Term Rewriting Systems

J. W. Klop[1]

Contents

Abstract

Term Rewriting Systems play an important role in various areas, such as abstract data type specifications, implementations of functional programming languages and automated deduction. In this chapter we introduce several of the basic concepts and facts for TRS's. Specifically, we discuss Abstract Reduction Systems; general Term Rewriting Systems including an account of Knuth-Bendix completion and (E-)unification; orthogonal TRS's and reduction strategies; strongly sequential orthogonal TRS's. The paper concludes with a discussion of conditional term rewriting systems. The emphasis throughout the chapter is on providing information of a syntactic nature.

[1] Research partially supported by ESPRIT project 432: Meteor (until Sept. 1989) and ESPRIT BRA projects 3020: Integration and 3074: Semagraph (since July 1989).

Introduction

The concept of a Term Rewriting System (TRS) is paradigmatic for the study of computational procedures. Already half a century ago, the λ-calculus, probably the most well-known Term Rewriting System, played a crucial role in mathematical logic with respect to formalizing the notion of computability; much later the same TRS figured in the fundamental work of Scott, Plotkin and others leading to a break-through in the denotational semantics of programming languages. More recently, the related system of Combinatory Logic was shown to be a very fruitful tool for the implementation of functional languages. Even more recently another related family of TRS's, that of Categorical Combinatory Logic, has emerged, yielding a remarkable connection between concepts from category theory and elementary steps in machine computations.

Term rewriting systems are attractive because of their simple syntax and semantics—at least those TRS's that do not involve bound variables such as λ-calculus, but involve the rewriting of terms from a first order language. This simplicity facilitates a satisfactory mathematical analysis. On the other hand they provide a natural medium for implementing computations, and in principle even for parallel computations. This feature makes TRS's interesting for the design of parallel reduction machines.

Another field where TRS's play a fundamental role concerns the analysis and implementation of abstract data type specifications (consistency properties, computability theory, decidability of word problems, theorem proving).

The aim of the present paper is to give an introduction to several key concepts in the theory of term rewriting, providing where possible some of the details. At various places some 'exercises' are included. These contain additional information for which proofs are relatively easy; they are not primarily meant to have an educational purpose, if only because the distribution of the exercises is not very uniform.

The present introduction starts at a level of 'rewriting' which is as abstract as possible and proceeds by considering term rewriting systems which have ever more 'structure'. Thus we start with Abstract Reduction Systems, which are no more than sets equipped with some binary ('rewrite') relations. A number of basic properties and facts can already be stated on this level.

Subsequently, the abstract reductions are specialized to reductions (rewritings) of terms. For such general Term Rewriting Systems a key issue is to prove the termination property; we present one of the major and most powerful termination proof methods, recursive path orderings, in a new formulation designed to facilitate human understanding (rather than practical

implementation). Proving termination is of great importance in the area of Knuth-Bendix completions. Here one is concerned, given an equational specification, to construct a TRS which is both confluent and terminating and which proves the same equations as the original specification. If the construction is successful, it yields a positive solution to the validity problem of the original equational specification. (Nowadays there are also several other applications of Knuth-Bendix-like completion methods, such as 'inductionless induction' and 'computing with equations'. For a survey of such applications we refer to Dershowitz & Jouannaud [90].)

Also in Chapter 1, we explain the basic ideas of Knuth-Bendix completion together with an interesting recent 'abstract' approach of Bachmair, Dershowitz & Hsiang [86] to prove the correctness of Knuth-Bendix completion algorithms. We also present an elegant unification algorithm, and likewise for 'E-unification'.

In Chapter 2 we impose more 'structure' on TRS's, in the form of an 'orthogonality' requirement (non-ambiguity and left-linearity). For such orthogonal TRS's a sizeable amount of theory has been developed, both syntactically and semantically. Here we will almost exclusively be concerned with the syntactical aspects; for semantical aspects we refer to Boudol [85], Berry & Lévy [79], Guessarian [81]. Basic theorems (confluence, the Parallel Moves Lemma, Church's theorem, O'Donnell's theorem) are presented, where possible with some proof sketch. Also in this section we survey the most important facts concerning reduction strategies for orthogonal TRS's, strategies aiming at finding normal forms whenever possible. Chapter 2 concludes with an explanation of the beautiful theory of Huet & Lévy [79] of (strongly) sequential TRS's. Such TRS's possess a 'good' reduction strategy.

In the final chapter (3) we consider TRS's with conditional rewrite rules.

Some important topics have not found their way into this introduction. Most notable are: rewriting modulo a set of equations, proof-by-consistency procedures, and graph rewriting. For information about the first two we refer to Bachmair [88] and Dershowitz & Jouannaud [90], for graph rewriting one may consult Barendregt e.a. [87].

This chapter is an extension of the short survey/tutorial Klop [87]; also most of the material in Klop [85] is included here.

1 Abstract Reduction Systems

Many of the basic definitions for and properties of TRS's (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 section about Abstract Reduction Systems. Moreover, the concepts and properties of Abstract Reduction Systems also apply to other rewrite systems than TRS's, such as string rewrite systems (Thue systems), tree rewrite systems, graph grammars. First we present a sequence of simple definitions.

Definition 1.0.1.

1. An *Abstract Reduction System* (ARS) is a structure $\mathcal{A} = \langle A, (\to_\alpha)_{\alpha \in I} \rangle$ consisting of a set A and a sequence of binary relations $\to_\alpha$ on A, also called reduction or rewrite relations. Sometimes we will refer to $\to_\alpha$ as α. In the case of just one reduction relation, we also use $\to$ without more. (An ARS with just one reduction relation is called 'replacement system' in Staples [75], and a 'transformation system' in Jantzen [88].) If for $a, b \in A$ we have $(a, b) \in \to_\alpha$, we write $a \to_\alpha b$ and call b a one-step (α-)*reduct* of a.
2. The transitive reflexive closure of $\to_\alpha$ is written as $\twoheadrightarrow_\alpha$. (More customary is the notation $\to_\alpha^*$, 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' $a \equiv a_0 \to_\alpha a_1 \to_\alpha \cdots \to_\alpha a_n \equiv b$. Here $\equiv$ denotes identity of elements of A. The element b is called an (α-)reduct of a. The equivalence relation generated by $\to_\alpha$ is $=_\alpha$, also called the *convertibility* relation generated by $\to_\alpha$. The reflexive closure of $\to_\alpha$ is $\to_\alpha^=$. The transitive closure of $\to_\alpha$ is $\to_\alpha^+$. The converse relation of $\to_\alpha$ is $\leftarrow_\alpha$ or $\to_\alpha^{-1}$. The union $\to_\alpha \cup \to_\beta$ is denoted by $\to_{\alpha\beta}$. The composition $\to_\alpha \circ \to_\beta$ is defined by: $a \to_\alpha \circ \to_\beta b$ if $a \to_\alpha c \to_\beta b$ for some $c \in A$.
3. If α, β are reduction relations on A, we say that α *commutes weakly* with β if the diagram of Figure 1.1a holds, i.e. if $\forall a, b, c \in A \, \exists d \in A \, (c \leftarrow_\beta a \to_\alpha b \Rightarrow c \twoheadrightarrow_\alpha d \twoheadleftarrow_\beta b)$, or in a shorter notation: $\leftarrow_\beta \circ \to_\alpha \subseteq \twoheadrightarrow_\alpha \circ \twoheadleftarrow_\beta$. Further, α *commutes* with β if $\twoheadrightarrow_\alpha$ and $\twoheadrightarrow_\beta$ commute weakly. (This terminology differs from that of Bachmair & Dershowitz [86], where α commutes with β if $\alpha^{-1} \circ \beta \subseteq \beta^{-1} \circ \alpha$.)
4. The reduction relation $\to$ is called *weakly confluent* or *weakly Church-Rosser* (WCR) if it is weakly self-commuting (see Figure 1.1b), i.e. if $\forall a, b, c \in A \, \exists d \in A \, (c \leftarrow a \to b \Rightarrow c \twoheadrightarrow d \twoheadleftarrow b)$. (The property WCR is also often called 'local confluence', e.g. in Jantzen [88].)
5. $\to$ is *subcommutative* (notation $\mathrm{WCR}^{\leq 1}$) if the diagram in Figure 1.1c holds, i.e. if $\forall a, b, c \in A \, \exists d \in A \, (c \leftarrow a \to b \Rightarrow c \to^= d \leftarrow^= b)$.
6. $\to$ is *confluent* or is *Church-Rosser*, has the Church-Rosser property (CR) if it is self-commuting (see Figure 1.1d), i.e. $\forall a, b, c \in A \, \exists d \in A \, (c \twoheadleftarrow a \twoheadrightarrow b \Rightarrow c \twoheadrightarrow d \twoheadleftarrow b)$. Sometimes (6) is called 'confluent' and the situation as in Proposition 1.0.2(6) 'Church-Rosser'.

Proposition 1.0.2. *The following are equivalent:*

1. $\rightarrow$ *is confluent*
2. $\twoheadrightarrow$ *is weakly confluent*
3. $\twoheadrightarrow$ *is self-commuting*
4. $\twoheadrightarrow$ *is subcommutative*
5. *the diagram in Figure 1.1e holds, i.e.*

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

6. $\forall a, b \in A \, \exists c \in A \, (a = b \Rightarrow a \twoheadrightarrow c \twoheadleftarrow b)$ *(Here '=' is the convertibility relation generated by* $\rightarrow$*. See diagram in Figure 1.1f.)*

Figure 1.1

Definition 1.0.3. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS.

1. 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 $a \in A$.
2. 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.

3. $\mathcal{A}$ (or $\rightarrow$) is *strongly normalizing* (SN) if every reduction sequence $a_0 \rightarrow a_1 \rightarrow \cdots$ 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}.
4. $\mathcal{A}$ (or $\rightarrow$) has the *unique normal form property* (UN) if $\forall a, b \in A (a = b$ & a, b are normal forms $\Rightarrow a \equiv b)$.
5. $\mathcal{A}$ (or $\rightarrow$) has the *normal form property* (NF) if $\forall a, b \in A (a$ is normal form & $a = b \Rightarrow b \twoheadrightarrow a)$.
6. $\mathcal{A}$ (or $\rightarrow$) is *inductive* (Ind) if for every reduction sequence (possibly infinite) $a_0 \rightarrow a_1 \rightarrow \cdots$ there is an $a \in A$ such that $a_n \twoheadrightarrow a$ for all n.
7. $\mathcal{A}$ (or $\rightarrow$) is *increasing* (Inc) if there is a map $|\ |\colon A \rightarrow \mathbb{N}$ 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 $<$.
8. $\mathcal{A}$ (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 [80], FB is called 'locally finite'.)

Exercise 1.0.4. Define: $\mathcal{A}$ (or $\rightarrow$) has the *unique normal form property with respect to reduction* ($\mathrm{UN}^{\rightarrow}$) if $\forall a, b, c \in A (a \twoheadrightarrow b$ & $a \twoheadrightarrow c$ & b, c are normal forms $\Rightarrow b \equiv c)$. Show that UN $\Rightarrow$ $\mathrm{UN}^{\rightarrow}$, but not conversely.

An ARS which is confluent and terminating (CR & SN) is also called *complete* (other terminology: 'canonical' or 'uniquely terminating').

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

Definition 1.0.5. Let $\mathcal{A} = \langle A, \rightarrow_\alpha \rangle$ and $\mathcal{B} = \langle B, \rightarrow_\beta \rangle$ be two ARS's. Then A is a *sub-ARS* of B, notation $\mathcal{A} \subseteq \mathcal{B}$, if:

1. $A \subseteq B$
2. α is the restriction of β to A, i.e. $\forall a, a' \in A\, (a \rightarrow_\beta a' \Leftrightarrow a \rightarrow_\alpha a')$
3. A is closed under β, i.e. $\forall a \in A\, (a \rightarrow_\beta b \Rightarrow b \in A)$.

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

Note that all properties introduced so far (CR, WCR, $\mathrm{WCR}^{\leq 1}$, 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.

Definition 1.0.6. Let $\mathcal{A} = \langle A, \rightarrow \rangle$ 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 $\rightarrow$ restricted to this set of reducts.

We will now collect in one theorem several implications between the various properties of ARS's. The first part (1) 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. Apart from the fundamental implication CR $\Rightarrow$ UN, the most important fact is (2), also known as Newman's Lemma. The property CP ('cofinality property') is defined in Exercise 1.0.8(13) below.

Theorem 1.0.7.

1. *CR* $\Rightarrow$ *NF* $\Rightarrow$ *UN*
2. *SN & WCR* $\Rightarrow$ *CR (Newman's Lemma)*
3. *UN & WN* $\Rightarrow$ *CR*
4. *UN & WN* $\Rightarrow$ *Ind*
5. *Ind & Inc* $\Rightarrow$ *SN*
6. *WCR & WN & Inc* $\Rightarrow$ *SN*
7. *CR* $\Leftrightarrow$ *CP for countable ARS's.*

Most of the proofs of (1)-(7) are easy. For Newman's Lemma a short proof is given in Huet [80]; an alternative proof, illustrating the notion of 'proof ordering', is given in Section 1.5 (Exercise 1.5.4). Proposition (5) is from Nederpelt [73]; (6) is proved in Klop [80a]; for (7) see Exercise 1.0.8(13) below. The propositions in the statement of the theorem (and some more—for these see Exercises 1.0.8) are displayed also in Figure 1.2; 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.)

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 1.3 (given by R. Hindley, see also Huet [80]).

There are several other facts about ARS's 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 of Exercises 1.0.8. For an understanding of the sequel these additional facts are not necessary. Some proofs require the notion of 'multiset ordering', explained in Exercise 1.3.15.

Exercises 1.0.8.

1. (Rosen [73]) If $\langle A, \rightarrow_1, \rightarrow_2 \rangle$ is an ARS such that $\twoheadrightarrow_1 = \twoheadrightarrow_2$ and $\rightarrow_1$ is subcommutative, then $\rightarrow_2$ is confluent.
2. (Hindley [64]) Let $\langle A, (\rightarrow_\alpha)_{\alpha \in I} \rangle$ be an ARS such that for all $\alpha, \beta \in I, \rightarrow_\alpha$ commutes with $\rightarrow_\beta$. (In particular, $\rightarrow_\alpha$ commutes with itself.) Then the

Figure 1.2

Figure 1.3

union $\rightarrow = \mathsf{U}_{\alpha \in I} \rightarrow_\alpha$ is confluent. (This proposition is sometimes referred to as the Lemma of Hindley-Rosen; see e.g. Barendregt [81], Proposition 3.3.5.)

3. (Hindley [64]) Let $\langle A, \rightarrow_1, \rightarrow_2 \rangle$ be an ARS. Suppose: $\forall a, b, c \in A \, \exists d \in A \, (a \rightarrow_1 b \;\&\; a \rightarrow_2 c \Rightarrow b \twoheadrightarrow_2 d \;\&\; c \rightarrow_1^{\equiv} d)$. (See Figure 1.4a.) Then $\rightarrow_1, \rightarrow_2$ commute.
4. (Staples [75]) Let $\langle A, \rightarrow_1, \rightarrow_2 \rangle$ be an ARS. Suppose: $\forall a, b, c \in A \, \exists d \in A \, (a \rightarrow_1 b \;\&\; a \twoheadrightarrow_2 c \Rightarrow b \twoheadrightarrow_2 d \;\&\; c \twoheadrightarrow_1 d)$. (See Figure 1.4b.) Then $\rightarrow_1, \rightarrow_2$ 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 1.4c.) 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 1.4d.) Then $\to_1$ requests $\to_2$.

7. (Staples [75]) Let $\langle A, \to_1, \to_2 \rangle$ be an ARS such that $\to_1$ requests $\to_2$ and $\to_2$ is confluent. 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 $\langle A, \to_1, \to_2 \rangle$ be an ARS where $\to_2$ is a compatible refinement of $\to_1$. Then: $\to_1$ is confluent iff $\to_2$ is confluent.

10. (Huet [80]) DEFINITION: Let $\langle A, \to \rangle$ be an ARS. Then $\to$ is called *strongly confluent* (see Figure 1.4e) if $\forall a,b,c \in A\, \exists d \in A\, (a \to b \;\&\; a \to c \Rightarrow b \twoheadrightarrow d \;\&\; c \to^{=} d)$. Prove that strong confluence implies confluence.

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

12. (Winkler & Buchberger [83]) Let $\langle A, \to, > \rangle$ be an ARS where the 'reduction' relation $>$ is a partial order and SN. (So $>$ is well-founded.) Suppose $a \to b$ implies $a > b$. Then the following are equivalent: (a) $\to$ is confluent, (b) whenever $a \to b$ and $a \to c$, there is a $\to$-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 $\to$ or $\leftarrow$. (See Figure 1.4f.) (Note that this strengthens Newman's Lemma.)

13. (Klop [80a]) Let $\mathcal{A} = \langle A, \to \rangle$ 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 $a \equiv a_0 \to a_1 \to \ldots$ such that $\{a_n \mid n \geq 0\}$ is cofinal in $\mathcal{G}(a)$. Then, for *countable* ARS's: $\mathcal{A}$ is CR $\Leftrightarrow$ $\mathcal{A}$ has CP.

14. Let $\mathcal{A} = \langle A, \to \rangle$ 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 $\mathcal{A} = \langle A, \to_\alpha \rangle, \mathcal{B} =$

Figure 1.4

$\langle B, \to_\beta \rangle$ be ARS's such that $A \subseteq B$. 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 ARS's $\langle A, \to \rangle$: $\forall a, b, c \in A\, \exists d \in A\, (c \leftarrow a \to b \;\&\; b \not\equiv c \Rightarrow c \to d \leftarrow b)$. (See Figure 1.5a.) Prove that WCR^1 & WN $\Rightarrow$ SN, and give a counterexample to the implication $\mathrm{WCR}^{\leq 1}$ & WN $\Rightarrow$ SN.

16. (Bachmair & Dershowitz [86]) Let $\langle A, \to_\alpha, \to_\beta \rangle$ be an ARS such that $\forall a, b, c \in A\, \exists d \in A\, (a \to_\alpha b \to_\beta c \Rightarrow a \to_\beta d \twoheadrightarrow_{\alpha\beta} c)$. (In the terminology of Bachmair & Dershowitz [86]: β *quasi-commutes over* α.) (See Figure 1.5b.) Prove that β/α is SN iff β is SN. (For the definition of β/α, see Exercise 1.0.8(19) below.)

17. (Klop [80a]) Let $\mathcal{A} = \langle A, \to_\alpha \rangle$ and $\mathcal{B} = \langle B, \to_\beta \rangle$ be ARS's. Let $\iota : A \to B$ and $\kappa : B \to A$ be maps such that

    (a) $\kappa(\iota(a)) = a$ for all $a \in A$,
    (b) $\forall a, a' \in A \forall b \in B\, \exists b' \in B\, (b \to_\kappa a \to_\alpha a' \Rightarrow b \to_\beta b' \to_\kappa a')$
    (Reductions in $\mathcal{A}$ can be 'lifted' to $\mathcal{B}$.) See Figure 1.5c.

    Prove that $\mathcal{B}$ is SN implies that $\mathcal{A}$ is SN.

18. (Geser [90]) Let $\langle A, \to_\alpha, \to_\beta \rangle$ be an ARS with two reduction relations α, β such that $\alpha \cup \beta$ is transitive. Then: $\alpha \cup \beta$ is SN $\Leftrightarrow$ α is SN and β is SN. (Hint: use the following infinite version of Ramsey's Theorem, in which for a set S the notation $[S]^2$ is used to denote the set $\{\{a, b\} \mid a, b \in S \;\&\; a \neq b\}$ of two-element subsets of S. Furthermore, $\mathbb{N}$ is the set of natural numbers. THEOREM: *Let* $[\mathbb{N}]^2$ *be partitioned into subsets* X *and* Y. *Then*

Figure 1.5

there is an infinite $A \subseteq \mathbb{N}$ *such that either* $[A]^2 \subseteq X$ *or* $[A]^2 \subseteq Y$.)

19. (Geser [90]) This exercise reformulates and slightly generalizes Exercise 1.0.8(11). Let $\langle A, \to_\alpha, \to_\beta \rangle$ be an ARS. DEFINITION: α/β ("α modulo β") is the reduction relation $\beta^*\alpha\beta^*$. So $a \to_{\alpha/\beta} b$ iff there are c, d such that $a \twoheadrightarrow_\beta c \to_\alpha d \twoheadrightarrow_\beta b$. Note that α is relatively terminating (in the sense of Exercise 1.0.8(11)) iff α/β is SN. DEFINITION: β is called *nonsplitting* (with respect to $\alpha \cup \beta$) if $\forall a, b, c \in A \exists d \in A (a \to_\beta b \;\&\; a \to_{\alpha\cup\beta} c \Rightarrow c \twoheadrightarrow_{\alpha\cup\beta} d \;\&\; b \,(\to_{\alpha\cup\beta})^{\equiv} d)$. Prove: If α/β is SN, α is WCR, and β is non-splitting, then $\alpha \cup \beta$ is confluent.

1.1 Basic notions

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:

1. a countably infinite set of *variables* $x_1, x_2, x_3, \ldots$ also denoted as $x, y, z, x', y', \ldots$
2. a non-empty set of *function symbols* or *operator symbols* $F, G, \ldots$, 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 $\mathrm{Ter}(\Sigma)$ and is defined inductively:

1. $x, y, z, \ldots \in \mathrm{Ter}(\Sigma)$,
2. if F is an n-ary function symbol and $t_1, \ldots, t_n \in \mathrm{Ter}(\Sigma)$ $(n \geq 0)$, then $F(t_1, \ldots, t_n) \in \mathrm{Ter}(\Sigma)$. The t_i $(i = 1, \ldots, n)$ are the arguments of the last term.

Terms not containing a variable are called *ground* terms (also: *closed* terms), and $\mathrm{Ter}_0(\Sigma)$ 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 C[ ]. If $t \in$ Ter(Σ) and t is substituted in $\Box$, the result is $C[t] \in$ Ter(Σ); t is said to be a subterm of $C[t]$, notation $t \subseteq C[t]$. 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.

Example 1.1.1. 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: $A(M(\mathbf{S(0)},0),\mathbf{S(0)})$.

A *substitution* σ is a map from Ter(Σ) to Ter(Σ) which satisfies $\sigma(F(t_1,\ldots,t_n)) = F(\sigma(t_1),\ldots,\sigma(t_n))$ for every n-ary function symbol F (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(Σ). 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:

1. the LHS (left-hand side) t is not a variable,
2. 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* (from 'reducible expression'), more precisely an r-redex. A redex t^σ may be replaced by its '*contractum*' s^σ inside a context C[ ]; this gives rise to *reduction steps* (or one-step rewritings)

$$C[t^\sigma] \to_r C[s^\sigma].$$

We call $\to_r$ the *one-step reduction relation* generated by r. Concatenating reduction steps we have (possibly infinite) *reduction sequences* $t_0 \to t_1 \to t_2 \to \cdots$ or *reductions* for short. If $t_0 \to \cdots \to t_n$ 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 1.0.

Example 1.1.2. Consider Σ as in Example 1.1.1. Let (Σ, R) be the TRS (specifying the natural numbers with addition, multiplication, successor and zero) with reduction rules R given in Table 1.1.
Now $M(S(S(0)),S(S(0))) \twoheadrightarrow S(S(S(S(0))))$, since we have the following reduction:

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 1.1

$$\begin{aligned}
\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{S}(\mathbf{0}))) &\to \mathbf{A}(\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{S}(\mathbf{0}))) \\
&\to S(\mathbf{A}(\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{0}))) \\
&\to S(S(\mathbf{A}(\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{0})),\mathbf{0}))) \\
&\to S(S(\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{S}(\mathbf{0})))) \\
&\to S(S(A(\mathbf{M}(\mathbf{S}(\mathbf{S}(\mathbf{0})),\mathbf{0}),S(S(0))))) \\
&\to S(S(\mathbf{A}(\mathbf{0},\mathbf{S}(\mathbf{S}(\mathbf{0}))))) \\
&\to S(S(S(\mathbf{A}(\mathbf{0},\mathbf{S}(\mathbf{0})))) \\
&\to S(S(S(S(\mathbf{A}(\mathbf{0},\mathbf{0}))))) \\
&\to S(S(S(S(0)))).
\end{aligned}$$

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 $(\mathrm{Ter}(\Sigma),(\to_r)_{r\in R})$. 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 $(\mathrm{Ter}_0(\Sigma),(\to_r)_{r\in R})$ is meant. Via the associated ARS, all notions considered in Section 1.0 (CR, UN, SN, ...) carry over to TRS's.

Example 1.1.3. Let (Σ,R) be the TRS of Example 1.1.2 and consider (Σ,R') where $R' = R \cup \{A(x,y) \to A(y,x)\}$; 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 1.6. The 'bottom' term in that reduction graph is a normal form.

Many-sorted Term Rewriting Systems

TRS's (Σ,R) as we just have defined are sometimes called *homogeneous* (Ganzinger & Giegerich [87]), as they correspond to algebraic data type specifications (by replacing '$\to$' by '$=$' in R) where the signature Σ has

Figure 1.6

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, at least with respect to the theory in this chapter; of course, the extension to the heterogeneous case presents a whole area of new features and problems (see e.g. Ehrig & Mahr [85], Drosten [89] for a treatment of many-sorted specifications and rewriting).

Semi-Thue systems

Semi-Thue Systems (STS's), as defined in Jantzen [88], can be 'viewed' in two ways as TRS's. We demonstrate this by the following:

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 $a(b(a(x))) \to b(a(b(x)))$. Now a reduction step in T, e.g.: $bbabaaa \to bbbabaa$, translates in R to the reduction step $b(b(a(b(a(a(o))))))) \to b(b(b(a(b(a(a(o))))))))$. 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* TRS's, which we will not do here. (See e.g. Bachmair & Plaisted [85] or Plaisted [85].)

Applicative Term Rewriting Systems

In some important TRS's there is a very special binary operator, called *application* (Ap). E.g. Combinatory Logic (CL), based on S, K, I, has the rewrite rules as in Table 1.2. Here S, K, I are constants. Often one uses

$Ap(Ap(Ap(S,x),y),z)$	$\rightarrow$	$Ap(Ap(x,z),Ap(y,z))$
$Ap(Ap(K,x),y)$	$\rightarrow$	x
$Ap(I,x)$	$\rightarrow$	x

Table 1.2

the infix notation $(t \cdot s)$ instead of $Ap(t,s)$, in which case the rewrite rules of CL read as follows:

$((S \cdot x) \cdot y) \cdot z$	$\rightarrow$	$(x \cdot z) \cdot (y \cdot z)$
$(K \cdot x) \cdot y$	$\rightarrow$	x
$I \cdot x$	$\rightarrow$	x

Table 1.3

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:

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

Table 1.4

Note that $xz(yz)$ restores to $(xz)(yz)$, 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 $t_1, \ldots, t_n (n \geq 0)$. But it needs, in the case of S, at least three arguments to use the rewrite rule for

S; e.g.: $St_1t_2t_3t_4t_5t_6 \to t_1t_3(t_2t_3)t_4t_5t_6$.

Example 1.1.4. We have $SII(SII) \to I(SII)(I(SII)) \to SII(I(SII)) \to SII(SII)$. The term $SII(SII)$ has many more reductions, which constitute an interesting reduction graph (see Figure 1.7).

Figure 1.7

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 programming languages. Actually, an extension of CL is used there, called SKIM (for S,K,I-Machine); it is also an applicative TRS (see Table 1.5). Note that this TRS has infinitely many constants: apart from the constants $S, K, \ldots, \underline{eq}$ there is a constant $\underline{n}$ for each $n \in \mathbb{N}$. There are also infinitely many reduction rules, because the last four rules are actually *rule schemes*; e.g. $\underline{plus}\ \underline{n}\ \underline{m} \to \underline{n+m}$ stands for all reduction rules like $\underline{plus}\ \underline{0}\ \underline{0} \to \underline{0}$, $\underline{plus}\ \underline{0}\ \underline{1} \to \underline{1}, \ldots, \underline{plus}\ \underline{37}\ \underline{63} \to \underline{100}, \ldots$. 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:

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

$Sxyz$	$\to$	$xz(yz)$	
Kxy	$\to$	x	
Ix	$\to$	x	
$Cxyz$	$\to$	xzy	
$Bxyz$	$\to$	$x(yz)$	
Yx	$\to$	$x(Yx)$	
$Uz(Pxy)$	$\to$	zxy	
$\underline{cond}\,\underline{true}\,xy$	$\to$	x	
$\underline{cond}\,\underline{false}\,xy$	$\to$	y	
$\underline{plus}\,\underline{n}\,\underline{m}$	$\to$	$\underline{n+m}$	
$\underline{times}\,\underline{n}\,\underline{m}$	$\to$	$\underline{n \cdot m}$	
$\underline{eq}\,\underline{n}\,\underline{n}$	$\to$	$\underline{true}$	
$\underline{eq}\,\underline{n}\,\underline{m}$	$\to$	$\underline{false}$	if $n \neq m$

Table 1.5

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 CL with test for syntactical equality in Table 1.6.

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

Table 1.6

However, some care should be taken: consider the TRS in Table 1.7.

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

Table 1.7

where D is now a *constant* (instead of a binary operator) subject to the rewrite rule, in full notation, $Ap(Ap(D,x),x) \to E$. These two TRS's 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, which plays an important role in implementations of functional languages (see Curien [86] and Hardin [89]):

$Id\,x$	$\rightarrow$	x
$(x \circ y)z$	$\rightarrow$	$x(yz)$
$Fst\,(x,y)$	$\rightarrow$	x
$Snd\,(x,y)$	$\rightarrow$	y
$\langle x,y\rangle z$	$\rightarrow$	(xz,yz)
$App\,(x,y)$	$\rightarrow$	xy
$\Lambda(x)yz$	$\rightarrow$	$x(y,z)$

Table 1.8

Here Id, Fst, Snd, App are constants, $\circ, \langle, \rangle$ and $(\ ,\ )$ are binary function symbols and Λ is a 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.

