with $\to_\beta$. Then the union $\to = \bigcup_{\alpha\in I} \to_\alpha$ is confluent.

(3) (Hindley [64]) Let $\langle A,\to_1,\to_2\rangle$ be an ARS. Suppose:
$\forall a,b,c\in A\ \exists d\in A\ (a \to_1 b \ \&\ a \to_2 c \Rightarrow b \twoheadrightarrow_2 d \ \&\ c \to_1^{=} d)$. (See Figure 4a.) Then $\to_1,\to_2$ commute.

(4) (Staples [75]) Let $\langle A,\to_1,\to_2\rangle$ be an ARS. Suppose:
$\forall a,b,c\in A\ \exists d\in A\ (a \to_1 b \ \&\ a \twoheadrightarrow_2 c \Rightarrow b \twoheadrightarrow_2 d \ \&\ c \twoheadrightarrow_1 d)$. (See Figure 4b.) Then $\to_1,\to_2$ commute.

(5) (Rosen [73]) Let $\langle A,\to_1,\to_2\rangle$ be an ARS.
DEFINITION: $\to_1$ *requests* $\to_2$ if $\forall a,b,c\in A\ \exists d,e\in A\ (a \twoheadrightarrow_1 b \ \&\ a \twoheadrightarrow_2 c \Rightarrow b \twoheadrightarrow_2 d \ \&\ c \twoheadrightarrow_1 e \twoheadrightarrow_2 d)$).
(See Figure 4c.) To prove: if $\to_1,\to_2$ are confluent and if $\to_1$ requests $\to_2$, then $\to_{12}$ is confluent.

(6) (Rosen [73]) Let $\langle A,\to_1,\to_2\rangle$ be an ARS such that $\to_2$ is confluent and:
$\forall a,b,c\in A\ \exists d,e\in A\ (a \twoheadrightarrow_1 b \ \&\ a \to_2 c \Rightarrow b \twoheadrightarrow_2 d \ \&\ c \twoheadrightarrow_1 e \twoheadrightarrow_2 d)$. (See Figure 4d.) Then $\to_1$ requests $\to_2$.

(7) (Staples [75]) Let $\langle A,\to_1,\to_2\rangle$ be an ARS such that $\to_2$ is confluent and $\to_1$ requests $\to_2$. Let $\to_3$ be the composition of $\twoheadrightarrow_1$ and $\twoheadrightarrow_2$, i.e. $a \to_3 b$ iff $\exists c\ a \twoheadrightarrow_1 c \twoheadrightarrow_2 b$. Suppose moreover that
$\forall a,b,c\in A\ \exists d\in A\ (a \twoheadrightarrow_1 b \ \&\ a \twoheadrightarrow_1 c \Rightarrow b \to_3 d \ \&\ c \to_3 d)$. Then $\to_{12}$ is confluent.

(8) (Staples [75]) DEFINITION: In the ARS $\langle A,\to_1,\to_2\rangle$ the reduction relation $\to_2$ is called a *refinement* of $\to_1$ if $\to_1 \subseteq \twoheadrightarrow_2$. If moreover $\forall a,b\in A\ \exists c\in A\ (a \twoheadrightarrow_2 b \Rightarrow a \twoheadrightarrow_1 c \ \&\ b \twoheadrightarrow_1 c)$, then $\to_2$ is a *compatible* refinement of $\to_1$.

Let in the ARS $\langle A,\to_1,\to_2\rangle$ the reduction relation $\to_2$ be a refinement of $\to_1$. Prove that $\to_2$ is a compatible refinement of $\to_1$ iff $\forall a,b,c\in A\ \exists d\in A\ (a \to_2 b \ \&\ b \twoheadrightarrow_1 c \Rightarrow c \twoheadrightarrow_1 d \ \&\ a \twoheadrightarrow_1 d)$.

(9) (Staples [75]) Let $\langle A, \rightarrow_1, \rightarrow_2 \rangle$ be an ARS where $\rightarrow_2$ is a compatible refinement of $\rightarrow_1$. Then: $\rightarrow_1$ is confluent iff $\rightarrow_2$ is confluent.

(10) (Huet [80]) DEFINITION: Let $\langle A, \rightarrow \rangle$ be an ARS. Then $\rightarrow$ is called *strongly confluent* (see Figure 4e) if: $\forall a,b,c \in A\ \exists d \in A\ (a \rightarrow b\ \&\ a \rightarrow c \Rightarrow b \rightarrow^{\equiv} d\ \&\ c \rightarrow^{\equiv} d)$. Prove that strong confluence implies confluence.

(11) Let $\langle A, (\rightarrow_\alpha)_{\alpha \in I} \rangle$ be an ARS such that for all $\alpha,\beta \in I$, $\rightarrow_\alpha$ commutes weakly with $\rightarrow_\beta$.
DEFINITION: (a) $\rightarrow_\alpha$ is *relatively terminating* if no reduction $a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots$ (where $\rightarrow = \bigcup_{\alpha \in I} \rightarrow_\alpha$) contains infinitely many α-steps.
(b) $\rightarrow_\alpha$ *has splitting effect* if there are $a,b,c, \in A$ such that for every $d \in A$ and every $\beta \in I$ with $a \rightarrow_\alpha b$, $a \rightarrow_\beta c$, $c \rightarrow_\alpha d$, $b \rightarrow_\beta d$, the reduction $b \rightarrow_\beta d$ consists of more than one step.
To prove: if every $\rightarrow_\alpha$ ($\alpha \in I$) which has splitting effect is relatively terminating, then $\rightarrow$ is confluent. (Note that this strengthens Newman's Lemma.)

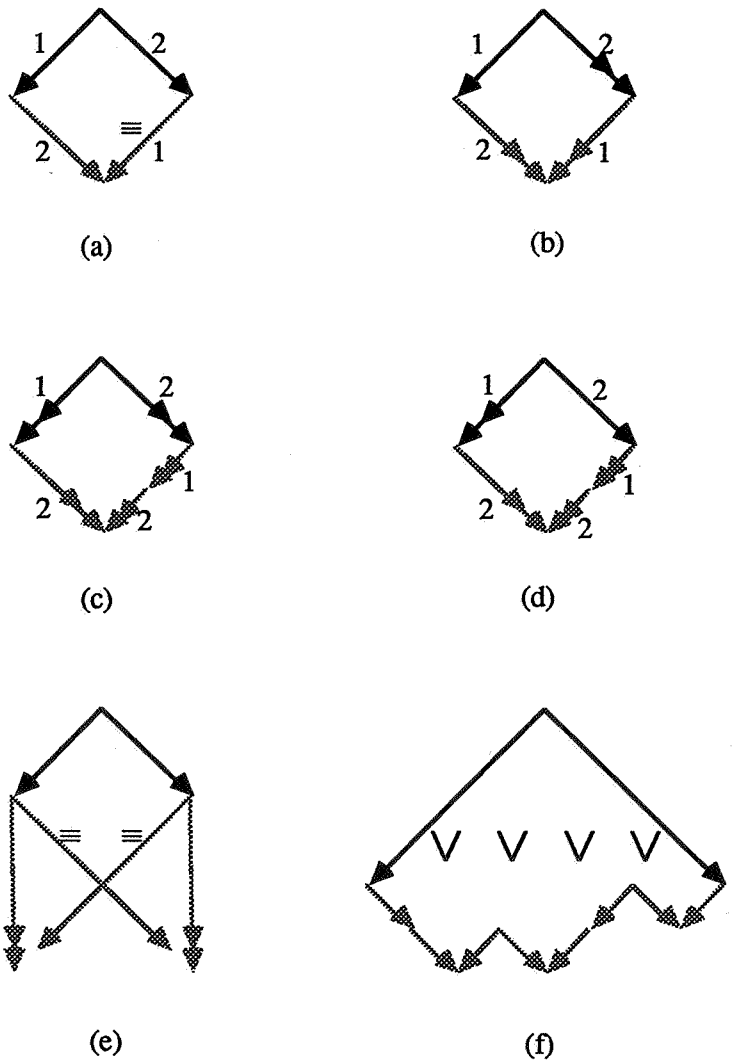


Figure 4

(12) (Le Chenadec [86]) Let $<A, \rightarrow, > >$ be an ARS where the 'reduction' relation $>$ is a partial order and SN. Suppose $a \rightarrow b$ implies $a > b$. Then the following are equivalent:
(a) $\rightarrow$ is confluent, (b) whenever $a \rightarrow b$ and $a \rightarrow c$, there is a $\rightarrow$-conversion $b \equiv d_1 \leftrightarrow d_2 \leftrightarrow \ldots \leftrightarrow d_n \equiv c$ (for some $n \geq 1$) between b,c such that $a > d_i$ $(i = 1,\ldots,n)$. Here each $\leftrightarrow$ is $\rightarrow$ or $\leftarrow$. (See Figure 4f.)
(Note that this strengthens Newman's Lemma.)

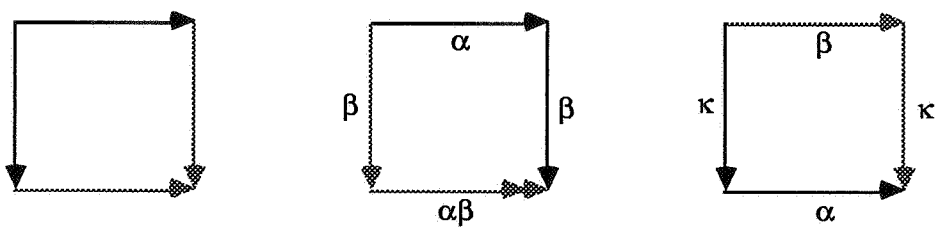
(13) (Klop [80]) Let $\mathcal{A} = <A, \rightarrow>$ be an ARS. Let $B \subseteq A$. Then B is *cofinal* in $\mathcal{A}$ if $\forall a \in A\ \exists b \in B\ a \twoheadrightarrow b$. Furthermore, $\mathcal{A}$ is said to have the *cofinality property* (CP) if in every reduction graph $\mathcal{G}(a)$, $a \in A$, there is a (possibly infinite) reduction sequence $a \equiv a_0 \rightarrow a_1 \rightarrow \ldots$ such that $\{a_n \mid n \geq 0\}$ is cofinal in $\mathcal{G}(a)$.
Then, for *countable* ARSs: $\mathcal{A}$ is CR $\Leftrightarrow$ $\mathcal{A}$ has CP.

(14) Let $\mathcal{A} = <A, \rightarrow>$ be an ARS. Define: A is *consistent* if not every pair of elements in A is convertible. Note that if $\mathcal{A}$ is confluent and has two different normal forms, $\mathcal{A}$ is consistent. Further, let $\mathcal{A} = <A, \rightarrow_\alpha>$, $\mathcal{B} = <B, \rightarrow_\beta>$ be ARSs such that $\mathcal{A} \subseteq \mathcal{B}$. Then we define: $\mathcal{B}$ is a *conservative* extension of $\mathcal{A}$ if $\forall a,a' \in A\ (a =_\beta a' \Leftrightarrow a =_\alpha a')$. Note that a conservative extension of a consistent ARS is again consistent. Further, note that a confluent extension $\mathcal{B}$ of $\mathcal{A}$ is conservative.

