TERM REWRITING SYSTEMS
FROM CHURCH-ROSSER TO KNUTH-BENDIX AND BEYOND

Jan Willem Klop
Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam;
Department of Mathematics and Computer Science, Free University,
de Boelelaan 1081, 1081 HV Amsterdam; jwk@cwi.nl

Term rewriting systems are important for computability theory of abstract data types, for automatic theorem proving, and for the foundations of functional programming. In this short survey we present, starting from first principles, several of the basic notions and facts in the area of term rewriting. Our treatment, which often will be informal, covers abstract rewriting, Combinatory Logic, orthogonal systems, strategies, critical pair completion, and some extended rewriting formats.
Note: Research partially supported by ESPRIT projects 3020: Integration and 3074: Semagraph.

Introduction

The intention of this paper is to present to a general audience of theoretical computer scientists some of the basic concepts, facts and developments in the area of Term Rewriting Systems. We do not aim at a complete historical account, even though the title may hint in that direction. Our aim would be fulfilled if some interest is raised, which subsequently may lead to consulting more extensive surveys such as Huet & Oppen [80], Klop [90], Dershowitz & Jouannaud [90]. Especially the last survey pays much attention to the more recent developments in Term Rewriting Systems.

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 breakthrough 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. (Turner [79], Barendregt [89]). Even more recently another related family of TRSs, that of Categorical Combinatory Logic, has emerged, yielding a remarkable connection between concepts from category theory and elementary steps in machine computations (Curien [86], Hardin [89]).

Term Rewriting Systems are attractive because of their simple syntax and semantics—at least those TRSs 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 TRSs provide a natural medium for implementing computations, and in principle even for parallel computations. This feature makes TRSs interesting for the design of parallel reduction machines.

Another field where TRSs play a fundamental role is the analysis and implementation of abstract data type specifications, with respect to consistency properties, computability theory (Goguen & Meseguer [85]), decidability of word problems (Knuth & Bendix [70]), theorem proving (Hsiang [85]).

In recent years, a strong impulse for the study of Term Rewriting Systems (including extensions of the usual rewriting format) is given by the design of functional programming languages such as Miranda (Turner [85]). Another strong impulse is given by efforts of many researchers to combine and integrate logic programming with functional programming (Dershowitz & Plaisted [87], Hölldobler [89]).

Contents

1. CHURCH-ROSSER

1.1. Combinatory Logic

A good starting point is the Term Rewriting System, or TRS for short, called Combinatory Logic (CL). Originally devised by Schönfinkel [24] in an attempt to eliminate bound variables from Predicate Logic, the system was rediscovered by H. Curry and played the central role in his foundational program of 'Illative Combinatory Logic'. Here we will not pay attention to the foundational aspects of CL, and merely consider the system of the following three equations in the first order language with binary function symbol Ap (*application*), constant symbols S, K, I and variables x, y, z, ... as in Table 1(a):

a $Ap(Ap(Ap(S, x), y), z) = Ap(Ap(x, z), Ap(y, z))$ $Ap(Ap(K, x), y) = x$ $Ap(I, x) = x$	b $((S\cdot x)\cdot y)\cdot z = (x\cdot z)\cdot(y\cdot z)$ $(K\cdot x)\cdot y = x$ $I\cdot x = x$
c $Sxyz = xz(yz)$ $Kxy = x$ $Ix = x$	d $Sxyz \to xz(yz)$ $Kxy \to x$ $Ix \to x$

Table 1

For better readability we write instead of the binary prefix operator Ap an infix dot (Table 1(b)). To enhance readability even more we leave away the dot and save several pairs of brackets under the convention of *association to the left*: restore missing bracket pairs as leftmost as possible (Table 1(c)). This is the form in which the equations of CL usually are presented. Actually, the equations have a direction, from left to right; applications of the equations consisting of replacing an instantiated left-hand side (in some context) by the corresponding instantiated right-hand side tend to simplify the term in some sense. For the K- and I-equation this is clear, for the S-equation it is less clear but may be clear after subsequent elaborations. Notationally the direction is represented as in Table 1(d); and this is our first example of a Term Rewriting System. Henceforth CL will denote this TRS.

So, in general, a TRS is a pair (Σ, R) where Σ is some signature, listing the function symbols (with their 'arities') and constant symbols, and R is a set of *reduction rules* (or rewrite rules) of the form $t \to s$. Here $t, s \in Ter(\Sigma)$, the set of terms over Σ. Two restrictions are imposed on the form of reduction rules: t must not be a variable, and s does not contain occurrences of variables that do not already occur in t. (It is not hard to think of an intuitive motivation for these restrictions; also, there are several points in the subsequent development of the theory where they are required.) The reduction rules are used as one expects from familiarity with equational logic: they induce *reduction steps* (rewrite steps) $C[t^\sigma] \to C[s^\sigma]$ for arbitrary contexts C[ ] and substitutions σ. A substitution σ is a map from Var, the set of variables, to $Ter(\Sigma)$; it is extended to a map from $Ter(\Sigma)$ to $Ter(\Sigma)$ in a homomorphic way. A *context* is a term with a 'hole' $\Box$; e.g. $SK(\Box I)x$ is a CL-context. The instantiated left-hand side t^σ of a reduction rule $t \to s$ is called a *redex* (*red*ucible *ex*pression) with *contractum* s^σ. Often we will write just R instead of (Σ, R) and Ter(R) instead of $Ter(\Sigma)$.

We have now defined the *one step reduction relation* $\to$ on Ter(R). The transitive reflexive closure of $\to$ is written as $\twoheadrightarrow$; also the notation $\to^*$ is often used. The reflexive closure of $\to$ is $\to^=$. The equivalence relation generated by $\to$ is '=', called *convertibility*. It should not be confused with $\equiv$, denoting syntactical identity.

Let us return to CL and play with some examples in order to appreciate the great expressive power of this TRS and to illustrate the concepts and notations introduced thus far. As CL was originally devised by Curry as a theory about functions, it is to be expected that function composition $\circ$ can be defined: indeed, abbreviating S(KS)K as B we have the reduction sequence in Table 2 (1), establishing that $Bxyz \twoheadrightarrow x(yz)$, and a fortiori that $Bxyz = x(yz)$. So Bxy is $x \circ y$ in prefix notation.