1.2 Disjoint 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 'disjoint sum' of TRS's:

Definition 1.2.1. Let R_1, R_2 be TRS's. Then the *disjoint 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 disjoint 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 [87b]:

Theorem 1.2.2. *$R_1 \oplus R_2$ 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 [87a] 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:

$$\begin{aligned} f(\underline{or}(0,1), \underline{or}(0,1), \underline{or}(0,1)) &\to f(0, \underline{or}(0,1), \underline{or}(0,1)) \\ &\to f(0,1, \underline{or}(0,1)) \\ &\to f(\underline{or}(0,1), \underline{or}(0,1), \underline{or}(0,1)) \\ &\to \cdots \end{aligned}$$

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 [87a]) shows: R_1 has the eleven rules

$$\begin{aligned} F(4,5,6,x) &\to F(x,x,x,x) \\ F(x,y,z,w) &\to 7 \end{aligned}$$

and R_2 has the three rules

$$\begin{aligned} G(x,x,y) &\to x \\ G(x,y,x) &\to x \\ G(y,x,x) &\to x. \end{aligned}$$

(Similar counterexamples with the additional property of being 'reduced' or 'irreducible'—meaning that both sides of every rule are normal forms with respect to the other rules (see Definition 1.4.18 below for a more accurate definition)—are given in Toyama [87a] and Ganzinger & Giegerich [87].) Now R_1 and R_2 are both complete, but $R_1 \oplus R_2$ is not:

$$\begin{aligned} &F(G(1,2,3), \ G(1,2,3), \ G(1,2,3), \ G(1,2,3)) \ \twoheadrightarrow \\ &F(G(4,4,3), \ G(5,2,5), \ G(1,6,6), \ G(1,2,3)) \ \twoheadrightarrow \\ &F(\ \ 4, \ \ 5, \ \ 6, \ G(1,2,3)) \ \to \\ &F(G(1,2,3), \ G(1,2,3), \ G(1,2,3), \ G(1,2,3)). \end{aligned}$$

Exercise 1.2.3. A simpler counterexample is given in Drosten [89]. Slightly adapted it reads:

$$
\begin{array}{lll}
R_1 & F(0,1,x) \to F(x,x,x) \\
 & F(x,y,z) \to 2 \\
 & 0 \to 2 \\
 & 1 \to 2
\end{array}
$$

and

$$
\begin{array}{lll}
R_2 & D(x,y,y) \to x \\
 & D(x,x,y) \to y.
\end{array}
$$

Now R_1, R_2 are complete; however, their disjoint sum is not. To see this, consider the term $F(M,M,M)$ where $M \equiv D(0,1,1)$ and show that $F(M,M,M)$ has a cyclic reduction.

The last counterexamples involve a non-leftlinear TRS. This is essential, as the following theorem indicates. First we define this concept:

Definition 1.2.4.

1. A reduction rule $t \to s$ is *left-linear* if t is a linear term.
2. A TRS is *left-linear* if all its reduction rules are left-linear.

Theorem 1.2.5. (Toyama, Klop & Barendregt [89]) *Let R_1, R_2 be left-linear TRS's. Then: $R_1 \oplus R_2$ is complete iff R_1 and R_2 are complete.*

Some useful information concerning the inference of SN for $R_1 \oplus R_2$ from the SN property for R_1 and R_2 separately is given in Rusinowitch [87a] and Middeldorp [89b], in terms of 'collapsing' and 'duplicating' rewrite rules.

Definition 1.2.6.

1. A rewrite rule $t \to s$ is a *collapsing* rule (c-rule) if s is a variable.
2. A rewrite rule $t \to s$ is a *duplicating* rule (d-rule) if some variable has more occurrences in s than it has in t.

Example 1.2.7. $F(x,x) \to G(x,x)$ is not a d-rule, but $F(x,x) \to H(x,x,x)$ is. Also $P(x) \to G(x,x)$ is a d-rule.

Theorem 1.2.8. *Let R_1 and R_2 be TRS's both with the property SN.*

1. *If neither R_1 nor R_2 contain c-rules, $R_1 \oplus R_2$ is SN.*
2. *If neither R_1 nor R_2 contain d-rules, $R_1 \oplus R_2$ is SN.*
3. *If one of the TRS's R_1, R_2 contains neither c- nor d-rules, $R_1 \oplus R_2$ is SN.*

Statements (1) and (2) are proved in Rusinowitch [87a]; statement (3) is proved in Middeldorp [89b].

Exercise 1.2.9. Prove that WN is a modular property.

Another useful fact, proved in Middeldorp [89a], is that UN is a modular

property.

Theorem 1.2.10. *$R_1 \oplus R_2$ is UN iff R_1 and R_2 are so.*

The proof of this theorem employs a lemma of independent interest; see the proof sketch in the following exercises.

Exercises 1.2.11. (Middeldorp [90])

1. Let R be a TRS. For $t \in \mathrm{Ter}(R)$, $[t]$ denotes the equivalence class of t with respect to convertibility in R: $[t] = \{t' \mid t =_R t'\}$. Further, $V(t)$ is the set of variables occurring in t. $EV(t)$ is the set of *essential* variables of t, defined as: $\cap_{t' \in [t]} V(t')$.
2. Now let $t(\vec{x}, \vec{y})$ be a term with essential variables $\vec{x} = x_1, \ldots, x_n$ and non-essential variables $\vec{y} = y_1, \ldots, y_m$. Prove that for arbitrary terms $\vec{s} = s_1, \ldots, s_m$ we have $t(\vec{x}, \vec{s}) =_R t(\vec{x}, \vec{y})$.
3. Let R have the property UN (unique normal forms). Show that a normal form has only essential variables.
4. Let R contain a ground term (i.e., R contains a constant symbol). Show that every convertibility class $[t]$ contains a term s having only essential variables.
5. Let R have the property UN and contain a ground term. Show that there is a choice function φ from $\{[t] \mid t \in \mathrm{Ter}(R)\}$ to $\mathrm{Ter}(R)$, selecting from each equivalence class $[t]$ a term such that
   (a) $\varphi([t]) \in [t]$;
   (b) if $[t]$ contains a normal form t', then $\phi([t]) \equiv t'$;
   (c) $\varphi([t])$ contains only essential variables.
6. LEMMA. *Let R be a TRS with property UN and containing a ground term. Then R can be extended to a confluent TRS R' with the same alphabet, the same convertibility and the same normal forms.*
   Prove the lemma by considering R', originating from R by adding the set of reduction rules $\{t \to \varphi([t]) \mid t \in \mathrm{Ter}(R)\ \&\ t \not\equiv \varphi([t])\}$. (Note that the $t \to \varphi([t])$ are added as reduction rules, not merely as reduction steps.)
7. LEMMA. *Let R be a TRS with property UN. Then R can be extended to a confluent TRS R' with the same convertibility and the same normal forms.*
   Prove the lemma as follows: in case R contains a constant, (6) applies; if not, we add a constant C and a rule $C \to C$ to yield R''. Now apply (6) on R''.

Exercise 1.2.12. (Middeldorp [90]) Let R_1, R_2 be disjoint TRS's, both having the property UN. Show that $R_1 \oplus R_2$ has property UN. (Proof sketch: Use the previous exercise to extend R_i to R_i' such that R_i' is confluent and has the same convertibility and the same normal forms as R_i $(i = 1, 2)$. Moreover, R_1' and R_2' can be taken disjoint from each other. By Toyama's theorem (1.2.2) $R_1' \oplus R_2'$ is confluent, and hence also UN. Now consider $t, t' \in \mathrm{Ter}(R_1 \oplus R_2)$ such that t, t' are normal forms and convertible in $R_1 \oplus R_2$. Obviously t, t' are also convertible in $R_1' \oplus R_2'$. The proof is concluded by showing that t, t' are also

normal forms in $R_1' \oplus R_2'$. Hence $t \equiv t'$, and $R_1 \oplus R_2$ is UN.)

Examples 1.2.13.

1. Consider CL $\oplus$ $\{D(x,x) \to E\}$, Combinatory Logic with binary test for syntactic equality as in Table 1.6. Note that this is indeed a disjoint sum. As we shall see in Section 2.1, CL is confluent. Trivially, the one rule TRS $\{D(x,x) \to E\}$ is confluent. Hence, by Toyama's theorem (1.2.2) the disjoint sum is confluent.
2. By contrast, the union CL $\cup$ $\{Dxx \to E\}$, Combinatory Logic with 'varyadic' test for syntactic equality as in Table 1.7, is *not* confluent. (See Klop [80a].) Note that this combined TRS is merely a union and not a disjoint sum, since CL and $\{Dxx \to E\}$ have the function symbol Ap in common, even though hidden by the applicative notation.
3. Another application of Toyama's theorem (1.2.2): let R consist of the rules

$$\begin{array}{lcl} \underline{if}\ \underline{true}\ \underline{then}\ x\ \underline{else}\ y & \to & x \\ \underline{if}\ \underline{false}\ \underline{then}\ x\ \underline{else}\ y & \to & y \\ \underline{if}\ z\ \underline{then}\ x\ \underline{else}\ x & \to & x \end{array}$$

(Here $\underline{true}, \underline{false}$ are constants and $\underline{if} - \underline{then} - \underline{else}$ is a ternary function symbol.) Then CL $\oplus$ R is confluent. Analogous to the situation in (2), it is essential here that the $\underline{if} - \underline{then} - \underline{else}$ construct is a ternary operator. For the corresponding varyadic operator, the resulting TRS would not be confluent.

Remark 1.2.14. A different approach to modularity is taken by Kurihara & Kaji [88]. If R_1 and R_2 are disjoint TRS's, it is not allowed in that approach to perform arbitrary interleaving of R_1-steps and R_2-steps; there is the obligation to use as long as possible the rules of the same TRS. Thus, if a rule of say R_1 is applied to term t, we must first normalize t with respect to R_1, before applying rules of R_2, and vice versa. Formally: define relations $\blacktriangleright_i$ $(i = 1, 2)$ for terms $s, t \in \mathrm{Ter}(R_1 \oplus R_2)$ by $s \blacktriangleright_i t$ if $s \to_i^+ t$ and t is a normal form of R_i. Furthermore, $\blacktriangleright$ is the union of $\blacktriangleright_1$ and $\blacktriangleright_2$. Now Kurihara & Kaji [88] prove the following theorem:

1. *Let R_1, R_2 be disjoint TRS's. Then the relation $\blacktriangleright$ is terminating (SN).*
2. *Let R_1, R_2 be disjoint complete TRS's. Then the relation $\blacktriangleright$ is complete.*

Note that in (1) R_1, R_2 need not be SN. We will sketch a proof of (2). Assuming (1), part (2) of the theorem follows in some easy steps: First observe that for $\blacktriangleright$ we have UN $\Leftrightarrow$ CR, using UN & SN $\Rightarrow$ CR, a general fact for ARS's. So to prove UN for $\blacktriangleright$. Consider reductions $s \blacktriangleright \cdots \blacktriangleright t_1$

and $s \blacktriangleright \cdots \blacktriangleright t_2$, where t_1, t_2 are $\blacktriangleright$-normal forms. Because the original reductions $\rightarrow_i$ $(i = 1, 2)$ in R_i are SN, the terms t_1, t_2 are normal forms with respect to $\rightarrow$, the union of $\rightarrow_i$ $(i = 1, 2)$. Hence by Toyama's theorem 1.2.2: $t_1 \equiv t_2$.

Exercises 1.2.15. (Middeldorp)

1. Show that the modularity of WN (Exercise 1.2.9) is a corollary of the theorem in Remark 1.2.14.
2. Give an example of disjoint confluent TRS's such that $\blacktriangleright$ is not confluent. (Solution by A. Middeldorp of this question in Kurihara & Kaji [88]: $R_1 = \{F(x,x) \rightarrow F(x,x), A \rightarrow B\}$; $R_2 = \{e(x) \rightarrow x\}$. Now $F(e(A),A) \blacktriangleright_1 F(e(B),B) \blacktriangleright_2 F(B,B)$ and $F(e(A),A) \blacktriangleright_2 F(A,A)$. The terms $F(A,A)$ and $F(B,B)$ are different $\blacktriangleright$-normal forms.)

In this introduction to TRS's we will not consider termination properties of combined TRS's $R_1 \cup R_2$ which are not disjoint sums. For results in that area see Dershowitz [81, 87], Bachmair & Dershowitz [86], Toyama [88] and, for heterogeneous TRS's, Ganzinger & Giegerich [87]. As to confluence properties of combined TRS's $R_1 \cup R_2$ which are not disjoint sums, we include two facts in the following exercises, which require some concepts from the sequel (namely, the notion of overlapping reduction rules, critical pairs, and λ-calculus).

Exercise 1.2.16. (Raoult & Vuillemin [80], Toyama [88]) Let R_1, R_2 be TRS's. Define: $R_1 - R_2$ (R_1 and R_2 are orthogonal to each other) if there is no overlap between a rule of R_1 and one of R_2. (There may be critical pairs due to overlap between R_1-rules, or between R_2-rules.) Prove:

Theorem. *Let R_1, R_2 be left-linear and confluent TRS's such that $R_1 - R_2$. Then $R_1 \cup R_2$ is confluent.*

(Proof sketch. Prove that in $R_1 \cup R_2$ we have: (1) R_1-reductions commute; (2) R_2-reductions commute; (3) R_1-reductions commute with R_2-reductions. In order to prove (3), it is sufficient to prove (4) as in Figure 1.8. To prove (4), we need the left-linearity and the orthogonality requirements. The result now follows by an application of the Hindley-Rosen lemma in Exercise 1.0.17(3). The orthogonality is obviously necessary. Note that also the left-linearity cannot be dropped—see Example 1.2.13(2).)

Figure 1.8

Exercises 1.2.17. Prove:
Theorem. *Let R be a left-linear, confluent TRS. Let the signature of R be disjoint from that of λ-calculus, i.e. R does not contain the application operator. Then $\lambda \oplus R$, the disjoint sum of λ-calculus and R, is confluent.*

Proof sketch: by the same strategy as used for Exercise 1.2.16.

Semantics of Term Rewriting Systems

Although we do not enter the subject of *semantics* of TRS's (see e.g. Boudol [85], Guessarian [81]), 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:

Theorem 1.2.18. *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 with respect to $\mathcal{A}$, 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 TRS's (not treated here).

Decidability of properties in Term Rewriting Systems

We adopt the restriction in this subsection to TRS's R with finite alphabet and finitely many reduction rules. It is undecidable whether for such TRS's 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 TRS's*, i.e. TRS's 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], Dauchet et al. [87], Oyamaguchi [87]).

For the termination property (SN) the situation is the same. It is undecidable for general TRS's, even for TRS's with only one rule (see for a proof Dauchet [89]). For ground TRS's termination is decidable (Huet & Lankford [78]).

For particular TRS's 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 1.4.

Exercise 1.2.19. If $t \in$ Ter(R), we say "t is SN" if t admits no infinite reduction $t \to t' \to t'' \to \cdots$. Prove: If R is not SN, then there is a redex of R

which is not SN. In fact, then there is a redex whose contractum is not SN.

Exercises 1.2.20. (Huet & Lankford [78])

1. Let R be a ground TRS with finitely many rules, $R = \{t_i \to s_i \mid i = 1, \ldots, n\}$. Prove: If R is not SN, then for some $i \in \{1, \ldots, n\}$ and some context $C[\,]$ we have $t_i \to^+ C[t_i]$. (Hint: Use the previous exercise and use induction on n.)
2. Conclude: *SN is decidable for finite ground TRS's.*

Exercise 1.2.21. (Undecidability of SN) In this exercise we will outline a proof that SN is an undecidable property for (finite) TRS's, via a translation of the problem to the (uniform) halting problem for Turing machines. The proof is a slight simplification of the one in Huet & Lankford [78]. (However, that proof employs only constants and unary function symbols; below we use also binary function symbols.) We will not be concerned with the number of reduction rules employed in the translation of a Turing machine to a TRS; for an undecidability proof using a TRS of only two reduction rules, thus establishing that SN is undecidable even for TRS's with only two rules, see Dershowitz [87]. For a (complicated) proof that even for one rule TRS's the property SN is undecidable, see Dauchet [89]. (Even more, for orthogonal one rule TRS's SN is undecidable, as shown in Dauchet [89]. The property 'orthogonal' is defined in Chapter 2.)

A (deterministic) Turing machine M consists of a triple $\langle Q, S, \delta \rangle$ where Q is a set $\{q_0, \ldots, q_n\}$ of *states*, $S = \{\Box, s_1, \ldots, s_m\}$ is the set of *tape symbols* ($\Box$ being the empty symbol or 'blank'), and δ is a partial function (the *transition function*) from $Q \times S$ to $Q \times S \times \{L, R\}$. Here L represents a move to the left, R to the right.

An *instantaneous description* or *configuration* is an element of S^*QS^* (in the well-known notation of regular expressions). E.g. in Figure 1.9(a) the configuration $\Box aqba\Box a$ is pictured; the understanding is that in the configuration w_1qw_2 the head is in state q and scans the first symbol to the right of it, i.e. of w_2. Furthermore, the infinite portions of tape which are to the left of w_1 and to the right of w_2, are supposed to be blank. *Equivalent* configurations arise by appending to the left or to the right of the configuration finite portions of empty tape, i.e. elements of $\{\Box\}^*$.

The transition function δ determines *transition rules*, of the form

$$qst \rightarrowtail s'q't \quad (\text{for all } t \in S) \text{ whenever } \delta(q, s) = (q', s', R)$$

and

$$tqs \rightarrowtail q'ts' \quad \text{for all } t \in S) \text{ whenever } \delta(q, s) = (q', s', L).$$

A transition rule of the first type ('R-type') is a move to the right (see Figure 1.9(b)), and of the second type ('L-type') a move to the left. A rule of the first type can also be rendered as

$$qs \rightarrowtail s'q' \text{ whenever } \delta(q, s) = (q', s', R).$$

Transition rules may be applied in a 'context', giving rise to *transitions* between configurations, by appending words $w_1, w_2 \in S^*$ to the left and the right. Thus

Figure 1.9

the transition rule $qst \rightarrowtail s'q'tR$ generates transitions $w_1qstw_2 \rightarrowtail w_1s'q'tw_2$ for all $w_1, w_2 \in S^*$. Note that transitions operate in fact on equivalence classes of configurations.

We will now translate all this in the terminology of TRS's. That is, we associate to the Turing machine $M = \langle Q, S, \delta \rangle$ a TRS R_M as follows. For each $q \in Q$ there is a binary function symbol which we will denote with the same letter. Each $s \in S$ corresponds to a unary function symbol, also denoted with the same letter. Furthermore, the alphabet of R_m contains a constant symbol $\blacksquare$. A word $w \in S^*$ is translated into the term $\phi(w)$ as follows:

$$\begin{aligned} &\phi(\varepsilon) = \blacksquare \quad (\varepsilon \text{ is the empty word}) \\ &\phi(\mathrm{sw}) = s(\phi(w)) \quad \text{for } s \in S, w \in S^*. \end{aligned}$$

E.g. the translation of $ba\Box a$ is $b(a(\Box(a(\blacksquare))))$. In the sequel of this exercise we will suppress parentheses by association to the right, thus rendering $b(a(\Box(a(\blacksquare))))$ as $ba\Box a\blacksquare$.

A configuration $w_1 q w_2$ will be translated to $q(\phi(w_1^{-1}), \phi(w_2))$. Here w_1^{-1} is w_1 reversed. The reason for this reversal will be clear later. E.g. the configuration $\Box aqba\Box a$ is translated to $q(a\Box\blacksquare, ba\Box a\blacksquare)$.

We will now define the translation of the transition rules of M into reduction rules of R_M. To transition rules of R-type, $qs \rightarrowtail s'q'$, we let correspond the reduction rule

$$q(x, sy) \rightarrow q'(s'x, y).$$

In the case that s is $\Box$, so that the rule reads $q\Box \rightarrowtail s'q'$, we add moreover the reduction rule

$$q(x, \blacksquare) \rightarrow q'(s'x, \blacksquare).$$

In some sense, the second rule is a degenerate case of the first one; conceiving $\blacksquare$ as a potentially infinite portion of tape, satisfying the equation $\blacksquare = \Box\blacksquare$, it is clear how this rule arises from the first one.

To a rule of L-type, $tqs \rightarrowtail q'ts'$, we let correspond the reduction rule

$$q(tx, sy) \rightarrow q'(x, ts'y).$$

Again we have some extra rules for the 'degenerate' cases. If $tqs \rightarrowtail q'ts'$ is in fact $\Box qs \rightarrowtail q'ts'$ we add moreover

$$q(\blacksquare, sy) \rightarrow q'(\blacksquare, \Box s'y).$$

If $tqs \rightarrowtail q'ts'$ is in fact $tq\Box \rightarrowtail q'ts'$ we add moreover

$$q(tx, \blacksquare) \rightarrow q'(x, ts'\blacksquare).$$

If $tqs \rightarrowtail q'ts'$ is $\Box q\Box \rightarrowtail q'\Box s'$ we add moreover

$$q(\blacksquare, \blacksquare) \rightarrow q'(\blacksquare, \Box s'\blacksquare).$$

(So the transition rule $\Box q\Box \rightarrowtail q'\Box s'$ corresponds to four reduction rules.)

1. Now it is not hard to prove that for configurations α, β we have:

$$\alpha \rightarrowtail \beta \Leftrightarrow \phi(\alpha) \rightarrow \phi(\beta).$$

2. Prove that, given a TRS R and a term t in R, the problem to determine whether t has an infinite reduction in R, is undecidable. This means: there is no algorithm that accepts as inputs pairs (R, t) of a TRS R (given by a finite set of rewrite rules) and a term $t \in \mathrm{Ter}(R)$, and that yields as output the answer 'yes' if t has an infinite reduction in R, and 'no' otherwise. (Using (1), reduce this problem to the well-known undecidable halting problem for Turing machines with empty tape as initial configuration.)

3. To each ground term in R_M of the form $q(t_1, t_2)$ where t_1, t_2 are terms in which no $q' \in Q$ occurs (call such a term 'restricted'), there corresponds a

configuration of M; but this is not so without that restriction. Prove that if some term t in R_M has an infinite reduction in R_M, then there is also a restricted ground term t' in R_M having an infinite reduction, and thus yielding a corresponding infinite run of the Turing machine M.

4. Prove, using (3) and referring to the well-known undecidable *uniform* halting problem for Turing machines, that the problem to determine whether a given TRS is SN (strongly normalizing) is undecidable. The uniform halting problem for Turing machines is the problem to decide whether a given Turing machine halts on every input as initial configuration.

1.3 A termination proof technique

As Newman's Lemma (WCR & SN $\Rightarrow$ CR) shows, termination (SN) is a useful property. In general, as noted in Exercise 1.2.21, 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 [87].) 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. (See also the similar concept of 'path of subterm ordering' in Plaisted [78], discussed in Rusinowitch [87b].) 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.

Definition 1.3.1.

1. Let $\mathbb{T}$ be the set of commutative finite trees with nodes labeled by natural numbers. Example: see Figure 1.10(a). 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.
2. 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 1.10(b); this tree will be denoted by $3^*(5, 7(9^*), 8^*(0(1, 5)))$.

Notation 1.3.2. $n(t_1, \ldots, t_k)$ will be written as $n(\vec{t})$. The t_i $(i = 1, \ldots, k)$ are elements of $\mathbb{T}^*$. Further, if $t \equiv n(t_1, \ldots, t_k)$ then t^* stands for $n^*(t_1, \ldots, t_k)$.

Definition 1.3.3. On $\mathbb{T}^*$ we define a reduction relation $\Rightarrow$ as follows.

1. *place marker at the top:*
   $n(\vec{t}) \Rightarrow n^*(\vec{t}) \qquad (\vec{t} = t_1, \ldots, t_k; k \geq 0)$

Figure 1.10

2. *make copies below lesser top:*
   if $n > m$, then $n^*(\vec{t}) \Rightarrow m(n^*(\vec{t}), \ldots, n^*(\vec{t}))$ $(j \geq 0$ copies of $n^*(\vec{t}))$
3. *push marker down:*
   $n^*(s, \vec{t}) \Rightarrow n(s^*, \ldots, s^*, \vec{t})$ $(j \geq 0$ copies of $s^*)$
4. *select argument:*
   $n^*(t_1, \ldots, t_k) \Rightarrow t_i$ $(i \in \{1, \ldots, k\}, k \geq 1)$

It is understood that these reductions may take place in a context, i.e. if $t \Rightarrow s$, then $n(—, t, —) \Rightarrow n(—, s, —)$

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

Example 1.3.4. Figure 1.11 displays a reduction in $\mathbb{T}^*$.

Clearly, the reduction $\Rightarrow$ is not SN in $\mathbb{T}^*$; for, consider the second step in Figure 1.11: there the right hand side contains a copy of the left-hand side. However:

Theorem 1.3.5. *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 $t_0 \Rightarrow^+ t_1 \Rightarrow^+ t_2 \Rightarrow^+ \cdots$ of terms t_i $(i \geq 0)$ without markers. The proof of Theorem 1.3.5 is based on Kruskal's Tree Theorem; we will give the main argument.

In order to introduce the next notion of 'embedding', we must make the definition of trees $t \in \mathbb{T}$ somewhat more precise. An element $t \in \mathbb{T}$ is a pair $(\langle D, \leq, \alpha_0 \rangle, \mathcal{L})$ where D is a finite set $\{\alpha_0, \beta, \gamma, \ldots\}$ with distinguished element α_0, called the root or the top of t, and partially ordered by $\leq$. We require that:

1. $\alpha_0 \geq \beta$ for all $\beta \in D$,
2. $\beta \leq \gamma$ and $\beta \leq \delta \Rightarrow \gamma \leq \delta$ or $\delta \leq \gamma$, for all $\beta, \gamma, \delta \in D$.

Figure 1.11

The set D is also called $\text{NODES}(t)$. Furthermore, $\mathcal{L}$: $D \to \mathbb{N}$ is a map assigning labels (natural numbers) to the nodes of t. Finally, we use the notation $\alpha \wedge \beta$ for the supremum (least upper bound) of $\alpha, \beta \in D$. (The actual names $\alpha, \beta, \ldots$ of the nodes are not important, which is why they were suppressed in the pictorial representation of $t \in \mathbb{T}$ above.)

Definition 1.3.6. Let $t, t' \in \mathbb{T}$. We say that t is *(homeomorphically) embedded in* t', notation $t \trianglelefteq t'$, if there is a map φ: $\text{NODES}(t) \to \text{NODES}(t')$ such that:

1. φ is *injective*,
2. φ is *monotonic* $(\alpha \leq \beta \Rightarrow \varphi(\alpha) \leq \varphi(\beta))$,
3. φ is *sup preserving* $(\varphi(\alpha \wedge \beta) = \varphi(\alpha) \wedge \varphi(\beta))$,
4. φ is *label increasing* $(\mathcal{L}(\alpha) \leq \mathcal{L}'(\varphi(\alpha))$, where $\mathcal{L}, \mathcal{L}'$ are the labeling maps of t, t' respectively; $\leq$ is the ordering on natural numbers).

Actually, (2) is superfluous as it follows from (3).

Example 1.3.7.

1. $2(9, 7(0, 4)) \trianglelefteq 1(3(8(0(5, 1)), 9, 5(9)), 2)$ as the embedding in Figure 1.12 shows.
2. Note that we do *not* have $1(0, 0) \trianglelefteq 1(0(0, 0))$.

Figure 1.12

Clearly, $\preceq$ is a partial order on $\mathbb{T}$. Moreover it satisfies the following remarkable property:

Theorem 1.3.8. (Kruskal's Tree Theorem) *Let $t_0, t_1, t_2, \ldots$ be a sequence of trees in $\mathbb{T}$. Then for some $i < j$: $t_i \preceq t_j$.*

The proof of this theorem, as given in Kruskal [60], is extremely complicated. Proofs are given in Dershowitz [79] and Dershowitz & Jouannaud [90]. See also Exercise 1.3.12 for a detailed proof sketch of a restricted case which is sufficient for the present purpose.

Now we have the following proposition (of which (1) is nontrivial to prove):

Proposition 1.3.9.

1. *$\Rightarrow^+$ is a strict partial order on $\mathbb{T}$,*
2. *if $s \preceq t$, then $t \Rightarrow^* s$.*

(Here $\Rightarrow^*$ is the transitive-reflexive closure of $\Rightarrow$.) Combining 1.3.8 and 1.3.9, we have Theorem 1.3.5. For, suppose there is an infinite sequence

$$t_0 \Rightarrow^+ t_1 \Rightarrow^+ t_2 \Rightarrow^+ \cdots \Rightarrow^+ t_i \Rightarrow^+ \cdots \Rightarrow^+ t_j \Rightarrow^+ \cdots$$

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

Application 1.3.10 (Dershowitz [87]). Let a TRS R as in Table 1.9 be given. To prove that R is SN. Choose a 'weight' assignment $\vee \mapsto 1$, $\wedge \mapsto 2$,

$\neg\neg x$	$\rightarrow$	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 1.9

$\neg \mapsto 3$. Now a reduction in R corresponds to a $\Rightarrow^+$ reduction in $\mathbb{T}$ (and hence it is also SN) as follows:

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

E.g. the second rule:

$$\begin{array}{lll} 3(1(t,s)) & \Rightarrow & 3^*(1(t,s)) \\ & \Rightarrow & 2(3^*(1(t,s)),3^*(1(t,s))) \\ & \Rightarrow^+ & 2(3(1^*(t,s)),3(1^*(t,s))) \\ & \Rightarrow^+ & 2(3(t),3(s)). \end{array}$$

Remark 1.3.11.