(15) (Newman [42]) Let WCR^1 be the following property of ARSs $<A, \rightarrow>$:
$\forall a,b,c \in A\ \exists d \in A\ (c \leftarrow a \rightarrow b \Rightarrow c \rightarrow d \leftarrow b)$. (See Figure 5a.) Prove that WCR^1 & WN $\Rightarrow$ SN, and give a counterexample to the implication $WCR^{\leq 1}$ & WN $\Rightarrow$ SN.

(16) (Bachmair & Dershowitz [86]) Let $<A, \rightarrow_\alpha, \rightarrow_\beta>$ be an ARS such that $\forall a,b,c \in A\ \exists d \in A\ (a \rightarrow_\alpha b \rightarrow_\beta c \Rightarrow a \rightarrow_\beta d \twoheadrightarrow_{\alpha\beta} c)$. (In the terminology of Bachmair & Dershowitz [86]: β *quasi-commutes over* α.) (See Figure 5b.) Prove that α is SN iff β is SN.

(17) (Klop [80]) Let $\mathcal{A} = <A, \rightarrow_\alpha>$ and $\mathcal{B} = <B, \rightarrow_\beta>$ be ARSs. Let $\iota: A \rightarrow B$ and $\kappa: B \rightarrow A$ be maps such that
(i) $\kappa(\iota(a)) = a$ for all $a \in A$,
(ii) $\forall a,a' \in A\ \forall b \in B\ \exists b' \in B\ (b \rightarrow_\kappa a' \rightarrow_\alpha a' \Rightarrow b \rightarrow_\beta b' \rightarrow_\kappa a')$ (Reductions in $\mathcal{A}$ can be 'lifted' to $\mathcal{B}$.) See Figure 5c.
Prove that $\mathcal{B}$ is SN implies that $\mathcal{A}$ is SN.



(a) (b) (c)

Figure 5

2. Term Rewriting Systems: basic notions

2.1. Syntax of Term Rewriting Systems.

A Term Rewriting System (TRS) is a pair (Σ,R) of an *alphabet* or *signature* Σ and a set of reduction rules (rewrite rules) R. The alphabet Σ consists of:

(i) a countably infinite set of *variables* $x_1,x_2,x_3,...$ also denoted as x,y,z,x',y',...

(ii) a non-empty set of *function symbols* or *operator symbols* F,G,..., each equipped with an 'arity' (a natural number), i.e. the number of 'arguments' it is supposed to have. We not only (may) have unary, binary, ternary, etc., function symbols, but also 0-ary: these are also called *constant symbols*.

The set of terms (or expressions) 'over' Σ is $Ter(\Sigma)$ and is defined inductively:

(i) $x,y,z,... \in Ter(\Sigma)$,

(ii) if F is an n-ary function symbol and $t_1,...,t_n \in Ter(\Sigma)$ $(n \geq 0)$, then $F(t_1,...,t_n) \in Ter(\Sigma)$.

The t_i $(i = 1,...,n)$ are the arguments of the last term.

Terms not containing a variable are called *ground* terms (also: *closed* terms), and $Ter_0(S)$ is the set of ground terms. Terms in which no variable occurs twice or more, are called *linear*.

Contexts are 'terms' containing one occurrence of a special symbol $\Box$, denoting an empty place. A context is generally denoted by $\mathbb{C}[\ ]$. If $t \in Ter(\Sigma)$ and t is substituted in $\Box$, the result is $\mathbb{C}[t] \in Ter(\Sigma)$; t is said to be a subterm of $\mathbb{C}[t]$, notation $t \subseteq \mathbb{C}[t]$. Since $\Box$ is itself a context, the trivial context, we also have $t \subseteq t$. Often this notion of subterm is not precise enough, and we have to distinguish *occurrences* of subterms (or symbols) in a term; it is easy to define the notion of occurrence formally, using sequence numbers denoting a 'position' in the term, but here we will be satisfied with a more informal treatment.

2.1.1. EXAMPLE. Let $\Sigma = \{A,M,S,0\}$ where the arities are 2,2,1,0 respectively. Then A(M(x,y),y)) is a (non-linear) term, A(M(x,y),z)) is a linear term, A(M(S(0),0),S(0)) is a ground term, A(M($\Box$,0),S(0)) is a context, S(0) is a subterm of A(M(S(0),0),S(0)) having two occurrences: A(M(**S(0)**,0),**S(0)**).

A *substitution* σ is a map from $Ter(\Sigma)$ to $Ter(\Sigma)$ satisfying $\sigma(F(t_1,...,t_n)) = F(\sigma(t_1),...,\sigma(t_n))$ for every n-ary function symbol (here $n \geq 0$). So, σ is determined by its restriction to the set of variables. We also write t^σ instead of $\sigma(t)$.

A *reduction rule* (or rewrite rule) is a pair (t,s) of terms $\in Ter(\Sigma)$. It will be written as $t \to s$. Often a reduction rule will get a name, e.g. r, and we write r: $t \to s$. Two conditions will be imposed:

(i) the LHS (left-hand side) t is not a variable,

(ii) the variables in the right-hand side s are already contained in t.

A reduction rule r: $t \to s$ determines a set of *rewrites* $t^\sigma \to_r s^\sigma$ for all substitutions σ. The LHS t^σ is called a *redex*, more precisely an r-redex. A redex t^σ may be replaced by its '*contractum*' s^σ inside a context $\mathbb{C}[\ ]$; this gives rise to *reduction steps* (or one-step rewritings)

$$\mathbb{C}[t^\sigma] \to_r \mathbb{C}[s^\sigma].$$

We call $\to_r$ the *one-step reduction relation* generated by r. Concatenating reduction steps we have (possibly infinite) *reduction sequences* $t_0 \to t_1 \to t_2 \to \ldots$ or *reductions* for short. If $t_0 \to \ldots \to t_n$ we also write $t_0 \twoheadrightarrow t_n$, and t_n is a *reduct* of t_0, in accordance with the notations and concepts introduced in Section 1.

2.1.2. EXAMPLE. Consider Σ as in Example 2.1.1. Let (Σ,R) be the TRS (specifying the natural numbers with addition, multiplication, successor and zero) with the following set R of reduction rules:

r_1	$A(x,0) \to x$
r_2	$A(x,S(y)) \to S(A(x,y))$
r_3	$M(x,0) \to 0$
r_4	$M(x,S(y)) \to A(M(x,y),x)$

Table 1

Now $M(S(S(0)), S(S(0))) \twoheadrightarrow S(S(S(S(0))))$, since we have the following reduction:

$\mathbf{M(S(S(0)),S(S(0)))} \to$
$\mathbf{A(M(S(S(0)),S(0)),S(S(0)))} \to$
$S(\ \mathbf{A(M(S(S(0)),S(0)),S(0))}\ ) \to$
$S(\ S(\ \mathbf{A(M(S(S(0)),S(0)),0)}\ )\ ) \to$
$S(\ S(\ \mathbf{M(S(S(0)),S(0))}\ )\ ) \to$
$S(S(A(\mathbf{M(S(S(0)),0)},S(S(0))))) \to$
$S(S(\mathbf{A(0,S(S(0)))})) \to$
$S(S(S(\mathbf{A(0,S(0))}))) \to$
$S(S(S(S(\mathbf{A(0,0)})))) \to$
$S(S(S(S(0))))$.

Here in each step the bold-face redex is rewritten. Note that this is not the only reduction from M(S(S(0)), S(S(0))) to S(S(S(S(0)))).

Obviously, for each TRS (Σ,R) there is a corresponding ARS, namely $(Ter(\Sigma), (\to_r)_{r\in R})$. Here we have to be careful: it may make a big difference whether one discusses the TRS (Σ,R) consisting of all terms, or the TRS restricted to the ground terms (see the next example). We will adopt the convention that (Σ,R) has as corresponding ARS the one mentioned already, and we write $(\Sigma,R)_0$ if the ARS $(Ter_0(\Sigma), (\to_r)_{r\in R})$ is meant. Via the associated ARS, all notions considered in Section 1 (CR, UN, SN, ...) carry over to TRSs.

2.1.3. EXAMPLE. Let (Σ,R) be the TRS of Example 2.1.1 and consider (Σ,R') where $R' = R \cup \{A(x,y) \rightarrow A(y,x)\}$; so the extra rule expresses commutativity of addition. Now (Σ,R') is not WN: the term A(x,y) has no normal form. However, $(\Sigma,R')_0$ (the restriction to ground terms) is WN.

Whereas $(\Sigma,R)_0$ is SN, $(\Sigma,R')_0$ is no longer so, as witnessed by the infinite reductions possible in the reduction graph in Figure 6. The 'bottom' term in that reduction graph is a normal form.

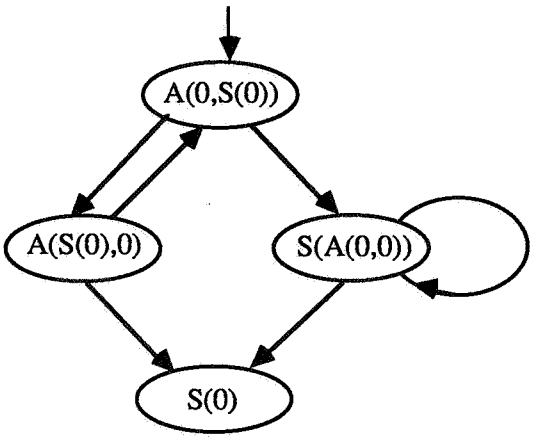


Figure 6

2.1.4. Many-sorted Term Rewriting Systems.

TRSs (Σ,R) as we have defined in 2.1 are sometimes called *homogeneous* (GANZINGER & GIEGERICH [87]), as they correspond to algebraic data type specifications (by replacing '$\rightarrow$' by '=' in R) where the signature Σ has just one sort (which therefore was not mentioned).

It is straightforward to extend our previous definitions to the *heterogeneous* or *many-sorted* case. The definition of term formation is as usual in many-sorted abstract data type specifications, and is left to the reader. We will stick to the homogeneous case, but note that 'everything' extends at once to the heterogeneous case.

2.1.5. Semi-Thue systems.

Semi-Thue Systems (STSs), as defined e.g. in JANTZEN [86], can be 'viewed' in two ways as TRSs. We demonstrate this by an example of a STS occurring in JANTZEN [86]:

(1) Let T = {(aba, bab)} be a one-rule STS. Then T corresponds to the TRS R with unary function symbols a,b and a constant o, and the reduction rule $a(b(a(x))) \rightarrow b(a(b(x)))$. Now a reduction step in T, e.g.: bbabaaa $\rightarrow$ bbbabaa, translates in R to the reduction step $b(b(a(b(a(a(o))))))) \rightarrow b(b(b(a(b(a(a(o))))))))$. It is easy to see that this translation gives an 'isomorphism' between T and R (or more precisely $(R)_0$, the restriction to ground terms).

(2) The second way to let a STS correspond to a TRS is by introducing an associative concatenation operator ., and letting the symbols of the STS correspond to constant symbols in the TRS. In fact, a 'natural' correspondence in this way requires that we introduce *equational TRSs*, which we will not do here. (See e.g. BACHMAIR & PLAISTED [85] or PLAISTED [85].)

2.1.6. PUZZLE. Find a ***one rule*** TRS which is weakly normalizing (WN), but not strongly normalizing (SN). An

answer can be found just before the References.