(1)	$Bxyz \equiv S(KS)Kxyz \to KSx(Kx)yz \to S(Kx)yz \to Kxz(yz) \to x(yz)$
(2)	$\omega x \equiv SIIx \to Ix(Ix) \to Ixx \to xx$
(3)	$\omega\omega \equiv SII(SII) \to I(SII)(I(SII)) \to I(SII)(SII) \to SII(SII)$
(4)	$P \equiv \omega(BF\omega) \twoheadrightarrow BF\omega(BF\omega) \twoheadrightarrow F(\omega(BF\omega)) \equiv FP$
(5)	$Cxyz \equiv S(BBS)(KK)xyz \twoheadrightarrow xzy$
(6)	$Wxy \equiv SS(KI)xy \twoheadrightarrow xyy$
(7)	$Yx \equiv WS(BWB)x \twoheadrightarrow BWBx(BWBx) \twoheadrightarrow x(BWBx(BWBx)) \twoheadleftarrow x(Yx)$

Table 2

Another aspect of CL is that it admits self-application. Abbreviating SII as ω, we have a 'self-applicator': $\omega x \twoheadrightarrow xx$ as in Table 2 (2). This leads to the term $\omega\omega$ admitting a cyclic reduction as in Table 2 (3).

CL-terms without variables (so, only built from S, K, I) are also called *combinators*. As observed earlier (Scott [75]), combinators are fun. Much of the fun derives from the fact that every CL-term F has a fixed point P: for, consider $P \equiv \omega(BF\omega)$, then we have $FP = P$, in fact $P \twoheadrightarrow FP$ (Table 2 (4)). This feature embodies the possibility of recursive definitions and is exploited to implement functional programming languages as in the TRS called SKIM (S-K-I-Machine) described in Turner [79] (see Table 3).

$Sxyz$	$\to xz(yz)$	$Uz(Pxy)$	$\to zxy$	
Kxy	$\to x$	$\underline{\text{cond}}\ T\,xy$	$\to x$	
Ix	$\to x$	$\underline{\text{cond}}\ F\,xy$	$\to y$	
$Cxyz$	$\to xzy$	$A\,\underline{n}\,\underline{m}$	$\to \underline{n+m}$	
$Bxyz$	$\to x(yz)$	$M\,\underline{n}\,\underline{m}$	$\to \underline{n.m}$	
Yx	$\to x(Yx)$	$E\,\underline{n}\,\underline{n}$	$\to T$	
$P_0(Pxy)$	$\to x$	$E\,\underline{n}\,\underline{m}$	$\to F$	*if* $n \neq m$
$P_1(Pxy)$	$\to y$			

Table 3

This TRS has infinitely many constants: apart from the constants S, K, ..., E there is a constant $\underline{n}$ for each natural number $n \in \mathbb{N}$. There are also infinitely many reduction rules, because the last four rules (for A, M, E) are actually *rule schemes*; e.g. $A\,\underline{n}\,\underline{m} \to \underline{n+m}$ stands for all reduction rules $A\,\underline{0}\,\underline{0} \to \underline{0}$, $A\,\underline{0}\,\underline{1} \to \underline{1}$, ..., $A\,\underline{37}\,\underline{63} \to \underline{100}$, ... (Historically, such reduction rules were called δ-*rules* by Church.) In fact, the extra constants in SKIM are introduced for reasons of efficient implementation; they can all be defined using only S and K, in such a way that 'reduction is respected'. Definitions of B, C are in Table 2. (To define the fixed point operator Y of SKIM, we should take a definition different from the one in Table 2 (7): note that there a conversion is established which is not a reduction.) For the other definitions one may consult Barendregt [84] or Hindley & Seldin [86].

CL derives its expressive power from the property of *combinatory completeness*: for every CL-term N containing no other variables than $x_1, ..., x_n$, there is a combinator M such that $Mx_1...x_n \twoheadrightarrow N$. Combinatory completeness implies the existence of a fixed point combinator, yielding the following 'recursive definition principle': for every 'reduction equation' $\xi x_1...x_n \twoheadrightarrow C[\xi]$, where $C[\ ]$ contains no other variables than $x_1, ..., x_n$, one can find a combinator as a solution for ξ. (It is not hard to extend this principle to multiple recursion.) In fact, combinatory completeness, and hence the recursive definition principle, holds for every *applicative* extension of CL, like SKIM; somewhat remarkably it does not hold as soon as a function symbol is added to CL with arity ≥ 1.

The salient fact about reduction in CL and its extension SKIM is:

1.1.1. THEOREM. (i) CL *and* SKIM *have the Church- Rosser property (or: are confluent). This means that* $\forall t, s, r\ \exists u\ (t \twoheadrightarrow s\ \&\ t \twoheadrightarrow r \Rightarrow s \twoheadrightarrow u\ \&\ r \twoheadrightarrow u)$.
Or, equivalently, $\forall r, s\ \exists u\ (r = s \Rightarrow s \twoheadrightarrow u\ \&\ r \twoheadrightarrow u)$. *(See Figure 1.)*
(ii) *Both TRSs have the unique normal form property.*

We have to explain the statement in (ii). A term is a *normal form*, or in normal form, if it does not contain redexes as subterms. Equivalently: t is in normal form if there does not exist an s such that t → s. A TRS has the *unique normal form property*, abbreviated UN, if whenever two normal forms are convertible, they must be identical: ∀t, s (t = s & t, s are normal forms ⇒ t ≡ s).

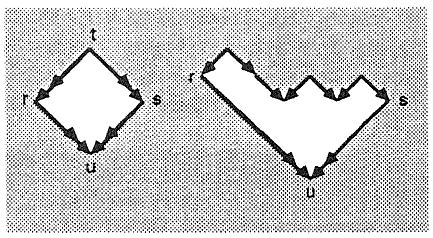


Figure 1

Statement (i) of 1.1.1, the Church-Rosser theorem for CL and SKIM, requires a nontrivial proof. Later on we will mention a confluence criterion by means of which the Church-Rosser property (or CR for short) can be derived very quickly from an inspection of only the left-hand sides of the reduction rules. (This confluence criterion is the property of 'orthogonality' enjoyed by both TRSs.) Statement (ii) follows almost trivially from (i), the confluence (CR) property. Indeed, we have in general: CR ⇒ UN.