1. 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 1.3.6) in the right-hand side, as in $f(s(x)) \rightarrow g(s(x), f(p(s(x))))$. For an extension of Kruskal's Theorem, leading to a method which also can deal with this case, see Kamin & Lévy [80] and Puel [86].
2. Another example where the method above does not work directly, is found in the TRS's corresponding to process algebra axiomatizations as in Bergstra & Klop [84, 85]. For instance in the axiom system PA there are the rewrite rules

$$\begin{array}{lll} x\|y & \rightarrow & (x \between y) + (y \between x) \\ (x+y) \between z & \rightarrow & (x \between z) + (y \between z) \\ (a \cdot x) \between y & \rightarrow & a \cdot (x\|y). \end{array}$$

Here one wants to order the operators as follows: $\| > \between > \cdot, +$, but then we get stuck at the third rule with the re-emergence of the 'heavy' operator $\|$. In Bergstra & Klop [85] the solution was adopted

to introduce infinitely many operators $\|_n$ and $\lfloor\!\lfloor_n$, 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 $x\|y$ as $g(x,y), x \lfloor\!\lfloor y$ as $h(x,y), a{\cdot}x$ as $f(x)$, we have Example 16 in Dershowitz [87] where this problem takes the following form and is solved by a lexicographical combination of recursive path orderings:

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

The termination proof as in Bergstra & Klop [85] amounts to the following for the present example. Define a norm $|\ |$ on terms by: $|\ t\ |$ is 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. Caution is required here: the norm must be chosen such that the norm of a term t is not increased by reduction of a subterm of t. (For this reason, taking $|\ t\ |$ as the length of t in symbols would not work for the process algebra example above.)

3. A third example were the proof method above does not work, is when an associativity rule

$$(x \cdot y) \cdot z \quad \to \quad x \cdot (y \cdot z)$$

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

$$\begin{array}{lcl} A(0,x) & \to & S(x) \\ A(S(x),0) & \to & A(x,S(0)) \\ A(S(x),S(y)) & \to & A(x,A(S(x),y)) \end{array}$$

What we need here is the *lexicographic path ordering* of Kamin & Lévy [80], see Dershowitz [87]. 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 1.3.6) respects also the left-right ordering; Kruskal's Tree Theorem works also for this case of noncommutative trees.
Next, the rules in Definition 1.3.3 are restricted such that the arities of the operators are respected; in Definition 1.3.3 the operators were treated 'varyadic'. So rule (3) becomes: $n^*(t_1,\ldots,t_i,\ldots,t_k) \Rightarrow$

$n(t_1, \ldots, t_i^*, \ldots, t_k)$ $(1 \leq i \leq k)$. Further, we add to the rules in Definition 1.3.3 (with (3) amended) the rule

5. *simplify left argument:*
$n^*(\vec{t}) \Rightarrow n(t_1^*, n^*(\vec{t}), \ldots, n^*(\vec{t}))$
$(\vec{t} = t_1, \ldots, t_k \ (k \geq 1)$; $k-1$ copies of $n^*(\vec{t}))$

Example:

$$\begin{array}{rcl} A(S(x), S(y)) & \Rightarrow & A^*(S(x), S(y)) \\ & \Rightarrow & A(S^*(x), A^*(S(x), S(y))) \\ & \Rightarrow & A(x, A^*(S(x), S(y))) \\ & \Rightarrow & A(x, A(S(x), S^*(y))) \\ & \Rightarrow & A(x, A(S(x), y)). \end{array}$$

Exercise 1.3.12. In this exercise we outline a short proof of a restricted version of Kruskal's Tree Theorem 1.3.8, which is sufficient for termination proofs of TRS's where the function symbols have arities uniformly bounded by some natural number N. (There may be infinitely many function symbols, as e.g. the g_n, h_n in the preceding Remark 1.3.11.) A fortiori this is the case for TRS's with finite alphabet.

The proof below is similar to that in Dershowitz [79]; the proof in Dershowitz & Jouannaud [90] is similar but for a short-cut there appealing to a special case of the Tree Theorem known as Higman's Lemma. These proofs are originally due to Nash-Williams [63]. First we define:

1. The *branching degree* of a node s in $t \in \mathbb{T}$ is the number of immediate successor nodes of s.
2. $\mathbb{T}_N$ is the subset of $\mathbb{T}$ consisting of trees where all nodes have branching degree $\leq N$. Likewise we define $\mathbb{T}_N^*$.

We will now outline a proof of Kruskal's Tree Theorem 1.3.8 where $\mathbb{T}$ is restricted to $\mathbb{T}_N$.

1. CLAIM. *Each infinite sequence of natural numbers $n_0, n_1, n_2, \ldots$ has a weakly ascending infinite subsequence.*
This means that there is a subsequence $n_{f(0)}, n_{f(1)}, n_{f(2)}, \ldots$ with $f(0) < f(1) < f(2) < \ldots$ such that $n_{f(0)} \leq n_{f(1)} \leq n_{f(2)} \leq \ldots$. The proof is simple.
2. DEFINITION.
   (a) Let $t \in \mathbb{T}_N$. Then $\mid t \mid$ is the number of nodes of t.
   (b) Notation: an infinite sequence of trees $t_0, t_1, \ldots$ will be written as $\mathbf{t}$. The initial segment $t_0, \ldots, t_{n-1}$ is $(\mathbf{t})_n$. The set of infinite sequences of trees from $\mathbb{T}_N$ is $\mathbb{T}_N^\omega$.
   (c) Let $D \subseteq \mathbb{T}_N^\omega$. Then the sequence $\mathbf{t} \in D$ is *minimal in* D if $\forall \mathbf{s} \in D\ (\mathbf{s})_n = (\mathbf{t})_n \Rightarrow \mid s_n \mid \geq \mid t_n \mid$.
   (d) Furthermore, we say that $\mathbf{s}, \mathbf{t} \in \mathbb{T}_N^\omega$ have distance 2^{-n} if $(\mathbf{s})_n = (\mathbf{t})_n$ but $(\mathbf{s})_{n+1} \neq (\mathbf{t})_{n+1}$. This induces a metric on $\mathbb{T}_N^\omega$.

3. CLAIM. *Let $D \subseteq \mathbb{T}^{\omega}_{N}$ be non-empty and closed w.r.t. the metric just defined. Then D contains a minimal element (with respect to D).*
The proof of Claim 3 is easy.

4. NOTATION.
   (a) Let $\mathbf{s}, \mathbf{t} \in \mathbb{T}^{\omega}_{N}$. Then $\mathbf{s} \subseteq \mathbf{t}$ means that $\mathbf{s}$ is a subsequence of $\mathbf{t}$.
   (b) Let $\mathbf{t} = t_0, t_1, \ldots$ and let $\mathbf{s} = s_{f(0)}, s_{f(1)}, \ldots$ be a subsequence of $\mathbf{t}$, such that for all $i, s_{f(i)}$ is a *proper* subtree of $t_{f(i)}$. Then we write $\mathbf{s} \subseteq\subseteq \mathbf{t}$ and call $\mathbf{s}$ a subsubsequence of $\mathbf{t}$. (See Figure 1.13.)

Figure 1.13

5. DEFINITION. $\mathbf{s} = s_0, s_1, s_2, \ldots$ is a *chain* if $s_0 \preceq s_1 \preceq s_2 \preceq \ldots$, where $\preceq$ is the embedding relation as in Kruskal's Tree Theorem.

We will now suppose, for a proof by contradiction, that there is a counterexample sequence to the restricted version of Kruskal's Tree Theorem that we want to prove. That is, the set $C \subseteq \mathbb{T}^{\omega}_{N}$ of sequences $\mathbf{s}$ such that for no $i < j$ we have $s_i \preceq s_j$, is supposed to be non-empty. Note that C is closed in the sense of Definition 2(d).

6. CLAIM. *Let $\mathbf{t}$ be a minimal element from C. Suppose $\mathbf{s} \subseteq\subseteq \mathbf{t}$.*
   (a) *Then for some $i < j$: $s_i \preceq s_j$.*
   (b) *Even stronger, $\mathbf{s}$ contains a subsequence which is a chain.*

PROOF of Claim 6(a). (Note that a minimal element $\mathbf{t}$ exists by the assumption $C \neq \varnothing$ and by Claim 3.) Let $\mathbf{s}$, $\mathbf{t}$ be as in Claim 6. Let s_0 be a proper subtree of $t_{f(0)} = t_k$. Consider the sequence $t_0, \ldots, t_{k-1}, s_0, s_1, s_2, \ldots$, that is, $(\mathbf{t})_k$ followed by $\mathbf{s}$. By minimality of $\mathbf{t}$, this sequence is not in C. Hence it contains an embedded pair of elements (the earlier one embedded in the later one). The embedded pair cannot occur in the prefix $(\mathbf{t})_k$ because $\mathbf{t} \in C$. It can also not be of the form $t_i \preceq s_j$, since then $\mathbf{t}$ would contain the embedded pair $t_i \preceq t_{f(j)}$. So, the embedded pair must occur in the postfix $\mathbf{s}$.

As to part (b) of the claim, suppose $\mathbf{s}$ does not contain an infinite chain as subsequence. Then $\mathbf{s}$ contains an infinite number of finite chains, each starting to the right of the end of the previous finite chain and each maximal in the sense

that it cannot be prolonged by an element occurring to the right of it in $\mathbf{s}$. Now consider the last elements of these finite chains. These last elements constitute an infinite subsubsequence of $\mathbf{t}$, containing by (a) of the claim an embedded pair. But that means that one of the maximal finite chains can be prolonged, a contradiction.

7. CLAIM. *Let* $\mathbf{t}$ *be minimal in* C *and suppose* $\mathbf{s} \subseteq\subseteq \mathbf{r} \subseteq \mathbf{t}$. *Then* $\mathbf{s}$ *contains an infinite chain as subsequence.*

The proof of Claim 7 is trivial. We will now apply a sieve procedure to the minimal counterexample sequence $\mathbf{t} \in C$. By Claim 1 we can take a subsequence $\mathbf{t}'$ of $\mathbf{t}$ such that the root labels are weakly ascending. Of $\mathbf{t}'$ we take a subsequence $\mathbf{t}^*$ with the property that the branching degrees of the roots are a weakly ascending sequence. By Claim 6 every subsubsequence of $\mathbf{t}^*$ still contains an infinite embedding chain.

Let us 'freeze' the elements in $\mathbf{t}'$, that is, we impose an ordering of the successors of each node in some arbitrary way. So the frozen trees in $\mathbf{t}'$ are no longer commutative trees, and we can speak of the first, second etc. 'arguments' of a node. (An argument of a node α is the subtree with as root a successor node β of α.)

The next step in the sieve procedure is done by considering the sequence of first arguments of (the roots of) the elements in t^*. As this is a subsubsequence, it contains an infinite chain. Accordingly, we thin t^* out, to the subsequence t^{**}. This sequence has the property that its first arguments form a chain. Next, t^{**} is thinned out by considering the sequence of the second arguments of t^{**}. Again, this sequence contains a chain, and thinning t^{**} accordingly yields the subsequence t^{***}.

Figure 1.14

After at most N steps of the last kind, we are through. The result is then a chain, since the roots already satisfied the embedding condition (they form a weakly ascending chain), and the arguments are also related as chains. (See Figure 1.14.) However, this contradicts the assumption that $\mathbf{t}$ contains no embedded pair. Hence C is empty, and the restricted version of Kruskal's Tree Theorem is proved.

Exercise 1.3.13. (Kruskal [60]) In this exercise we introduce the terminology of *well-quasi-orders* which is often used to formulate Kruskal's Tree Theorem.

1. DEFINITION. The binary relation $\leq$ is a *quasi-order* (qo) if it is reflexive and transitive. (So the relation $\twoheadrightarrow$ in a TRS is a qo.) If in addition $\leq$ is anti-symmetric (i.e. $x \leq y \;\&\; y \leq x \Rightarrow x = y$ for all x, y) then $\leq$ is a partial order (po).
2. DEFINITION. Let $\langle X, \leq \rangle$ be a qo. A subset $Y \subseteq X$ is called a *cone* if $x \in Y \;\&\; x \leq y \Rightarrow y \in Y$ for all $x, y \in X$. The cone *generated by* $Y \subseteq X$, notation $Y\uparrow$, is the set $\{x \in X \mid \exists y \in Y\ y \leq x\}$. (It is the intersection of all cones containing Y.) A cone Z is *finitely generated* if $Z = Y\uparrow$ for some finite Y.
3. DEFINITION. Let $\langle X, \leq \rangle$ be a qo (po, respectively). Then $\langle X, \leq \rangle$ is a *well-quasi-order* (wqo) or *well-partial-order* (wpo) respectively, if every cone of X is finitely generated.
4. DEFINITION. Let $\langle X, \leq \rangle$ be a qo. A subset $Y \subseteq X$ is an *anti-chain* if the elements of Y are pairwise incomparable, i.e. for all $x, y \in Y$ such that $x \neq y$ we have neither $x \leq y$ nor $y \leq x$.

Prove the following lemma:

5. LEMMA. *Let $\langle X, \leq \rangle$ be a qo. Then the following conditions are equivalent:*
   (a) *$\langle X, \leq \rangle$ is a wqo;*
   (b) *X contains no infinite descending chains $x_0 > x_1 > x_2 > \cdots$ and all anti-chains of X are finite;*
   (c) *for every infinite sequence of elements $x_0, x_1, x_2, \ldots$ in X there are i, j such that $i < j$ and $x_i \leq x_j$.*

So, Kruskal's Tree Theorem as stated in 1.3.8 can be reformulated as follows: *$\langle \mathbb{T}, \trianglelefteq \rangle$ is a well-quasi-order.* Prove that $\langle \mathbb{T}, \trianglelefteq \rangle$ is in fact a partial order; so Kruskal's theorem states that $\langle \mathbb{T}, \trianglelefteq \rangle$ is even a well-partial-order.

Exercise 1.3.14.

1. Show that the well-partial-order $\langle \mathbb{T}, \trianglelefteq \rangle$ is not a linear order.
2. Show that $\Rightarrow^+$ is a linear order. As it is well-founded (Theorem 1.3.5), it corresponds to an ordinal. For connections with the ordinal Γ_0, the first impredicative ordinal, see Dershowitz [87]. For more about Kruskal's Tree Theorem and the connection with large ordinals, as well as a version of the Tree Theorem which is independent from Peano's Arithmetic, see Smoryński [82] and Gallier [87].

Exercise 1.3.15. (Multiset orderings) Very useful for termination proofs (used in some of the Exercises 1.0.8) are the *multiset orderings*; these are particular cases of the well-founded ordering $\langle \mathbb{T}, \Rightarrow^+ \rangle$ discussed above, namely by restricting the domain $\mathbb{T}$.

1. *Multisets.* Let $\langle X, < \rangle$ be a strict partial order. Then the p.o. of *multisets over X*, or the *multiset extension of X*, notation $\langle X^\mu, <^\mu \rangle$, is obtained as follows. The elements of X^μ are finite "sets" of elements from X with the understanding that multiplicity of occurrences is taken into account, other than in ordinary sets. A multiset will be denoted by square brackets [ ].

E.g. if $a, b \in X$ then $[a, a, b]$ and $[a, b]$ are different multisets; but $[a, a, b]$ and $[a, b, a]$ denote the same multiset. Stated differently, a multiset is a finite sequence of elements where the order of occurrences in the sequence is disregarded. Giving a more formal definition is left to the reader. A multiset is also known as a *bag*. We use in this exercise $\alpha, \beta, \ldots$ as variables for multisets.

Now we define the following relation $>_1$ between elements of X^μ by the two clauses:

(a) $[a] >_1 [b_1, \ldots, b_n]$ for all $a, b_1, \ldots, b_n \in X$ $(n \geq 0)$ such that $a > b_i$ $(i = 1, \ldots, n)$;

(b) $\alpha >_1 \beta \Rightarrow \alpha \cup \gamma >_1 \beta \cup \gamma$. Here $\cup$ denotes multiset union, defined in the obvious way as a union where the multiplicities of the elements are respected. E.g. $[a, a, b] \cup [a, b, c] = [a, a, a, b, b, c]$. Thus, a multiset gets smaller by replacing an element in it by arbitrarily many (possibly 0) elements which are less in the original ordering. The converse of $>_1$ is $<_1$.

Furthermore, we define:

(c) $<^\mu$ is the transitive closure of $<_1$.

Now prove the following statements:

(a) If $\langle X, < \rangle$ is a strict partial order, then so is its multiset extension $\langle X^\mu, <^\mu \rangle$. If $\langle X, < \rangle$ is moreover a linear order, then so is $\langle X^\mu, <^\mu \rangle$.

(b) (Dershowitz & Manna [79]) $\langle X, < \rangle$ is a well-founded p.o. $\Leftrightarrow$ $\langle X^\mu, <^\mu \rangle$ is a well-founded p.o. (The p.o. $\langle X, < \rangle$ is well-founded if there are no infinite descending chains $x_0 > x_1 > \ldots$.)

(c) Let $\langle X, < \rangle$ be a well-founded linear order with order type α. Then $\langle X^\mu, <^\mu \rangle$ has order type ω^α.

2. *Nested multisets.* Let $\langle X, < \rangle$ be a p.o. Then the p.o. of *nested multisets over* X, notation: $\langle X^{\mu^*}, <^{\mu^*} \rangle$, is defined as follows. The domain X^{μ^*} is the least set Y such that $X \subseteq Y$ and $Y^\mu = Y$. Or, inductively:

(a) $X_0 = X$;

(b) $X_{n+1} = (X_0 \cup \cdots \cup X_n)^\mu$;

(c) $X^{\mu^*} = \cup_{n \geq 0} X_n$.

Note that the elements of X^{μ^*} can be represented as finite commutative trees, with terminal nodes labeled by elements from X, and non-terminal nodes with a label representing the multiset-operator. The *depth* of $\alpha \in X^{\mu^*}$ is the stage of the inductive definition in which it is generated, or in the tree representation, the maximum of the lengths of the branches of the tree corresponding to α.

Furthermore, the ordering $<^{\mu^*}$ is the least relation R extending $<$ and satisfying:

(a) $x \, R \, \alpha$ for all $x \in X$ and multisets $\alpha \in X^{\mu^*} - X$;

(b) $[\alpha] \, R \, [\beta_1, \ldots, \beta_n]$ for all $\alpha, \beta_1, \ldots, \beta_n \in X^{\mu^*}$ $(n \geq 0)$ such that $\alpha_i \, R \, \beta_i$ $(i = 1, \ldots, n)$;

(c) $\alpha \, R \, \beta \Rightarrow \alpha \cup \gamma \, R \, \beta \cup \gamma$ for all multisets $\alpha, \beta, \gamma \in X^{\mu^*} - X$.

Now:

(a) Let $\langle X, < \rangle$ be a p.o. Prove that $\langle X^{\mu^*}, <^{\mu^*} \rangle$ is a p.o. If moreover $\langle X, < \rangle$ is a linear order, then so is $\langle X^{\mu^*}, <^{\mu^*} \rangle$.

(b) Let $\alpha, \beta \in \langle X^{\mu^*}, <^{\mu^*} \rangle$. Prove that if the depth of α is greater than the depth of β, we have $\alpha <^{\mu^*} \beta$.

(c) (Dershowitz & Manna [79])
$\langle X, < \rangle$ is well-founded $\Leftrightarrow$ $\langle X^{\mu^*}, <^{\mu^*} \rangle$ is well-founded.

(d) Let $\langle \mathbb{N}, < \rangle$ be the natural numbers with the usual ordering. Prove that $\langle \mathbb{N}^{\mu^*}, >^{\mu^*} \rangle$, the nested multisets over the natural numbers, is in fact a restriction of the recursive path ordering $\langle \mathbb{T}, \Rightarrow^+ \rangle$ if the non-terminal nodes of the tree representation of $\alpha \in \mathbb{N}^{\mu^*}$ are taken to be 0. That is, for $\alpha, \beta \in \mathbb{N}^{\mu^*} \subseteq \mathbb{T}$ we then have: $\alpha >^{\mu^*} \beta \Leftrightarrow \alpha \Rightarrow^+ \beta$.

(e) Show that the order type of the well-founded linear ordering $\langle \mathbb{N}^{\mu^*}, <^{\mu^*} \rangle$ is ϵ_0. Note that $\langle \mathbb{N}^{\mu^*}, <^{\mu^*} \rangle$ is isomorphic to $\langle \{0\}^{\mu^*}, <^{\mu^*} \rangle$. Here '$<$' in the last occurrence of $<^{\mu^*}$ is the restriction of $<$ to $\{0\}$ (which in fact is the empty relation). Figure 1.15 gives an example of two multisets α, β over $\{0\}$, such that $\alpha >^{\mu^*} \beta$, or equivalently, $\alpha \Rightarrow^+ \beta$. All labels at the nodes can be taken 0, and are omitted in the figure. Note that the procedure using the markers may employ all clauses in Definition 1.3.3 except clause (2).

Figure 1.15

1.4 Completion of equational specifications

In this section we will give an introduction to Knuth-Bendix completion of equational specifications. First we will introduce the latter concept.

Equational specifications: syntax and semantics

We can be short about introducing the syntax of equational specifications: an equational specification is just a TRS "without orientation". More precisely, an equational specification is a pair (Σ, E) where the signature

(or alphabet) Σ is as in Section 1.1 for TRS's (Σ, R), and where E is a set of equations $s = t$ between terms $s, t \in \mathrm{Ter}(\Sigma)$.

If an equation $s = t$ is *derivable* from the equations in E, we write $(\Sigma, E) \vdash s = t$ or $s =_E t$. Formally, derivability is defined by means of the inference system of Table 1.10.

$(\Sigma, E) \vdash s = t$	if $s = t \in E$
$\dfrac{(\Sigma, E) \vdash s = t}{(\Sigma, E) \vdash s^\sigma = t^\sigma}$	for every substitution σ
$\dfrac{(\Sigma, E) \vdash s_1 = t_1, \ldots, (\Sigma, E) \vdash s_n = t_n}{(\Sigma, E) \vdash F(s_1, \ldots, s_n) = F(t_1, \ldots, t_n)}$	for every n-ary $F \in \Sigma$
$(\Sigma, E) \vdash t = t$	
$\dfrac{(\Sigma, E) \vdash t_1 = t_2, (\Sigma, E) \vdash t_2 = t_3}{(\Sigma, E) \vdash t_1 = t_3}$	
$\dfrac{(\Sigma, E) \vdash s = t}{(\Sigma, E) \vdash t = s}$	

Table 1.10

Exercise 1.4.1. (Equational deduction systems) Often the inference system in Table 1.10 is presented slightly different, as follows. Prove the equivalence of the two versions below with the system above. Axioms (in addition to the equations in E):

$$t = t \quad \textit{reflexivity}$$

Rules:

$$\frac{t_1 = t_2}{t_2 = t_2} \quad \textit{symmetry}$$

$$\frac{t_1 = t_2,\ t_2 = t_3}{t_1 = t_3} \quad \textit{transitivity}$$

$$\frac{t_1 = t_2}{t_1[x := t] = t_2[x := t]} \quad \textit{substitution}(1)$$

$$\frac{t_1 = t_2}{t[x := t_1] = t[x := t_2]} \quad \textit{substitution}(2)$$

Here $[x := t]$ denotes substitution of t for all occurrences of x. (The assignment notation is chosen to avoid the usual confusion between $[x/t], [t/x], [x \backslash t], [t \backslash x]$.) An equivalent formulation is to combine the two substitution rules in one:

$$\frac{t_1 = t_2,\ t = t'}{t_1[x := t] = t_2[x := t']} \quad \textit{substitution}$$

If Σ is a signature, a Σ-*algebra* $\mathcal{A}$ is a set A together with functions $F^{\mathcal{A}} : A^n \to A$ for every n-ary function symbol $F \in \Sigma$. (If F is 0-ary, i.e. F is a constant, then $F^{\mathcal{A}} \in A$.) An equation $s = t$ $(s, t \in \mathrm{Ter}(\Sigma))$ is assigned a meaning in $\mathcal{A}$ by interpreting the function symbols in s, t via the corresponding functions in $\mathcal{A}$. Variables in $s = t$ are (implicitly) universally quantified. If the universally quantified statement corresponding to $s = t$ $(s, t \in \mathrm{Ter}(\Sigma))$ is true in $\mathcal{A}$, we write $\mathcal{A} \models s = t$ and say that $s = t$ is *valid* in $\mathcal{A}$. $\mathcal{A}$ is called a *model* of a set of equations E if every equation in E is valid in $\mathcal{A}$. Abbreviation: $\mathcal{A} \models E$. The *variety* of Σ-algebras defined by an equational specification (Σ, E), notation $\mathrm{Alg}(\Sigma, E)$, is the class of all Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \models E$. Instead of $\forall \mathcal{A} \in \mathrm{Alg}(\Sigma, E)\ \mathcal{A} \models F$, where F is a set of equations between Σ-terms, we will write $(\Sigma, E) \models F$. There is the well-known completeness theorem for equational logic of Birkhoff [35]:

Theorem 1.4.2. *Let (Σ, E) be an equational specification. Then for all $s, t \in Ter(\Sigma)$:*

$$(\Sigma, E) \vdash s = t \ \Leftrightarrow\ (\Sigma, E) \models s = t.$$

Now the *validity problem* or *uniform word problem* for (Σ, E) is:

> *Given an equation $s = t$ between Σ-terms, decide whether or not $(\Sigma, E) \models s = t$.*

According to Birkhoff's completeness theorem for equational logic this amounts to deciding $(\Sigma, E) \vdash s = t$. Now we can state why *complete* TRS's (i.e. TRS's which are SN and CR) are important. Suppose for the equational specification (Σ, E) we can find a complete TRS (Σ, R) such that for all terms $s, t \in \mathrm{Ter}(\Sigma)$:

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

Then (if R has finitely many rewrite rules only) we have a positive solution of the validity problem. The decision algorithm is simple:

1. *Reduce s and t to their respective normal forms s', t'*
2. *Compare s' and t': $s =_R t$ iff $s' \equiv t'$.*

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 (even if finite) has a solvable validity problem. The most famous example of such an E with unsolvable validity problem is the set of equations obtained from CL, Combinatory Logic, in Tables 1.3, 1.4 above after replacing '$\rightarrow$' by '$=$': see Table 1.11.

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

Table 1.11

(For a proof of the unsolvability see Barendregt [81].) So the validity problem of (Σ, E) can be solved by providing a complete TRS (Σ, R) for (Σ, E). Note however, that there are equational specifications (Σ, E) with decidable validity problem but without a complete TRS (Σ, R) satisfying $(*)$: see the following Exercise.

Exercise 1.4.3. Let (Σ, E) be the specification given by the equations

$$\begin{array}{lcl} x+0 & = & x \\ x+S(y) & = & S(x+y) \\ x+y & = & y+x \end{array}$$

Prove that there is no complete TRS R 'for' E, i.e. such that for all terms $s, t \in \mathrm{Ter}(\Sigma) : s =_R t \Leftrightarrow s =_E t$. (Consider in a supposed complete TRS R, the normal forms of the open terms $x + y$ and $y + x$.)

It is important to realize that we have considered up to now equations $s = t$ between possibly *open* Σ-terms (i.e. possibly containing variables). If we restrict attention to equations $s = t$ between *ground* terms s, t, we are considering the *word problem* for (Σ, E), which is the following decidability problem:

> *Given an equation $s = t$ between ground terms $s, t \in \mathrm{Ter}(\Sigma)$, decide whether or not $(\Sigma, E) \models s = t$ (or equivalently, $(\Sigma, E) \vdash s = t$).*

Also for the word problem, complete TRS's provide a positive solution. In fact, we require less than completeness (SN and CR) for all terms, but only for ground terms. (See Example 1.1.3 for an example where this makes a difference.) It may be (as in Exercise 1.4.3) that a complete TRS for E cannot be found with respect to all terms, while there does exist a TRS which is complete for the restriction to ground terms.

Exercise 1.4.4. Consider the specification as in the previous exercise and find a TRS (Σ, R) such that $(\Sigma, R)_0$ (i.e. the restriction of (Σ, R) to ground terms) is complete.

Remark 1.4.5. Note that there are finite equational specifications (Σ, E) which have a decidable word problem (so for ground terms) for which no complete TRS R (complete with respect to ground terms) exists. This strengthens the observation in Exercise 1.4.3. The simplest such (Σ, E) is the specification consisting of a single binary commutative operator $+$ and a constant 0, and equations $E = \{x + y = y + x\}$. According to Exercise 1.4.3 (which also works for the present simpler specification) no complete TRS R can be found such that for all (open) s, t we have $s =_R t \Leftrightarrow s =_E t$. According to the next exercise, we also have the stronger result that no TRS R exists which is complete for ground terms and such that for ground terms s, t we have $s =_R t \Leftrightarrow s =_E t$.

Exercise 1.4.6. (Bergstra & Klop) Prove the following fact:

THEOREM. *Let (Σ, E) be the specification with $\Sigma = \{0, +\}$ and $E = \{x + y = y + x\}$. Then there is no finite TRS R such that the restriction to ground terms, $(R)_0$, is complete and such that $=_R$ and $=_E$ coincide on ground terms.*

PROOF SKETCH. Define terms $t_0 \equiv 0$, $t_{n+1} \equiv t_n + t_n$ $(n \geq 0)$. Suppose R is a TRS with finitely many rewrite rules such that $=_R$ and $=_E$ coincide on ground terms. Let N be the maximum of the depths of the LHS's of the rewrite rules in R. (Here 'depth' refers to the height of the corresponding term formation tree.)

Consider the terms $t^* \equiv t_N + t_{2N}$ and $t^{**} \equiv t_{2N} + t_N$. Clearly, $t^* =_E t^{**}$. In fact, $\{t^*, t^{**}\}$ is an E-equivalence class, hence also an R-convertibility class. Therefore there must be a rewrite rule r such that t^* is an r-redex or t^{**} is an r-redex (since there are only two elements in the convertibility class) and such that $t^* \to_r t^{**}$. Say t^* is an r-redex. Now one can easily show that $t^* \to_r t^{**} \to_r t^*$. Hence R is not even SN on ground terms.