2.2. Applicative Term Rewriting Systems.

In some important TRSs there is a very special binary operator, called *application* (Ap). E.g. Combinatory Logic (CL), based on S,K,I, has the rewrite rules

Ap(Ap(Ap(S,x),y),z)	→ Ap(Ap(x,z), Ap(y,z))
Ap(Ap(K,x),y)	→ x
Ap(I,x)	→ x

Table 2

Here S,K,I are constants. Often one uses the infix notation (t.s) instead of Ap(t,s), in which case the rewrite rules of CL read as follows:

((S.x).y).z	→ (x.z).(y.z)
(K.x).y	→ x
I.x	→ x

Table 3

As in ordinary algebra, the dot is mostly suppressed; and a further notational simplification is that many pairs of brackets are dropped in the convention of *association to the left.* That is, one restores the missing brackets choosing in each step of the restoration the leftmost possibility. Thus the three rules become:

Sxyz	→ xz(yz)
Kxy	→ x
Ix	→ x

Table 4

Note that xz(yz) restores to (xy)(yz), not to x(z(yz)). Likewise Kxy restores to (Kx)y, not K(xy). Of course not all bracket pairs can be dropped: xzyz is when restored ((xz)y)z, which is quite different from xz(yz). Note that e.g. SIx does not contain a redex Ix.

It is a convenient fiction to view the S,K,I in the last three equations as "operators with variable "arity" or *varyadic* operators, since they may be followed by an arbitrary number of arguments $t_1,...,t_n$ ($n \geq 0$). But it needs, in the case of S, at least three arguments to use the rewrite rule for S; e.g.: $St_1t_2t_3t_4t_5t_6 \rightarrow t_1t_3(t_2t_3)t_4t_5t_6$.

2.2.1. EXAMPLE. SII(SII) → I(SII)(I(SII)) → SII(I(SII)) → SII(SII). The term SII(SII) has many more reductions, which constitute an interesting reduction graph (see Figure 7).

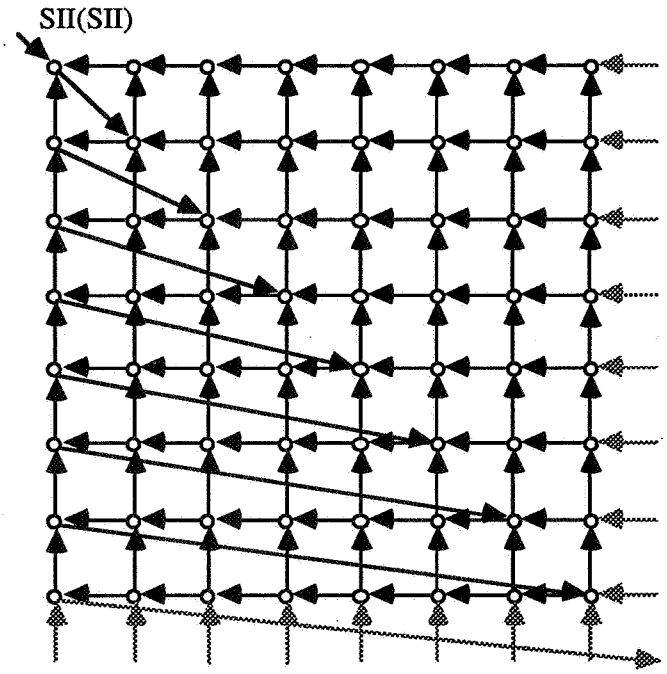


Figure 7

The TRS CL has 'universal computational power': every (partial) recursive function on the natural numbers can be expressed in CL. This feature is used in TURNER [79], where CL is used to implement functional languages. Actually, an extension of CL is used there, called SKIM (for S,K,I-Machine); it is also an applicative TRS:

SKIM

Sxyz	→ xz(yz)	
Kxy	→ x	
Ix	→ x	
Cxyz	→ xzy	
Bxyz	→ x(yz)	
Yx	→ x(Yx)	
Uz(Pxy)	→ zxy	
cond true xy	→ x	
cond false xy	→ y	
plus n m	→ n+m	
times n m	→ n.m	
eq n n	→ true	
eq n m	→ false	if n ≠ m

Table 5

In fact, the extra constants in SKIM are there for reasons of efficient implementation; they can all be defined using only S and K. E.g. defining B as S(KS)K we have:

$$\begin{aligned} Bxyz \equiv\ & S(KS)Kxyz && \rightarrow \\ & KSx(Kx)yz && \rightarrow \\ & S(Kx)yz && \rightarrow \\ & Kxz(yz) && \rightarrow \\ & x(yz) \end{aligned}$$

as we should have. Likewise, defining C as S(BBS)(KK), we have $Cxyz \twoheadrightarrow xzy$ as the reader may check. For the other definitions one may consult BARENDREGT [81] or HINDLEY & SELDIN [86].

It is harmless to mix the applicative notation with the usual one, as in the following TRS, CL with test for syntactical equality:

Sxyz	$\rightarrow$ xz(yz)
Kxy	$\rightarrow$ x
Ix	$\rightarrow$ x
D(x,x)	$\rightarrow$ E

Table 6

However, some care should be taken: consider the following TRS

Sxyz	$\rightarrow$ xz(yz)
Kxy	$\rightarrow$ x
Ix	$\rightarrow$ x
Dxx	$\rightarrow$ E

Table 7

where D is now a *constant* (instead of a binary operator) subject to the rewrite rule, in full notation, $Ap(Ap(D,x),x) \rightarrow E$. These two TRSs have very different properties, as we shall see later (the first TRS is confluent, the second is not).

Another interesting example of a TRS in such a mixed notation is Weak Categorical Combinatory Logic in Table 8, which plays an important role in implementations of functional languages (see CURIEN [86]). Here Id, Fst, Snd, App are constants, $\circ$, < , > and (,) are binary function symbols and Λ is an unary function symbol. Note that Fst, Snd are not binary symbols and that App is not the 'underlying' application operator which was called in CL above Ap.

Id x	= x
(x∘y)z	= x(yz)
Fst(x,y)	= x
Snd(x,y)	= y
<x,y>z	= (xz,yz)
App(x,y)	= xy
Λ(x)yz	= x(y,z)

Table 8

2.3. Direct sums of Term Rewriting Systems.

In view of the need for modularisation of abstract data type specifications, it would be very helpful if some properties of a TRS could be inferred from their validity for 'parts' of that TRS. The simplest possible definition of 'parts' is that obtained by the concept of 'direct sum' of TRSs:

2.3.1. DEFINITION. Let R_1,R_2 be TRSs. Then the *direct sum* $R_1 \oplus R_2$ of R_1,R_2 is the TRS obtained by taking the disjoint union of R_1 and R_2. That is, if the alphabets of R_1,R_2 are disjoint (R_1,R_2 have no function or constant symbols in common), then the direct sum is the ordinary union; otherwise we take renamed copies R_1',R_2' of R_1,R_2 such that these copies have disjoint alphabets and define $R_1 \oplus R_2$ to be the union of these copies.

We have the following useful fact from TOYAMA [87]:

2.3.2. THEOREM. $R_1 \oplus R_2$ *is confluent iff* R_1 *and* R_2 *are confluent.*

So, confluence is a 'modular' property. One might think that the same is true for termination (SN), but TOYAMA [87] gives a simple counterexample: take

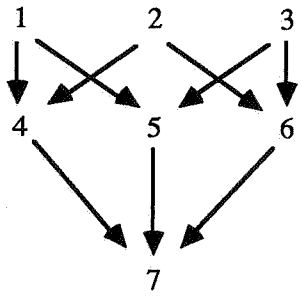
$$R_1 = \{f(0,1,x) \rightarrow f(x,x,x)\}$$
$$R_2 = \{\underline{or}(x,y) \rightarrow x, \underline{or}(x,y) \rightarrow y\}$$

then R_1,R_2 are both SN, but $R_1 \oplus R_2$ is not, since there is the infinite reduction:

$$f(\underline{or}(0,1), \underline{or}(0,1), \underline{or}(0,1)) \rightarrow f(0, \underline{or}(0,1), \underline{or}(0,1)) \rightarrow$$
$$f(0, 1, \underline{or}(0,1)) \rightarrow f(\underline{or}(0,1), \underline{or}(0,1), \underline{or}(0,1)) \rightarrow ....$$

In this counterexample R_2 is not confluent and thus one may conjecture that 'confluent and terminating' (or CR & SN, or complete) is a modular property (i.e. $R_1 \oplus R_2$ is complete iff R_1,R_2 are so). Again this is not the case, as a counterexample given by Barendregt and Klop (adapted by Toyama, see TOYAMA [86]) shows: R_1 has the eleven rules

$F(4,5,6,x) \to F(x,x,x,x)$
$F(x,y,z,w) \to 7$



and R_2 has the three rules

$G(x,x,y) \to x$
$G(x,y,x) \to x$
$G(y,x,x) \to x.$

(Similar counterexamples with the additional property of being 'reduced' or 'irreducible' - meaning that both sides of every rule are normal forms w.r.t. the other rules - are given in TOYAMA [86] and GANZINGER & GIEGERICH [87]. A correction to the last reference: there the diagram on p.23 has one arrow too many, namely $b \to k(0)$; otherwise R is not terminating as shown by the term f(b,b,b,g(0,b)), reducing to itself.)

Now R_1 and R_2 are both complete, but $R_1 \oplus R_2$ is not:

$F(G(1,2,3), G(1,2,3), G(1,2,3), G(1,2,3)) \twoheadrightarrow$
$F(G(4,4,3), G(5,2,5), G(1,6,6), G(1,2,3)) \twoheadrightarrow$
$F(\ 4,\ \ 5,\ \ 6,\ \ G(1,2,3)) \to$
$F(G(1,2,3), G(1,2,3), G(1,2,3), G(1,2,3)).$

2.3.3. EXAMPLE. (i) Consider CL $\oplus$ $\{D(x,x) \to E\}$, Combinatory Logic with binary test for syntactic equality as in Table 6. Note that this is indeed a direct sum. As we shall see in Section 5, CL is confluent. Trivially, the one rule TRS $\{D(x,x) \to E\}$ is confluent. Hence, by Toyama's theorem (2.3.2) the direct sum is confluent.

(ii) By contrast, the union CL $\cup$ $\{Dxx \to E\}$, Combinatory Logic with 'varyadic' test for syntactic equality as in Table 7, is *not* confluent. (See KLOP [80].) Note that this combined TRS is merely a union and not a direct sum, since CL and $\{Dxx \to E\}$ have the function symbol Ap in common, even though hidden by the applicative notation.

(iii) Another application of Toyama's theorem (2.3.2): let R consist of the rules

$$\underline{\text{if}}\ \underline{\text{true}}\ \underline{\text{then}}\ x\ \underline{\text{else}}\ y \to x$$
$$\underline{\text{if}}\ \underline{\text{false}}\ \underline{\text{then}}\ x\ \underline{\text{else}}\ y \to y$$
$$\underline{\text{if}}\ z\ \underline{\text{then}}\ x\ \underline{\text{else}}\ x \to x.$$

(Here true, false are constants and if - then - else is a ternary function symbol.) Then CL $\oplus$ R is confluent.