1.2. Repeated variables

Above, in the system SKIM, we encountered the rules E $\underline{n}$ $\underline{n}$ → T for all 'numerals' $\underline{n}$. One may ask why we do not simply replace this infinity of reduction rules by the single rule Exx → T. This reduction rule is called 'non-left-linear'. A term is linear when it contains no repeated variables; a rule is left-linear when its left-hand side is linear.

Rather surprisingly, SKIM with the rule scheme E $\underline{n}$ $\underline{n}$ → T replaced by the single rule Exx → T would no longer be confluent—even though the resulting TRS is *weakly confluent* (has the weak CR property, WCR). A TRS is weakly confluent if ∀t, s, r ∃u (t → s & t → r ⇒ s ↠ u & r ↠ u). (See Figure 2.)

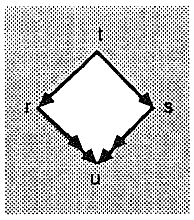


Figure 2

In fact, the same state of affairs holds also for CL extended with the non-left-linear rule Dxx → T (we use D to avoid confusion with the E of SKIM; D can be read as 'discriminator').

1.2.1. THEOREM. *Let* R *be* CL *extended with the rule* Dxx → T. *Then* R *is weakly confluent, but not confluent. Yet,* R *has the unique normal form property. Furthermore, the equational theory of* CL ∪ {Dxx → T} *is conservative over that of* CL.

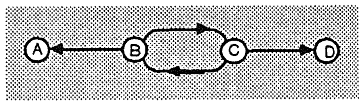
The weak confluence is easily established by simple casuistics. The essence of the proof of non-confluence is as follows. Consider the 'ad hoc' TRS (ad hoc in contrast with the general purpose character of CL) with rules: Dxx → T, Fx → Dx(Fx), P → FP. (So the signature consists of the application operator Ap, not visible due to the 'applicative notation' explained above, and of constants D, T, F, P, and nothing else.) Now we have P → FP → DP(FP) → D(FP)(FP)) → T, hence FP ↠ T and FP ↠ FT. For the confluence property, the terms T and FT should have a common reduct. However, T is a normal form, and FT admits as only reductions: FT → DT(FT) → DT(DT(FT) → DT(DT(DT(FT))) → ... .

Therefore confluence fails—for the ad hoc TRS. Now, using fixed point constructions as hinted at in Section 1.1, one can define the ad hoc TRS in CL ∪ {Dxx → T}, thereby respecting the reduction relation, and obtain the non-confluence of CL ∪ {Dxx → T}.

For a more precise account of theorem 1.2.1 and its proof, see Klop [80] and Klop & de Vrijer [89].

1.3. Newman's Lemma and Abstract Reduction Systems

As we have seen in 1.2, weak confluence does not necessarily imply confluence. This fact is already apparent by considering the TRS with only the constants A, B, C, D and reduction rules B → A, B → C, C → B, C → D. This TRS does also not have the property UN.


$$\begin{tikzcd} A & B \arrow[l] \arrow[r, bend left] & C \arrow[l, bend left] \arrow[r] & D \end{tikzcd}$$
Figure 3

An obvious thought is to use WCR in a tiling procedure to find common reducts. If the tiling succeeds, we get a successfully completed *reduction diagram* as in Figure 4(a). Here we may have to use some 'empty steps' to keep matters in a rectangular shape (these are the shaded steps). Some experiments show that if the reduction diagram, constructed by tiling on the basis of the WCR property, grows into an infinite reduction diagram, *there always arise infinite reductions*. (See Figure 4(b) for the simplest infinite reduction diagram, and Figure 4(c) for an infinite diagram with a curious fractal-like border.)

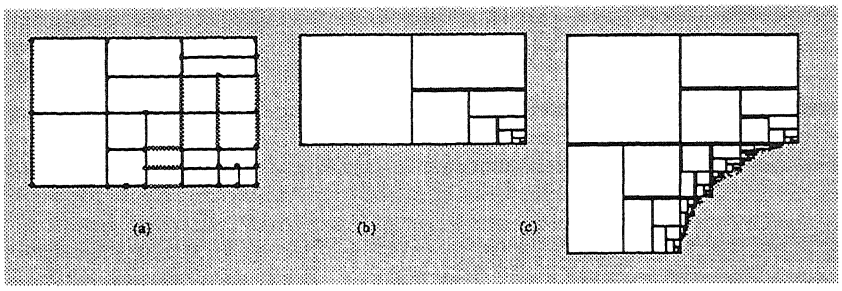


Figure 4

Indeed, the following lemma states that if there are no infinite reductions, then the tiling procedure must succeed. We use the abbreviation SN (*Strong Normalization*) to indicate that there are no infinite reductions. A TRS with the property SN is also said to be *terminating* or (Huet [80]) *noetherian*.

1.3.1. NEWMAN'S LEMMA. SN & WCR $\Rightarrow$ CR.

Various proofs have been given of this fundamental lemma. The (too) complicated proof in Newman [42] (Theorem 3) uses numerical estimations. Another proof can be given using a version of König's Lemma for infinite, finitely branching "dag's" (directed acyclic graphs). Klop [90] contains a proof by means of 'proof orderings', using multisets. Huet [80] contains an elegant proof using noetherian induction. A rather similar proof, equally elegant, is in Barendregt [84]: assuming SN, we need only to prove for confluence that no point is 'bad', where a point is bad if it reduces to different normal forms. From WCR it follows that if t is bad, then $t \to t'$ for some bad t'. Hence, by SN, no bad points exist.