Term rewriting and initial algebra semantics

We will now make more explicit the connection between term rewriting and initial algebra semantics. We suppose familiarity with the concept of an initial algebra in the class of models of an equational specification (Σ, E), i.e. the variety $\mathrm{Alg}(\Sigma, E)$, as defined by universal properties in terms of homomorphisms. (See e.g. Meinke & Tucker [91], Goguen & Meseguer [85].) Although the initial algebra is only determined up to isomorphism, we will speak of 'the' initial algebra and use the notation $I(\Sigma, E)$ for it. It is well-known that $I(\Sigma, E)$ can be obtained from the set of ground terms $\mathrm{Ter}_0(\Sigma)$ by dividing out the congruence relation $=_E$. Thus we can equate the initial algebra $I(\Sigma, E)$ with the quotient algebra $\mathrm{Ter}_0(\Sigma)/=_E$.

Now suppose that (Σ, R) is a TRS 'for' (Σ, E), that is, $=_R$ coincides with $=_E$. (So the initial algebra of (Σ, E) van also be written as $\mathrm{Ter}_0(\Sigma)/=_R$.) If R is a *complete* TRS, then $I(\Sigma, E)$ is in fact a *computable* algebra. This is merely a rephrasing of: the word problem (for ground terms) for (Σ, E) is solvable. As noted in Exercise 1.4.6, the reverse is not necessarily the case; for some (Σ, E) with computable initial algebra there does not exist a

complete TRS—at least not *in the same signature*. However, a remarkable theorem of Bergstra and Tucker states that if we allow an extension of the signature with some functions and constants (no new sorts), then a complete TRS can always be found. (This result also follows from the simulation of Turing Machines by a TRS—consisting of two rules—as in Dershowitz [87].) More precisely:

Definition 1.4.7.

1. The algebra $\mathcal{A} \in \mathrm{Alg}(\Sigma, E)$ is *minimal*, if it is (isomorphic to) a quotient algebra $\mathrm{Ter}(\Sigma)/\equiv$ for some congruence $\equiv$. In particular, $I(\Sigma, E)$ is a minimal algebra. In other words, an algebra is minimal if its elements are generated by functions and constants in the signature.
2. A minimal algebra $\mathcal{A}$ is *computable*, if its equality is decidable, i.e. if the relation $\mathcal{A} \models t = s$ for ground terms $t, s \in \mathrm{Ter}(\Sigma)$ is decidable.

Theorem 1.4.8. (Bergstra & Tucker [80]) *Let $\mathcal{A}$ be a minimal Σ-algebra, Σ a finite signature. Then the following are equivalent:*

1. *$\mathcal{A}$ is a computable algebra;*
2. *there is an extension of Σ to a finite Σ', obtained by adding some function and constant symbols, and there is a complete TRS (Σ', R) such that*
$$\mathcal{A} \equiv I(\Sigma', R^{=}) \mid_{\Sigma} .$$

Here $R^{=}$ is the equational specification obtained by viewing the reduction rules in R as equations, and $\mid_{\Sigma}$ is the restriction to the signature Σ. So $\mathcal{A}$ is a '*reduct*' (see Meinke & Tucker [91]) of an initial algebra given by a complete TRS. (The TRS R as in the theorem is not only ground complete, but complete with respect to all terms. Actually, it is an orthogonal TRS as defined in the next chapter; and for orthogonal TRS's possessing at least one ground term, ground completeness implies completeness.) The functions (including the constants as 0-ary functions) to be added to Σ are sometimes referred to as '*hidden functions*'. Note that according to the statement in the theorem no new sorts are needed, thus the present theorem has also a bearing on the homogeneous (i.e. one-sorted) case that we are considering in this chapter.

For more information concerning the connection between term rewriting and computability aspects of initial algebra semantics (and 'final' algebra semantics), also for the heterogeneous (many-sorted) case, we refer to the very complete survey Goguen & Meseguer [85].

Critical pair completion

We resume the question how to find a complete TRS (for the case of open terms, henceforth) for an equational specification (Σ, E). This is in fact

what the Knuth-Bendix completion algorithm is trying to do. We will now explain the essential features of the completion algorithm first by an informal, "intuition-guided" completion of the equational specification E of groups:

$e \cdot x$	$=$	x
$I(x) \cdot x$	$=$	e
$(x \cdot y) \cdot z$	$=$	$x \cdot (y \cdot z)$

Table 1.12

First we give these equations a 'sensible' orientation:

1. $e \cdot x \to x$
2. $I(x) \cdot x \to e$
3. $(x \cdot y) \cdot z \to x \cdot (y \cdot z)$

(Note that the orientation in rules 1, 2 is forced, by the restrictions on rewrite rules in Section 1.1. As to the orientation of rule 3, the other direction is just as 'sensible'.) These rules are not confluent, as can be seen by superposition of e.g. 2 and 3. Redex $I(x) \cdot x$ can be unified with a *non-variable* subterm of redex $\underline{(x \cdot y)} \cdot z$ (the underlined subterm), with result $(I(x) \cdot x) \cdot z$. This term is subject to two possible reductions: $(I(x) \cdot x) \cdot z \to_2 e \cdot z$ and $(I(x) \cdot x) \cdot z \to_3 I(x) \cdot (x \cdot z)$. The pair of reducts $\langle e \cdot z, I(x) \cdot (x \cdot z) \rangle$ is called a *critical pair*, since the confluence property depends on the reduction possibilities of the terms in this pair. Formally, we have the following definition which at a first reading is not easily digested. For the concept of a 'most general unifier' we refer to Section 1.6 below.

Definition 1.4.9. Let $\alpha \to \beta$ and $\gamma \to \delta$ be two rewrite rules such that α is *unifiable* (after renaming of variables) with a subterm of γ which is not a variable (a non-variable subterm). This means that there is a context $C[\,]$, a non-variable term t and a 'most general unifier' σ such that $\gamma \equiv C[t]$ and $t^\sigma \equiv \alpha^\sigma$. The term $\gamma^\sigma \equiv C[t]^\sigma$ can be reduced in two possible ways: $C[t]^\sigma \to C[\beta]^\sigma$ and $\gamma^\sigma \to \delta^\sigma$. Now the pair of reducts $\langle C[\beta]^\sigma, \delta^\sigma \rangle$ is called a *critical pair* obtained by the superposition of $\alpha \to \beta$ on $\gamma \to \delta$. If $\alpha \to \beta$ and $\gamma \to \delta$ are the same rewrite rule, we furthermore require that α is unifiable with a proper (i.e. $\not\equiv \alpha$) non-variable subterm of $\gamma \equiv \alpha$.

Definition 1.4.10. A critical pair $\langle s, t \rangle$ is called *convergent* if s and t have a common reduct.

Our last critical pair $\langle e \cdot z, I(x) \cdot (x \cdot z) \rangle$ is not convergent: $I(x) \cdot (x \cdot z)$ is

a normal form and $e \cdot z$ only reduces to the normal form z. So we have the problematic pair of terms $z, I(x) \cdot (x \cdot z)$; problematic because their equality is derivable from E, but they have no common reduct with respect to the reduction available so far. Therefore we adopt a new rule

4. $I(x) \cdot (x \cdot z) \to z$

Now we have a superposition of rule 2 and 4: $I(I(y)) \cdot (I(y) \cdot y) \to_4 y$ and $I(I(y)) \cdot (I(y) \cdot y) \to_2 I(I(y)) \cdot e$. This yields the critical pair $\langle y, I(I(y)) \cdot e \rangle$ which cannot further be reduced. Adopt new rule:

5. $I(I(y)) \cdot e \to y$ *cancelled 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: $I(e) \cdot (e \cdot z) \to_4 z$ and $I(e) \cdot (e \cdot z) \to_1 I(e) \cdot z$. Adopt new rule:

6. $I(e) \cdot z \to z$ *cancelled later*

Superposition of 3, 5: $(I(Iy)) \cdot e) \cdot x \to_3 I(I(y)) \cdot (e \cdot x)$ and $(I(Iy)) \cdot e) \cdot x \to_5 y \cdot x$. Adopt new rule:

7. $I(Iy)) \cdot x \to y \cdot x$ *cancelled later*

Superposition of 5, 7: $I(I(y)) \cdot e \to_7 y.e$ and $I(I(y)) \cdot e \to_5 y$. Adopt new rule:

8. $y \cdot e \to y$

Superposition of 5, 8: $I(I(y)) \cdot e \to_5 y$ and $I(I(y)) \cdot e \to_8 I(I(y))$. Adopt new rule

9. $I(I(y)) \to y$ *cancel 5 and 7*

(Rule 5 is now no longer necessary to ensure that the critical pair $\langle y, I(I(y)) \cdot e \rangle$ has a common reduct, because: $I(I(y)) \cdot e \to_9 y \cdot e \to_8 y$. Likewise for rule 7.) Superposition of 6, 8: $I(e) \cdot e \to_6 e$ and $I(e) \cdot e \to_8 I(e)$. Adopt new rule

10. $I(e) \to e$ *cancel 6*

Superposition of 2, 9: $I(I(y)) \cdot I(y) \to_2 e$and $I(I(y)) \cdot I(y) \to_9 y \cdot I(y)$. Adopt new rule

11. $y \cdot I(y) \to e$

Superposition of 3, 11: $(y \cdot I(y)) \cdot x \to_3 y \cdot (I(y) \cdot x)$ and $(y \cdot I(y)) \cdot x \to_{11} e \cdot x$. Adopt new rule

12. $y \cdot (I(y) \cdot x) \to x$

Superposition (again) of 3, 11: $(x \cdot y) \cdot I(x \cdot y) \to_{11} e$ and $(x \cdot y) \cdot I(x \cdot y) \to_3 x \cdot (y \cdot I(x \cdot y))$. Adopt new rule

13. $x \cdot (y \cdot (y \cdot I(x \cdot y)) \to e$ *cancelled later*

Superposition of 13, 4: $I(x) \cdot (x \cdot (y \cdot I(x \cdot y))) \to_4 y \cdot I(x \cdot y)$ and $I(x) \cdot (x \cdot (y \cdot I(x \cdot y))) \to_{13} I(x) \cdot e$. Adopt new rule

14. $y \cdot I(x \cdot y) \to I(x)$ *cancelled later* *cancel 13*

Superposition of 4, 14: $I(y)\cdot(y\cdot I(x\cdot y)) \to_4 I(x\cdot y)$ and $I(y)\cdot(y\cdot I(x\cdot y)) \to_{14} I(y)\cdot I(x)$. Adopt new rule

15. $I(x \cdot y) \to I(y) \cdot I(x)$ *cancel 14*

At this moment the TRS has only convergent critical pairs. The significance of this fact is stated in the following lemma.

Lemma 1.4.11. (Critical Pair Lemma; Knuth & Bendix [70], Huet [80]) *A TRS R is WCR iff all critical pairs are convergent.*

Exercise 1.4.12. Prove the Critical Pair Lemma. (The proof is not hard, after distinguishing cases as in Figure 1.16, after Le Chénadec [86] where the proof also can be found. Some care has to be taken to deal with repeated variables in left-hand sides of reduction rules.)

Exercise 1.4.13. Prove, using the Critical Pair Lemma: if the TRS R has finitely many rules and is SN, then WCR and CR are decidable.

So the TRS R_c with rewrite rules as in Table 1.13 is WCR.

1.	$e \cdot x$	$\to$	x
2.	$I(x) \cdot x$	$\to$	e
3.	$(x \cdot y) \cdot z$	$\to$	$x \cdot (y \cdot z)$
4.	$I(x) \cdot (x \cdot z)$	$\to$	z
8.	$y \cdot e$	$\to$	y
9.	$I(I(y))$	$\to$	y
10.	$I(e)$	$\to$	e
11.	$y \cdot I(y)$	$\to$	e
12.	$y \cdot (I(y) \cdot x)$	$\to$	x
15.	$I(x \cdot y)$	$\to$	$I(y) \cdot I(x)$

Table 1.13

Furthermore, one can prove SN for R_c by a 'Knuth-Bendix ordering' (not treated here) or by the recursive path ordering explained in Section 1.3. (In fact we need the extended lexicographic version of Remark 1.3.11(3), due to the presence of the associativity rule.) According to Newman's Lemma (1.0.7(2)) R_c is therefore CR and hence complete. We conclude that the validity problem for the equational specification of groups is solvable.

The following theorem of Knuth and Bendix is an immediate corollary of the Critical Pair Lemma 1.4.11 and Newman's Lemma:

Corollary 1.4.14. (Knuth & Bendix [70]) *Let R be a TRS which is SN. Then R is CR iff all critical pairs of R are convergent.*

Figure 1.16

The completion procedure above 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: $z \to I(x) \cdot (x \cdot z)$ 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 $I(y) \cdot I(x) \to I(x \cdot y)$ 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 'reduction ordering' on the terms.

Definition 1.4.15. A *reduction ordering* $>$ is a well-founded partial ordering on terms, which is closed under substitutions and contexts, i.e. if $s > t$ then $s^\sigma > t^\sigma$ for all substitutions σ, and if $s > t$ then $C[s] > C[t]$ for all contexts C[ ].

We now have immediately the following fact (noting that if R is SN, then $\rightarrow_R^+$ satisfies the requirements of Definition 1.4.15):

Proposition 1.4.16. *A TRS R is SN iff there is a reduction ordering $>$ such that $\alpha > \beta$ for every rewrite rule $\alpha \rightarrow \beta$ of R.*

Simple version of the Knuth-Bendix completion algorithm

Input: – an equational specification (Σ, E)
– a reduction ordering $>$ on $\mathrm{Ter}(\Sigma)$
(i.e. a program which computes $>$)
Output: - a complete TRS R such that for all
$s, t \in \mathrm{Ter}(\Sigma) : s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

$R := \varnothing$;
while $E \neq \varnothing$ **do**
  choose an equation $s = t \in E$;
  reduce s and t to respective normal forms s' and t'
  with respect to R;
  if $s' \equiv t'$ **then**
    $E := E - \{s = t\}$
  else
    if $s' > t'$ **then**
      $\alpha := s'$; $\beta := t'$
    else if $t' > s'$ **then**
      $\alpha := t'$; $\beta := s'$
    else
      failure
    fi;
    $CP := \{P = Q \mid \langle P, Q \rangle$ is a critical pair between
        the rules in R and $\alpha \rightarrow \beta\}$;
    $R := R \cup \{\alpha \rightarrow \beta\}$;
    $E := E \cup CP - \{s = t\}$
  fi
od;
success

Figure 1.17

In Figure 1.17 a simple version of the Knuth-Bendix completion algorithm is presented. As to the reduction ordering $>$ on $\mathrm{Ter}(\Sigma)$ which is an input to the algorithm: finding this is a matter of ingenuity, or experimentation. (Also without reduction ordering, computer systems for Knuth-Bendix completion equipped with an interactive question for orientation of equations into rewrite rules are of great help.)

The program of Figure 1.17 has three possibilities: it may (1) terminate successfully, (2) loop infinitely, or (3) fail because a pair of terms s, t cannot be oriented (i.e. neither $s > t$ nor $t > s$). The third case gives the most important restriction of the Knuth-Bendix algorithm: equational specifications with commutative operators cannot be completed.

Exercise 1.4.17. Show that there exists no complete TRS for the specification of abelian groups as in Table 1.14. (Consider in a supposed complete TRS the normal forms of the open terms $x + y$ and $y + x$.)

$0 + x$	$=$	x
$(-x) + x$	$=$	0
$(x + y) + z$	$=$	$x + (y + z)$
$x + y$	$=$	$y + x$

Table 1.14

If one still wants to deal with equational specifications having commutative/associative operators as in Exercise 1.4.17, one has to work modulo the equations of associativity and commutativity. For completion modulo such equations we refer to Peterson & Stickel [81] and Jouannaud & Kirchner [86].

In case (1) the resulting TRS is complete. To show this requires a non-trivial proof, see e.g. Huet [81]. In the next section we will give an abstract formulation of Knuth-Bendix completion, following Bachmair, Dershowitz & Hsiang [86], which streamlines considerably this kind of correctness proofs.

The completion program of Figure 1.17 does not 'simplify' the rewrite rules themselves. Such an optimization can be performed after termination of the program, as follows.

Definition 1.4.18. A TRS R is called *irreducible* if for every rewrite rule $\alpha \to \beta$ of R the following holds:

1. β is a normal form with respect to R,
2. α is a normal form with respect to $R - \{\alpha \to \beta\}$.

Exercise 1.4.19. Prove that every irreducible ground TRS is complete. (Hint: use Exercise 1.2.19 and Corollary 1.4.14.)

Theorem 1.4.20. (Métivier [83]) *Let R be a complete TRS. Then we can find an irreducible complete TRS R' such that the convertibilities $=_R$ and $=_{R'}$ coincide.*

Exercise 1.4.21. A proof of Theorem 1.4.20 can be given along the following line. Let R_1 be the TRS $\{\alpha \to \beta' \mid \alpha \to \beta \in R$ and β' is the normal form of β with respect to $R\}$. We may assume that R_1 does not contain rewrite rules that are a renaming of another rewrite rule. Further, define $R' = \{\alpha \to \beta \in R_1 \mid \alpha$ is a normal form with respect to $R_1 - \{\alpha \to \beta\}\}$. Now the proof that $s =_R t \Leftrightarrow s =_{R'} t$ follows from the (easy) proofs of the sequence of statements:

1. if $s \to_{R_1} t$ then $s \to_R^+ t$;
2. R and R_1 define the same set of normal forms;
3. R_1 is SN;
4. if $s \twoheadrightarrow_R t$ and t is a normal form then $s \twoheadrightarrow_{R_1} t$;
5. $s =_R t \;\Leftrightarrow\; s =_{R_1} t$;
6. R_1 is CR;
7. if $s \to_{R'} t$ then $s \to_{R_1} t$;
8. R_1 and R' define the same set of normal forms;
9. R' is SN;
10. if $s \twoheadrightarrow_{R_1} t$ and t is a normal form then $s \twoheadrightarrow_{R'} t$;
11. $s =_{R_1} t \;\Leftrightarrow\; s =_{R'} t$;
12. R' is CR;
13. R' is irreducible.

Instead of optimizing the TRS which is the output of the above simple completion algorithm *after* the completion, it is more efficient to do this *during* the completion. Figure 1.18 contains a more efficient Knuth-Bendix completion algorithm, which upon successful termination yields irreducible TRS's as output.

We conclude this section with a theorem stating that the Knuth-Bendix completion algorithm, given an equational specification and a reduction ordering, cannot generate two different complete irreducible TRS's. According to Dershowitz, Marcus & Tarlecki [88] the theorem is originally due to M. Ballantyne, but first proved in Métivier [83].

Definition 1.4.22. Let $>$ be a reduction ordering. We call a TRS R *compatible* with $>$ if for every rewrite rule $\alpha \to \beta$ of R we have $\alpha > \beta$.

More efficient version of the Knuth-Bendix completion algorithm

Input: – an equational specification (Σ, E)
– a reduction ordering $>$ on $\mathrm{Ter}(\Sigma)$
Output: - a complete irreducible TRS R such that for all $s,t \in \mathrm{Ter}(\Sigma) : s =_R t \Leftrightarrow (\Sigma, E) \vdash s = t$

$R := \varnothing$;
while $E \neq \varnothing$ **do**
choose an equation $s = t \in E$;
reduce s and t to respective normal forms

while $E \neq \varnothing$ **do**
choose an equation $s = t \in E$;
reduce s and t to respective normal forms s' and t'
with respect to R;
if $s' \equiv t'$ **then**
$E := E - \{s = t\}$
else
if $s' > t'$ **then**
$\alpha := s'$; $\beta := t'$
else if $t' > s'$ **then**
$\alpha := t'$; $\beta := s'$
else
failure
fi;
$R := \{\gamma \to \delta' \mid \gamma \to \delta \in R$ and δ' is a normal form of δ with respect to $R \cup \{\alpha \to \beta\}\}$;
$CP := \{P = Q \mid \langle P, Q\rangle$ is a critical pair between the rules in R and $\alpha \to \beta\}$;
$E := E \cup CP \cup \{\gamma = \delta \mid \gamma \to \delta \in R$ and γ is reducible by $\alpha \to \beta\} - \{s = t\}$;
$R := R \cup \{\alpha \to \beta\} - \{\gamma \to \delta \mid \gamma$ is reducible by $\alpha \to \beta\}$
fi
od;
success

Figure 1.18

Theorem 1.4.23. (Métivier [83]) *Let R_1 and R_2 be two complete irreducible TRS's compatible with a given reduction ordering $>$. Suppose R_1 and R_2 define the same convertibility. Then R_1 and R_2 are equal (modulo a renaming of variables).*

Exercise 1.4.24. (Huet [80]) In this exercise we collect some criteria for confluence in terms of properties of critical pairs, as well as some counterexamples,

from Huet [80]. Also some questions are listed which are, as far as we know, open. See Table 1.15.

Table 1.15

1. In row 1 of the table the Critical Pair Lemma 1.4.11 is stated: if every critical pair $\langle t, s\rangle$ is convergent (notation: $t \downarrow s$), then WCR holds. However, CR need not to hold; a counterexample is given by the TRS with four constants a, b, c, d and rules as in Figure 1.3.
2. Row 2 of the table is Theorem 1.4.14 of Knuth and Bendix.
3. In row 3, LL means that the TRS is left-linear, RL right-linear (i.e. no right-hand side of a reduction rule contains repetitions of a variable). Strongly confluent is defined in Exercise 1.0.8(10).

We furthermore define:
DEFINITION. A TRS is *strongly closed* if for every every critical pair $\langle t, s\rangle$ there are t', t'' such that $t \twoheadrightarrow t' \leftarrow^{\equiv} s$ and $s \twoheadrightarrow t'' \leftarrow^{\equiv} t$. Prove that 'strongly closed' is not sufficient to guarantee CR, by considering the non-left-linear TRS $\{F(x,x) \to A,\ F(x, G(x)) \to B,\ C \to G(C)\}$. However, if the TRS is left-linear, right-linear and strongly closed, then CR holds (for a proof see Huet [80]); in fact, we then have strong confluence.

4. In 3, RL cannot be dropped. A nice counterexample is in Huet [80], given by J.-J. Lévy: it contains the following eight left-linear rules. See also Figure 1.19.

$$\begin{array}{lcllcl} F(A,A) & \to & G(B,B) & G(B,B) & \to & F(A,A) \\ A & \to & A' & B & \to & B' \\ F(A',x) & \to & F(x,x) & G(B',x) & \to & G(x,x) \\ F(x,A') & \to & F(x,x) & G(x,B') & \to & G(x,x) \end{array}$$

Check that CR does not hold, and that the TRS is strongly closed.

Figure 1.19

5. This is a remarkable fact: if the TRS is left-linear, and for every critical pair $\langle t, s \rangle$ we have $t \rightarrow_{||} s$, then $\mathrm{WCR}^{\leq 1}$ holds, and hence CR. Here $\rightarrow_{||}$ (parallel reduction) denotes a sequence of redex contractions at disjoint occurrences.

6,7,8. If in 5 we replace $t \rightarrow_{||} s$ by $s \rightarrow_{||} t$, then the CR question is open. Likewise (7) if $t \rightarrow_{||} s$ is replaced by $s \rightarrow^{\equiv} t$, or (8) replaced by: "$t \rightarrow^{\equiv} s$ or $s \rightarrow^{\equiv} t$".

Exercise 1.4.25. Knuth & Bendix [70] contains completions of two specifications which closely resemble the specification of groups (see Table 1.16), called 'L-R theory' and 'R-L theory'. Prove, using the completions, that $x \cdot e = x$ is not derivable in L-R theory and that in R-L theory the equations $e \cdot x = x$ and $x \cdot I(x) = e$ are not derivable. Furthermore, in L-R theory the equation $x \cdot e = x$ is not derivable. Hence the three theories are different, i.e. determine different varieties of algebras. In fact, note that the variety of groups is a proper subset of both the variety of L-R algebras and that of R-L algebras, and that the latter two varieties are incomparable with respect to set inclusion.

1.5 An abstract formulation of completion

(This section is taken from Klop & Middeldorp [88].)

There are many completion algorithms such as the two above (in Figures 1.17 and 1.18), differing in order of execution or ways of optimization. The question is, how to prove that these algorithms are correct, i.e. deliver upon successful termination indeed a TRS R with the same equality as the one generated by the original set of equations E. As there is a whole family of completion algorithms, one needs to extract the 'abstract principles' of such algorithms; and this is done indeed by Bachmair, Dershowitz & Hsiang [86]. Their method for proving correctness of completion algorithms starts with the introduction of a derivation system where the objects are pairs

group theory	*L-R theory:*	*R-L theory:*
$e \cdot x = x$	$e \cdot x = x$	$x \cdot e = x$
$I(x) \cdot x = e$	$x \cdot I(x) = e$	$I(x) \cdot x = e$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
completion:	*completion:*	*completion:*
$e \cdot x \to x$	$e \cdot x \to x$	
$x \cdot e \to x$		$x \cdot e \to x$
$I(x) \cdot x \to e$		$I(x) \cdot x \to e$
$x \cdot I(x) \to e$	$x \cdot I(x) \to e$	
$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$	$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$	$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$
$I(e) \to e$	$I(e) \to e$	$I(e) \to e$
$I(x \cdot y) \to I(y) \cdot I(x)$	$I(x \cdot y) \to I(y) \cdot I(x)$	$I(x \cdot y) \to I(y) \cdot I(x)$
$x \cdot (I(x) \cdot y) \to y$	$x \cdot (I(x) \cdot y) \to y$	
		$e \cdot x \to I(I(x))$
$I(x) \cdot (x \cdot y) \to y$	$I(x) \cdot (x \cdot y) \to y$	
		$x \cdot I(I(y)) \to x \cdot y$
$I(I(x)) \to x$		
	$x \cdot e \to I(I(x))$	
	$I(I(I(x))) \to I(x)$	$I(I(I(x))) \to I(x)$
		$x \cdot (y \cdot I(y)) \to x$
	$I(I(x)) \cdot y \to x \cdot y$	
		$x \cdot (I(I(y)) \cdot z) \to x \cdot (y \cdot z)$
		$x \cdot (y \cdot (I(y) \cdot z)) \to x \cdot z$
		$I(x) \cdot (x \cdot y) \to I(I(y))$

Table 1.16

(E, R); each derivation step from (E, R) to (E', R') preserves equality: $=_{E \cup R}$ coincides with $=_{E' \cup R'}$, and moreover, along a sequence of derivations the actual proofs of equations $s = t$ will be getting 'better and better', with as optimal proof format that of a "rewrite proof". See Figure 1.20, where it is shown how E (that is the pair $(E, \varnothing)$) is gradually transformed via pairs (E', R') to a TRS R (that is the pair $(\varnothing, R)$); along the way the two example proofs in Figure 1.20 get more and more oriented until they are in rewrite form. (Here direction is downward; horizontal steps are without direction.)

There are two crucial ideas in this recent approach. One is the concept of a derivation system on pairs (E, R) as discussed above. The other is the concept of ordering the proofs of equations $s = t$ according to their degree of orientation. We will now proceed to a more formal explanation.

Figure 1.20

Definition 1.5.1. Let (Σ, E) be an equational specification. If $s =_E t$ by application of exactly one equation in E we write $s \leftrightarrow_E t$. So $s \leftrightarrow_E t$ iff there exists a context $C[\ ]$, a substitution σ and an equation $u = v$ (or $v = u$) in E such that $s \equiv C[u^\sigma]$ and $t \equiv C[v^\sigma]$.

Definition 1.5.2. Let (Σ, E) be an equational specification and R a TRS with signature Σ.

1. A *proof* in $E \cup R$ of an equation $s = t$ between terms $s, t \in \mathrm{Ter}(\Sigma)$ is a sequence of terms $(s_0, \ldots, s_n)$ such that $s_0 \equiv s, s_n \equiv t$, and for all $0 < i \leq n$ we have $s_{i-1} \leftrightarrow_E s_i$, $s_{i-1} \rightarrow_R s_i$ or $s_{i-1} \leftarrow_R s_i$.
2. A *subproof* of $P \equiv (s_0, \ldots, s_n)$ is a proof $P' \equiv (s_i, \ldots, s_j)$ with $0 \leq i \leq j \leq n$. The notation $P[P']$ means that P' is a subproof of P. (Actually, as occurrence of a subproof of P.)
3. A proof of the form $s_0 \twoheadrightarrow_R s_k \twoheadleftarrow_R s_n$ is called a *rewrite proof*.

By definition, $P \equiv (s)$ is a proof of $s = s$. Figure 1.21 contains an example of a proof.

Knuth-Bendix completion aims at transforming every proof $(s_0, \ldots, s_n)$

Figure 1.21

into a rewrite proof $s_0 \twoheadrightarrow t \twoheadleftarrow s_n$. We now present an inference system for Knuth-Bendix completion. The objects of this system are pairs (E, R). The inference system $\mathcal{BC}$ (*basic completion*) has the following rules (see Table 1.17); $>$ is a reduction ordering.

(C_1)	*orienting an equation*			
	$(E \cup' \{s =' t\}, R)$	$\Rightarrow$	$(E, R \cup \{s \to t\})$	if $s > t$
(C_2)	*adding an equation*			
	(E, R)	$\Rightarrow$	$(E \cup \{s = t\}, R)$	if $s \leftarrow_R u \rightarrow_R t$
(C_3)	*simplifying an equation*			
	$(E \cup' \{s =' t\}, R)$	$\Rightarrow$	$(E \cup \{u = t\}, R)$	if $s \rightarrow_R u$
(C_4)	*deleting a trivial equation*			
	$(E \cup' \{s = s\}, R)$	$\Rightarrow$	(E, R)	

Table 1.17

The notation $s =' t$ means $s = t$ or $t = s$; the symbol $\cup'$ denotes disjoint union. A $\mathcal{BC}$-derivation is a finite or infinite sequence $(E_0, R_0) \Rightarrow (E_1, R_1) \Rightarrow (E_2, R_2) \Rightarrow \cdots$. We write $\Rightarrow^+$ for the transitive closure of $\Rightarrow$.