In this introduction to TRSs we will not consider termination properties of combined TRSs $R_1 \cup R_2$ which are not direct sums. For results in that area see DERSHOWITZ [81,85], BACHMAIR & DERSHOWITZ [86], TOYAMA [86] and, for heterogeneous TRSs, GIEGERICH & GANZINGER [87].

2.4. Semantics of Term Rewriting Systems.

Although we do not enter the subject of *semantics* of TRSs (see e.g. BOUDOL [85]), there is one simple remark that should be made. It concerns a semantical consideration that can be of great help in a proof of UN or CR:

2.4.1. THEOREM. *Let* $\mathcal{A}$ *be an algebra 'for' the TRS* R *such that for all normal forms* t,t' *of* R:

$$\mathcal{A} \models t = t' \ \Rightarrow\ t \equiv t'.$$

Then R *has the property UN (uniqueness of normal forms).*

Here the phrase '$\mathcal{A}$ is an algebra for the TRS R' means that $\mathcal{A}$ has the same signature as R, and that reduction in R is sound w.r.t. $\mathcal{A}$, i.e. $t \twoheadrightarrow_R s$ implies $\mathcal{A} \models t = s$. The terms t,s need not be ground terms.

More 'semantic confluence tests' can be found in PLAISTED [85], in the setting of equational TRSs.

2.5. Decidability of properties in Term Rewriting Systems.

We adopt the restriction in this subsection to TRSs R with finite alphabet and finitely many reduction rules. It is undecidable whether for such TRSs the property confluence (CR) holds. (This is so both for R, the TRS of all terms, and $(R)_0$, the TRS restricted to ground terms.)

For *ground TRSs*, i.e.TRSs where in every rule $t \to s$ the terms t,s are ground terms (not to be confused with $(R)_0$ above), confluence is decidable (DAUCHET & TISON [84]).

For the termination property (SN) the situation is the same. It is undecidable for general TRSs, even for TRSs with two rules only (see for a proof DERSHOWITZ [85]); for one rule TRSs the question is open. For ground TRSs termination is decidable (HUET & LANKFORD [78]).

For particular TRSs it may also be undecidable whether two terms are convertible, whether a term has a normal form, whether a term has an infinite reduction. A TRS where all these properties are undecidable is Combinatory Logic (CL), in Table 4.

We mention a recent result which is as far as we know new: let R_1,R_2 be left-linear TRSs (see for 'left-linear' section 5), and let $NF_0(R_i)$ (i = 1,2) be their sets of ground normal forms. Then it is decidable whether $NF_0(R_1) = NF_0(R_2)$ (see WIEDIJK [87]). Work in progress indicates that the condition of left-linearity can be removed.

3. A termination proof technique

As Newman's Lemma (WCR & SN $\Rightarrow$ CR) shows, termination (SN) is a useful property. In general, as noted in 2.5, it is undecidable whether a TRS is SN; but in many instances SN can be proved and various techniques have been developed to do so. (See HUET & OPPEN [80], DERSHOWITZ [85].) We will present in this section one of the most powerful of such termination proof techniques: the method of *recursive path orderings*, as developed by Dershowitz on the basis of a beautiful theorem of Kruskal. In fact we will use the presentation of BERGSTRA & KLOP [85], where the rather complicated inductive definitions of the usual presentation are replaced by a reduction procedure which is to our taste easier to grasp.

3.1. DEFINITION. (i) Let $\mathbb{T}$ be the set of commutative finite trees with nodes labeled by natural numbers. Example: see Figure 8a. This tree will also be denoted by: 3(5,7(9),8(0(1,5))). Commutativity means that the 'arguments' may be permuted; thus 3(8(0(5,1)),5,7(9)) denotes the same commutative tree.

(ii) Let $\mathbb{T}^*$ be the set of such trees where some of the nodes may be marked with (a single) *. So $\mathbb{T} \subseteq \mathbb{T}^*$. Example: see Figure 8b; this tree will be denoted by 3*(5,7(9*),8*(0(1,5))).

3.2. NOTATION. $n(t_1,...,t_k)$ will be written as $n(\mathbb{t})$. The t_i (i = 1,...,k) are elements of $\mathbb{T}^*$. Further, if $t \equiv n(t_1,...,t_k)$ then t* stands for $n^*(t_1,...,t_k)$.

3.3. DEFINITION. On T^* we define a reduction relation $\rightarrow$ as follows.

(i) *place marker at the top:*

$n(\mathbb{t}) \rightarrow n^*(\mathbb{t})$ $\quad (\mathbb{t} = t_1,...,t_k;\ k \geq 0)$

(ii) *make copies below lesser top:*

if $n > m$, then $n^*(\mathbb{t}) \rightarrow m(n^*(\mathbb{t}),...,n^*(\mathbb{t}))$ $\quad (j \geq 0$ copies of $n^*(\mathbb{t}))$

(iii) *push marker down:*

$n^*(s,\mathbb{t}) \rightarrow n(s^*,...,s^*,\mathbb{t})$ $\quad (j \geq 0$ copies of $s^*)$

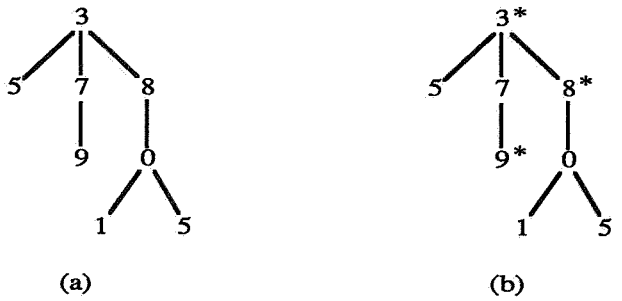
(iv) *select argument:*

$n^*(t_1,...,t_k) \rightarrow t_i$ $\quad (i \in \{1,...,k\},\ k \geq 1)$

(v) *in context:*

if $t \rightarrow s$, then $n(---,t,---) \rightarrow n(---,s,---)$

We write $\rightarrow^+$ for the transitive (but not reflexive) closure of $\rightarrow$.



(a) (b)

Figure 8

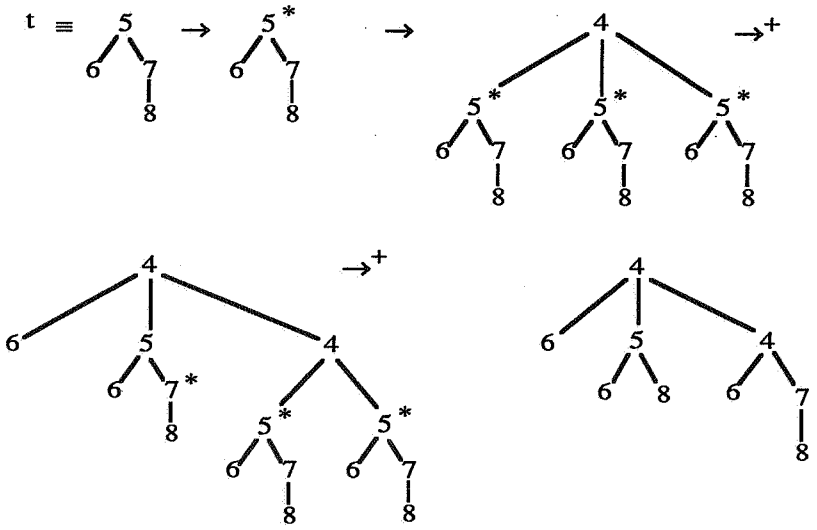


Figure 9

3.4. EXAMPLE. Figure 9 above displays a reduction in $\mathbb{T}^*$.

Clearly, the reduction $\rightarrow$ is not SN in $\mathbb{T}^*$; for, consider the second step in Figure 9: there the right hand side contains a copy of the left-hand side. However:

3.5. THEOREM. *The relation $\rightarrow^+$, restricted to $\mathbb{T}$, is a well-founded partial ordering. Or, rephrased, the relation $\rightarrow^+$, restricted to $\mathbb{T}$, is SN.*

So there is no infinite sequence $t_0 \rightarrow^+ t_1 \rightarrow^+ t_2 \rightarrow^+ \ldots$ of terms t_i ($i \geq 0$) without markers. The proof of Theorem 3.5 is based on Kruskal's Tree Theorem; we will give the main argument.

3.6. DEFINITION. Let $t,s \in \mathbb{T}$. We say that s is *homeomorphically embedded in* t, notation $s \trianglelefteq t$, if there is an injection φ: NODES(s) $\rightarrow$ NODES(t) which is an order-preserving isomorphism and such that for all nodes $\alpha \in$ NODES(s) we have: label(α) $\leq$ label($\varphi(\alpha)$) where $\leq$ is the usual ordering on $\mathbb{N}$.

3.7. EXAMPLE. $2(9,7(0,4)) \trianglelefteq 1(3(8(0(5,1)),9,5(9)),2)$ as the embedding in Figure 10 shows.

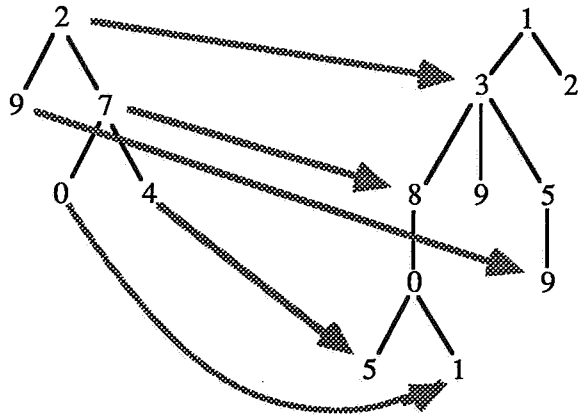


Figure 10

Clearly, $\trianglelefteq$ is a partial order on T. Moreover it satisfies the following property (for a short proof see DERSHOWITZ [79]):

3.8. KRUSKAL'S TREE THEOREM. *Let* $t_0,t_1,t_2,...$ *be a sequence of trees in* $\mathbb{T}$. *Then for some* $i < j$: $t_i \trianglelefteq t_j$.

It is not hard to prove the following proposition:

3.9. PROPOSITION. (i) $\rightarrow^+$ *is a strict partial order on* $\mathbb{T}$, (ii) *if* $s \trianglelefteq t$, *then* $t \rightarrow^+ s$.

Combining 3.8 and 3.9, we have Theorem 3.5. For, suppose there is an infinite sequence

$$t_0 \rightarrow^+ t_1 \rightarrow^+ t_2 \rightarrow^+ ... \rightarrow^+ t_i \rightarrow^+ ... \rightarrow^+ t_j \rightarrow^+ ...$$

then for some $i < j$ we have $t_i \trianglelefteq t_j$, hence $t_j \rightarrow^+ t_i$ which is impossible as $\rightarrow^+$ is a strict partial order.