Just like the implication CR $\Rightarrow$ UN that we noted above, Newman's Lemma is a proposition about *Abstract Reduction Systems*; the term structure of a TRS does not play a role. An Abstract Reduction System or ARS for short, is a set equipped with one or more binary relations, that are called reduction relations in view of the applications to TRSs or other rewriting systems. So an ARS has the form $\mathcal{A} = \langle A, \to\rangle$ in case it has only one reduction relation $\to$, or $\mathcal{A} = \langle A, (\to_i)_{i\in I}\rangle$ in the general case. The notions CR, WCR, SN apply already to ARSs. For a collection of facts (mostly criteria for confluence) holding for ARSs, we refer to Staples [75] and Klop [90]. Here we mention a few of the most important of them.

1.3.2. LEMMA . *Let* $\mathcal{A} = \langle A, (\to_\alpha)_{\alpha\in I}\rangle$ *be an ARS such that for all* $\alpha, \beta \in I$ *we have that* $\to_\alpha$ *commutes with* $\to_\beta$. *(This means:* $\forall a, b, c \in A\ \exists d\in A\ (a \twoheadrightarrow_\alpha b\ \&\ a \twoheadrightarrow_\beta c \Rightarrow b \twoheadrightarrow_\beta d\ \&\ c \twoheadrightarrow_\alpha d$*; see Figure 5(a).) Then* $\mathcal{A}$ *(i.e. the union* $\to = \bigcup_{\alpha\in I} \to_\alpha$*) is confluent.*

This lemma is known as the Lemma of Hindley-Rosen. The proof is trivial, but the lemma is very useful.

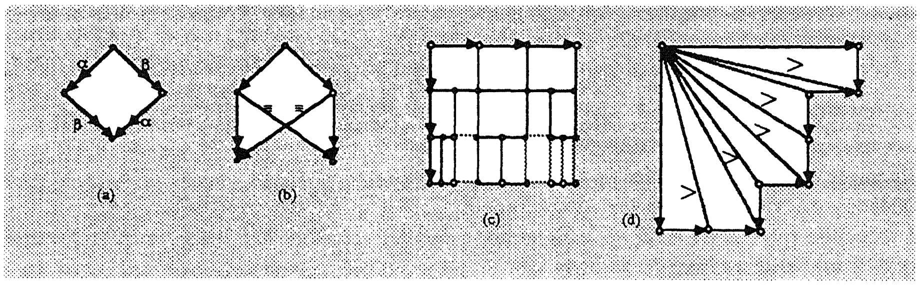


Figure 5

Another simple but useful lemma is:

1.3.3. LEMMA (Huet [80]). *Let* $\langle A, \to\rangle$ *be an ARS. Suppose that* $\forall a, b, c \in A\ \exists d \in A\ (a \to b\ \&\ a \to c \Rightarrow b \twoheadrightarrow d\ \&\ c \to^{\equiv} d)$. *(A reduction relation* $\to$ *with this property is called 'strongly confluent'; see Figure 5(b).) Then* $\to$ *is confluent (see Figure 5(c)).*

The next lemma strengthens Newman's Lemma (see Figure 5(d)):

1.3.4. LEMMA (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* $\leftarrow$.

To conclude this sample of confluence criteria for ARSs we mention an unpublished result of N.G. de Bruijn that recently was brought to our attention and which is a considerable strengthening of Huet's strong confluence lemma 1.3.3. In contrast with the previous lemma's, this one is not easy to prove. We use the following notation:

Let $\mathcal{A} = \langle A, (\to_n)_{n\in I}\rangle$ be an ARS with I a partial order. Then, for a, b $\in$ A, $a \twoheadrightarrow_{<n} b$ means that there is a sequence of reduction steps from a to b, each reduction step having index < n. Analogously $a \twoheadrightarrow_{\leq n} b$ is defined. Furthermore, $\to_n^{\equiv}$ is the reflexive closure of $\to_n$.

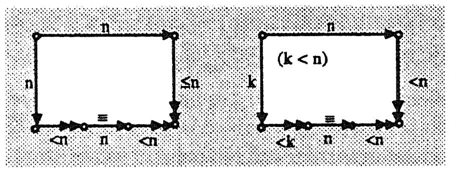


Figure 6

1.3.5. LEMMA (De Bruijn [78]).
Let $\mathcal{A} = \langle A, (\to_n)_{n\in I}\rangle$ *be an ARS with* I *a well-founded linear order. Suppose that*
(i) $\forall a, b, c, n\ \exists d, e, f\ (a \to_n b\ \&\ a \to_n c \Rightarrow b \twoheadrightarrow_{\leq n} f\ \&\ c \twoheadrightarrow_{<n} d \to_n^{\equiv} e \twoheadrightarrow_{<n} f)$, *and*
(ii) $\forall a, b, c, n, k\ \exists d, e, f\ (k < n\ \&\ a \to_n b\ \&\ a \to_k c \Rightarrow b \twoheadrightarrow_{<n} f\ \&\ c \twoheadrightarrow_{<k} d \to_n^{\equiv} e \twoheadrightarrow_{<n} f)$.
Then $\mathcal{A}$ *is confluent.*

1.4. Disjoint unions of Term Rewriting Systems

After the intermezzo about Abstract Reduction Systems we return to Term Rewriting Systems. Often a TRS can be partitioned in some parts that have a *disjoint alphabet* (or signature). Let us denote for TRSs R_1, R_2 having disjoint alphabets with $R_1 \oplus R_2$ the TRS that results by taking the union of R_1, R_2, both with respect to the alphabets and the sets of reduction rules. (If it is not required that the alphabets are disjoint we denote the union just by $R_1 \cup R_2$.) If for a property $\mathcal{P}$ we have: $R_1 \oplus R_2 \models \mathcal{P} \Leftrightarrow R_1 \models \mathcal{P}\ \&\ R_2 \models \mathcal{P}$, we call $\mathcal{P}$ a *modular* property. (Here $\models$ is informally used as abbreviation for 'satisfies'.) A pleasant state of affairs is that we have:

1.4.1. THEOREM (Toyama [87b]). *Confluence is a modular property of TRSs.*

This theorem has useful applications. For instance, consider the extension of CL with a *binary* discriminator:

$$\mathsf{Sxyz} \to \mathsf{xz(yz)}, \quad \mathsf{Kxy} \to \mathsf{x}, \quad \mathsf{Ix} \to \mathsf{x}, \quad \mathsf{D(x, x)} \to \mathsf{T}.$$

This extension CL $\oplus$ {D(x, x) $\to$ T} is confluent, because CL is and because the one rule TRS {D(x, x) $\to$ T} also is confluent, as is easily seen by an application of Newman's Lemma. This should be contrasted with our earlier observation that the extension CL $\cup$ {Dxx $\to$ T} is not confluent. Indeed, the latter is not a disjoint union since Dxx $\to$ T is in fact Ap(Ap(D, x), x) $\to$ T, revealing that there is an overlap in alphabets between CL and Dxx $\to$ T.