It is easily seen that, given a derivation step $(E, R) \Rightarrow (E', R')$, if R is SN then so is R' and furthermore $=_{E \cup R}$ coincides with $=_{E' \cup R'}$. However, proofs in $E' \cup R'$ are in general 'simpler' than in $E \cup R$. For example, by adding equations to E by inference rule C_2 some subproofs $s \leftarrow_R u \rightarrow_R t$ can be replaced by $s \leftrightarrow_{E'} t$. To formalize this reduction in complexity we introduce orderings on proofs.

Definition 1.5.3. A binary relation $\rightarrowtail$ on proofs is *monotonic* if $Q \rightarrowtail Q'$ implies $P[Q] \rightarrowtail P[Q']$ for all proofs P, Q and Q'. The relation $\rightarrowtail$ is *stable* if

$$P \equiv (s, \ldots, u_i, \ldots, t) \rightarrowtail (s, \ldots, v_j, \ldots, t) \equiv Q$$

implies that

$$(C[s^\sigma], \ldots, C[u_i^\sigma], \ldots, C[t^\sigma]) \rightarrowtail (C[s^\sigma], \ldots, C[v_j^\sigma], \ldots, C[t^\sigma])$$

for all proofs P and Q, contexts $C[\,]$ and substitutions σ. A *proof ordering* is a stable, monotonic, well-founded partial ordering on proofs.

Exercise 1.5.4. To illustrate the concept of proof ordering we will give an alternative proof of Newman's Lemma 1.0.7(2) using this notion. ('Alternative' with respect to the proofs that we have seen in the literature. The present proof is nevertheless well-known.) See also Exercise 1.3.15 for our multiset notations.

Let R be a TRS which is SN and WCR. Let $P \equiv (s_0, \ldots, s_n)$ be a proof of the conversion $s_0 = s_n$. We define the *complexity* $\mid P \mid$ of the proof P as the multiset $[s_0, \ldots, s_n]$. The ordering $\rightarrowtail$ which we will use is induced by the multiset extension of $\rightarrow_R^+$, notation: $(\rightarrow_R^+)^\mu$. So

$$P \rightarrowtail P' \text{ iff } \mid P \mid (\rightarrow_R^+)^\mu \mid P' \mid .$$

(This means that $P \rightarrowtail P'$ if the multiset $\mid P' \mid$ arises from the multiset $\mid P \mid$ by repeatedly replacing an element of the multiset by arbitrarily many elements which are less in the sense of the well-founded ordering $\rightarrow_R^+$. I.e. by repeatedly replacing a term t in the multiset of terms by a number (≥ 0) of proper reducts of t.)

1. Prove that $\rightarrowtail$ is a proof ordering.
2. If $P \equiv (s_0, \ldots, s_n)$ is not a rewrite proof, then there is a proof P' of the equation $s_0 = s_n$ such that $P \rightarrowtail P'$. (Hint: consider a 'peak' in the conversion P, and replace it by a 'valley', using WCR. See Figure 1.22.)
3. Conclude that R is CR.

Figure 1.22

The proof ordering which we use for completion is based on the given reduction ordering and on the elementary steps ($\rightarrow_R, \leftarrow_R$ or $\leftrightarrow_E$) in a proof.

Definition 1.5.5.

1. The *complexity* $\mid P \mid$ of a proof $P \equiv (s_0, \ldots, s_n)$ is the multiset $[c(s_0, s_1), \ldots, c(s_{n-1}, s_n)]$ where $c(s_{i-1}, s_i)$, the complexity of an elementary proof step, is defined by

$$c(s_{i-1}, s_i) = \begin{cases} [s_{i-1}] & \text{if } s_{i-1} \to_R s_i \\ [s_i] & \text{if } s_{i-1} \leftarrow_R s_i \\ [s_{i-1}, s_i] & \text{if } s_{i-1} \leftrightarrow_E s_i \end{cases}$$

2. To compare the complexities of the elementary proof steps we use the multiset extension $>^\mu$ of the reduction ordering $>$. (See Exercise 1.3.15.) To compare proof complexities we use the multiset extension of $>^\mu$, notation: $>^{\mu\mu}$. Now we define:

$$P \rightarrowtail_{\mathcal{BC}} P' \;\Leftrightarrow\; \mid P \mid >^{\mu\mu} \mid P' \mid .$$

Definition 1.5.6. A proof ordering $\rightarrowtail$ is *compatible with* $\mathcal{BC}$ if $(E, R) \Rightarrow^+ (E', R')$ implies that for every proof P in $E \cup R$ of an equation $s = t$ there exists a proof P' of $s = t$ in $E' \cup R'$ such that $P \rightarrowtail P'$ or $P \equiv P'$.

The following proposition has a straightforward proof, which follows from considering Figure 1.23 and applying stability and monotonicity of $\rightarrowtail_{\mathcal{BC}}$. Figure 1.23 suggests how proofs are reduced in complexity by application of a transformation step according to $C_1, \ldots, C_4$. For instance, in the case of C_2 (see Figure 1.23) the complexity of the subproof $t \leftarrow_R s \to_R u$ is $[[s], [s]]$ which decreases to the complexity of the subproof $t \leftrightarrow_R u$, namely $[[t, u]]$. This is indeed a decrease since $[s] >^\mu [t, u]$.

Proposition 1.5.7. *The ordering* $\to_{\mathcal{BC}}$ *is a proof ordering, which moreover is compatible with* $\mathcal{BC}$.

So in a $\mathcal{BC}$-derivation $(E_0, R_0) \Rightarrow (E_1, R_1) \Rightarrow (E_2, R_2) \Rightarrow \cdots$ the proofs in $E_j \cup R_j$ are no more difficult than corresponding proofs in $E_i \cup R_i$, for all $j > i$. The following fairness property of BC-derivations implies that moreover every proof in $E_i \cup R_i$ of an equation $s = t$ which is not yet a rewrite proof, can be simplified to a rewrite proof of $s = t$ in $E_j \cup R_j$ for some $j > i$.

Definition 1.5.8. A $\mathcal{BC}$-derivation $(E_0, R_0) \Rightarrow (E_1, R_1) \Rightarrow (E_2, R_2) \Rightarrow \cdots$ is called *fair* if

1. $\bigcap_{j>i} E_j = \varnothing$ for all $i \geq 0$, and
2. if $\langle c, d \rangle \in \bigcap_{j \geq i} CP_j$ for some $i \geq 0$ then $c = d \in E_k$ for some $k \geq 0$. (CP_j is the set of all critical pairs between the rewrite rules of R_j.)

So, according to (2) every critical pair which arises will be (or was) an equation at some time, and by (1) every equation will be 'considered'

Figure 1.23

eventually, that is, oriented in a rewrite rule, simplified, or deleted. The following fact can now be proved routinely.

Proposition 1.5.9. *Let* $(E_0, R_0) \Rightarrow (E_1, R_1) \Rightarrow (E_2, R_2) \Rightarrow \cdots$ *be a fair* $\mathcal{BC}$*-derivation and let* P *be a proof of* $s = t$ *in* $E_i \cup R_i$*. If* P *is not yet a rewrite proof then for some* $j \geq i$ *there exists a proof* P' *in* $E_j \cup R_j$ *of* $s = t$ *such that* $P \rightarrowtail_{\mathcal{BC}} P'$*.*

By a *completion procedure* we mean a strategy for applying the inference rules of $\mathcal{BC}$ to inputs (Σ, E) and reduction ordering $>$, in order to generate a $\mathcal{BC}$-derivation $(E_0, R_0) \Rightarrow (E_1, R_1) \Rightarrow \cdots$ with $(E_0, R_0) = (E, \varnothing)$. Because for some inputs a fair derivation may not be possible, we allow for a completion procedure to fail. We say that a completion procedure is *fair* if it generates only fair derivations unless it fails. We now have:

Theorem 1.5.10. (Bachmair, Dershowitz & Hsiang [86]) *Let* **C** *be a fair completion procedure that does not fail on input* (Σ, E) *and* $>$*.*

1. *If* $s =_E t$ *then* **C** *will generate a pair* (E_i, R_i) *such that* s *and* t *have*

a common reduct in R_i.

2. R^∞ $(= \bigcup_n R_n)$ *is a complete TRS.*

1.6 Unification

In the preceding sections about completion algorithms, we have used as a 'subroutine' the determination of a *most general unifier* of two terms. In the present section we will describe a version of a unification algorithm, due to Martelli & Montanari [82]; this nondeterministic algorithm to compute mgu's is itself phrased in the terminology of rewriting. We start with presenting the rewrite rules for 'syntactic unification', and afterwards extend these rules to include 'semantic unification' or 'E-unification'.

Syntactic unification

Before presenting the syntactic unification algorithm, we introduce some more concepts about substitutions, which were defined in Section 1.1 as homomorphisms (with respect to term formation) from the set of terms $\mathrm{Ter}(R)$ of the TRS R to $\mathrm{Ter}(R)$. The composition of substitutions σ, τ is the usual one for functions: $(\tau \circ \sigma)(t) = \tau(\sigma(t))$ for $t \in \mathrm{Ter}(R)$; however, $\tau \circ \sigma$ will be written as $\sigma\tau$, and in accordance with our earlier notation convention, $\sigma(t)$ as t^σ. Note that this notation is unambiguous: $(t^\sigma)^\tau = t^{(\sigma\tau)}$.

The *support* of substitution σ is the restriction of σ to the set of those variables x_i for which $x_i \not\equiv x_i^\sigma$. Usually, the support will be finite, and in this case we write σ (by some 'abus de langage') as its support, which is a finite list of 'bindings' of terms to variables:

$$\{x_{i_1} := t_i, \ldots, x_{i_n} := t_n\}.$$

A *renaming substitution* is a bijective substitution. This implies that a renaming, restricted to the set of variables $Var = \{x_i \mid i \geq 0\}$, is a permutation of Var. Note that the composition $\sigma\tau$ of renamings σ, τ is again a renaming, and that the inverse σ^{-1} of a renaming σ exists and is again a renaming. Terms s, t differing a renaming, i.e. $t^\sigma \equiv s$ for some renaming σ, are called *variants* (of each other).

If t, s are terms such that $t^\sigma \equiv s$ for some substitution σ, we write $t \leq s$. The relation $\leq$ is not yet a partial ordering; it is a quasi-ordering, also called the *subsumption* relation. One easily proves for all $s, t \in \mathrm{Ter}(R)$: $s \leq t \;\&\; t \leq s \Leftrightarrow s, t$ are variants. For substitutions σ, τ we write $\sigma \leq \tau$ if $\tau = \sigma\rho$ for some substitution ρ. In this case σ is called *more general than* τ. (The 'overloading' of the symbol $\leq$ will cause no confusion.) Analogous to the case of terms, one easily proves: $\sigma \leq \tau \;\&\; \tau \leq \sigma \Leftrightarrow \sigma, \tau$ differ a renaming ($\sigma\rho = \tau$ for some renaming ρ).

We call σ a *unifier* of a set of terms $\mathcal{T} = \{t_1, \ldots, t_n\}$ if $t_1^\sigma \equiv t_n^\sigma$. It is a *most general unifier* (mgu) of $\mathcal{T}$ if for every unifier τ of $\mathcal{T}$ we have $\sigma \leq \tau$. Each finite set of terms which can be unified (has a unifier) has a mgu; it is unique *modulo renamings*.

The task of finding a most general unifier of two terms $F(t_1, \ldots, t_n)$ and $F(s_1, \ldots, s_n)$ can be viewed as the task of solving the set of equations $\{t_1 = s_1, \ldots, t_n = s_n\}$. A very elegant algorithm exploiting this representation was given by Martelli-Montanari [82]. It consists of rules which transform one set of equations into another one. To conform with the notation in 'equational logic programming' as in Hölldobler [89], we write instead of $\{t_1 = s_1, \ldots, t_n = s_n\}$:

$$\Leftarrow t_1 = s_1, \ldots, t_n = s_n,$$

called also an *equational goal*. The empty goal (empty set of equations) will be denoted as $\Box$. The algorithm to be presented transforms, nondeterministically, goals into goals; just as in logic programming we intend to end a sequence of transformations in the empty goal:

$$G_0 \rightarrowtail G_1 \rightarrowtail \cdots \rightarrowtail \Box.$$

Here $\rightarrowtail$ denotes an elementary 'derivation' step; $G_0, G_1, \ldots$ are equational goals. Actually, at some of the $\rightarrowtail$-steps we may obtain as a 'side-effect' a substitution σ; it will be denoted as a subscript, so that such a step has the form $G \rightarrowtail_\sigma G'$. So a derivation may have the form, e.g.:

$$G_0 \rightarrowtail G_1 \rightarrowtail_{\sigma_1} G_2 \rightarrowtail_{\sigma_2} G_3 \rightarrowtail G_4 \rightarrowtail G_5 \rightarrowtail_{\sigma_5} G_6 \rightarrowtail \cdots \rightarrowtail \Box.$$

Derivation sequences ending in $\Box$ are *successful*; it will also be possible that a derivation is stuck and cannot be prolonged to reach $\Box$, because no transformation rule applies. In that case we conclude the sequence after the goal where the sequence got stuck, with the symbol $\blacksquare$ (for 'failure'):

$$G_0 \rightarrowtail G_1 \rightarrowtail \cdots \rightarrowtail \quad \blacksquare.$$

In the case of a successful derivation, we can obtain the 'harvest' by composing all the substitutions that are found, in their order of appearance; in the example above: $\sigma_1\sigma_2\sigma_5\cdots$. This substitution is the *computed answer substitution* of the successful derivation that we are considering.

We will now present the four derivation rules for equational goals that together constitute a unification algorithm. With some adaptations, these 'Martelli-Montanari rules' (MM-rules) are as follows. Here $\Leftarrow t = s,\ E$ stands for an equational goal containing the equation $t = s$; with E we denote the remaining equations in the goal.

1. **Term decomposition**

$$\begin{array}{l} \Leftarrow F(t_1,\ldots,t_n) = F(s_1,\ldots,s_n),\ E \\ \qquad\qquad \rightarrowtail \\ \Leftarrow t_1 = s_1,\ldots,t_n = s_n,\ E \end{array}$$

2. **Removal of trivial equations**

$$\Leftarrow x = x,\ E \quad \rightarrowtail \quad \Leftarrow E$$

3. **Swap**

$$\begin{array}{l} \Leftarrow t = x,\ E \quad \rightarrowtail \quad \Leftarrow x = t,\ E \\ \textit{if } t \textit{ is not a variable} \end{array}$$

4. **Variable elimination**

$$\begin{array}{l} \Leftarrow x = t,\ E \quad \rightarrowtail_{\{x:=t\}} \quad \Leftarrow E^{\{x:=t\}} \\ \textit{if } x \notin t \end{array}$$

(If E is $t_1 = s_1,\ldots,t_n = s_n$, then E^σ is $t_1^\sigma = s_1^\sigma,\ldots,t_n^\sigma = s_n^\sigma$. With '$\in$' we abbreviate 'occurs in'. Note that only in transformation rule (4) a substitution is delivered.)

We have the following well-known 'completeness' theorem:

Theorem 1.6.1. (Unification Theorem) *Let G be an equational goal $\Leftarrow t_1 = s_1,\ldots,t_n = s_n$. Then the following are equivalent:*

1. *the equations in G can be unified;*
2. *there is a mgu σ such that $t_1^\sigma \equiv s_1^\sigma,\ldots,t_n^\sigma \equiv s_n^\sigma$;*
3. *the derivation tree with root G and constructed with the MM rules is finite and has only success branches, all yielding an mgu of the equations in G as computed answer substitution.*

Furthermore, if the equations in G cannot be unified, the MM-derivation tree with root G is also finite, but now with all branches ending unsuccessfully.

(It will be clear what is meant in the statement of the theorem above with derivation *tree*; it arises because the rules can be applied nondeterministically.) In the original presentation of Martelli and Montanari, the following two rules are also included; they enhance efficiency, by pruning the MM-derivation tree of some unsuccessful subtrees. But we don't need them for the completeness of this (nondeterministic) unification algorithm. (Also, when extending the set of rules to deal with E-unification, as we will do below, (5) and (6) must be omitted.)

5. **Failure rule**

$$\Leftarrow\ F(t_1,\ldots,t_n) = G(s_1,\ldots,s_m),\ E \ \ \rightarrowtail\ \ \blacksquare$$

6. **Occur check**

$$\Leftarrow\ x = t,\ E \ \ \rightarrowtail\ \ \blacksquare$$
$$\textit{if } x \not\equiv t \textit{ and } x \in t$$

It is not hard to prove that the MM rules are indeed terminating, as stated by the Unification Theorem. (See Martelli-Montanari [82], Apt [90], or Dershowitz & Jouannaud [90].)

If t, s are unifiable terms we will denote with $mgu(s,t)$ a particular mgu of $\{s,t\}$, obtained by performing the MM transformations according to some fixed strategy.

Example 1.6.2.

1. We want to determine 'the' mgu of the terms $F(G(x), H(x,u))$ and $F(z, H(F(y,y),z))$. The MM rules yield the following successful derivation:

$$\begin{array}{ll}
\Leftarrow\ F(G(x),H(x,u)) = F(z,H(F(y,y),z)) & \rightarrowtail_{(1)} \\
\Leftarrow\ G(x) = z,\ H(x,u) = H(F(y,y),z) & \rightarrowtail_{(3)} \\
\Leftarrow\ z = G(x),\ H(x,u) = H(F(y,y),z) & \rightarrowtail_{(4),\{z:=G(x)\}} \\
\Leftarrow\ H(x,u) = H(F(y,y),G(x)) & \rightarrowtail_{(1)} \\
\Leftarrow\ x = F(y,y),\ u = G(x) & \rightarrowtail_{(4),\{x:=F(y,y)\}} \\
\Leftarrow\ u = G(F(y,y)) & \rightarrowtail_{(4),\{u:=G(F(y,y))\}} \\
\Box &
\end{array}$$

with computed answer substitution $\{z := G(x)\}\{x := F(y,y)\}\{u := G(F(y,y))\} = \{z := G(F(y,y)), x := F(y,y), u := G(F(y,y))\}$. Indeed this is a mgu of the original pair of terms.

2. A failing unification attempt:

$$\begin{array}{ll}
\Leftarrow\ F(x,y) = F(y,G(x)) & \rightarrowtail_{(1)} \\
\Leftarrow\ x = y,\ y = G(x) & \rightarrowtail_{(2),\{x:=y\}} \\
\Leftarrow\ y = G(y) & \rightarrowtail_{(6)} \\
\blacksquare &
\end{array}$$

Semantic unification

In the previous section we have presented an algorithm to solve equations $t_1 = t_2$ 'syntactically'; this is a particular case of the important problem to solve equations 'semantically', i.e. modulo some equational theory E (for this reason semantic unification is also called E-unification). More

precisely, in the presence of an equational theory E, and given an equation $t_1 = t_2$, we want to find substitutions σ such that $E \models t_1^\sigma = t_2^\sigma$ or equivalently (see Theorem 1.4.2) $E \vdash t_1^\sigma = t_2^\sigma$. So syntactic unification is E-unification with empty E.

The situation is now much more complicated than for the case of syntactic unification, since in general there will not be a most general unifier σ for t_1, t_2. We will not really enter the vast area of unification theory (see Siekmann [84]), but will mention two algorithms for E-unification which are pertinent to term rewriting. Both algorithms operate under the assumption that E, the underlying equational theory, is a *complete* TRS (or rather corresponds to one after orienting the equality axioms of E into rewrite rules). So here we have another important application of Knuth-Bendix completion: it prepares the way for equation solving over E, by delivering a complete TRS for E (if possible).

Narrowing

A well-known technique to solve equations $t_1 = t_2$ in the presence of an equational theory E uses the 'narrowing' transformation on terms. We will give an 'intuitive' explanation first, which also explains why narrowing is called 'narrowing'.

If (Σ, E) is an equational theory, we write $[t = s]_E$ for the set of solutions of the equation $t = s$ in E, i.e. $\{\sigma \mid E \vdash t^\sigma = s^\sigma\}$. A solution σ is a substitution as defined earlier, i.e. a map from Var, the set of variables, to $\mathrm{Ter}(\Sigma)$. Let SUB be the set of all substitutions, and if $X \subseteq$ SUB, let σX denote $\{\sigma\tau \mid \tau \in X\}$. Now noting that for every σ we have $[t = s]_E \supseteq \sigma[t^\sigma = s^\sigma]_E$, there is in principle the possibility of a stepwise determination of $[t = s]_E$. This stepwise determination consists of two kind of steps. The first is as just described: guess a component σ of a solution and narrow $[t = s]_E$ to $\sigma[t^\sigma = s^\sigma]_E$. The second is: apply an equation of E in one of the sides of the equation $t = s$ under consideration. Clearly, a step of the second kind preserves equality of the solution set. By an iteration of such steps, alternating between steps of the first kind and steps of the second kind, we may reach the solution set of a trivial equation $r = r$ (which is SUB):

$$\begin{aligned}
[t = s]_E &\supseteq \sigma[t^\sigma = s^\sigma]_E = \sigma[r = s^\sigma]_E \\
&\supseteq \sigma\sigma^1[r^{\sigma^1} = s^{\sigma\sigma^1}]_E = \cdots \\
&\supseteq \cdots \\
&\supseteq \sigma\sigma^1 \cdots \sigma^n[r = r]_E.
\end{aligned}$$

The last solution set $\sigma\sigma^1 \cdots \sigma^n[r = r]_E$ of this 'narrowing' chain has as a most general element the substitution $\sigma\sigma^1 \cdots \sigma^n$. The word 'narrowing' has been given a formal content: it denotes a certain method, based on

term rewriting, to perform a stepwise determination of $[t = s]_E$ as described. A narrowing step combines a step of the first kind and one of the second. Actually, the narrowing relation is first defined on terms rather than equations, as in the following definition, where we suppose that R is a TRS equivalent to E (i.e. $=_R$ coincides with $=_E$). Note that narrowing is a generalization of reduction: any reductions step in a TRS is also a narrowing step. Formally:

Definition 1.6.3. Let term t contain the subterm u, so $t \equiv C[u]$ for some context $C[\,]$. In the presence of a TRS R we say that t is *narrowable* to t' at the (nonvariable) subterm $u \subseteq t$ using rewrite rule $r : t_1 \to t_2 \in R$, via $\sigma = mgu(u, t_1)$, if $t' \equiv C[t_2]^\sigma$. Notation: $t \rightsquigarrow_{u,r,\sigma} t'$. (Sometimes we will drop mention of u, r; but not of σ.)

Figure 1.24

We now extend the narrowing transformation, which was defined on terms, to equations: if $t \rightsquigarrow_\sigma t'$, then $t = s \rightsquigarrow_\sigma t' = s$ and likewise $s = t \rightsquigarrow_\sigma s = t'$ are said to be narrowing steps (on equations). As we have seen, the word narrowing actually refers to the solution sets: if $t = s \rightsquigarrow_\sigma t' = s^\sigma$ then $[t = s]_R \supseteq \sigma[t^\sigma = s^\sigma]_R$. Note how narrowing cuts down the search space for determining the solution set, first by using the directional aspect of a TRS, and second by performing substitutions which are as 'small' (as general) as possible. However, there is a price to be paid: to ensure completeness

of the narrowing method for solving equations, we must require that the underlying TRS is ... complete. More precisely (as stated in Hullot [80]): in order to solve an equation $t_1 = t_1'$ in an equational theory E, corresponding to a complete TRS R, we can construct all possible narrowing derivations starting from the given equation until an equation $t_n = t_n'$ is obtained such that t_n and t_n' are syntactically unifiable. In fact, we are sure to find all possible solutions of the equation. We will make this more precise, via the following definition.

Definition 1.6.4. Let τ, σ be substitutions and E an equational theory. Then $\tau \leq_E \sigma$ if for some ρ we have $\tau\rho =_E \sigma$. (Here $\tau\rho =_E \sigma$ means: $E \vdash x^{\tau\rho} = x^{\sigma}$ for all x.)

Now we have the following completeness theorem for narrowing plus syntactic unification. (See Martelli, Moiso & Rossi [86], Theorem 2. See also Hölldobler [89] for a proof of this theorem and many related facts.) The formulation of the theorem refers to a slightly more general setting than in our discussion of narrowing above: the narrowing procedure may be applied not only to single equations, but to equational goals $\Leftarrow t_1 = s_1, \ldots, t_n = s_n$.

Theorem 1.6.5. *Let R be a complete TRS. Suppose $t_1^\sigma =_R t_2^\sigma$ (i.e. σ is a solution of the equation $t_1 = t_2$). Then there is a successful derivation sequence starting with $\Leftarrow t_1 = t_2$ and using narrowing steps and MM steps (1–4), such that the computed answer substitution τ of this sequence 'improves' σ, i.e. $\tau \leq_R \sigma$.*

Remark 1.6.6.

1. Note that the subscript R in $\tau \leq_R \sigma$ is necessary. (Example: $R = \{f(b) \to g(b),\ a \to b\}$. Now $\sigma = \{x/a\}$ is a solution for $\Leftarrow f(x) = g(x)$, but as computed answer substitution we only find $\tau = \{x/b\}$.)
2. Also completeness of R is necessary.
   (a) To see that *confluence* of R is necessary, consider the TRS $R = \{a \to b,\ a \to c\}$ (so R is not confluent). Now the equation $\Leftarrow b = c$ cannot be solved, i.e. we do not find the expected computed answer substitution ϵ (the identity substitution). However, if we turn R into a confluent system, e.g. by adding the rewrite rules $b \to d$ and $c \to d$, then narrowing (together with syntactic unification) gives a refutation of $\Leftarrow b = c$:

$$\Leftarrow \mathbf{b} = c \rightsquigarrow_\epsilon \Leftarrow d = \mathbf{c} \rightsquigarrow_\epsilon \Leftarrow \mathbf{d} = \mathbf{d} \rightarrowtail_1 \square.$$

   (b) To see that *termination* of R is necessary, consider the confluent but nonterminating TRS with one rule: $c \to f(c)$. Now narrowing plus syntactic unification is not complete: the equation $x = f(x)$ has a solution, $\{x := c\}$, but cannot be resolved,

because the only subterm where narrowing may be applied is $f(x)$ (narrowing may not be performed on a variable) and this does not unify with c. (Also syntactic unification does not help, since x occurs in $f(x)$.) So we do not find a computed answer substitution.

3. Theorem 1.6.5 can be improved in the following sense: we can drop the termination requirement on R, thus only requiring R to be confluent, if we consider only *normalizable* solutions σ (as in the statement of the theorem above). Here σ is called normalizable if all terms x^σ (x a variable) have a normal form. (Note that the solution $\{x := c\}$ in 2(b) was not normalizable.) If moreover we consider not only normalizable solutions σ, but *normal* σ (meaning that every x^σ is in normal form), then we can even drop the subscript R in $\tau \leq_R \sigma$, in the statement of the theorem above.

Lazy term rewriting as a complete E-unification algorithm

An interesting complete E-unification algorithm is given by Martelli, Moiso & Rossi [86], also for the case where E corresponds after orienting the equations to a *complete* TRS R. The nondeterministic algorithm consists of the four derivation rules (1)–(4) for syntactic unification as given above together with a single rule called 'term rewriting' in Martelli e.a. [86]. Of course derivation rules 5 (failure rule) and 6 (occur check) are not included now. Actually, this rule does not resemble what is usually called term rewriting. Here we will call the present rule 'lazy term rewriting'.

7. **Lazy term rewriting**

$$\begin{array}{l} \Leftarrow\ C[F(t_1,\ldots,t_n)] = s,\ E \\ \qquad\qquad \rightarrowtail \\ \Leftarrow\ C[t] = s,\ t_1 = s_1,\ldots,t_n = s_n,\ E \\ \qquad \textit{if } F(s_1,\ldots,s_n) \to t \end{array}$$

and likewise with the reverse of the equations $C[F(t_1,\ldots,t_n)] = s$ and $C[t] = s$. Here $C[\,]$ is some context, and $F(s_1,\ldots,s_n) \to t$ is a rewrite rule from the complete TRS R.

Note how amazingly little is 'done' in a lazy term rewriting step, as compared to the rather complicated narrowing procedure.

2 Orthogonal Term Rewriting Systems

In the preceding sections we have considered general properties of TRS's and how these properties are related; among them the most important property, confluence, with its consequence of uniqueness of normal forms.

We will now consider a special class of TRS's, the orthogonal ones (in the literature mostly known as non-ambiguous and left-linear TRS's), which all have the confluence property as well as various other desirable properties concerned with reduction strategies.

A remark concerning the choice of the word 'orthogonal': to avoid the cumbersome phrase 'non-ambiguous and left-linear', Klop [80a] introduced the abbreviation 'regular'. This terminology is also used in e.g. O'Donnell [85], Kennaway [89], Klop [87], and in early versions of Dershowitz & Jouannaud [90]. On a proposal of Dershowitz and Jouannaud the word 'regular' has been replaced in the present paper by 'orthogonal'; this in view of the fact that many authors found the terminology 'regular' objectionable. Indeed, the word 'orthogonal' has the right intuitive connotations.

2.1 Basic theory of orthogonal TRS's

Definition 2.1.1.

1. A TRS R is *orthogonal* if the reduction rules of R are *left-linear* (R is left-linear) and there are *no critical pairs.*
2. R is *weakly orthogonal* if R is left-linear and R contains only trivial critical pairs, i.e. $\langle t, s\rangle$ is a critical pair then $t \equiv s$.