3.10. APPLICATION. Let a TRS R as in Table 9 be given. To prove that R is SN.

$\neg\neg x$	$\rightarrow$	x
$\neg(x \vee y)$	$\rightarrow$	$(\neg x \wedge \neg y)$
$\neg(x \wedge y)$	$\rightarrow$	$(\neg x \vee \neg y)$
$x \wedge (y \vee z)$	$\rightarrow$	$(x \wedge y) \vee (x \wedge z)$
$(y \vee z) \wedge x$	$\rightarrow$	$(y \wedge x) \vee (z \wedge x)$

Table 9

Choose a 'weight' assignment $\vee \to 1$, $\wedge \to 2$, $\neg \to 3$. Now a reduction in R corresponds to a $\to^+$ reduction in $\mathbb{T}$ (and hence it is also SN) as follows:

$$\begin{array}{lll} 3(3(t)) & \to^+ & t \\ 3(1(t,s)) & \to^+ & 2(3(t),3(s)) \\ 3(2(t,s)) & \to^+ & 1(3(t),3(s)) \\ 2(t,1(s,r)) & \to^+ & 1(2(t,s),2(t,r)) \\ 2(1(s,r),t) & \to^+ & 1(2(s,t),2(r,t)) \end{array}$$

E.g. the second rule:

$$3(1(t,s)) \to 3^*(1(t,s)) \to 2(3^*(1(t,s)), 3^*(1(t,s))) \to^+ 2(3(1^*(t,s)), 3(1^*(t,s))) \to^+ 2(3(t),3(s)).$$

3.11. REMARK. (i) The termination proof method above does not work when a rule is present of which the left-hand side is embedded (in the sense of Definition 3.6) in the right-hand side, as in $f(s(x)) \to g(s(x), f(p(s(x))))$. For an extension of Kruskal's Theorem, leading to a method which also can deal with this case, see PUEL [86].

(ii) Another example where the method above does not work directly, is found in the TRSs corresponding to process algebra axiomatisations as in BERGSTRA & KLOP [84,85]. For instance in the axiom system PA there are the rewrite rules

$$\begin{array}{lll} x \parallel y & \to & (x \between y) + (y \between x) \\ (x+y) \between z & \to & (x \between z) + (y \between z) \\ (a.x) \between y & \to & a.(x \parallel y). \end{array}$$

Here one wants to order the operators as follows: $\parallel > \between > .,+$, but then we get stuck at the third rule with the re-emergence of the 'heavy' operator $\parallel$. In BERGSTRA & KLOP [85] the solution was adopted to introduce infinitely many operators $\parallel_n$ and $\between_n$, where n refers to some complexity measure of the actual arguments of the operators in a reduction. In fact, the operator + does not contribute to the problem, and forgetting about it and writing $x \parallel y$ as $g(x,y)$, $x \between y$ as $h(x,y)$, $a.x$ as $f(x)$, we have exactly Example 16 in DERSHOWITZ [85] where the same problem is noted and solved by a lexicographical combination of recursive path orderings:

$$\begin{array}{lll} g(x,y) & \to & h(x,y) \\ h(f(x),y) & \to & f(g(x,y)). \end{array}$$

The termination proof as in BERGSTRA & KLOP [85] amounts to the following. Define a norm $|\ |$ on terms by: $|t|$ = the length of t in symbols; then introduce normed operators g_n and h_n ($n \geq 2$); order the operators thus: $g_n > h_n > f$, $h_{n+1} > g_n$. Then replace in a term t every subterm $h(s,r)$ by $h_{|s|+|r|}(s,r)$ and likewise for $g(s,r)$. Now the recursive path ordering as before is applicable.

(iii) A third example were the proof method above does not work, is when an associativity rule

$$(x.y).z \rightarrow x.(y.z)$$

is present. The same problem occurs in the TRS for Ackermann's function:

$$\begin{aligned} A(0,x) &\rightarrow S(x) \\ A(S(x),0) &\rightarrow A(x,S(0)) \\ A(S(x),S(y)) &\rightarrow A(x,A(S(x),y)) \end{aligned}$$

What we need here is the *lexicographic path ordering* of Kamin and Lévy, see DERSHOWITZ [85]. Essentially this says that a reduction in complexity in the first argument of A outweighs an increase (strictly bounded by the complexity of the original term) in the second argument. In fact, an ordering with the same effect can easily be described in the framework of reduction with markers * as explained above: all one has to do is give up the commutativity of the trees in $\mathbb{T}$ and $\mathbb{T}^*$ and require that an embedding (Definition 3.6) respects also the left-right ordering; Kruskal's Tree Theorem works also for this case of noncommutative trees. Next, the rules in Definition 3.3. are restricted such that the arities of the operators are respected; in Definition 3.3. the operators were treated 'varyadic'. So rule (iii) becomes: $n^*(t_1,...,t_i,...,t_k) \rightarrow n(t_1,...,t_i^*,...,t_k)$ $(1 \leq i \leq k)$. Further, we add to the rules in Definition 3.3 (with (iii) amended) the rule

(vi) *simplify left argument*

$$n^*(\mathbb{t}) \rightarrow n(t_1^*, n^*(\mathbb{t}),..., n^*(\mathbb{t})) \qquad (\mathbb{t} = t_1,...,t_k\ (k \geq 1);\ k\text{-}1 \text{ copies of } n^*(\mathbb{t}))$$

Example: $A(S(x),S(y)) \rightarrow A^*(S(x),S(y)) \rightarrow A(S^*(x),A^*(S(x),S(y))) \rightarrow A(x,A^*(S(x),S(y)))$
$\rightarrow A(x,A(S(x),S^*(y))) \rightarrow A(x,A(S(x),y))$.

4. Knuth-Bendix completion

Complete TRSs, i.e. TRSs that have the properties SN and CR, are important to solve the *word problem* of equational specifications. E.g. consider the equational specification E of *groups* :

E

e.x	= x
I(x).x	= e
(x.y).z	= x.(y.z)

Table 10

The word problem (for the closed term algebra of E, that is the algebra whose elements are the

ground terms modulo equality derivable from E) is the *decision problem* for the set of equations t = s between ground terms which can be derived from the equations in E. If there is an algorithm to determine whether such an equation is indeed derivable, the word problem is called solvable.

Now suppose that a complete TRS can be found such that for all ground terms t,s:

$$t =_R s \Leftrightarrow E \vdash t = s \qquad (*)$$

(Here $=_R$ is convertibility in R.) Then the word problem for E is solvable; the algorithm is simple: reduce t,s to their normal forms t*,s* and compare. If and only if $t^* \equiv s^*$ we have $E \vdash t = s$.

We are now faced with the question how to find a complete TRS R for a given set of equations E such that (*) holds. In general this is not possible, since not every E has a solvable word problem. The most famous example of such an E with unsolvable word problem is the set of equations obtained from CL, Combinatory Logic, in Tables 2,3,4 above after replacing '→' by '=':

Sxyz	= xz(yz)
Kxy	= x
Ix	= x

Table 11

(The TRS CL itself does not qualify, as it is not complete.)

4.0.1. REMARK. Even if E has a decidable word problem for ground terms, there does not always exist a *finite complete* TRS R *using the same alphabet* equivalent to E (in the sense of (*)).

Cf. the similar situation for Semi-Thue Systems (see 2.1.5): according to KAPUR & NARENDRAN [85], there is no finite complete STS over $\Sigma = \{a,b\}$ equivalent to the STS {(aba, bab)}, which easily can be seen to have a decidable word problem. In KAPUR & NARENDRAN [85] it is stated that this result is also pertinent to TRSs, but the result does not seem to transfer to TRSs immediately, in the correspondence of 2.1.5 (1). (It does transfer immediately to ETRSs or equational TRSs, not treated here.) The problem is the constant o. Translated to TRSs (as in 2.1.5 (1)) the result of Kapur and Narendran reads: there is no finite complete TRS R with unary functions a,b and constant o equivalent (in the sense of (*)) to the TRS { a(b(a(x))) → b(a(b(x))) }, if R is forbidden to have reduction rules where LHS and RHS are ground terms (i.e. all the terms in the rules must end in x, not o). (Rules where LHS and RHS end in o, would correspond in the STS to rules where the LHS may only be replaced in some word w by the RHS if the LHS is a *postfix* of w.)

Nevertheless, also for the TRS case there are examples of E with decidable word problem but without finite complete TRS over the same signature as E. A two-sorted example given by J.A. Bergstra (personal communication): let E be the specification of finite multisets of integers, as follows. The signature contains 0 and successor S for the sort of integers, and for the sort of multisets the constant Ø (the empty multiset) and the operation ins (insertion of an integer into a multiset). The set of equations E consists of the single commutativity axiom ins(x, ins(y,X)) = ins(y, ins(x,X)).

But for many E's, including the one in Table 10, a complete TRS satisfying (*) can be found. To illustrate the idea of the 'completion' algorithm, first described in KNUTH & BENDIX [70], we embark on a 'naive' attempt to complete the set of equations E for groups by hand. (For an explanation of a similar completion procedure on STSs, see JANTZEN [86].) We start with giving the equations a 'sensible' orientation:

1. e.x → x
2. I(x).x → e
3. (x.y).z → x.(y.z)

These rules are not confluent, as can be seen by superposition of e.g. 2 and 3. Redex I(x).x can be unified with a *non-variable* subterm of redex (x.y).z (the underlined subterm), with result (I(x).x).z. This term is subject to two possible reductions: (I(x).x).z $\rightarrow_2$ e.z and (I(x).x).z $\rightarrow_3$ I(x).(x.z). The pair of reducts

<e.z, I(x).(x.z)>

is called a *critical pair* . After reduction e.z → z we have the problematic pair of terms z, I(x).(x.z); problematic because their equality is derivable from E, but they have no common reduct w.r.t. the reduction available so far. Therefore we adopt a new
rule

4. I(x).(x.z) → z

Now we have a superposition of rule 2 and 4: I(I(y)).(I(y).y) $\rightarrow_4$ y and I(I(y)).(I(y).y) $\rightarrow_2$ I(I(y)).e. This yields the critical pair <y, I(I(y)).e> which cannot further be reduced. Adopt new rule:

5. I(I(y)).e → y canceled later

As it will turn out, in a later stage this last rule will become superfluous. We go on searching for critical pairs:
Superposition of 4,1: I(e).(e.z) $\rightarrow_4$ z and I(e).(e.z) $\rightarrow_1$ I(e).z.
Adopt new rule:

6. I(e).z → z canceled later

Superposition of 3,5: (I(Iy)).e).x $\rightarrow_3$ I(I(y)).(e.x) and (I(Iy)).e).x $\rightarrow_5$ y.x.
Adopt new rule:

7. I(Iy)).x → y.x canceled later

Superposition of 5,7: I(I(y)).e $\rightarrow_7$ y.e and I(I(y)).e $\rightarrow_5$ y.
Adopt new rule:

8. y.e → y

Superposition of 5,8: I(I(y)).e $\rightarrow_5$ y and I(I(y)).e $\rightarrow_8$ I(I(y)).
Adopt new rule

9. I(I(y)) → y cancel 5 and 7

(Rule 5 is now no longer necessary to ensure that the critical pair <y, I(I(y)).e> has a common reduct, because: $I(I(y)).e \rightarrow_9 y.e \rightarrow_8 y$. Likewise for rule 7.)
Superposition of 6,8: $I(e).e \rightarrow_6 e$ and $I(e).e \rightarrow_8 I(e)$.
Adopt new rule

10. $I(e) \rightarrow e$ cancel 6

Superposition of 2,9: $I(I(y)).I(y) \rightarrow_2 e$ and $I(I(y)).I(y) \rightarrow_9 y.I(y)$.
Adopt new rule