Is termination (SN) also modular? No: Toyama [87a] gives the following simple counterexample. Take R_1 = {F(0, 1, x) $\to$ F(x, x, x)} and R_2 = {or(x, y) $\to$ x, or(x, y) $\to$ y}; both TRSs are terminating. They are also disjoint. However $R_1 \oplus R_2$ has an infinite reduction:

$$\mathsf{F(or(0, 1), or(0, 1), or(0,1))} \to \mathsf{F(0, or(0, 1), or(0, 1))} \to \mathsf{F(0, 1, or(0, 1))} \to \mathsf{F(or(0, 1), or(0, 1), or(0,1))}.$$

In view of the fact that R_2 is not confluent, one may conjecture that '*confluent & terminating*' (usually called '*complete*', sometimes also '*canonical*') is a modular property, but this also fails as a more complicated counterexample shows (Toyama [87a]). However:

1.4.2. THEOREM (Toyama, Klop & Barendregt [89]).
Let R_1, R_2 *be left-linear TRSs. Then* $R_1 \oplus R_2$ *is complete iff* R_1 *and* R_2 *are complete.*

(Reminder: A TRS is left-linear if no rule contains repeated variables in its left-hand side.)

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 Theorem 1.4.4.

1.4.3. DEFINITION. (i) A rewrite rule $t \to s$ is a *collapsing* rule (c-rule) if s is a variable. (ii) 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: $F(x, x) \to G(x, x)$ 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.

1.4.4. THEOREM. *Let* R_1 *and* R_2 *be TRSs both with the property* SN.
(i) *If neither* R_1 *nor* R_2 *contain c-rules,* $R_1 \oplus R_2$ *is* SN.
(ii) *If neither* R_1 *nor* R_2 *contain d-rules,* $R_1 \oplus R_2$ *is* SN.
(iii) *If one of the TRSs* R_1, R_2 *contains neither c- nor d-rules,* $R_1 \oplus R_2$ *is* SN.

Statements (i) and (ii) are proved in Rusinowitch [87]; statement (iii) is proved in Middeldorp [89b]. Another useful fact, proved in Middeldorp [89a], is that UN is a modular property. More on the theme of modular properties can be found in recent work of Kurihara & Kaji [88] and Kurihara & Ohuchi [89].

1.5. Decidability

As is to be expected, most properties of TRSs are undecidable. Consider only TRSs with finite signature and finitely many reduction rules. Then it is undecidable whether confluence holds, and also whether termination holds (Huet & Lankford [78], Klop [90]). (Even for TRSs with only one rule termination is undecidable.) However, for *ground* TRSs (where all rules are between ground terms, i.e. no rule contains variables), confluence is decidable (Dauchet et al. [87], Oyamaguchi [87]). Also termination is decidable for ground TRSs (Huet & Lankford [78]).

For particular TRSs it may also be undecidable whether two terms are convertible, whether a term has a normal form, whether a term has an infinite reduction. A TRS where all these properties of terms are undecidable is CL (Barendregt [84]).

2. KNUTH-BENDIX

2.1. Equational Logic

We will now explain two important applications that TRSs, and especially *complete* TRSs, have in Equational Logic: to *decide word problems*, and to *solve equations* in some equational theory. Equational Logic is concerned with equational theories (or equational specifications) of the form (Σ, E) where Σ is a signature as before and E is a set of equations over Σ.

We suppose familiarity with the *semantics* of Equational Logic, that is, with the concept of a Σ-algebra $\mathcal{A}$ and the notion $\mathcal{A} \models t = s$, expressing validity of the equation $t = s$ between Σ-terms in $\mathcal{A}$. If all equations of E are valid in the algebra $\mathcal{A}$, we write $\mathcal{A} \models E$. The *variety* of Σ-algebras defined by an equational specification (Σ, E), notation $Alg(\Sigma, E)$, is the class of all Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \models E$.
Instead of $\forall \mathcal{A} \in Alg(\Sigma, E)\ \mathcal{A} \models t = s$, we write $(\Sigma, E) \models t = s$.

A simple *inference system* for Equational Logic is given in Table 4. If an equation $t = s$ between Σ-terms is derivable by means of this inference system, using the equations of (Σ, E) as axioms, we write $(\Sigma, E) \vdash t = s$. We then have the well-known completeness theorem 2.1.1.

2.1.1. THEOREM (Birkhoff [35]). *Let* (Σ, E) *be an equational specification. Then for all* $t, s \in Ter(\Sigma)$:

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

Axioms *(in addition to the equations in* E*)*:	
$t = t$	*(reflexivity)*
Inference rules:	
from $t_1 = t_2$ *infer* $t_2 = t_1$	*(symmetry)*
from $t_1 = t_2$, $t_2 = t_3$ *infer* $t_1 = t_3$	*(transitivity)*
from $t_1 = t_2$ *infer* $t_1[x := t] = t_2[x := t]$	*(substitution (1))*
from $t_1 = t_2$ *infer* $t[x := t_1] = t[x := t_2]$	*(substitution (2))*

Table 4

The *validity problem* or *uniform word problem* for (Σ, E) is: given an equation $t = s$ between Σ-terms, decide whether or not $(\Sigma, E) \models t = s$. According to Birkhoff's completeness theorem this amounts to deciding $(\Sigma, E) \vdash t = s$. Now we can state why complete TRSs (i.e. TRSs which are SN and CR, terminating and confluent) are important. Suppose for the specification (Σ, E) we can find a complete TRS (Σ, R) such that for all terms $t, s \in Ter(\Sigma)$: $t =_R s \Leftrightarrow (\Sigma, E) \vdash t = s$. (Here $=_R$ is the convertibility relation of R.) Then, provided R has only finitely many rewrite rules, we have a positive solution for the validity problem, obtained by this simple algorithm: reduce s and t to their respective normal forms s', t' and compare; $t =_R s$ iff $t' \equiv s'$.