We recall that a reduction rule $t \to s$ is left-linear if t is linear, i.e. no variable occurs twice or more in t. E.g. the rule $D(x,x) \to E$ is not left-linear; nor is the rule $\underline{\text{if}}\ x\ \underline{\text{then}}\ y\ \underline{\text{else}}\ y \to y$. 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.

Exercise 2.1.2. Let R consist of the rules $D(x,x) \to E$, $C(x) \to D(x, C(x))$, $A \to C(A)$. To show: 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 orthogonal TRS's are confluent, the non-confluence of R is caused by the non-left-linear rule $D(x,x) \to E$.

In the preceding section (Definition 1.4.9) we have already defined the notion of 'critical pair'. Since that definition is often found difficult, we will now explain the *absence* of critical pairs in a more 'intuitive' way. Let R be the TRS as in Table 2.1:

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 2.1

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

Figure 2.1

Figure 2.2

The TRS R in Table 2.1 has the property that *in no term patterns can overlap,* i.e. R has the non-overlapping or non-ambiguity property. Figure 2.2 shows a term in R with all patterns indicated, and indeed they do not overlap.

Overlap can already occur in one rule, e.g. in the rule $L(L(x)) \to 0$; see Figure 2.3(a). An overlap at the root (of the tree corresponding to a term), arising from the rules $F(0,x,y) \to 0$, $F(x,1,y) \to 1$, is shown in Figure 2.3(b). Another overlap at the root, arising from the rules for the non-deterministic $\underline{or}$: $\underline{or}(x,y) \to x$, $\underline{or}(x,y) \to y$, is shown in Figure 2.3(c).

Figure 2.3

We will now formulate and sketch the proofs of the basic theorems for orthogonal TRS's. To that end, we need the notion of '*descendant*' in a reduction. Somewhat informally, this notion can be introduced as follows: Let t be a term in a orthogonal TRS R, and let $s \subseteq t$ be a redex whose head symbol we will give a marking, say by underlining it, to be able to 'trace' it during a sequence of reduction (rewrite) steps. Thus if $s \equiv F(t_1,\ldots,t_n)$, it is marked as $\underline{F}(t_1,\ldots,t_n)$. First consider the rewrite step $t \to_{s'} t'$, obtained by contraction of redex s' in t. We wish to know what has happened in this step to the marked redex s. The following cases can be distinguished, depending on the relative positions of s and s' in t:
Case 1. The occurrences of s' and s in t are disjoint. Then we find back the marked redex s, unaltered, in t'.
Case 2. The occurrences of s and s' coincide. Then the marked redex has disappeared in t'; t' does not contain an underlined symbol.
Case 3. s' is a proper subterm of the marked redex s (so s' is a subterm of one of the arguments of s). Then we find back the marked redex, with some possible change in one of the arguments. (Here we need the orthogonality of R; otherwise the marked redex could have stopped being a redex in t' after the 'internal' contraction of s'.)
Case 4. s is a proper subterm of s'. Then the marked redex s is n times multiplied for some $n \geq 0$; if $n = 0$, s is erased in t'. The reduct t' now contains n copies of the marked redex, all of them still marked.

Now the marked redexes in t' are called the *descendants* of $s \subseteq t$ in the reduction step $t \to_{s'} t'$. It is obvious how to extend this definition

by transitivity to sequences of rewrite steps $t \to_{s'} t' \to_{s''} t'' \to \cdots \to t^{(n-1)} \to_{s^{(n)}} t^{(n)}$.

Proposition 2.1.3. *Let R be a orthogonal TRS, $t \in Ter(R)$. Let t contain, possibly among others, the mutually disjoint redexes $s_1, \ldots, s_n$. Let these redexes be marked by underlining their head symbol. Furthermore, suppose that $t \twoheadrightarrow t'$ is the sequence of n rewrite steps obtained by contraction of all redexes s_i (in some order), and let $t \to_s t''$ be a rewrite step obtained from contracting redex s. (See Figure 2.4(a).) Then a common reduct t''' of t', t'' can be found by contracting in t'' all marked redexes (which still are mutually disjoint). The reduction $t' \twoheadrightarrow t'''$ consists of the contraction of all descendants of s in t' after the reduction $t \twoheadrightarrow t'$.*

Figure 2.4

The proof is a matter of easy casuistics, left to the reader. An immediate corollary is the 'Parallel Moves Lemma':

Lemma 2.1.4. (Parallel Moves Lemma) *We consider reductions in an orthogonal TRS. Let $t \twoheadrightarrow t''$, and let $t \to_s t'$ be a one step reduction by contraction of redex s. Then a common reduct t''' of t', t'' can be found by contraction in t'' of all descendants of redex s, which are mutually disjoint. (See Figure 2.4(b).)*

By repeated application of the Parallel Moves Lemma we now have:

Theorem 2.1.5. *Every orthogonal TRS is confluent.*

Theorem 2.1.5 also holds for weakly orthogonal TRS's. The earliest proof of Theorem 2.1.5 is probably that of Rosen [73]; but earlier proofs of the confluence of CL (Combinatory Logic), work just as well for orthogonal TRS's in general. The confluence theorem for (weakly) orthogonal TRS's is also a special case of a theorem of Huet (mentioned already in Exercise 1.4.24), stated here without proof. We need a definition first:

Definition 2.1.6. (Parallel reduction) $t \to_{\|} s$ if $t \twoheadrightarrow s$ via a reduction of disjoint redexes.

Theorem 2.1.7. (Huet [80]) *Let R be a left-linear TRS. Suppose for every critical pair $\langle t, s\rangle$ we have $t \to_{\|} s$. Then $\to_{\|}$ is strongly confluent, hence R is confluent.*

(For the definition of 'strongly confluent' see Exercise 1.0.8(10). In fact, the proof in Huet [80] yields more: $\to_{\|}$ is even *subcommutative*—see Definition 1.0.1(5).)

Exercises 2.1.8.

1. Combinatory Logic (Table 1.4) has rule patterns as in Figure 2.5; they cannot overlap. As CL is left-linear, it is therefore orthogonal and hence confluent.

Figure 2.5

2. SKIM, in Table 1.5, is orthogonal. Likewise for the TRS's CL with test for equality, binary or applicative, in Tables 1.6, 1.7 respectively. Also Weak Categorical Combinatory Logic in Table 1.8 is orthogonal.
3. A Recursive Program Scheme (RPS) is a TRS with
   (a) a finite set of function symbols $\mathcal{F} = \{F_1, \ldots, F_n\}$ (the 'unknown' functions), where F_i has arity $m_i \geq 0$ $(i = 1, \ldots, n)$, and
   (b) a finite set $\mathcal{G} = \{G_1, \ldots, G_k\}$ (the 'known' or 'basic' functions), disjoint from $\mathcal{F}$, where G_j has arity $p_j \geq 0$ $(j = 1, \ldots, k)$,
   (c) reduction rules of the form
   $$F_i(x_1 \ldots, x_{m_i}) \to t_i \qquad (i = 1, \ldots, n)$$
   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 $(i = 1, \ldots, n)$ there is exactly one rule.

Every RPS is orthogonal, hence confluent. For an extensive treatise on semantical aspects of Recursive Program Schemes, see Courcelle [90].

Exercise 2.1.9. For a deterministic Turing machine M, the TRS R_M as defined in Exercise 1.2.21 is orthogonal.

Apart from confluence, many interesting facts can be proved for orthogonal TRS's.

Definition 2.1.10.

1. A TRS is *non-erasing* if in every rule $t \to s$ the same variables occur in t and in s. (E.g. CL is not non-erasing, due to the rule $Kxy \to x$.)
2. 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.)

The next theorem was proved in Church [41] for the case of the non-erasing version of λ-calculus, the λI-calculus, where the restriction on term formation is adopted saying that in every abstraction term $\lambda x.M$ the variable x must have a free occurrence in M.

Theorem 2.1.11. *Let R be orthogonal and non-erasing. Then: R is WN iff R is SN.*

Another useful theorem, which also reduces the burden of a termination (SN) proof for orthogonal TRS's, is:

Theorem 2.1.12. (O'Donnell [77]) *Let R be an orthogonal 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 2.1.11 then says that for a term in an orthogonal, non-erasing TRS the properties WN and SN coincide. Likewise there is a local version of Theorem 2.1.12. Thus, if in CL a term can be normalized via an innermost reduction, all its reductions are finite.

Exercise 2.1.13. In this exercise we sketch a proof of Theorem 2.1.11 and O'Donnell's Theorem 2.1.12.

1. The following proposition has an easy proof:

   PROPOSITION. *Let t be a term in an orthogonal TRS, containing mutually disjoint redexes $s_1, \ldots, s_n$, and a redex s. Let $t \twoheadrightarrow t'$ be the n-step reduction obtained by contraction, in some order, of the redexes $s_1, \ldots, s_n$. Suppose s has after the reduction $t \twoheadrightarrow t'$ no descendants in t'. Then for some $i \in \{1, \ldots, n\}$: $s \subseteq s_i$. This means that either s coincides with some s_i, or is contained in an argument of some s_i.*

2. We write '$\infty(t)$' if the term t has an infinite reduction $t \to\to \cdots$. So $\infty(t)$ iff t is not SN. Using the proposition in (1) one can now prove (the proof is non-trivial):

   LEMMA. *Let t be a term in an orthogonal TRS such that $\infty(t)$. Let $t \to_s t'$ be a reduction step such that $\neg\infty(t')$. Then the redex s must contain a proper subterm p with $\infty(p)$ that is erased in the step $t \to_s t'$ (i.e. has no*

descendants in t').

3. Using the Lemma it is now easy to prove Theorem 2.1.11: 'critical' steps $t \to t'$ in which $\infty(t)$ but $\neg\infty(t')$, cannot occur in a non-erasing TRS.
4. Theorem 2.1.12 follows from the Lemma in (2) by observing that an innermost contraction cannot erase a proper subterm which admits an infinite reduction, since otherwise the contracted redex would not have been innermost.

Exercise 2.1.14. STS's (Semi-Thue Systems), viewed as TRS's as explained in Section 1.1, are always non-erasing (since left-hand side and right-hand side of every rule end in x, in their TRS version). Also, if there are no critical pairs in the STS, it is orthogonal. 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 Exercise 1.0.8(15), whence WN $\Leftrightarrow$ SN.

Exercise 2.1.15. (Klop [80a]) Let t_0 be a term in an orthogonal TRS. Suppose t_0 has a normal form, but has also an infinite reduction $t_0 \to t_1 \to t_2 \to \cdots$. Show that t_0 has also an infinite 'expansion' (the reverse of a reduction) $\cdots \to t_{-2} \to t_{-1} \to t_0$. (Hint: use the lemma in Exercise 2.1.13, and note that an 'erasing redex' can be used to 'pump' an infinite expansion.)

Exercise 2.1.16. (Klop [85])

1. Let R be orthogonal, and suppose R is WN (i.e. every term has a normal form), but not SN. Let $t \in \mathrm{Ter}(R)$ be a term with an infinite reduction. Then $\mathcal{G}(t)$, the reduction graph of t, contains an infinite expansion (by confluence, there must then also be an infinite expansion of the normal form t_0 of t inside $\mathcal{G}(t)$). In fact, $\mathcal{G}(t)$ contains reductions as follows:

$$t \twoheadrightarrow t_1 \twoheadrightarrow t_2 \twoheadrightarrow \cdots \quad \text{and} \quad t_0 \twoheadleftarrow t_1' \twoheadleftarrow t_2' \twoheadleftarrow \cdots$$

such that $t_n \to t_n'$ for all $n \geq 1$ and such that the t_i $(n \geq 1)$ are pairwise distinct, and likewise the t_j' $(n \geq 1)$ are pairwise distinct.

2. Let t be a term in an orthogonal TRS. Prove: if $\mathcal{G}(t)$ contains an infinite reduction but contains no infinite acyclic expansion, then t reduces to a context $C[\,]$ of a term s without normal form. (The set of s such that $t \twoheadrightarrow C[s]$ for some $C[\,]$, is called in Barendregt [81] the *family* of t.) (In particular the conclusion holds if $\mathcal{G}(t)$ is finite but contains a reduction cycle. Curiously, in CL as in Table 1.4 this is impossible, i.e. finite reduction graphs in CL based on S, K, I are acyclic; see Klop [80b].)
3. The following figure displays the reduction graph of a term t in an orthogonal TRS R. Give an example of such t, R. Conclude, using (2), that t must have a term without normal form in its family. A fortiori, such a reduction graph cannot occur in an orthogonal TRS which is WN.

Figure 2.6

Exercise 2.1.17. (Klop [80a]) Prove for all orthogonal TRS's R with finite alphabet and finitely many rules:

1. R is non-erasing $\Leftrightarrow$ R has the property FB^{-1}. (See Definition 1.0.3(8) for FB^{-1}.)
2. R has the property SN^{-1} $\Leftrightarrow$ R has the property Inc. (See Definition 1.0.3.) (Hint: Prove SN^{-1} $\Rightarrow$ non-erasing, use (1) and use the implication FB^{-1} & SN^{-1} $\Rightarrow$ Inc; see Figure 1.2.)

2.2 Reduction strategies for orthogonal TRS's

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 orthogonal TRS's; for general TRS's 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 or sequential 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.

Definition 2.2.1. Let R be a TRS.

1. A *one step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: $\mathrm{Ter}(R) \to \mathrm{Ter}(R)$ such that
   (a) $t \equiv \mathbb{F}(t)$ if t is a normal form,
   (b) $t \to \mathbb{F}(t)$ otherwise.
2. A *many step reduction strategy* $\mathbb{F}$ *for* R is a map $\mathbb{F}$: $\mathrm{Ter}(R) \to \mathrm{Ter}(R)$ such that
   (a) $t \equiv \mathbb{F}(t)$ if t is a normal form,
   (b) $t \to^+ \mathbb{F}(t)$ otherwise.

Here $\to^+$ is the transitive (but not reflexive) closure of $\to$. Instead of 'one step strategy' we will also say 'sequential strategy'.

Definition 2.2.2.

1. 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.
2. $\mathbb{F}$ 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 Exercise 1.0.8(13) for 'cofinal' and see Figure 2.7.)

Figure 2.7

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 $t \equiv t_0 \to t_1 \to t_2 \to \cdots$ 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 class 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.

Definition 2.2.3.

1. 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).
2. The *parallel-innermost* (many step) strategy reduces simultaneously all innermost redexes. Since these are pairwise disjoint, this is no problem.
3. The *leftmost-outermost* (one step) strategy: in each step the leftmost redex of the maximal (or outermost) redexes is reduced. Notation: $\mathbb{F}_{\mathrm{lm}}$.
4. The *parallel-outermost* (many step) strategy reduces simultaneously all maximal redexes; since these are pairwise disjoint, this is no problem. Notation: $\mathbb{F}_{\mathrm{po}}$.
5. The *full substitution rule* (or *Kleene reduction*, or *Gross-Knuth reduction*): this is a many step strategy in which all redexes are simultaneously reduced. Notation: $\mathbb{F}_{\mathrm{GK}}$.

Strategies (1)–(4) are well-defined for general TRS's. Strategy (5) is only defined for orthogonal TRS's, since for a general TRS it is not possible to define an unequivocal result of simultaneous reduction of a set of possibly nested redexes. The five strategies are illustrated in Figure 2.8 (taken from Bergstra, Heering & Klint [89]), for the following TRS:

$$\begin{array}{lcl} and(true, x) & \rightarrow & x \\ and(false, x) & \rightarrow & false \\ or(true, x) & \rightarrow & true \\ or(false, x) & \rightarrow & true \end{array}$$

We will be mainly interested here in the strategies (3)–(5), for a reason that will be clear by inspection of Table 2.2 below. We have the following facts (for proofs see O'Donnell [77] or Klop [80a]):

Figure 2.8 (after Bergstra, Heering & Klint [89])

Theorem 2.2.4. *For orthogonal TRS's:*

1. $\mathbb{F}_{\mathrm{GK}}$ *is a cofinal reduction strategy.*
2. $\mathbb{F}_{\mathrm{po}}$ *is a normalizing reduction strategy.*

Remark 2.2.5. For λ-calculus this theorem also holds. Moreover, $\mathbb{F}_{\mathrm{lm}}$ is there also a normalizing strategy, just as it is for the orthogonal TRS CL (Combinatory Logic). However, in general $\mathbb{F}_{\mathrm{lm}}$ is not a normalizing strategy for orthogonal TRS's (see Exercise 2.2.6).

Exercise 2.2.6.

1. An example showing that the leftmost-outermost strategy is not normalizing in general, is given in Huet & Lévy [79]: take the orthogonal TRS $\{F(x, B) \to D,\ A \to B,\ C \to C\}$ and consider the term $F(C, A)$. Check that this term has a normal form which is not found by the leftmost-outermost strategy.

2. An example (by N. Dershowitz) showing that parallel-outermost reduction need not be cofinal can be found in the TRS $\{A \to F(A),\ G(x) \to G(x)\}$.

Even though $\mathbb{F}_{\mathrm{lm}}$ is in general for orthogonal TRS's not normalizing, there is a large class of orthogonal TRS's for which it is:

Definition 2.2.7. (O'Donnell [77]) An orthogonal TRS is *left-normal* if in every reduction rule $t \to s$ the constant and function symbols in the left-hand side t precede (in the linear term notation) the variables.

Example 2.2.8.

1. CL (Combinatory Logic) is left-normal.
2. RPS's (Recursive Program Schemes) as defined in Exercise 2.1.8(3) are all left-normal.
3. $F(x, B) \to D$ is not left-normal; $F(B, x) \to D$ is left-normal.

Exercise 2.2.9. (Primitive Recursive Functions) The primitive recursive functions from $\mathbb{N}$ to $\mathbb{N}$ are defined by the following inductive definition (Shoenfield [67]):

1. The *constant* functions $C_{n,k}$, the *projection* functions $P_{n,i}$ and the *successor* function S are primitive recursive. (Here $C_{n,k}(x_1, \ldots, x_n) = k$, $P_{n,i}(x_1, \ldots, x_n) = x_i$, $S(x) = x + 1$.)
2. If $G, H_1, \ldots, H_k$ are primitive recursive, then F defined by

$$F(\vec{x}) = G(H_1(\vec{x}), \ldots, H_k(\vec{x}))$$

   (where $\vec{x} = x_1, \ldots, x_n$) is primitive recursive.
3. If G and H are primitive recursive, then F defined by

$$\begin{array}{lcl} F(0, \vec{x}) & = & G(\vec{x}) \\ F(S(y), \vec{x}) & = & H(F(y, \vec{x}), y, \vec{x}) \end{array}$$

   is primitive recursive.

Show that, by replacing every '=' by '$\to$' in the defining equations, every primitive recursive function is defined by a terminating, left-normal, orthogonal constructor TRS. (For the definition of 'constructor TRS', see the end of Section 2.3.)

Theorem 2.2.10. (O'Donnell [77], Klop [80a]) *Let R be a left-normal orthogonal TRS. Then $\mathbb{F}_{\mathrm{lm}}$ is a normalizing reduction strategy for R.*

Exercise 2.2.11. (Hindley)

1. Consider CL extended with Recursor, where Recursor $= \{Rxy0 \to x,\ Rxy(Sz) \to yz(Rxyz)\}$. Note that this applicative TRS is not left-normal, and show that $\mathbb{F}_{\mathrm{lm}}$ is not normalizing.
2. However, for the TRS CL $\cup$ Recursor* where Recursor$^* = \{R^*0xy \to x,\ R^*(Sz)xy \to yz(Rzxy)\}$ the strategy $\mathbb{F}_{\mathrm{lm}}$ is normalizing.

In the reduction strategy $\mathbb{F}_{\mathrm{GK}}$ (full substitution) every redex is 'killed' *as soon as it arises*, and this repeatedly. Suppose we relax this requirement, and allow ourselves some time (i.e. some number of reduction steps) before getting rid of a particular redex—but with the obligation to deal with it *eventually*. The reductions arising in this way are all cofinal.

Definition 2.2.12.

1. Let $\mathcal{R} = t_0 \to t_1 \to \cdots$ be a finite or infinite reduction sequence. Consider some redex s in some term t_n of $\mathcal{R}$. We say that s is *secured in* $\mathcal{R}$ if eventually there are no descendants of s left, i.e.

$$\exists m > n\ (t_m \text{ contains no descendants } s', s'', \ldots \text{ of } s \subseteq t_n).$$

2. $\mathcal{R}$ is *fair* if every redex in $\mathcal{R}$ is secured.

Theorem 2.2.13. *For reductions $\mathcal{R}$ in orthogonal TRS's: $\mathcal{R}$ is fair $\Rightarrow$ $\mathcal{R}$ is cofinal.*

Note that Theorem 2.2.4(1) is a corollary of the present theorem, since evidently reductions obtained by applying $\mathbb{F}_{\mathrm{GK}}$ are fair. A similar relaxation of constraints applies to the other two strategies $\mathbb{F}_{\mathrm{po}}$ and $\mathbb{F}_{\mathrm{lm}}$:

Definition 2.2.14.

1. A reduction $\mathcal{R}$ is *leftmost-fair* if $\mathcal{R}$ ends in a normal form or infinitely many times the leftmost outermost redex is contracted in $\mathcal{R}$.
2. $\mathcal{R} = t_0 \to t_1 \to \cdots$ is *outermost-fair* if $\mathcal{R}$ does not contain a term t_n with an outermost redex which infinitely long stays an outermost redex but which is never contracted.

Theorem 2.2.15. *Let R be an orthogonal TRS. Then:*

1. *Outermost-fair reductions are normalizing.*
2. *If R is moreover left-normal, then leftmost-fair reductions are normalizing.*

We will now summarize some of the main properties of the various reduction strategies (and their 'relaxed' versions) in Table 2.2. Before doing so, we introduce one more property of strategies:

Definition 2.2.16. A reduction strategy $\mathbb{F}$ for R is *perpetual*, if for all t: $\infty(t) \Rightarrow \infty(\mathbb{F}(t))$.

Here $\infty(t)$ means that t has an infinite reduction, i.e. not $\mathrm{SN}(t)$. So a perpetual strategy is the opposite of a normalizing one; it tries to avoid normal forms whenever possible, and could therefore also be called 'anti-normalizing'.

In Table 2.2 p, n, c stand for perpetual, normalizing, cofinal respectively. In case a property is not mentioned, it does not hold generally. Note that for the leftmost-outermost strategy, when applied to orthogonal TRS's in general, none of the three properties holds generally. Proofs that leftmost-outermost reduction is normalizing for left-normal orthogonal TRS's and that parallel-outermost reduction is normalizing for all orthogonal TRS's can be found in O'Donnell [77]. The latter fact is also proved in Bergstra & Klop [86] (Appendix).

Table 2.2

Computable reduction strategies

A strategy is *recursive* or *computable* if it is, after a coding of the terms into natural numbers, a recursive function. Obviously we are primarily interested in computable strategies; and indeed all five strategies in Definition 2.2.3 are computable. We may now ask whether there is always for an orthogonal TRS a *computable one-step normalizing* reduction strategy. A priori this is not at all clear, in view of TRS's such as the one given by G. Berry: CL extended with rules

$$\begin{aligned} FABx &\to C \\ FBxA &\to C \\ FxAB &\to C \end{aligned}$$

which is an orthogonal TRS. This TRS seems to require a parallel reduction strategy (so, not a one-step or sequential strategy), because in a term of the form $FMNL$ we have no way to see the 'right' argument for computation: a step in the third argument may be unnecessary, namely if the first and second argument evaluate to A and B respectively (which is undecidable due to the presence of CL); likewise a step in the other arguments may be unnecessary. In the next section about sequential TRS's this problem will be analyzed extensively.

When we want to be more liberal, we can consider the same problem for the weakly orthogonal TRS obtained by extending CL with Parallel-or:

$$\begin{array}{lcl} or(true, x) & \rightarrow & true \\ or(x, true) & \rightarrow & true \end{array}$$

It has been claimed by some authors that such TRS's require a parallel evaluation. However, there is the following surprising fact.

Theorem 2.2.17. (Kennaway [89]) *For every weakly orthogonal TRS there exists a computable sequential normalizing reduction strategy.*

The algorithm involved is however too complicated to be of more than theoretical interest.

Standard reductions in orthogonal TRS's

For λ-calculus and CL there is a very convenient tool: the Standardization Theorem (see Barendregt [81], Klop [80a]). For orthogonal TRS's there is unfortunately not a straightforward generalization of this theorem (however, see Huet & Lévy [79] for a generalization). The obstacle is the same as for the normalizing property of the leftmost reduction strategy, discussed in the previous section. When we restrict ourselves again to left-normal orthogonal TRS's, there is a straightforward generalization.

Definition 2.2.18. (Standard Reductions) Let R be a TRS and $\mathcal{R} = t_0 \rightarrow t_1 \rightarrow \cdots$ be a reduction in R. Mark in every step of $\mathcal{R}$ all redex head symbols to the left of the head symbol of the contracted redex, with '$*$'. Furthermore, markers are persistent in subsequent steps.

Then $\mathcal{R}$ is a *standard reduction* if in no step a redex is contracted with a marked head operator.

Exercise 2.2.19. Consider CL $\cup$ Pairing, where Pairing $= \{D_0(Dxy) \rightarrow x,\ D_1(Dxy) \rightarrow y\}$.

1. Show that the reduction $D_0(D(KII)I) \rightarrow D_0(DII) \rightarrow I$ is not standard.
2. Show that $D_0(D(KII)I) \rightarrow KII \rightarrow I$ is standard.

Exercise 2.2.20. (Hindley) Consider in the applicative TRS $R = \{PxQ \rightarrow xx,\ R \rightarrow S,\ Ix \rightarrow x\}$ the reduction

$$\mathcal{R} = PR(IQ) \rightarrow PRQ \rightarrow RR \rightarrow SR$$

and show that there is no standard reduction for $\mathcal{R}$ (i.e. a reduction $PR(IQ) \twoheadrightarrow SR$ which is standard).

Theorem 2.2.21. (Standardization theorem for left-normal orthogonal TRS's) *Let R be a left-normal orthogonal TRS. Then: if $t \twoheadrightarrow s$ there is a standard reduction in R from t to s.*

For a proof see Klop [80a]. A corollary is our earlier theorem stating that $\mathbb{F}_{\mathrm{lm}}$ is a normalizing strategy for left-normal orthogonal TRS's; this

fact is in λ-calculus and CL literature also known as the *Normalization Theorem.*

Exercise 2.2.22. Prove the Normalization Theorem for left-normal orthogonal TRS's from the Standardization Theorem. (Hint: suppose t has a normal form t_0. By the Standardization Theorem, there is a standard reduction from t to t_0. This is in fact the reduction as given by $\mathbb{F}_{\mathrm{lm}}$.)

2.3 Sequential orthogonal Term Rewriting Systems

An important feature of orthogonal TRS's is whether they are 'sequential'. The property of sequentiality is relevant both for the existence of normalizing reduction strategies and for the definability (implementability) in λ-calculus or CL.

That a TRS is sequential, does of course not mean that it is impossible to rewrite redexes in a parallel way. It means that there are also adequate sequential reduction strategies, i.e. it is not *necessary* to rewrite in a parallel way in order to find normal forms. Sequentiality is a desirable property, but unfortunately it is an undecidable property. However, there is a stronger version, 'strong sequentiality', which is decidable and which guarantees the existence of a recursive one-step normalizing reduction strategy. This was shown in Huet & Lévy [79]. In this section we follow this paper, as well as Klop & Middeldorp [89]. We note that here we are primarily interested in 'mathematical' properties of strong sequentiality, and are not concerned with *efficiency* of decision algorithms; for the latter see Huet & Lévy [79].

As remarked in Kennaway [89], one can ask whether 'sequential' is the right terminology, in view of his theorem (2.2.17) stating that every orthogonal TRS has a computable, sequential, normalizing strategy. Yet we feel that the terminology is right, if we are interested in 'feasibly sequential' (admitting a sequential normalizing strategy that is computable in a 'feasible' way).

Definition 2.3.1. Let $t \in \mathrm{Ter}R)$, R orthogonal. Let $s \subseteq t$ be a redex. Then s is a *needed* redex (needed for the computation of the normal form, if it exists) iff in all reductions $t \to \cdots \to t'$ such that t' is a normal form, some descendant of s is contracted. (So, trivially, any redex in a term without normal form is needed.)

Exercise 2.3.2. Show that the leftmost outermost redex in $t \in \mathrm{Ter}R)$ where R is a left-normal orthogonal TRS, is a needed redex.

Theorem 2.3.3. (Huet & Lévy [79]) *Let t be a term in an orthogonal TRS R.*

1. *If t is not in normal form, t contains a needed redex.*
2. *Repeated contraction of a needed redex leads to the normal form, if it exists. (So, needed reduction is normalizing.)*

The proof involves quite some effort and is not included here. (For a proof different from the one of Huet and Lévy, see Kennaway & Sleep [89].)

Exercise 2.3.4.

1. The present theory about needed reductions in orthogonal TRS's trivializes for non-erasing TRS's: Show that in a non-erasing orthogonal TRS *every* redex in a term is needed.
2. (Kennaway [89]) Furthermore the present theory does not have a straightforward generalization to *weakly* orthogonal TRS's: Show that Theorem 2.3.3 does not hold for weakly orthogonal TRS's, by considering $\{or(true, x) \to true, or(x, true) \to true\}$. (However, see O'Donnell [85].)