11. $y.I(y) \rightarrow e$

Superposition of 3,11: $(y.I(y)).x \rightarrow_3 y.(I(y).x)$ and $(y.I(y)).x \rightarrow_{11} e.x$.
Adopt new rule

12. $y.(I(y).x) \rightarrow x$

Superposition (again) of 3,11: $(x.y).I(x.y) \rightarrow_{11} e$ and $(x.y).I(x.y) \rightarrow_3 x.(y.I(x.y))$.
Adopt new rule

13. $x.(y.(y.I(x.y)) \rightarrow e$ canceled later

Superposition of 13,4: $I(x).[x.(y.I(x.y))] \rightarrow_4 y.I(x.y)$ and $I(x).[x.(y.I(x.y))] \rightarrow_{13} I(x).e$.
Adopt new rule

14. $y.I(x.y) \rightarrow I(x)$ canceled later cancel 13

Superposition of 4,14: $I(y).(y.I(x.y)) \rightarrow_4 I(x.y)$ and $I(y).(y.I(x.y)) \rightarrow_{14} I(y).I(x)$.
Adopt new rule

15. $I(x.y) \rightarrow I(y).I(x)$ cancel 14

The rewrite system is complete now:

1.	e.x	→ x
2.	I(x).x	→ e
3.	(x.y).z	→ x.(y.z)
4.	I(x).(x.z)	→ z
8.	y.e	→ y
9.	I(I(y))	→ y
10.	I(e)	→ e
11.	y.I(y)	→ e
12.	y.(I(y).x)	→ x
15.	I(x.y)	→ I(y).I(x)

Table 12

This completion procedure by hand was naive, since we were not very systematic in searching for critical pairs, and especially since we were guided by an intuitive sense only of what direction to adopt when generating a new rule. In most cases there was no other possibility (e.g. at 4: $z \rightarrow I(x).(x.z)$ is not a reduction rule due to the restriction that the LHS is not a single variable), but in

case 15 the other direction was at least as plausible, as it is even length-decreasing. However, the other direction I(y).I(x) → I(x.y) would have led to disastrous complications (described in KNUTH & BENDIX [70]).

The problem of what direction to choose is solved in the actual Knuth-Bendix algorithm and its variants by preordaining a *monotonic well-founded partial ordering* > on the terms. ('Monotonic' means: $t > s \Rightarrow \mathbb{C}[t] > \mathbb{C}[s]$ for all contexts $\mathbb{C}[\ ]$; 'well-founded' means that there is no infinite descending sequence $t > t' > t'' > \ldots$.) Then we only generate a new rule $t \to s$ if $t > s$. This guarantees that in each stage the reduction rules generated thus far, are SN. If the procedure of generating new rules terminates succesfully, then we have also a confluent set of reduction rules, basically by the following theorem:

4.1. THEOREM (KNUTH & BENDIX [70]). *Let the TRS* R *be SN. Then* R *is confluent iff each critical pair has a common reduct.*

For the example above, leading to the ten rules in Table 12 (which occur also in KNUTH & BENDIX [70]) one can prove SN by a 'Knuth-Bendix ordering' (not treated here) or by the recursive path ordering explained in Section 3. (In fact we need the extended lexicographic version of Remark 3.11 (iii), due to the presence of the associativity rule.) Further, Knuth and Bendix proved that CR follows because all the critical pairs have a common reduct; e.g.:

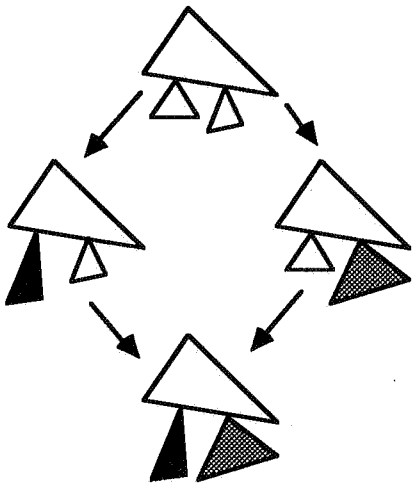
$$\begin{array}{lll} I(y.I(y)) & \to_{15} & I(I(y)).I(y) \\ & & \downarrow_{9} \\ \downarrow_{11} & & y.I(y) \\ & & \downarrow_{11} \\ I(e) & \to_{10} & e \end{array}$$

(As the finitely many rules generate only finitely many critical pairs it is not hard to check this for Table 12.)

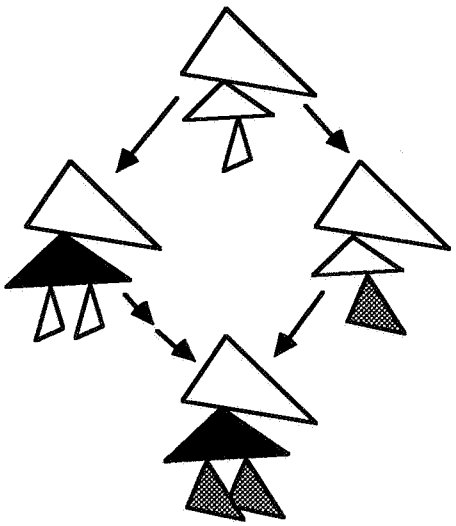
Actually, Theorem 4.1 is a corollary of Newman's Lemma and a theorem of Huet who eliminated the assumption of SN:

4.2. THEOREM (HUET [80]). *A TRS is weakly confluent iff each critical pair of terms has a common reduct.*

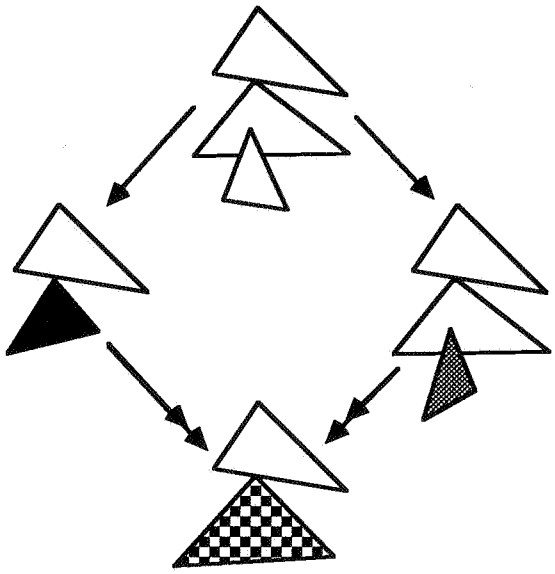
Since the proof (also in LE CHENADEC [86]) gives a good insight in what happens in terms, we will spend some words (and figures) on it. In checking weak confluence, we have two 'diverging' one step reductions given. The two redexes involved in those steps can be in three cases, as in Figure 11. In the first two cases (disjoint redexes and nested redexes) there is no problem to find 'converging' reductions as required for weak confluence. The only problem is the third case, overlapping redexes; but there the hypothesis that each critical pair of terms has a common reduct, does the work.



(a) *disjoint redexes*



(b) *nested redexes*



(c) *overlapping redexes*

Figure 11 (after LE CHENADEC [86])

Knuth and Bendix gave an algorithm to do what was done above by hand, to compute a complete TRS from the given set of equations E - if possible. (The algorithm may fail.) There are several versions around at present; for a correctness proof of some versions see HUET [81]. We now give a simple version, as in DERSHOWITZ [85a], in Table 13.

Knuth-Bendix completion algorithm

The procedure has as input

- *a finite set* R *of rules*
- *a finite set* E *of equations*
- *a program to compute a well-founded monotonic ordering* > *on the terms.*

The reduction rules in R *are 'sound' w.r.t.* E.

The critical pairs <t,s> *of* R *are present in* E *as equations* t = s.

The procedure now generates new rules, again sound w.r.t. E.

==========

Repeat while E is not empty. If E is empty, we have successful termination.

(1) Remove an equation t = s (or s = t) from E such that t > s. If such an equation does not exist: failure.

(2) Add the rule t → s to R.

(3) Use t → s (and other rules in R) to normalize the RHSs of the present rules.

(4) Extend E with all critical pairs in R caused by t → s.

(5) Remove all old rules from R whose LHS contains an instance of t.

(6) Use R to normalize both sides of equations in E.
Remove each equation that becomes an identical equation.

Table 13

5. Regular Term Rewriting Systems

We will now consider a restricted (but for many purposes sufficiently large) class of TRSs with pleasant properties: the regular TRSs.

5.1. DEFINITION. A TRS R is *regular* if the reduction rules of R are *left-linear* and there are *no critical pairs* .

A reduction rule $t \rightarrow s$ is left-linear if t is linear, i.e. no variable occurs twice or more in t. E.g. the rule $D(x,x) \rightarrow E$ is not left-linear; nor is the rule if x then y else $y \rightarrow y$. A TRS R without critical pairs is also called *non-ambiguous* or *non-overlapping*.

One problem with non-left-linear rules is that their application requires a test for syntactic equality of the arguments substituted for the variables occurring more than once. As terms may be very large, this may be very laborious. Another problem is that the presence of non-left-linear rules may destroy the CR property.

5.2. EXAMPLE. Let R consist of the rules $D(x,x) \rightarrow E$, $C(x) \rightarrow D(x,C(x))$, $A \rightarrow C(A)$. Then R is WCR, but not CR; for, we have reductions $C(A) \twoheadrightarrow E$ and $C(A) \twoheadrightarrow C(E)$ but C(E), E have no common reduct. There are no critical pairs in R. Hence, in view of our later theorem stating that regular TRSs are confluent, the non-confluence of R is caused by the non-left-linear rule $D(x,x) \rightarrow E$.

We will now be somewhat more precise about the definition of 'no critical pairs' or 'non-ambiguous'. Let R be the TRS as in Table 14:

r_1	$F(G(x, S(0)), y, H(z))$	$\rightarrow x$
r_2	$G(x, S(S(0)))$	$\rightarrow 0$
r_3	$P(G(x, S(0)))$	$\rightarrow S(0)$

Table 14

Call the context $F(G(\Box, S(0)), \Box, H(\Box))$ the *pattern* of rule r_1. (Earlier, we defined a context as a term with exactly one hole $\Box$, but it is clear what a context with more holes is.) In tree form the pattern is the shaded area as in Figure 12.

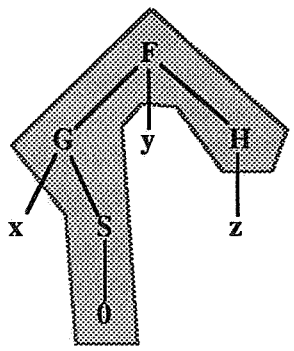


Figure 12

For a left-linear rule it is only its pattern that 'counts'.

The TRS R in Table 14 has the property that *in no term patterns can overlap*, i.e. R has the non-overlapping or non-ambiguity property. The following figure (13) shows a term in R with all

patterns indicated, and indeed they do not overlap.