2.1.2. Groups, L-R algebras and R-L algebras: an example

To illustrate the use of complete TRSs for equational specifications we turn to the classical example given in Knuth & Bendix [70]. Apart from the three axioms for group theory as in the first column of Table 5, one may consider two closely related theories: the three axioms for L-R theory, and three axioms for R-L theory. At first sight it is not clear whether these different theories determine different varieties. Let us call an algebra satisfying L-R theory an L-R algebra, and likewise for a R-L algebra. Now Knuth & Bendix [70] find complete TRSs for these theories as in the table.

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 5

From these completions it is immediately clear that the three varieties indeed are different. Clearly, it follows that each group is also an L-R algebra and a R-L algebra. Furthermore, the equation $x\cdot e = x$ is not derivable in L-R theory, because the normal forms of left-hand side and right-hand side of this equation, with respect to the L-R completion, are I(I(x)) and x, so syntactically different. Hence (by Birkhoff's completeness theorem) there is an L-R algebra which is not a R-L algebra. Also, there is a R-L algebra which is not an L-R algebra because in R-L theory the equations e·x = x and x·I(x) = e are not derivable. Finally, it is clear that the variety of groups is the intersection of the other two varieties. The strategy of deciding validity problems in a positive way by providing a complete TRS does not work always; even for very simple (Σ, E) with solvable validity problem it may be impossible to find a complete TRS R with the same equality. A typical obstacle is the presence in E of equations expressing commutativity of some operator. E.g., consider the specification with signature: a constant 0, and a binary function A, and suppose E = {A(x, y) = A(y, x)}. Then there is no complete TRS 'for' E with the same signature Σ. Discovering the one-line proof is left to the reader.

2.2. Critical pair completion

So, in spite of the drawback that we just mentioned, it is important to be able to find complete TRSs for equational specifications. The seminal paper Knuth & Bendix [70] demonstrated a method, *Knuth-Bendix completion* or *critical pair completion*, that does the job—at least fairly often. The best way to get an understanding of the method is to complete by hand the specification of groups as in Table 5, which amounts to two pages of computation. Here we will consider a simpler example, which is less spectacular than the group completion, but shorter.

Consider the following equational specification (or theory) E of the integers (ℤ) with 0, +, successor S and predecessor P (left column, (a)):

(a)		(b)	
	0 + x = x	(1)	0 + x → x
	x + 0 = x	(2)	x + 0 → x
	S(x) + y = S(x + y)	(3)	S(x) + y → S(x + y)
	x + S(y) = S(x + y)	(4)	x + S(y) → S(x + y)
	x + P(y) = P(x + y)	(5)	x + P(y) → P(x + y)
	P(S(x)) = x	(6)	P(S(x)) → x

(Since this specification is intended to be symmetrical with respect to permuting S and P one might expect also the equations S(P(x) = x and P(x) + y = P(x + y) to be included in E. Actually these equations are derivable.)

We will now perform an 'intuition guided' completion of E. First let us adopt as rewrite rules all equations from E, oriented from left to right. This yields rules (1 - 6) as above in column (b). This is not yet a complete TRS; though it is terminating (as will be seen later), it is not confluent. The reason is that there is an overlap between the left-hand sides of (3), (5) that is harmful: S(x) + P(y) reduces with (3) to S(x + P(y)) and with (5) to P(S(x) + y). These two terms form what is called a *critical pair*. The terms can be reduced further: S(x + P(y)) → S(P(x + y)) and P(S(x) + y) → P(S(x + y)) → x + y, but then we are stuck since S(P(x + y)) and x + y are normal forms with respect to (1 - 6). So confluence fails.

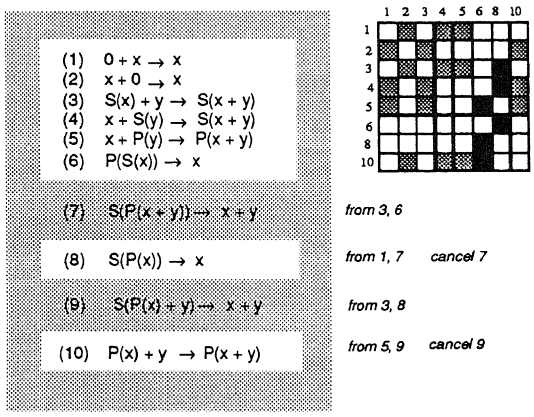


Figure 7

Actually, this is not the only critical pair generated by (1 - 6); there are also overlaps between (1), (2), between (1), (4), between (1), (5), between (2), (3), between (3), (4), between (5), (6). But these critical pairs are harmless. For instance, (5), (6) yield the critical pair P(x + S(y)), x + y. These terms have a common reduct: P(x + S(y)) → P(S(x + y)) → x + y.

We try to solve the problem of non-confluence posed by the terms S(P(x + y)) and x + y in a drastic way: we simply add as a new rule (7) S(P(x + y)) → x + y. Now the critical pair given by (3), (5) is harmless too. However, new critical pairs arise: overlap between (1), (7) yields S(P(0 + y)) with as reducts 0 + y, S(P(y)). This causes us to adopt a new rule: (8) S(P(y)) → y. We can cancel (7) now, as it is a consequence of (8). We consider the possible overlaps: the overlap between (8), (6) is harmless; likewise between (8), (4). But not the one between (8), (3), which causes the introduction of rule (9): S(P(x) + y) → x + y. In this way we continue, and luckily after a few more steps as in Figure 7 we reach a successful conclusion: a TRS R where all critical pairs are harmless. Moreover, R is terminating; this can be seen by noting that all our rules were chosen such that they respected the recursive path ordering (to be explained in the next section) obtained by putting + > S and + > P.