Thus, Theorem 2.3.3 gives us a normalizing reduction strategy: just contract some needed redex. However, the definition of 'needed' refers to all reductions to normal form, so in order to determine what the needed redexes are, we have to inspect the normalizing reductions first, which is not a very good recipe for a normalizing reduction strategy. In other words, the determination of needed redexes involves *look-ahead*, and it is this necessity for look-ahead that we wish to eliminate. Before we do so, first the following observation, which is easy to prove:

Proposition 2.3.5. *Let $t \in TerR)$, R orthogonal. Let s and s' be redexes in t such that $s \subseteq s'$. Then: s is needed $\Rightarrow$ s' is needed.*

Corollary 2.3.6. *Let t be a term not in normal form. Then some outermost redex of t is needed.*

Now let $C[\ ,\ldots,\ ]$ denote a context with n holes. Denote by σ a substitution of redexes $s_1, \ldots, s_n$ in the holes $1, \ldots, n$. Then the last corollary states:

$$\forall C[\ ,\ldots,\ ] \text{ in normal form } \forall\sigma\ \exists i\ s_i \text{ is needed in } C[s_1, \ldots, s_n].$$

So, which s_i is needed, may depend on σ, i.e. from the other s_j. A more pleasant state of affairs would be when the TRS R would satisfy the following property:

Definition 2.3.7. Let R be an orthogonal TRS. Then R is *sequential** if

$$\forall C[\ ,\ldots,\ ] \text{ in normal form } \exists i\ \forall\sigma;\ s_i \text{ is needed in } C[s_1, \ldots, s_n].$$

Exercise 2.3.8. (Middeldorp)

1. Show that the orthogonal TRS (where CL is Combinatory Logic)

$$\mathrm{CL} \oplus \{F(A,B,x) \to C,\ F(B,x,A) \to C,\ F(x,A,B) \to C\}$$

(due to G. Berry) is not sequential*.

2. Show that the TRS $\{F(A,B,x) \to C,\ F(B,x,A) \to C,\ F(x,A,B) \to C\}$ is sequential*.

3. Conclude that 'sequential*' is not a modular property.

The concept 'sequential*' is only introduced for expository purposes. It is not a satisfactory property as it is undecidable. As it will turn out, a more satisfactory property is 'strongly sequential*', defined as follows.

Definition 2.3.9.

1. Let R be an orthogonal TRS and $C[\,]$ a context. Then a reduction relation $\to_?$ (*possible reduction*) is defined as follows. For every redex s and every term t:

$$C[s] \to_? C[t]$$

As usual, $\twoheadrightarrow_?$ is the transitive reflexive closure. The concept of 'descendant' is defined for $\to_?$ in the obvious way.

2. Let s be a redex in t. Then s is *strongly needed* if in every reduction $t \to_? \ldots \to_? t'$ where t' is a normal form, a descendant of s is contracted. Clearly: s is strongly needed $\Rightarrow$ s is needed.

3. R is *strongly sequential** if $\forall C[\ ,\ldots,\ ]$ in normal form $\exists i\ \forall \sigma\ s_i$ is strongly needed in $C[s_1,\ldots,s_n]$.

This property of 'strong sequentiality*' may be rather subtle, as the following example of Huet & Lévy [79] in Exercise 2.3.10 shows.

Exercise 2.3.10. Let R have rewrite rules, written in tree notation, as in Figure 2.9(a) (the RHS's are irrelevant). Show that R is not strongly sequential*, by considering the context as in Figure 2.9(b).

Now the situation takes a pleasant turn, since as we will prove, it is decidable whether a orthogonal TRS is strongly sequential*, and moreover, there is a simple algorithm to compute an i as in the definition. Actually, Huet & Lévy define concepts 'sequential' and 'strongly sequential' in a different way; our 'sequential*' does not exactly coincide with 'sequential' but 'strongly sequential*' is equivalent with 'strongly sequential'. We will define these concepts now. We need some preliminary definitions:

Definition 2.3.11.

1. Let R be a orthogonal TRS. Then the set $\mathrm{Ter}_\Omega(R)$ of Ω-*terms* of R consists of those terms that can be built from function and constant

Figure 2.9

symbols from R together with a new constant Ω. Reduction relations $\rightarrow$ and $\rightarrow_?$ are extended to $\mathrm{Ter}_\Omega(R)$ in the obvious way. As before, we say that t is a normal form if $t \in \mathrm{Ter}_\Omega(R)$, t contains no Ω and t contains no redexes. Further, t is an Ω-normal form if t contains no redexes (but t may contain occurrences of Ω).

2. On $\mathrm{Ter}_\Omega(R)$ we define a partial order $\preceq$ by:
   (a) $\Omega \preceq t$ for all $t \in \mathrm{Ter}_\Omega(R)$,
   (b) $t_i \preceq t_i'$ $(i = 1, \ldots, n) \Rightarrow F(t_1, \ldots, t_n) \preceq F(t_1', \ldots, t_n')$.
   Clearly, $t \preceq s$ iff $t \equiv C[\Omega, \ldots, \Omega]$ and $s \equiv C[t_1, \ldots, t_n]$ for some context $C[\ , \ldots, \ ]$ not containing Ω, and some $t_i \in \mathrm{Ter}_\Omega(R)$ $(i = 1, \ldots, n;\ n \geq 0)$.
3. A predicate P on $\mathrm{Ter}_\Omega(R)$ is *monotone* if $t \preceq t'$ implies $P(t) \Rightarrow P(t')$.
4. The predicate nf is defined on $\mathrm{Ter}_\Omega(R)$ as follows:

   $nf(t)$ holds if $t \twoheadrightarrow n$ where n is a normal form (so without Ω).

5. The predicate $nf_?$ is defined on $\mathrm{Ter}_\Omega(R)$ as follows:

   $nf_?(t)$ holds if $t \twoheadrightarrow_? n$ where n is a normal form.

6. Let P be a monotone predicate on $\mathrm{Ter}_\Omega(R)$. Let $t \equiv C[\Omega, \ldots, \underline{\Omega}, \ldots, \Omega]$ where all Ω's in t are displayed. Then the underlined occurrence of Ω is an *index with respect to* P (or P-index) if for all s such that $t \preceq s$, where $s \equiv C[t_1, \ldots, t_i, \ldots, t_n]$, we have: $P(s) \Rightarrow t_i \neq \Omega$. (Note that in particular, if t has a P-index, then $P(t)$ does not hold.)

It is easily proved that the predicates nf and $nf_?$ are monotone. Now we define (after Huet & Lévy [79]):

Definition 2.3.12.

1. The orthogonal TRS R is *sequential* if every $t \in \mathrm{Ter}_\Omega(R)$ in Ω-normal form, but not in normal form, has a nf-index.
2. The orthogonal TRS R is *strongly sequential* if every $t \in \mathrm{Ter}_\Omega(R)$ in Ω-normal form, but not in normal form, has a $nf_?$-index.

Exercise 2.3.13. (Middeldorp)

1. Show that: R is sequential $\Rightarrow$ R is sequential*, but not vice versa. Hint: consider the TRS as in Exercise 2.3.8(2), with the term $F(\Omega, \Omega, \Omega)$.
2. Show that: R is strongly sequential $\Leftrightarrow$ R is strongly sequential*.

Exercise 2.3.14. Let $t \equiv C[\Omega, \ldots, \underline{\Omega}, \ldots, \Omega] \in \mathrm{Ter}_\Omega(R)$, R not necessarily strongly sequential. The i-th occurrence of Ω in t is underlined. Suppose that this underlined occurrence is a $nf_?$-index of t. Show then that in $C[s_1, \ldots, s_i, \ldots, s_n]$, where s_i is a redex and the other s_j are arbitrary terms, the redex s_i is strongly needed.

To link the beginning of this section, which used the terminology of contexts, with the present set-up via Ω-terms, we note that a context in normal form, containing at least one hole, corresponds with an Ω-term in Ω-normal form, but not in normal form. Before devoting the rest of this section to an exposition of the long proof that strong sequentiality is a decidable property, we will first show how to find a $nf_?$-index. First, we need some definitions.

Definition 2.3.15. Let $t \in \mathrm{Ter}_\Omega(R)$.

1. t is a *redex compatible* Ω-term if t can be refined to a redex (i.e. $t \preceq t'$ for some redex t').
2. *Ω-reduction* replaces a redex compatible subterm $\not\equiv \Omega$ by Ω, notation: $\to_\Omega$. So, $C[t] \to_\Omega C[\Omega]$ if t is redex compatible and $t \not\equiv \Omega$.
3. The *fixed part* $\omega(t)$ of an Ω-term t is the result of maximal application of Ω-reductions. (In other words, the normal form with respect to Ω-reduction.)

Exercise 2.3.16. Show that $\omega(t)$ is well-defined, by proving that Ω-reduction is confluent and terminating.

Now let $t \equiv C[\Omega, \ldots, \underline{\Omega}, \ldots, \Omega]$ be an Ω-normal form containing at least one Ω. We wish to test whether the i-th occurrence of Ω, the underlined one, is a $nf_?$-index of t. To this end we replace it by a fresh constant symbol, p. Result: $t' \equiv C[\Omega, \ldots, p, \ldots, \Omega]$.

Claim 2.3.17. $\underline{\Omega}$ is a $nf_?$-index in $t \Leftrightarrow p$ occurs in $\omega(t')$. (See Figure 2.10.)

Figure 2.10

The proof of the claim is routine and we omit it. Intuitively, the persistence of the test symbol p in $\omega(t')$ means that whatever the redexes (or even general terms, cf. Exercise 2.3.14) in the other places are, and whatever their reducts might be, the p does not vanish, is not erasable by the action of the other redexes (or general terms). So if instead of p an actual redex s_i was present, the only way to normalize the term at hand is to reduce s_i itself, eventually. Huet & Lévy [79] gives an efficient algorithm for executing the 'p-test', i.e. for finding $nf_?$-indexes (and hence, strongly needed redexes, cf. Exercise 2.3.14).

The decision procedure for the strong sequentiality property itself is much more difficult. We will now present a proof (in a slightly informal way) which is a simplification of the one in Huet & Lévy [79], but where we do not pay any attention to the efficiency of the decision procedure.

In the following we will refer to a $nf_?$-index as an 'index' for short. An Ω-occurrence which is not an index, will be called 'free'. A term in which all Ω's are free, is called free.

The main 'problem' is that we do *not* have the following transitivity property for indexes, which on first sight one might expect to hold: if in the Ω-terms $C_1[\Omega], C_2[\Omega]$, where in both terms the displayed occurrence of Ω is an index (there may be other Ω's occurring), then the displayed Ω in $C_1[C_2[\Omega]]$ is again an index.

Example 2.3.18. Counterexample to transitivity for indexes. Consider the TRS as in Exercise 2.3.10, and the term $F(G(\Omega,\Omega),\underline{\Omega})$. The underlined occurrence is an index, as is easily seen by applying the 'p-test': $\omega(F(G(\Omega,\Omega),p)) = F(\Omega,p)$. However, substituting the same term in the index position, with result $F(G(\Omega,\Omega),F(G(\Omega,\Omega),\Omega))$, we have the 'context' in Figure 3.12(b), which is as shown, essentially, in Exercise 2.3.10, a free term.

However, some 'partial' transitivity properties for the propagation of indexes do hold, notably the one in Proposition 2.3.21 below. We will now

make explicit some properties of index propagation. To this end we employ the following notational convention: instead of "*the displayed occurrence of* Ω *in* $C[\Omega]$ *is an index* "(here the Ω-term $C[\Omega]$ may contain other Ω's) we will just write "$C[\Omega\downarrow]$". However, the absence of an arrow as e.g. in $C[\Omega, \Omega\downarrow]$ does not mean that (in this case) the first Ω is not an index. Furthermore we stipulate that in $C[\Omega, \ldots, \Omega]$ (or a version with arrow annotations) more occurrences of Ω may occur than the ones displayed, unless specified explicitly otherwise. Finally, the notations $s, t, C[\Omega, \ldots, \Omega]$ (possibly with arrow annotations) will refer to Ω-terms, which we sometimes call just 'terms'.

Proposition 2.3.19.

1. $C_1[C_2[\Omega\downarrow]] \Rightarrow C_1[\Omega\downarrow]$ *and* $C_2[\Omega\downarrow]$.
2. *The reverse implication does not hold generally.*

Figure 2.11

Proof. See Figure 2.11, where an arrow points to an index-Ω. Part (2) is the counterexample in 2.3.18. Part (1) follows by an easy argument using the p-test criterion for indexes.

Proposition 2.3.20.

1. *Let* $\omega(t) = \Omega$. *Then* $C[t, \Omega\downarrow] \Rightarrow [\Omega, \Omega\downarrow]$.
2. *The reverse implication holds for all* t: $C[t, \Omega\downarrow] \Leftarrow C[\Omega, \Omega\downarrow]$.

Proof. (See Figure 2.12.) Simple applications of the p-test.

The following proposition (from Klop & Middeldorp [89]) states the 'partial transitivity' for index propagation mentioned before. Here ρ refers to the *maximal height* of the trees corresponding to the redex schemes (i.e., the left-hand-sides of reduction rules) of R. Furthermore, the *depth* of an occurrence in a term is the length of the branch leading from the root symbol to that occurrence.

Proposition 2.3.21. *Let the depth of* Ω *in* $C_2[\Omega]$ *be at least* ρ. *Then:*

$$C_1[C_2[\Omega\downarrow]] \text{ and } C_2[C_3[\Omega\downarrow]] \Rightarrow C_1[C_2[C_3[\Omega\downarrow]]].$$

Figure 2.12

Figure 2.13

Proof sketch. Suppose contexts $C_i[\Omega]$ $(i = 1, 2, 3)$ as in the proposition are given. Consider $\omega(C_1[C_2[C_3[p]]])$. We claim that p is still present in this term. For if not, consider an Ω-reduction leading to $\omega(C_1[C_2[C_3[p]]])$ and especially the Ω-reduction step in which the symbol p is lost. The redex compatible subterm which is removed in this step, has a root symbol s. Now s cannot occur in the subterm $C_2[C_3[p]]$ of $C_1[C_2[C_3[p]]]$, for otherwise p would not occur in $\omega(C_2[C_3[p]])$. But s can also not occur in the C_1-part of $C_1[C_2[C_3[p]]]$, for then p would not occur in $\omega(C_1[C_2[p]])$ due to the assumption referring to ρ.

In the following propositions, a *rigid* term t is a term t such that $\omega(t) = t$. Terms t such that $\omega(t) = \Omega$, will be called *soft*; they 'melt away' completely by Ω-reduction. It is not hard to prove that every term has a unique decomposition in a top part which is rigid and some subterms which are soft. (The top context may be trivial, i.e. equal to Ω.)

Proposition 2.3.22. *Every term $t \in Ter_\Omega(R)$ can be written, uniquely, as $C[t_1, \ldots, t_n]$ such that $C[\Omega, \ldots, \Omega]$ is rigid and the t_i $(i = 1, \ldots, n)$ are soft.*

Proposition 2.3.23. *Suppose $C[t_1, \ldots, t_n]$ is a term such that $C[\Omega, \ldots, \Omega]$ is rigid and t_k is soft for $k = 1, \ldots, n$. Let $t_i \equiv C'[\Omega]$. Then:*

$$C'[\Omega\downarrow] \Rightarrow C[t_1, \ldots, t_{i-1}, C'[\Omega\downarrow], t_{i+1}, \ldots, t_n].$$

Proof. (See Figure 2.14.) By routine arguments involving the p-test.

Figure 2.14

In an attempt to decide whether the TRS R is strongly sequential, we will try to construct a term $t \in \mathrm{Ter}_\Omega(R)$ in Ω-normal form but not in normal form, which is free, i.e. has no indexes. If such a free term exists, then and only then R is not strongly sequential. Especially we will look for a *minimal* free term, minimal with respect to the length. According to the last proposition, we may suppose that a minimal free term, if it exists, is soft. So, such a minimal free term is built from redex compatible terms (i.e. originates, starting from a redex compatible term, by repeatedly substituting redex compatible terms for Ω's)—this follows at once from the definition of 'soft' and Ω-reduction. (See Figure 2.15(a).) However, this observation is not yet sufficient for a sensible attempt to construct a minimal free term, for there are in general infinitely many redex compatible terms which may be the building blocks of the minimal free term we are looking for. Fortunately, we may even suppose that a minimal free term is built from a special kind of redex compatible terms, the *preredexes*, of which only finitely many exist if the TRS R has finitely many reduction rules as was our assumption. (See Figure 2.15(b).)

Definition 2.3.24.

1. A *redex scheme* (or *redex pattern*) is a left-hand side of a reduction rule where all variables are replaced by Ω.
2. A *preredex* is a term which can be refined to a redex scheme. (See Figure 2.16.)

Figure 2.15

Figure 2.16

So, a redex scheme itself is a preredex; every preredex is also a redex compatible term. If the TRS R has finitely many rules, there are only finitely many preredexes. The Ω's in a redex scheme are all free; the Ω's arising by 'truncating' a redex scheme and thus forming a preredex, may be free or an index depending on other redex schemes. The 'old' Ω's in the truncation, if there are any, remain free. All this follows immediately from the definitions and the p-test.

We have already noted that a minimal free term t may be supposed to be built from redex compatible terms, as in Figure 2.15(a). This 'partition' in redex compatible terms need not be unique, but that does not matter.

Suppose a certain partition of t is given, corresponding to some Ω-reduction from t to Ω. Each redex compatible term from which t is built, and which is removed in this Ω-reduction, consists of a preredex refined with some 'extra' subterms. (The subterms that make the difference between Figure 2.15(a) and (b).) Now we remove from t all these extra subterms. (See Figure 2.17.)

Figure 2.17

We claim that the term t', originating after removing all 'extra' subterms, is again free. Namely, consider the example in Figure 2.16, and remove the two extra subterms of the redex compatible subterm s. The Ω's that arise after this removal are free in s; this follows easily by applying the p-test and noting that subterm r is soft. But then these Ω's are also free in t; this follows from Proposition 2.3.19(1). Furthermore, the present removal of the extra subterms of s also does not turn free Ω's at other places into indexes, by Proposition 2.3.20(2).

We will now try to construct a minimal free term t in a tree-like procedure, as suggested in Figure 2.18.

Of course, we want t to be in Ω-normalform—cf. Definition 2.3.12(2). We start, therefore, with the finitely many proper preredexes, where a preredex is 'proper' if it is not a redex scheme. Now at every index Ω, we attach in the next construction step, again a proper preredex. This nondeterministic procedure is repeated. A branch in the thus originating tree of construction terminates 'successfully' if a free term is reached. In that case the TRS R is not strongly sequential. However, there may arise infinite branches in the construction tree. But these we may 'close', eventually, by some form of 'loop checking' in the following way. First a definition.

Definition 2.3.25.

1. Let $C_i[\Omega]$ be preredexes $(i = 1, \ldots, n)$. Then the term

$$\tau \equiv C_1[C_2[\ldots[C_n[\Omega]]\ldots]]$$

Figure 2.18

is called a *tower of preredexes*. If $1 \leq i < j \leq n$, we say that tower τ contains the subtower $\tau' \equiv C_i[C_{i+1}[\ldots[C_j[\Omega]]\ldots]]$.

2. Let ρ be the maximal height of redex schemes of R. A tower $\tau \equiv C_1[\ldots[C_n[\Omega]]\ldots]$ of preredexes is *sufficiently high* if the depth of the displayed Ω in τ is at least ρ.

3. Let t be a term built from preredexes. A *main tower* in τ is a tower (arising after removing some subterms of t) containing a complete branch in the tree corresponding to t (so, from root to some 'final' symbol).

Now if in the construction tree we observe at some construction branch the arising of a term which has a main tower containing two disjoint sufficiently high *identical* subtowers, that construction branch is stopped unsuccessfully.

So every branch of the construction tree terminates, either successfully in a free term, or unsuccessfully. Because the construction tree is finitely branching, the result is a finite construction tree. Now if all construction branches terminate unsuccessfully, the TRS R is strongly sequential; otherwise the presence of a free term at the end of a successful branch reveals that TRS R is not strongly sequential. Hence strong sequentiality is decidable.

We still have to prove that our decision procedure is correct, in particular we have to justify the correctness of the 'loop check' for unsuccessfully closing branches at which a repetition of subtowers occurs. To this end, consider the term s at some point (node) in the construction tree, and consider a successor s' obtained by adjoining a proper preredex π at some index position of s. In general, π will contain some free Ω's as well as some index Ω's (with respect to π). The free Ω's remain free with respect to the whole term s' (Proposition 2.3.19(1)). What about the indexes of π? They may become free in s'. Now what happens with them is entirely determined by the main tower of proper preredexes in s' leading to the Ω in s where π will be adjoined. This follows from Proposition 2.3.20 stating that removal of soft terms does not affect the index or non-index status of other Ω's.

In fact, what happens with the indexes of π is already determined by the subtower of height $\geq \rho$ immediately above the adherence point Ω. This follows from Proposition 2.3.21. But then it is easy to see that in a minimal free term there will not be a repetition of two identical sufficiently large disjoint subtowers (see Figure 2.19). For, if such a repetition occurs in a minimal free term, we can construct a smaller one by cutting away part of the term as in Figure 2.19, contradicting the minimality. This ends the proof of decidability of strong sequentiality.

Many TRS's arising in 'practice' are *constructor* TRS's. For such TRS's it is easy to decide strong sequentiality. A constructor TRS is a TRS in which the set of function symbols can be partitioned into a set $\mathcal{D}$ of *defined* function symbols and a set $\mathcal{C}$ of *constructors*, such that for every rewrite rule $t \to s$, the left-hand side t has the form $F(t_1, \ldots, t_n)$ with $F \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathrm{Ter}(\mathcal{C}, \mathcal{V})$, the set of terms built from variables and constructors.

Figure 2.19

The reason that for constructor TRS's deciding strong sequentiality is easy, is that we do have transitivity of indexes now, in contrast with the case of general TRS's (cf. Counterexample 2.3.18).

Proposition 2.3.26. *Let R be an orthogonal constructor TRS. Let $C_2[\Omega]$ start with a defined function symbol. Then: $C_1[\Omega\downarrow]$ and $C_2[\Omega\downarrow]$ implies $C_1[C_2[\Omega\downarrow]]$.*

Proof. Straightforward, using the p-test for finding indexes.

Corollary 2.3.27. *A constructor TRS is strongly sequential iff every proper preredex has an index.*

So, in order to decide whether a constructor TRS R is strongly sequential, we only have to compute the indexes of its finitely many proper preredexes. (R is supposed to have only finitely many rewrite rules.) Also, the computation of these indexes is very easy: Let $C[\Omega, \ldots, \Omega, \ldots, \Omega]$ be a preredex of R. Now it is not difficult to see that $C[\Omega, \ldots, \Omega\downarrow, \ldots, \Omega]$ iff $C[\Omega, \ldots, p, \ldots, \Omega]$ is not redex compatible.

Exercise 2.3.28. Let R be an orthogonal constructor TRS. Show that R is strongly sequential if every proper preredex P has an Ω-occurrence which is in *all*

redex schemes S such that $P \preceq S$, more defined (i.e. replaced by an Ω-term $\not\equiv \Omega$).

Exercise 2.3.29. (Huet & Lévy [79]) Let R be a left-normal orthogonal TRS. Show that R is strongly sequential. Show that, in fact, in $C[\Omega, \ldots, \Omega]$ (where $C[, \ldots,]$ is an Ω-free context in normal form) the leftmost occurrence of Ω is an index.

Exercise 2.3.30. (Klop & Middeldorp [89]) Show that strong sequentiality is a modular property of orthogonal TRS's, i.e. if R_1, R_2 are orthogonal TRS's with disjoint alphabet, then:

$R_1 \oplus R_2$ is strongly sequential $\Leftrightarrow$ R_1 and R_2 are strongly sequential.

Exercise 2.3.31. (Thatte [87]) Let R_1, R_2 be orthogonal TRS's. Define R_1, R_2 to be *left-equivalent* if the rewrite rules of R_1, R_2 have identical left-hand sides. An orthogonal TRS R is called *left-sequential* if all TRS's which are left-equivalent with R, are sequential (Definition 2.3.12(1)).

Let R be an orthogonal TRS and $C[\Omega, \ldots, \Omega]$ a context in Ω-normal form. The i-th occurrence of Ω is called an *index with respect to left-sequentiality* if this Ω is an index with respect to sequentiality for all TRS's left-equivalent with R.

1. Let R be the TRS with rules $\{F(A,B,x) \to x,\ F(x,A,B) \to x,\ F(B,x,A) \to x,\ G(A) \to A\}$. Show that the third occurrence of Ω in $F(G(\Omega), G(\Omega), \Omega)$ is an index with respect to left-sequentiality, but not with respect to strong sequentiality.
2. Prove that strong sequentiality implies left-sequentiality.
*3. (Open problem.) Does the reverse of (2) also hold, i.e. is every left-sequential TRS strongly sequentialΓ

3 Conditional Term Rewriting Systems

Of growing importance in the field of term rewriting are the *conditional* Term Rewriting Systems (CTRS's). CTRS's have originally arisen from Universal Algebra (see Meinke & Tucker [91]) and in the theory of Abstract Data Types, as implementations of specifications containing *positive conditional equations*

$$t_1(\vec{x}) = s_1(\vec{x}) \wedge \ldots \wedge t_n(\vec{x}) = s_n(\vec{x}) \Rightarrow t_0(\vec{x}) = s_0(\vec{x}) \qquad (*)$$

(If $n = 0$, the equation is called unconditional.) Here $\vec{x} = x_1, \ldots, x_k$; not every t_i, s_i needs to contain all those variables. In $(*)$ we implicitly use universal quantification over $\vec{x}$, i.e. $(*)$ is meant to be

$$\forall \vec{x} \,(\bigwedge_{i=1,..,n} t_i(\vec{x}) = s_i(\vec{x}) \Rightarrow t_0(\vec{x}) = s_0(\vec{x})).$$

Hence the variables appearing in the conditions $t_i(\vec{x}) = s_i(\vec{x})$, $i = 1, \ldots, n$, but not in the consequent $t_0(\vec{x}) = s_0(\vec{x})$ have an 'existential' meaning; e.g.

$$E(x, y) = true \ \wedge \ E(y, z) = true \ \Rightarrow \ E(x, z) = true$$

is by elementary predicate logic equivalent to

$$\exists y \, (E(x, y) = true \ \wedge \ E(y, z) = true) \ \Rightarrow \ E(x, z) = true.$$

Henceforth we will, conform the notation often used in 'equational logic programming', write instead of $(*)$:

$$t_0(\vec{x}) = s_0(\vec{x}) \ \Leftarrow \ t_1(\vec{x}) = s_1(\vec{x}), \ldots, t_n(\vec{x}) = s_n(\vec{x}).$$

Example 3.0.1. A specification of gcd on natural numbers with 0 and successor S, using conditional equations:

$$\begin{array}{lcll}
0 < 0 & = & 0 & \\
0 < S(x) & = & S(0) & \\
S(x) < 0 & = & 0 & \\
S(x) < S(y) & = & x < y & \\
gcd(x, y) & = & gcd(x - y, y) & \Leftarrow \ y < x = S(0) \\
gcd(x, y) & = & gcd(x, y - x) & \Leftarrow \ x < y = S(0) \\
gcd(x, x) & = & x &
\end{array}
\qquad
\begin{array}{lcl}
S(x) - S(y) & = & x - y \\
0 - x & = & 0 \\
x - 0 & = & x
\end{array}$$

(To keep the specification one-sorted, 0 and $S(0)$ are used as booleans $false$ and $true$ respectively. Furthermore, '$-$' is cut-off subtraction.)

The satisfaction relation $\mathcal{A} \models \varphi$, for an equational implication or as we will call them henceforth, *conditional equation* φ, is clear; see also Meinke & Tucker [91], where it is also shown that analogous to the equational case we can develop initial algebra semantics for conditional equations. Conditional equations not only facilitate some specifications, they also are a strictly stronger specification mechanism. In Bergstra & Meyer [84] a conditional specification is given with an initial algebra that cannot be specified (in the same signature) by means of equations.

Again we can ask whether there exists a deduction system and a corresponding completeness theorem, in analogy with Birkhoff's theorem 1.4.2 for equational logic.

Conditional equational deduction

Selman [72] presents a sound and complete deduction system for, as they are called there, *equation conjunction implication (ECI) languages*, or as we will say, for conditional equational logic. We state this deduction system in a considerably simplified way, by considering in a conditional equation

$t = s \Leftarrow E$ where $E = t_1 = s_1, \ldots, t_n = s_n$ $(n \geq 0)$, the sequence of conditions E as a set rather than an ordered tuple as in Selman [72], and by admitting empty E. (See Table 3.1.) Adapting the inference system to the case where E is a multiset or even an ordered tuple is straightforward.

---

axioms

$t = s \Leftarrow t = s,\ t' = s'$

$t = t \Leftarrow$

$t = s \Leftarrow t = r,\ s = r$

$F(t_1, \ldots, t_n) = F(s_1, \ldots, s_n) \Leftarrow t_1 = s_1, \ldots, t_n = s_n$
for every n-ary F

rules

$$\frac{t = s \Leftarrow t' = s',\ E,\ t' = s' \Leftarrow F}{t = s \Leftarrow E,\ F}$$

$$\frac{t = s \Leftarrow E}{t^\sigma = s^\sigma \Leftarrow E^\sigma} \text{ for every substitution } \sigma$$

Table 3.1

Here $E = \{t_1 = s_1, \ldots, t_n = s_n\}$ $(n \geq 0)$, $F = \{t'_1 = s'_1, \ldots, t'_m = s'_m\}$ $(m \geq 0)$; $E^\sigma = \{t^\sigma_1 = s^\sigma_1, \ldots, t^\sigma_n = s^\sigma_n\}$.

Operational semantics of conditional equations

In the unconditional case, there is no problem in the transition from equations to directed equations, i.e. rewrite rules: $t_0(\vec{x}) = s_0(\vec{x})$ is replaced by $t_0(\vec{x}) \to s_0(\vec{x})$, provided the left-hand side is not a single variable and variables occurring in the right-hand side do also occur in the left-hand side. (Of course, choosing the 'right' direction may be a problem—see our discussion of Knuth-Bendix completion.)