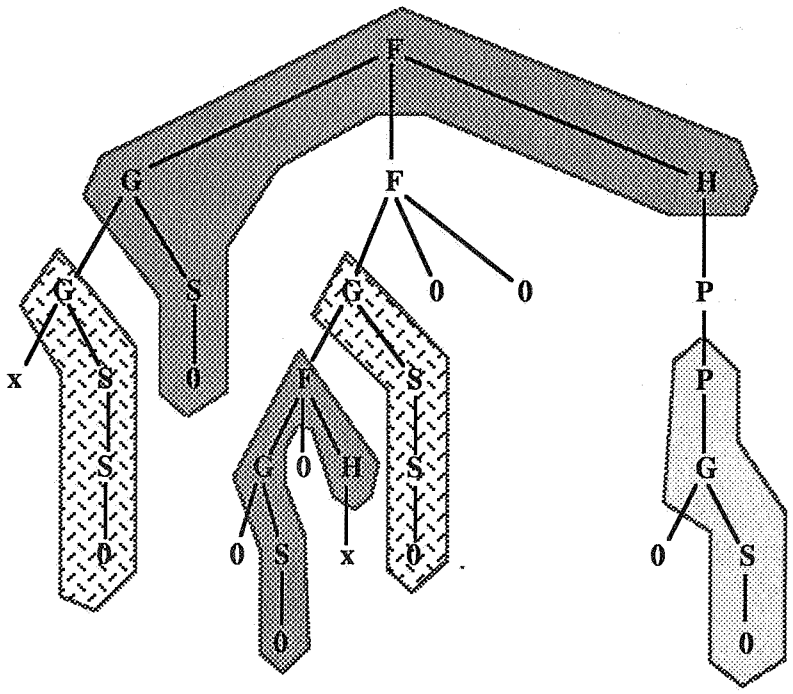
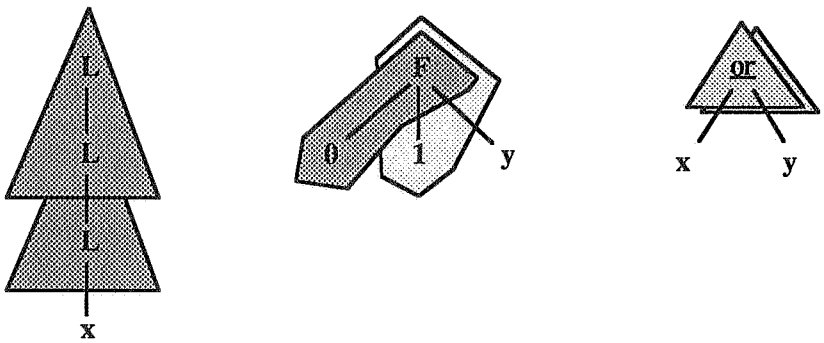


Figure 13

Overlap can already occur in one rule, e.g. in the rule $L(L(x)) \rightarrow 0$; see Figure 14a. An overlap at the root (of the tree corresponding to a term), arising from the rules $F(0,x,y) \rightarrow 0$, $F(x,1,y) \rightarrow 1$, is shown in Figure 14b. Another overlap at the root, arising from the rules for the non-deterministic $\underline{or}$: $\underline{or}(x,y) \rightarrow x$, $\underline{or}(x,y) \rightarrow y$, is shown in Figure 14c.



(a) (b) (c)

Figure 14

We now have the following important fact:

5.3. THEOREM. *Regular TRSs are confluent.*

The earliest proof is probably that of ROSEN [73]; but earlier proofs of the confluence of CL (Combinatory Logic), work just as well for regular TRSs in general. The theorem is also a special case of a theorem of Huet, for which we need a definition first:

5.4. DEFINITION. (Parallel reduction) $t \twoheadrightarrow_{||} s$ if $t \twoheadrightarrow s$ via a reduction of disjoint redexes.

5.5. THEOREM (HUET [80]). *Let* R *be a left-linear TRS. Suppose for every critical pair* <t,s> *we have* $t \twoheadrightarrow_{||} s$. *Then* $\twoheadrightarrow_{||}$ *is strongly confluent, hence* R *is confluent.*

(For the definition of 'strongly confluent' see Ex. 1.7 (10) above.)

5.6. EXAMPLES. (i) Combinatory Logic (Table 4) has rule patterns as in Figure 15; they cannot overlap. As CL is left-linear, it is therefore regular and hence confluent.

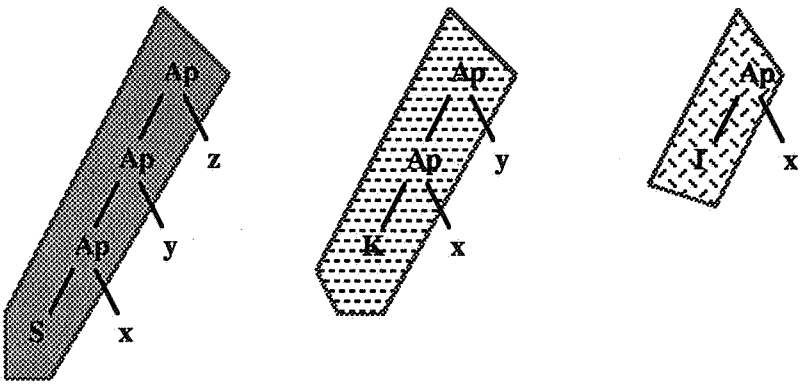


Figure 15

(ii) SKIM, in Table 5, is regular. Likewise for the TRSs CL with test for equality, binary or applicative, in Tables 6,7 respectively. Also Weak Categorical Combinatory Logic in Table 8 is regular.

(iii) A *Recursive Program Scheme (RPS)* is a TRS with

- a finite set of function symbols $\mathcal{F} = \{F_1,...,F_n\}$ (the 'unknown' functions), where F_i has arity $m_i \geq 0$ (i = 1,...,n), and
- a finite set $\mathcal{G} = \{G_1,...,G_k\}$ (the 'known' or 'basic' functions), disjoint from $\mathcal{F}$, where G_j has arity $p_j \geq 0$ (j = 1,...,k).
- The reduction rules of R have the form

$$F_i(x_1,...,x_{m_i}) \rightarrow t_i \qquad (i = 1,...,n)$$

where all the displayed variables are pairwise different and where t_i is an arbitrary term built

from operators in $\mathcal{F},\mathcal{G}$ and the displayed variables. For each F_i (i = 1,...,n) there is exactly one rule.

Example: $F_1(x) \rightarrow G_1(x,F_1(x),F_2(x,x))$
$F_2(x,y) \rightarrow G_2(F_2(x,x),F_1(G_3))$

Every RPS is regular, hence confluent.

Apart from confluence, many interesting facts can be proved for regular TRSs.

5.7. DEFINITION. (i) A TRS is *non-erasing* (NE) if in every rule $t \rightarrow s$ the same variables occur in t and in s. (E.g. CL is not NE, due to the rule $Kxy \rightarrow x$.)
(ii) A TRS is *weakly innermost normalizing* (WIN) if every term has a normal form which can be reached by an *innermost* reduction. (In an innermost reduction a redex may only be 'contracted' if it contains no proper subredexes.)

5.8. THEOREM. *Let* R *be regular and NE. Then:* R *is WN iff* R *is SN.*

5.9. THEOREM (O'DONNELL [77]). *Let* R *be a regular TRS. Then:* R *is WIN iff* R *is SN.*

The last two theorems can be refined to terms: call a term WN if it has a normal form, SN if it has no infinite reductions, WIN if it has a normal form reachable by an innermost reduction. The 'local' version of Theorem 5.8 then says that for a term in a regular, non-erasing TRS the properties WN and SN coincide. Likewise there is a local version of Theorem 5.9. Thus, if in CL a term can be normalized via an innermost reduction, all its reductions are finite.

5.10. REMARK. STSs (Semi-Thue Systems), viewed as TRSs as explained in 2.1.5 (1), are always non-erasing (since LHS and RHS of every rule end in x, in their TRS version). Also, if there are no critical pairs in the STS, it is regular in the present sense of Definition 5.1 (not in the sense of formal language theory or STS theory). So if a STS has no critical pairs, the properties SN and WN coincide. This rather trivial observation could have been more easily made by noting that for a STS without critical pairs the property WCR^1 holds, as defined in Ex. 1.7 (15), whence WN $\Leftrightarrow$ SN.

6. Reduction strategies for regular Term Rewriting Systems

Terms in a TRS may have a normal form as well as admitting infinite reductions. So, if we are interested in finding normal forms, we should have some strategy at our disposal telling us what redex to contract in order to achieve that desired result. We will in this section present some strategies which are guaranteed to find the normal form of a term whenever such a normal form exists. We will adopt the restriction to regular TRSs; for general TRSs there does not seem to be

any result about the existence of 'good' reduction strategies.

The strategies below will be of two kinds: one step strategies (which point in each reduction step to just one redex as the one to contract) and many step strategies (in which a set of redexes is contracted simultaneously). Of course all strategies must be computable.

Apart from the objective of finding a normal form, we will consider the objective of finding a 'best possible' reduction even if the term at hand does not have a normal form.

6.1. DEFINITION. (i) If R is a TRS, a *one step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: Ter(R) $\to$ Ter(R) such that

(1) $t \equiv \mathbb{F}(t)$ if t is a normal form,

(2) $t \to \mathbb{F}(t)$ else.

(ii) A *many step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: Ter(R) $\to$ Ter(R) such that

(1) $t \equiv \mathbb{F}(t)$ if t is a normal form,

(2) $t \to^+ \mathbb{F}(t)$ else.

Here $\to^+$ is the transitive (but not reflexive) closure of $\to$.

6.2. DEFINITION. (i) A reduction strategy (one step or many step) $\mathbb{F}$ for R is *normalizing* if for each term t in R having a normal form, the sequence $\{\mathbb{F}^n(t) \mid n \geq 0\}$ contains a normal form.

(ii) $\mathbb{F}$ is *cofinal* if for each t the sequence $\{\mathbb{F}^n(t) \mid n \geq 0\}$ is cofinal in $\mathcal{G}(t)$, the reduction graph of t. (See Ex.1.7 (13) for 'cofinal' and see Figure 16.)

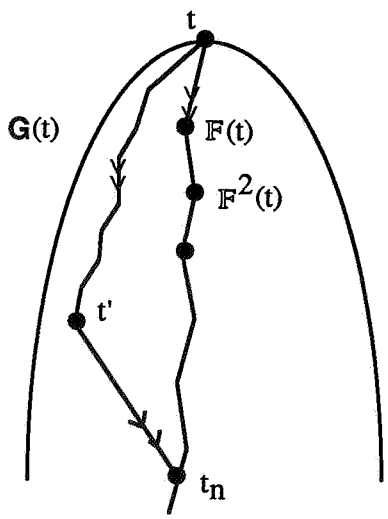


Figure 16

A normalizing reduction strategy is good, but a cofinal one is even better: it finds, when applied on term t, the best possible reduction sequence starting from t (or rather, a best possible) in the following sense. Consider a reduction $t \to s$ as a gain in information; thus normal forms have maximum information. In case there is no normal form in $\mathcal{G}(t)$, one can still consider infinite reductions as developing more and more information. Now the cofinal reductions $t \equiv t_0 \to t_1 \to t_2 \to \ldots$ are optimal since for every t' in $\mathcal{G}(t)$ they contain a t_n with information content no less than

that of t' (since t' $\twoheadrightarrow$ t_n for some t_n, by definition of 'cofinal'). In a sense, a cofinal reduction plays the role of a kind of 'infinite normal form'. See e.g. BERRY & LÉVY [79] and BOUDOL [85], where spaces of finite and infinite reductions modulo the so-called permutation equivalence are studied; this give rise to cpo's or even complete lattices where the bottom point corresponds to the empty reduction of t, i.e. to t itself, and the top point corresponds to the normal form (or rather the equivalence clas of reductions to the normal form), if it exists, and otherwise to the equivalence class of cofinal reductions.