Let us give a precise definition of critical pairs:

2.2.1. DEFINITION. Let the TRS R contain the rewrite rules r: t → s and r': t' → s'. Suppose r, r' are 'standardized apart', i.e. renamed such that they have no variables in common. Suppose furthermore that t ≡ C[u], u not a variable, and that u and t' can be unified with most general unifier σ. Then $t^\sigma \equiv C^\sigma[u^\sigma] \equiv C^\sigma[t'^\sigma]$ is subject to an r'-reduction as well as an r-reduction, with result: $C^\sigma[s'^\sigma]$ respectively s^σ. Now $\langle C^\sigma[s'^\sigma], s^\sigma\rangle$ is called a *critical pair* of R.

If r, r' are (renamed versions of) the same rewrite rule, we moreover require that the context C[ ] is not the trivial context.

Note that if C[ ] in the above situation is trivial (so that t, t' unify 'at the root') two critical pairs are obtained which are mirror-images: $\langle s'^\sigma, s^\sigma\rangle$ and $\langle s^\sigma, s'^\sigma\rangle$. Such critical pairs are sometimes called 'overlays'. (See the chess-board-like table in Figure 7, where it is mentioned which pairs of rules of our example above give rise to critical pairs. The grey squares denote overlays.)

A critical pair ⟨s, t⟩ is *convergent* if s, t have a common reduct (∃r s ↠ r & t ↠ r), notation: s ↓ t. The significance of the fact that all critical pairs ⟨s, t⟩ are 'harmless', i.e. convergent, is expressed by the following lemma.

2.2.1. CRITICAL PAIR LEMMA (Huet [80]). *A TRS is weakly confluent iff all its critical pairs are convergent.*

Convergence of all critical pairs is not sufficient for confluence; a counterexample is the ABCD-TRS in Figure 3 (Section 1.3), with critical pairs (overlays) ⟨A, C⟩, ⟨C, A⟩, ⟨B, D⟩, ⟨D, B⟩. However, in addition to termination, it is sufficient for confluence, according to Newman's Lemma, and we have:

2.2.2. THEOREM (Knuth & Bendix [70]). *A terminating TRS is confluent iff all its critical pairs are convergent.*

As we noted above, convergence of all critical pairs is sufficient for weak confluence, but not for confluence. Huet [80] gave a criterion for critical pairs, stronger than convergence, which does imply confluence while not requiring termination as in the Knuth-Bendix theorem. First define *parallel reduction* as follows: $t \to_{||} s$ if t reduces to s via a reduction sequence consisting of contracting a set of disjoint redexes in t. Thus, if $t_i^{\sigma_i} \to s_i^{\sigma_i}$ (i = 1, ..., n) are contractions, i.e. instances of reduction rules $t_i \to s_i$ (i = 1, ..., n), then $C[t_1^{\sigma_1}, ..., t_n^{\sigma_n}] \to_{||} C[s_1^{\sigma_1}, ..., s_n^{\sigma_n}]$.

2.2.3. THEOREM (Huet [80]). *Let* R *be a left-linear TRS such that we have* $s \to_{||} t$ *for every critical pair* ⟨s, t⟩. *Then* R *is confluent.*

Note the direction involved here. As far as we know, it is an open problem whether the reverse condition also implies confluence: *for all critical pairs* ⟨s, t⟩ *we have* $t \to_{||} s$. Also open seems to be the problem whether confluence is implied by the property: *for all critical pairs* ⟨s, t⟩ *we have* $s \to^{\equiv} t$ or $t \to^{\equiv} s$.

The example of a completion above merely intended to give the flavour of a critical pair completion. Actual completion algorithms (for some simple versions see Klop [90]) would not start with adopting oriented versions of all equations that are initially given, as we did above. Rather, there will be a step by step conversion of E into a complete TRS R (if possible; the algorithm may fail), with as intermediate stages pairs (E', R'), where E' contains some equations and R' contains some rewrite rules.

An important aspect in critical pair completion algorithms is that we need to have an ordering of terms at our disposal, guiding us (or the algorithm) how to choose orientations. Thus, at the start of a completion procedure, one must provide the algorithm with a so-called *reduction ordering* on terms; this is a well-founded partial order among terms which is closed under substitutions and contexts, i.e. if $s > t$ then $C[s^\sigma] > C[t^\sigma]$ for all substitutions σ and contexts $C[\ ]$.

The original specification E does not prove $x + y = y + x$ (this follows immediately from the fact that $x + y$ and $y + x$ are different normal forms of R); yet for all ground terms t, s we have $E \vdash t + s = s + t$. That is: $x + y = y + x$ is valid in the initial algebra of E. Such an equation is called an *inductive theorem* (since its validity is usually proved with induction to the structure of ground terms). Completion techniques provide the means to prove inductive theorems without using induction ("inductionless induction"); another phrase in this respect is "proof by consistency". For an account of proof by consistency applications, see Dershowitz & Jouannaud [90] and Bachmair [88].

To conclude this section about completion, we mention an important recent development in proving correctness of completion algorithms. This is the method of *proof orderings*, introduced by Bachmair, Dershowitz & Hsiang [86]; for details we refer also to Dershowitz & Jouannaud [90], Klop [90].

2.3. Termination

Clearly, termination is an important property of TRSs (see Newman's Lemma), and therefore it is important to have methods to prove termination. In general, it is undecidable whether a TRS is terminating, but some quite sophisticated methods have been devised to prove termination for many TRSs. We present the most powerful of these methods (in a simplified version), known as the method of *recursive path orderings*. (Actually, there are some refinements of the method which are even stronger; they are not discussed here.) The method is based on a powerful theorem of Kruskal [60], which is too beautiful not to mention even in a short survey.

Let $t, s \in \mathrm{Ter}(\Sigma)$. We say that t is *embeddable in* s, notation $t << s$, if $s \twoheadrightarrow_S t$ with respect to the TRS (Σ, S) consisting of the rules $F(t_1, ..., t_n) \to t_i$ for all $1 \leq i \leq n$ and all n-ary $F \in \Sigma$. (S stands for simplification.) Example: $F(H(A), B) << F(G(H(A), A), H(B))$. Note that not $F(A, B) << F(G(A, B), A)$.