In the conditional case the transition from conditional equations to conditional rewrite rules does present a problem, or at least some choices. Dershowitz, Okada & Sivakumar [88] make the following distinctions, thereby extending a classification introduced in Bergstra & Klop [86]. First we introduce some notation.

Definition 3.0.2. Let $\rightarrow$ be a rewrite relation.

1. $t \downarrow s$ (t, s are *joinable*) if $t \twoheadrightarrow u$ and $s \twoheadrightarrow u$ for some term u. (So $\rightarrow$ is confluent if $=$ (convertibility) and $\downarrow$ coincide.)
2. $s \twoheadrightarrow_! t$ if $s \twoheadrightarrow t$ and t is a ground normal form.

Now there are several choices for evaluating the conditions of CTRS's. In the terminology of Dershowitz, Okada & Sivakumar [88] we can distinguish (among others) the following types of CTRS's:

1. *semi-equational* systems

$$t_0 \rightarrow s_0 \Leftarrow t_1 = s_1, \ldots, t_n = s_n$$

2. *join* systems

$$t_0 \rightarrow s_0 \Leftarrow t_1 \downarrow s_1, \ldots, t_n \downarrow s_n$$

3. *normal* systems

$$t_0 \rightarrow s_0 \Leftarrow t_1 \twoheadrightarrow_! s_1, \ldots, t_n \twoheadrightarrow_! s_n$$

4. *generalized* systems

$$t_0 \rightarrow s_0 \Leftarrow P_1, \ldots, P_n.$$

In the last type of CTRS's, the P_i ($i = 1, \ldots, n$) are conditions formulated in a general mathematical framework, e.g. in some first order language, involving the variables occurring in the consequent (and possibly others).

In Bergstra & Klop [86] semi-equational systems were called to be of Type I, join systems of Type II, and normal systems of Type III_n. Actually, Bergstra & Klop [86] define: $t \twoheadrightarrow_! s$ if s is a ground normal form even with respect to the *unconditional part* from the CTRS R (obtained by removing all conditions). This is necessary since otherwise the reduction relation may not be well-defined.

Note that in the cases (1)–(3), the definition of $\rightarrow$ is circular since it depends from conditions involving in some way or another a reference to $\rightarrow$; but it is not hard to see that in fact $\rightarrow$ is well-defined since all conditions of type (1)–(3) are positive. Hence the rewrite rules constitute a positive induction definition of $\rightarrow$. In the case of generalized CTRS's we have to take care in formulating the conditions, in order to ensure that $\rightarrow$ is well-defined.

Remark 3.0.3. In a rewrite rule $t \rightarrow s$ one requires that in s no new variables appear with respect to t. The same requirement is made for conditional rewrite rules $t \rightarrow s \Leftarrow C$. But, as observed in Dershowitz, Okada & Sivakumar [88], for CTRS's it would make good sense to lift this

requirement, as e.g. in the following perfectly natural conditional rewrite specification of the Fibonacci numbers:

$$\begin{array}{lcll} Fib(0) & \to & \langle 0,1\rangle & \\ Fib(x+1) & \to & \langle z, y+z\rangle & \Leftarrow \quad Fib(x)\downarrow\langle y,z\rangle. \end{array}$$

We will not study this more liberal format here, since it introduces a considerable complication of the theory.

We will now discuss several confluence criteria for CTRS's. The first one is a generalization due to Middeldorp [91] (also in Middeldorp [90]) of Toyama's theorem 1.2.2, stating that confluence is a modular property of TRS's, to CTRS's:

Theorem 3.0.4. *Let R_1, R_2 be both semi-equational CTRS's or both join CTRS's or both normal CTRS's with disjoint alphabet. Then:*

$$R_1, R_2 \text{ are confluent} \Leftrightarrow R_1 \oplus R_2 \text{ is confluent.}$$

(The disjoint sum $R_1 \oplus R_2$ is defined analogously to the unconditional case: simply join the sets of rewrite rules.) The proof is a nontrivial application of Toyama's theorem 1.2.2.

Orthogonal Conditional Term Rewriting Systems

We will now state some confluence criteria for orthogonal CTRS's.

Definition 3.0.5.

1. Let R be a CTRS (of any type, semi-equational, join, ...). Then R_u, the *unconditional version* of R, is the TRS which arises from R by deleting all conditions.
2. The CTRS R is called *(non-)left-linear* if R_u is so; likewise for *(weakly) orthogonal.* (See Section 2.1 for orthogonal TRS's.)

Definition 3.0.6.

1. Let R be a CTRS with rewrite relation $\to$, and let P be an n-ary predicate on the set of terms of R. Then P is *closed with respect to* $\to$ if for all terms t_i, t_i' such that $t_i \twoheadrightarrow t_i'$ $(i = 1, \ldots, n)$:

$$P(t_1, \ldots, t_n) \;\Rightarrow\; P(t_1', \ldots, t_n').$$

2. Let R be a CTRS with rewrite relation $\to$. Then R is *closed* if all conditions (appearing in some conditional rewrite rule of R), viewed

as predicates with the variables ranging over R-terms, are closed with respect to $\rightarrow$.

Theorem 3.0.7. (O'Donnell [77]) *Let R be a generalized, weakly orthogonal CTRS which is closed. Then R is confluent.*

The proof is a rather straightforward generalization of the confluence proof for weakly orthogonal TRS's.

Obviously, the convertibility conditions $t_i = s_i$ $(i = 1, \ldots, n)$ in a rewrite rule of a semi-equational CTRS are closed. Hence:

Corollary 3.0.8. *Weakly orthogonal semi-equational CTRS's are confluent.*

Example 3.0.9. Let R be the orthogonal, semi-equational CTRS obtained by extending Combinatory Logic with a 'test for convertibility':

$$\begin{array}{lcll} Sxyz & \rightarrow & xz(yz) & \\ Kxy & \rightarrow & x & \\ Ix & \rightarrow & x & \\ Dxy & \rightarrow & E & \Leftarrow \quad x = y. \end{array}$$

Then R is confluent.

The question now arises whether analogous facts hold for the other types of CTRS's. Indeed, this is the case for normal conditions. The following theorem is a slight generalization of a result in Bergstra & Klop [86]:

Theorem 3.0.10. *Weakly orthogonal normal CTRS's are confluent.*

Remark 3.0.11.

1. Orthogonal join CTRS's are in general not confluent, and even in general not weakly confluent. In Bergstra & Klop [86] the following counterexample is given:

$$\begin{array}{lcll} C(x) & \rightarrow & E & \Leftarrow \quad x \downarrow C(x) \\ B & \rightarrow & C(B). & \end{array}$$

   See Figure 3.1. $C(E) \downarrow E$ does not hold, since this would require $C(E) \rightarrow E$, i.e. $C(E) \downarrow E$.

2. The counterexample in (1) exhibits an interesting phenomenon, or rather, makes a pitfall explicit. According to Corollary 3.0.8 above, the semi-equational CTRS with rules

$$\begin{array}{lcll} C(x) & \rightarrow & E & \Leftarrow \quad x = C(x) \\ B & \rightarrow & C(B) & \end{array}$$

Figure 3.1

is confluent. Hence its convertibility, $=$, coincides with the joinability relation, $\downarrow$. So $x = C(x)$ iff $x \downarrow C(x)$. Yet the join CTRS obtained by replacing the condition $x = C(x)$ by $x \downarrow C(x)$, is according to (1) of this remark *not* confluent.

The complexity of normal forms

Whereas in the unconditional case, being in 'normal form' is an easily decidable property, this needs no longer to be so in the case of CTRS's. In fact, there are semi-equational orthogonal CTRS's for which the set of normal forms is undecidable (and hence not even r.e., since the complement of the set of normal forms is r.e.). The same holds for normal orthogonal CTRS's, and for join CTRS's. The proof is short and instructive enough to be included:

Consider CL (Combinatory Logic); it is well-known (cf. Barendregt [81]) that there is a *representation* $\underline{n}$, a ground CL-term in normal form, of the natural number n for each $n \geq 0$, together with a *computable coding* $\#$ from the set of ground CL-terms into natural numbers, and an '*enumerator*' E (also a ground CL-term in normal form) such that $E\#(M) \twoheadrightarrow M$ for every ground CL-term M. Now let R be the normal CTRS obtained by extending CL with a new constant symbol F and the rule

$$Fx \to \underline{1} \Leftarrow Ex \twoheadrightarrow \underline{0}.$$

(Note that the reduction relation $\to$ of R satisfies $Fx \to \underline{1} \Leftrightarrow Ex \twoheadrightarrow \underline{0}$.) If R had decidable normal forms, then in particular the set $\{\underline{n} \mid F\underline{n} \twoheadrightarrow \underline{1}\}$ would be decidable, i.e. the set $\{\underline{n} \mid E\underline{n} \twoheadrightarrow \underline{0}\}$ would be so. However, then the set

$$\mathcal{X} = \{M \text{ a ground CL-term} \mid M \twoheadrightarrow \underline{0}\}$$

is decidable; for, given M we compute $\#(M)$ and decide whether $E(\underline{\#(M)}) \twoheadrightarrow \underline{0}$ or not. (By confluence for R it follows from $E(\underline{\#(M)}) \twoheadrightarrow \underline{0}$ and $E\underline{\#(M)} \twoheadrightarrow M$ that $M \twoheadrightarrow \underline{0}$.) But this contradicts the fact that $\mathcal{X}$ is undecidable; this follows from a theorem of Scott stating that *any nonempty proper subset of the set of ground CL-terms which is closed under convertibility in CL, must be undecidable.*

For a condition guaranteeing decidability of normal forms, we refer to the notion 'decreasing' below.

Exercise 3.0.12. Adapt the proof above such that it holds for normal CTRS's, and for join CTRS's.

Exercise 3.0.13. (Bergstra & Klop [86]) In this exercise we give a criterion for decidability of normal forms which does not imply termination (as the criterion 'decreasing' does).

Let R be a normal CTRS. If r: $t \to s \Leftarrow t_1 \twoheadrightarrow n_1, \ldots, t_k \twoheadrightarrow n_k$ is a rule of R, then an instance t^σ (σ some substitution) is called a *candidate r-redex* of R. (Of course it depends on the validity of the instantiated conditions $t_i^\sigma \twoheadrightarrow n_i$ of r whether t^σ is an actual r-redex or not.)

We define inductively the set NF_n of normal forms of order n for all $n \geq 0$ as follows: NF_0 is the set of normal forms of R_u, the unconditional part of R. Suppose NF_i ($i \leq n$) have been defined. Then $M \in \mathrm{NF}_{n+1}$ if for every candidate r-redex $t^\sigma \subseteq M$, r as above, the left-hand side of one of the conditions, t_i^σ, evaluates to a 'wrong' normal form m_i, i.e. $m_i \not\equiv n_i$, such that m_i is of order $\leq n$. Furthermore, NF is the set of all normal forms of R. We say that $\mathrm{NF} - \bigcup_{n \geq 0} \mathrm{NF}_n$ contains the normal forms of *infinite order*.

1. Show that if NF is undecidable, then there must be some normal form of infinite order.
2. Suppose for every rule r (as above) of R we have $t_i \subset t$ (t_i is a proper subterm of t), $i = 1, \ldots, k$. Then we say that R *has subterm conditions*. Show that if R has subterm conditions, there are no normal forms of infinite order. Hence NF is decidable.

Non-orthogonal conditional rewriting

Following Dershowitz, Okada & Sivakumar [88] (see also Dershowitz & Okada [90]), we will now consider CTRS's which are not orthogonal (i.e. may have 'critical pairs') and formulate some conditions ensuring confluence.

Definition 3.0.14. (Critical Pairs)

1. Let R be a CTRS containing the two conditional rewrite rules

$$t_i \to s_i \Leftarrow E_i \quad (i = 1, 2).$$

(Suppose these are 'standardized apart', i.e. have no variables in common.) Suppose t_2 can be unified with the nonvariable subterm u in $t_1 \equiv C[u]$, via $\sigma = mgu(t_2, u)$. Then the conditional *equation*(!)

$$s_1^\sigma = C[t_2]^\sigma \Leftarrow (E_1, E_2)^\sigma$$

is a *critical pair* of the two rules.

2. A critical pair is an *overlay* if in (1), t_1 and t_2 unify at the root, i.e. $t_1 \equiv u$.
3. A CTRS is *non-overlapping* (or non-ambiguous) if it has no critical pairs.
4. A critical pair $s = t \Leftarrow E$ is *joinable* if for all substitutions σ such that E^σ is true, we have $s^\sigma \downarrow t^\sigma$.

Theorem 3.0.15. (Dershowitz, Okada & Sivakumar [88])

1. *Let R be a semi-equational CTRS. Then: If R is terminating and all critical pairs are joinable, R is confluent.*
2. *Let R be a join system. Then: If R is decreasing and all critical pairs are joinable, R is confluent.*
3. *Let R be a join system. If R is terminating and all critical pairs are overlays and joinable, R is confluent.*

This theorem contains the unexplained notion of a 'decreasing' CTRS:

Definition 3.0.16. (Dershowitz, Okada & Sivakumar [88]) Let R be a CTRS.

1. $>$ is a *decreasing* ordering for R if
   (a) $>$ is a well-founded ordering on the set of terms of R (i.e. there are no descending chains $t_0 > t_1 > t_2 > \cdots$);
   (b) $t \subset s \Rightarrow t < s$ (here $\subset$ is the proper subterm ordering);
   (c) $t \to s \Rightarrow t > s$;
   (d) for each rewrite rule $t \to s \Leftarrow t_1 \downarrow s_1, \ldots, t_n \downarrow s_n$ $(n \geq 0)$ and each substitution σ we have: $t^\sigma > t_i^\sigma, s_i^\sigma$ $(i = 1, \ldots, n)$. (Likewise for other CTRS's than join CTRS's.)
2. A CTRS is decreasing if it has a decreasing ordering.

A consequence of 'decreasing' is termination. Moreover, the notions $\to, \twoheadrightarrow, \downarrow$, and normal form are decidable.

Remark 3.0.17. Related notions are *fair* or *simplifying* CTRS's (Kaplan [84, 87]) and *reductive* CTRS's (Jouannaud & Waldmann [86]). In fact: reductive $\Rightarrow$ simplifying $\Rightarrow$ decreasing; see also Dershowitz & Okada [90].

We conclude this section by mentioning a useful fact:

Theorem 3.0.18. (Dershowitz, Okada & Sivakumar [88]) *Let $R^=$ be a decreasing semi-equational CTRS. Let $R^\downarrow$ be the corresponding join CTRS (where conditions $t_i = s_i$ are changed into $t_i \downarrow s_i$). Then:*

$$R^= \text{ is confluent } \Rightarrow R^\downarrow \text{ is confluent.}$$

Acknowledgements

I am grateful to several persons for their support in writing this chapter. In particular I like to thank Henk Barendregt, Nachum Dershowitz, Ronan Sleep, Roel de Vrijer, as well as editors and co-authors of this Handbook. Special thanks to Aart Middeldorp for several contributions, and to Jean-Jacques Lévy for his close scrutiny of a previous version including many helpful comments. Finally, many thanks to Aart Middeldorp, Vincent van Oostrom, Jane Spurr and Fer-Jan de Vries for the heroic struggle to transform an early Macintosh version into LaTeX.

4 References

Apt, K.R. (1990). *Logic Programming.* In: Formal models and semantics, Handbook of Theoretical Computer Science, Vol. B (ed. J. van Leeuwen), 495-574.

Bachmair, L. (1988). *Proof by consistency in equational theories.* In: Proceedings of the 3rd IEEE Symposium on Logic in Computer Science, Edinburgh, 228-233.

Bachmair, L. (1989). *Canonical Equational Proofs.* Birkhäuser, Boston, 1991.

Bachmair, L. & Dershowitz, N. (1986). *Commutation, transformation, and termination.* Proceedings of the 8th Conference on Automated Deduction (ed. J.H. Siekmann), Oxford, Springer LNCS 230, 5-20.

Bachmair, L., Dershowitz, N. & Hsiang, J. (1986). *Orderings for equational proofs.* In: Proceedings of the 1st IEEE Symposium on Logic in Computer Science, Cambridge, Massachusetts, 346-357.

Bachmair, L. & Plaisted, D.A. (1985). *Associative path orderings.* In: Proceedings of the 1st International Conference 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 edn. 1981, 2nd edn. 1984). North-Holland, Amsterdam.

Barendregt, H.P. (1989). *Functional programming and lambda calculus.* In: Handbook of Theoretical Computer Science (ed. J. van Leeuwen), North-Holland, Amsterdam.

Barendregt, H.P., van Eekelen, M.C.J.D., Glauert, J.R.W., Kennaway, J.R., Plasmeijer, M.J. & Sleep, M.R. (1987). *Term graph rewriting.* In: Proceedings of the 1st Conference on Parallel Architectures and Languages Europe (PARLE), Eindhoven, Vol. II, Springer LNCS 259, 141-158.

Bergstra, J.A., Heering, J. & Klint, P. (eds.) (1989). *Algebraic specification.* Addison-Wesley, Reading, Massachusetts.

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

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

Bergstra, J.A. & Klop, J.W. (1986). *Conditional rewrite rules: confluence and termination.* JCSS **32** (3), 323-362.

Bergstra, J.A. & Meyer, J.-J.Ch. (1984). *On specifying sets of integers.* EIK (Elektronische Informationsverarbeitung und Kybernetik) **20** (10/11), 531-541.

Bergstra, J.A. & Tucker, J.V. (1980). *A characterisation of computable data types by means of a finite equational specification method.* In: Proceedings of of the 7th International Colloquium on Automata, Languages and Programming, (eds. J.W. de Bakker & J. van Leeuwen), Amsterdam, Springer LNCS 85, 76-90.

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

Birkhoff, G. (1935). *On the structure of abstract algebras.* In: Proceedings of the Cambridge Philosophical Society **31**, 433-454.

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

Courcelle, B. (1990). *Recursive application schemes.* In: Formal models and semantics, Handbook of Theoretical Computer Science, Vol. B, (ed. J. van Leeuwen), Elsevier - The MIT Press, Amsterdam, 459-492.

Church, A. (1941). *The calculi of lambda conversion.* Annals of Mathematics Studies, Vol. 6, Princeton University Press.

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

Dauchet, M. (1989). *Simulation of Turing machines by a left-linear rewrite rule.* In: Proceedings of the 3rd International Conference on Rewriting Techniques and Applications, Chapel Hill, Springer LNCS 355, 109-120.

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

Dauchet, M., Tison, S., Heuillard, T. & Lescanne, P. (1987). *Decidability of the confluence of ground term rewriting systems.* In: Proceedings of the 2nd Symposium on Logic in Computer Science, Ithaca, NY, 353-359.

Dershowitz, N. (1979). *A note on simplification orderings.* Information Processing Letters **9** (5), 212-215.

Dershowitz, N. (1981). *Termination of linear rewriting systems.* Proceedings of the 8th International Colloquium on Automata, Languages and Programming, (Eds. S. Even and O. Kariv), Springer LNCS 115, 448-458.

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

Dershowitz, N. (1987). *Termination of rewriting.* J. of Symbolic Computation **3** (1), 69-116. Corrigendum: **4** (3), 409-410.

Dershowitz, N. & Jouannaud, J.-P. (1990). *Rewrite systems.* In: Formal models and semantics, Handbook of Theoretical Computer Science, Vol. B, (ed. J. van Leeuwen), Elsevier - The MIT Press, Amsterdam, 243-320.

Dershowitz, N. & Manna, Z. (1979). *Proving termination with multiset orderings.* Comm. of the ACM **22** (8), 465-476.

Dershowitz, N., Marcus, L. & Tarlecki, A. (1988). *Existence, uniqueness, and*

construction of rewrite systems. SIAM J. Comput. **17** (4), 629-639.

Dershowitz, N. & Okada, M. (1990). *A rationale for conditional equational programming.* TCS **75**, 111-138.

Dershowitz, N., Okada, M. & Sivakumar, G. (1987). *Confluence of Conditional Rewrite Systems.* In: Proceedings of the 1st International Workshop on Conditional Term Rewrite Systems, Orsay, Springer LNCS 308, 31-44.

Dershowitz, N., Okada, M. & Sivakumar, G. (1988). *Canonical Conditional Rewrite Systems.* In: Proceedings of the 9th Conference on Automated Deduction, Argonne, Springer LNCS 310, 538-549.

Drosten, K. (1989). *Termersetzungssysteme.* Informatik-Fachberichte **210**, Springer. (In German.)

Ehrig, H. & Mahr, B. (1985). *Fundamentals of Algebraic Specification 1. Equations and Initial Semantics.* Springer-Verlag, Berlin.

Gallier, J.H. (1987). *What's so special about Kruskal's Theorem and the ordinal* Γ_0. Technical report MS-CIS-87-27, University of Pennsylvania, Philadelphia.

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) **31**, 22-28.

Geser, A. (1990). *Relative Termination.* Ph.D. Thesis, University of Passau, 1990.

Goguen, J.A. & Meseguer, J. (1985). *Initiality, induction, and computability.* In: Algebraic methods in semantics (eds. M. Nivat & J.C. Reynolds), Cambridge University Press 1985, 459-542.

Guessarian, I. (1981). *Algebraic semantics.* Springer LNCS 99.

Hardin, T. (1989). *Confluence results for the pure Strong Categorical Logic CCL; λ-calculi as subsystems of CCL.* TCS, Fundamental Studies **65** (3), 291-342.

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

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

Hölldobler, S. (1989). *Foundations of Equational Logic Programming.* Springer LNCS 353.

Huet, G. (1980). *Confluent reductions: Abstract properties and applications to term rewriting systems.* JACM **27** (4), 797-821.

Huet, G. (1981). *A complete proof of correctness of the Knuth-Bendix completion algorithm.* JCSS **23**, 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 359. To appear as: *Computations in orthogonal term rewriting systems* in: Computational logic, essays in honour of Alan Robinson (eds. J.-L. Lassez & G. Plotkin), MIT Press, Cambridge, Massachusetts.

Huet, G. & Oppen, D.C. (1980). *Equations and rewrite rules: A survey.* In:

Formal Language Theory: Perspectives and Open Problems (ed. R.V. Book), Academic Press, London, 349-405.

Hullot, J.M. (1980) *Canonical forms and unification.* In: Proceedings 5th Conference on Automated Deduction, Les Arcs, France, 318-334.

Jantzen, M. (1988). *Confluent string rewriting and congruences.* EATCS (European Association for Theoretical Computer Science) Monographs on Theoretical Computer Science **14**, Springer-Verlag, Berlin.

Jouannaud, J.-P. & Kirchner, H. (1986). *Completion of a set of rules modulo a set of equations.* SIAM J. Comp. **15** (4), 1155-1194.

Jouannaud, J.-P. & Waldmann, B. (1986). *Reductive conditional Term Rewriting Systems.* In: Proceedings of the 3rd IFIP Working Conference on Formal Description of Programming Concepts, Ebberup, 223-244.

Kamin, S. & Lévy, J.-J. (1980). *Two generalizations of the recursive path ordering.* Unpublished manuscript, University of Illinois.

Kaplan, S. (1984). *Conditional Rewrite Rules.* TCS **33** (2,3).

Kaplan, S. (1987). *Simplifying conditional term rewriting systems: Unification, termination and confluence.* J. of Symbolic Computation **4** (3), 295-334.

Kennaway, J.R. (1989). *Sequential evaluation strategies for parallel-or and related reduction systems.* Annals of Pure and Applied Logic **43**, 31-56.

Kennaway, J.R. & Sleep, M.R. (1989). *Neededness is hypernormalizing in regular combinatory reduction systems.* Preprint, School of Information Systems, University of East Anglia, Norwich.

Klop, J.W. (1980a). *Combinatory Reduction Systems.* Mathematical Centre Tracts 127, CWI, Amsterdam.

Klop, J.W. (1980b). *Reduction cycles in Combinatory Logic.* In: Festschrift 'To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism' (eds. J.P. Seldin & J.R. Hindley), Academic Press, London, 193-214.

Klop, J.W. (1985). *Term Rewriting Systems.* Notes for the Seminar on Reduction Machines, Ustica. Unpublished.

Klop, J.W. (1987) *Term rewriting systems: a tutorial,* Bulletin of the EATCS **32**, 143-182.

Klop, J.W. & Middeldorp, A. (1988). *An Introduction to Knuth-Bendix Completion.* CWI Quarterly **1**(3), Centre for Mathematics and Computer Science, Amsterdam, 31-52.

Klop, J.W. & Middeldorp, A. (1989). *Sequentiality in Orthogonal Term Rewriting Systems.* Report CS-R8932, CWI, Centre for Mathematics and Computer Science, Amsterdam. To appear in J. of Symbolic Computation.

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

Kruskal, J.B. (1960). *Well-Quasi-Ordering, the Tree Theorem, and Vazsonyi's Conjecture.* Transactions of the AMS **95**, 210-225.

Kurihara, M. & Kaji, I. (1988). *Modular Term Rewriting Systems: Termination, Confluence and Strategies.* Report, Hokkaido University. Abridged version:

Modular term rewriting systems and the termination. Information Processing Letters 34 **34**, 1-4.

Lankford, D.S. (1979). *On proving term rewriting systems are Noetherian.* Memo MTP-3, Mathematical Department, Louisiana Technical University, Ruston, Louisiana.

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

Martelli, A. & Montanari, U. (1982). *An efficient unification algorithm.* Transactions on Programming Languages and Systems **4**(2), 258-282.

Martelli, A., Moiso, C. & Rossi C.F. (1986). *An Algorithm for Unification in Equational Theories.* In: Proceedings Symposium on Logic Programming, 180-186.

Meinke, K. & Tucker, J.V. (1991). *Universal algebra.* In: Handbook of Logic in Computer Science (eds. S. Abramsky, D. Gabbay & T. Maibaum), Oxford University Press, this volume.

Métivier, Y. (1983). *About the rewriting systems produced by the Knuth-Bendix completion algorithm.* Information Processing Letters **16**, 31-34.

Middeldorp, A. (1989a). *Modular aspects of properties of term rewriting systems related to normal forms.* In: Proceedings of 3rd International Conference on Rewriting Techniques and Applications, Chapel Hill, Springer LNCS 355, 263-277.

Middeldorp, A. (1989b). *A sufficient condition for the termination of the direct sum of term rewriting systems.* In: Proceedings of the 4th IEEE Symposium on Logic in Computer Science, Pacific Grove, 396-401.

Middeldorp, A. (1990). *Modular properties of term rewriting systems.* Ph.D. Thesis, Vrije Universiteit, Amsterdam.

Middeldorp, A. (1991). *Modular properties of conditional term rewriting properties.* To appear in Information and Computation.

Nash-Williams, C.St.J.A. (1963). *On well-quasi-ordering finite trees.* In: Proceedings of the Cambridge Philosophical Society **59**(4), 833-835.

Nederpelt, R.P. (1973). *Strong normalization for a typed lambda calculus with lambda structured types.* Ph.D. Thesis, Technische Hogeschool, Eindhoven, the Netherlands.

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

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

O'Donnell, M.J. (1985). *Equational logic as a programming language.* The MIT Press, Cambridge, Massachusetts.

Oyamaguchi, M. (1987). *The Church-Rosser property for ground term rewriting systems is decidable.* TCS **49**(1), 43-79.

Peterson, G.E. & Stickel, M.E. (1981). *Complete sets of reductions for some equational theories.* J. of the ACM, **28**(2), 233-264.

Plaisted, D.A. (1978). *A recursively defined ordering for proving termination of*

term rewriting systems. Report R-78-943, University of Illinois, Urbana, Illinois.

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

Puel, L. (1986). *Using unavoidable sets of trees to generalize Kruskal's theorem.* J. of Symbolic Computation **8**(4), 335-382.

Raoult, J.-C. & Vuillemin, J. (1980). *Operational and semantic equivalence between recursive programs.* Journal of the ACM **27**(4),772-796.

Rosen, B.K. (1973). *Tree-manipulating systems and Church-Rosser theorems.* Journal of the ACM **20**(1), 160-187.

Rusinowitch, M. (1987a). *On termination of the direct sum of term rewriting systems* Information Processing Letters **26**, 65-70.

Rusinowitch, M. (1987b). *Path of subterms ordering and recursive decomposition ordering revisited.* Journal of Symbolic Computation **3**, 117-131.

Selman, A. (1972). *Completeness of calculii for axiomatically defined classes of algebras.* Algebra Universalis **2**, 20-32.

Shoenfield, J.R. (1967). *Mathematical Logic.* Addison-Wesley, Reading, Massachusetts.

Siekmann, J. (1984). *Universal unification.* In: Proceedings of the 7th International Conference on Automated Deduction (ed. R.E. Shostak), Napa, California, Springer LNCS 170, 1-42.

Smoryński, C. (1982). *The variety of arboreal experience.* The Mathematical Intelligencer, **4**(4), 182-189.

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

Thatte, S. (1987). *A refinement of strong sequentiality for term rewriting with constructors.* Information and Computation **72**, 46-65.

Toyama, Y. (1987a). *Counterexamples to termination for the direct sum of Term Rewriting Systems.* Information Processing Letters **25**, 141-143.

Toyama, Y. (1987b). *On the Church-Rosser property for the direct sum of term rewriting systems.* Journal of the ACM **34**(1), 128-143.

Toyama, Y. (1988). *Commutativity of Term Rewriting Systems.* In: Programming of Future Generation Computer II (eds. K. Fuchi and L. Kott), North-Holland, Amsterdam, 393-407.

Toyama, Y., Klop, J.W. & Barendregt, H.P. (1989). *Termination for the direct sum of left-linear term rewriting systems.* In: Proceedings of the 3rd International Conference on Rewriting Techniques and Applications, Chapel Hill, Springer LNCS 355, 477-491. Extended version: Report CS-R8923, CWI, Amsterdam.

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

Winkler, F. & Buchberger, B. (1983). *A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm.* In: Proceedings of the Colloquium on Algebra, Combinatorics and Logic in Computer Science, Györ, Hungary.