We now present some well-known reduction strategies.

6.3. DEFINITION. (i) The *leftmost-innermost* (one step) strategy is the strategy in which in each step the leftmost of the minimal or innermost redexes is contracted (reduced).
(ii) The *parallel-innermost* (many step) strategy reduces simultaneously all innermost redexes. Since these are pairwise disjoint, this is no problem.
(iii) The *leftmost-outermost* (one step) strategy: in each step the leftmost redex of the maximal (or outermost) redexes is reduced.
(iv) The *parallel-outermost* (many step) strategy reduces simultaneously all maximal redexes; since these are pairwise disjoint, this is no problem.
(v) The *full substitution rule* (or *Kleene reduction*, or *Gross-Knuth reduction*): this is a many step strategy in which all redexes are simultaneously reduced.

Strategies (i)-(iv) are well-defined for general TRSs. Strategy (v) is only defined for regular TRSs, since for a general TRS it is not possible to define an unequivocal result of simultaneous reduction of a set of possibly nested redexes.

It is worth-while to distinguish a class of regular TRSs as follows:

6.4. DEFINITION. A regular TRS is *left-normal* if in every reduction rule t $\rightarrow$ s the constant and function symbols in the LHS t precede (in the linear term notation) the variables.

6.4.1. EXAMPLE. (i) CL (Combinatory Logic) is left-normal. (ii) RPSs (Recursive Program Schemes) as defined in Examples 5.6 (iii) are all left-normal.

We can now summarize some of the main properties of the reduction strategies in Definition 6.3; see Table 14.

	regular TRSs	regular left-normal TRSs
leftmost-innermost	-	-
parallel-innermost	-	-
leftmost-outermost	-	+
parallel-outermost	+	+
full substitution	++	++

Table 14

Here '-' means: in general not normalizing, hence in general not cofinal; '+' means normalizing but not cofinal, '++' means cofinal, hence normalizing.

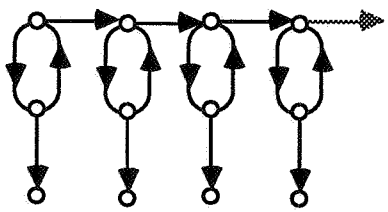
An example showing that the leftmost-outermost strategy is not normalizing in general, is given in HUET & LÉVY [79]: take the regular TRS $\{F(x,B) \to D,\ A \to B,\ C \to C\}$ and consider the term F(C,A). This term has a normal form, D. But the leftmost-outermost reduction is cyclic: $F(C,A) \to F(C,A)$, since it contracts the leftmost-outermost redex C.

Proofs that leftmost-outermost reduction is normalizing for left-normal regular TRSs and that parallel-outermost reductions is normalizing for all regular TRSs can be found in O'DONNELL [77]. The latter fact is also proved in BERGSTRA & KLOP [86] (Appendix).

An example showing that parallel-outermost reduction need not be cofinal, can be found in CL (in Table 4). Namely, define the term I_t as SKt, and check that $I_t x \twoheadrightarrow x$. Furthermore, define the term Ω_t as $SI_tI_t(SI_tI_t)$. Now the parallel-outermost strategy, applied on Ω_{II}, yields a cyclic reduction sequence $\Omega_{II} \twoheadrightarrow \Omega_{II}$ which is not cofinal since $\Omega_{II} \twoheadrightarrow \Omega_I$ but not $\Omega_I \twoheadrightarrow \Omega_{II}$.

As said in the Introduction, this tutorial can only present an initial part of the theory of TRSs and we have to stop short of some very interesting work about sequentiality of TRSs (in HUET & LÉVY [79]). Huet and Lévy prove that every term t in a regular TRS has a *needed* redex, that is a redex which has to be contracted in every reduction to the normal form of t. Unfortunately, for regular TRSs in general there is no algorithm to pick out such a needed redex. But Huet and Lévy go on and define a class of *strongly sequential* TRSs where a needed redex can be found efficiently; and strong sequentiality is a decidable property. Thus for strongly sequential regular TRSs there is a normalizing reduction strategy.

ANSWER TO PUZZLE 1.6.1 (by R. van Glabbeek, Centre for Mathematics and Computer Science, Amsterdam):



ANSWER TO PUZZLE 2.1.6 (by G.J. Akkerman, Free University Amsterdam):

$R = \{F(a, F(x,y)) \to F(x, F(x, F(b,b)))\}$. R is not SN, as F(a, F(a, F(b,b))) reduces to itself by outermost reduction; on the other hand every term has a normal form, e.g. F(a, F(a, F(b,b))) has, by the innermost reduction, normal form F(b, F(b, F(b,b))).

References

BACHMAIR, L. & DERSHOWITZ, N. (1986). *Commutation, transformation, and termination.* Proc. of 8th International Conference on Automated Deduction (ed. J.H. Siekmann), Oxford, Springer LNCS 230, 5-20.

BACHMAIR, L. & PLAISTED, D.A. (1985). *Associative path orderings.* Proc. of 1st Intern. Conf. on Rewriting

Techniques and Applications (ed. J.-P. Jouannaud), Dijon, Springer LNCS 202, 241-254.

BARENDREGT, H.P. (1981). *The Lambda Calculus, its Syntax and Semantics.* 1st ed. North-Holland 1981, 2nd ed. North-Holland 1984.

BERGSTRA, J.A. & KLOP, J.W. (1984). *Process algebra for synchronous communication.* Information and Control 60 (1/3), 1984, 109-137.

BERGSTRA, J.A. & KLOP, J.W. (1985). *Algebra of communicating processes with abstraction.* TCS 37 (1), 1985, 171-199.

BERGSTRA, J.A. & KLOP, J.W. (1986). *Conditional rewrite rules: confluence and termination.* JCSS Vol.32, No.3, 1986, 323-362.

BERRY, G. & LÉVY, J.-J. (1979) *Minimal and optimal computations of recursive programs.* JACM, 26 (1979), 148-175.

BOUDOL, G. (1985). *Computational semantics of term rewriting systems.* In: Algebraic methods in semantics (eds. M. Nivat and J.C. Reynolds), Cambridge University Press 1985, 169-236.

CURIEN, P.-L. (1986). *Categorical combinators, sequential algorithms and functional programming.* Research Notes in Theoretical Computer Science, Pitman, London 1986.

DERSHOWITZ, N. (1979). *A note on simplification orderings.* Information Processing Letters, Vol.9, No.5, 1979, 212-215.

DERSHOWITZ, N. (1981). *Termination of linear rewriting systems.* Proc. 8th ICALP (Eds. S. Even and O. Kariv), Springer LNCS 115, 448-458.

DERSHOWITZ, N. (1985). *Termination.* Proc. of 1st Intern. Conf. on Rewriting Techniques and Applications (ed. J.-P. Jouannaud), Dijon, Springer LNCS 202, 180-224.

DERSHOWITZ, N. (1985a). *Computing with rewrite systems.* Information and Control 65 (1985), 122-157.

DAUCHET, M. & TISON, S. (1984). *Decidability of confluence for ground term rewriting systems.* Report, Université de Lille I.

GANZINGER, H. & GIEGERICH, R. (1987). *A note on termination in combinations of heterogeneous term rewriting systems.* Bulletin of the EATCS (European Association for Theoretical Computer Science), No.31, Febr. 1987, 22-28.

HINDLEY, J.R. (1964). *The Church-Rosser property and a result in combinatory logic.* Ph.D. Thesis, Univ. Newcastle-upon-Tyne, 1964.

HINDLEY, J.R. & SELDIN, J.P. (1986). *Introduction to Combinators and λ-Calculus.* London Mathematical Society Student Texts, Nr.1, Cambridge University Press 1986.

HUET, G. (1980). *Confluent reductions: Abstract properties and applications to term rewriting systems.* JACM, Vol.27, No.4 (1980), 797-821.

HUET, G. (1981). *A complete proof of correctness of the Knuth-Bendix completion algorithm.* JCSS 23 (1981), 11-21.

HUET, G. & LANKFORD, D.S. (1978). *On the uniform halting problem for term rewriting systems.* Rapport Laboria 283, IRIA, 1978.

HUET, G. & LÉVY, J.-J. (1979). *Call-by-need computations in non-ambiguous linear term rewriting systems.* Rapport INRIA nr.359.

HUET, G. & OPPEN, D.C. (1980). *Equations and rewrite rules: A survey.* In: Formal Language Theory: Perspectives and Open Problems (ed. R. Book), Academic Press, 1980, 349-405.

JANTZEN, M. (1986). *Confluent string rewriting and congruences.* Bull. of the EATCS (European Association for Theoretical Computer Science), Nr.28, Febr.1986, 52-72.

KAPUR, D. & NARENDRAN, P. (1985). *A finite Thue system with decidable word problem and without equivalent finite canonical system.* TCS 35 (1985), 337-344.

KLOP, J.W. (1980). *Combinatory Reduction Systems.* Mathematical Centre Tracts Nr.127, Centre for Mathematics and Computer Science, Amsterdam.

KNUTH, D.E. & BENDIX, P.B. (1970). *Simple word problems in universal algebras.* In: Computational Problems in Abstract Algebra (ed. J. Leech), Pergamon Press, 1970, 263-297.

LE CHENADEC, P. (1986). *Canonical forms in finitely presented algebras.* Research Notes in Theoretical Computer Science, Pitman, London 1986.

NEDERPELT, R.P. (1973). *Strong normalization for a typed lambda calculus with lambda structured types.* Ph.D. Thesis, Eindhoven 1973.

NEWMAN, M.H.A. (1942). *On theories with a combinatorial definition of "equivalence".* Annals of Math. 43, Nr.2 (1942), 223-243.

O'DONNELL, M.J. (1977). *Computing in systems described by equations.* Springer LNCS 58.

PLAISTED, D.A. (1985). *Semantic confluence tests and completion methods.* Information and Control 65 (1985), 182-215.

PUEL, L. (1986). *Using unavoidable sets of trees to generalize Kruskal's theorem.* Rapport de Récherche du LIENS (Laboratoire d'Informatique de l'Ecole Normale Supérieure), Paris.

ROSEN, B.K. (1973). *Tree-manipulating systems and Church-Rosser theorems.* JACM, Vol.20 (1973), 160-187.

STAPLES, J. (1975). *Church-Rosser theorems for replacement systems.* In: Algebra and Logic (ed. J. Crosley), Springer Lecture Notes in Mathematics 450, 291-307.

TOYAMA, Y. (1986). *Counterexamples to terminating for the direct sum of Term Rewriting Systems,* preprint NTT Electrical Communication Laboratories, Tokyo, to appear in IPL.

TOYAMA, Y. (1987). *On the Church-Rosser property for the direct sum of term rewriting systems.* JACM, Vol.34, No.1, 1987, 128-143.

TURNER, D.A. (1979). *A new implementation technique for applicative languages.* Software Practice and Experience, Vol.9, 1979, 31-49.

WIEDIJK, F. (1987). *Normal forms in left-linear term rewrite systems.* Report Univ. of Amsterdam, 1987.