2.3.1. KRUSKAL'S TREE THEOREM. *Let* $t_1, t_2, ...$ *be a a sequence of terms, such that in the sequence only finitely many symbols (function symbols, constants, variables) appear. Then for some* i, j *with* $i < j$ *we have* $t_i << t_j$.

Let us now define the recursive path ordering. We will define it using some auxiliary terms using markers. Let $\Sigma^* = \Sigma \cup \{F^* \mid F \in \Sigma\}$ (F a function or constant symbol from Σ; F^* has the same arity as F). Example: if $F(H(A), B) \in \mathrm{Ter}(\Sigma)$, then $F^*(H^*(A), B) \in \mathrm{Ter}(\Sigma^*)$. Note that $\mathrm{Ter}(\Sigma) \subseteq \mathrm{Ter}(\Sigma^*)$. Now suppose Σ finite and suppose that function and constant symbols of Σ are partially ordered by $>$. We define a reduction relation $\Rightarrow$ on $\mathrm{Ter}(\Sigma^*)$, with the following reduction rules.

(1)	$F(\mathbf{t})$	$\Rightarrow F^*(\mathbf{t})$
(2)	$F^*(\mathbf{t})$	$\Rightarrow G(F^*(\mathbf{t}), ..., F^*(\mathbf{t}))$ *if* $F > G$
(3)	$F^*(\mathbf{t})$	$\Rightarrow t_i \quad (i = 1, ..., n)$
(4)	$F^*(\mathbf{p}, G(\mathbf{s}), \mathbf{q})$	$\Rightarrow F(\mathbf{p}, G^*(\mathbf{s}), \mathbf{q})$

Table 6

Here $\mathbf{t} = t_1, ..., t_n$ and $\mathbf{s} = s_1, ..., s_m$ with $t_i, s_j \in \mathrm{Ter}(\Sigma^*)$. Furthermore, $F, G \in \Sigma$ are function symbols with arities $n, m \geq 0$ respectively (so in rule (2) there are in the right-hand side m copies of $F^*(\mathbf{t})$). In rule (1), (2) the arity of F may be 0; in rule (3), (4) it is clear that the arity of F has to be at least 1. In (4), $\mathbf{p}$, $G(\mathbf{s})$, $\mathbf{q}$ is a sequence of n elements from $\mathrm{Ter}(\Sigma^*)$, where $\mathbf{p}$, $\mathbf{q}$ may be empty sequences. Since $\Rightarrow$ is a reduction relation, it is understood that reduction steps according to (1-4) may be performed within a Σ^*-context. With $\Rightarrow^*$ we denote the transitive reflexive closure of $\Rightarrow$, with $\Rightarrow^+$ the transitive closure. Note that the simplification reduction $\twoheadrightarrow_S$ is contained in $\Rightarrow^*$, i.e. if $s \twoheadrightarrow_S t$ then $s \Rightarrow^* t$.

Intuitively, attaching a marker as in rule (1) signifies a command to make the term smaller. The other rules express one step of the execution of this command, which is fully executed if all markers have disappeared. Clearly, $\Rightarrow$ on $\mathrm{Ter}(\Sigma^*)$ is not terminating; see the reduction rule (2). However, the restriction of $\Rightarrow^+$ to $\mathrm{Ter}(\Sigma)$, the set of unmarked terms, is terminating (SN). This is proved by first showing that $\Rightarrow^+$ is a strict partial order. (The difficult part here is to show the acyclicity.) Now termination follows at once from Kruskal's Tree Theorem: suppose there is an infinite sequence $t_1 \Rightarrow^+ t_2 \Rightarrow^+ \ldots$ of Σ-terms. Then for some $i < j$ we have $t_i \ll t_j$, i.e. $t_j \twoheadrightarrow_S t_i$, and therefore $t_j \Rightarrow^* t_i$. But then we have a cycle: $t_i \Rightarrow^+ t_j \Rightarrow^* t_i$.

So $\Rightarrow^+$ is a well-founded ordering on $\mathrm{Ter}(\Sigma)$. This ordering is called the *recursive path ordering.* (Usually, the ordering is defined such that it is preserved under permutations of the arguments $\mathbf{t}$ of a term $F(\mathbf{t})$; we will not do so here.) The recursive path ordering can be used for termination proofs of TRSs as follows. Let (Σ, R) be a TRS with finite Σ. (The method can be extended to deal with infinite signatures, however.) Suppose the function and constant symbols of Σ can be partially ordered in such a way that for the corresponding recursive path order $\Rightarrow^+$ we have, for every reduction rule $s \to t$ of R, that $s \Rightarrow^+ t$. Then (since $s \Rightarrow^+ t$ implies $C[s^\sigma] \Rightarrow^+ C[t^\sigma]$ for every context $C[\,]$ and instantiation σ) it follows immediately that $\to$ must be terminating too. It is instructive to see the method at work in the following example of Dershowitz:

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

Table 7

Order the operators as follows: $\neg > \wedge > \vee$. Now we have e.g. for the second rule $\neg(x \vee y) \Rightarrow^+ (\neg x) \wedge (\neg y)$, since:

$$\neg(x \vee y) \Rightarrow \neg^*(x \vee y) \Rightarrow (\neg^*(x \vee y)) \wedge (\neg^*(x \vee y)) \Rightarrow (\neg(x \vee^* y)) \wedge (\neg^*(x \vee y)) \Rightarrow$$
$$(\neg x) \wedge (\neg^*(x \vee y)) \Rightarrow (\neg x) \wedge (\neg(x \vee^* y)) \Rightarrow (\neg x) \wedge (\neg y).$$

Likewise for the other rules. Hence the reduction relation $\to$, computing disjunctive normal forms, is terminating.

Just as we encountered the presence of commutativity axioms $x + y = y + x$ in E as an obstacle for finding a complete TRS R for E, we encounter problems in proving termination via the recursive path ordering (rpo) method of a TRS containg a rewrite rule expressing associativity of an operator: $(x + y) + z \to x + (y + z)$. In fact, the rpo method will not work in this case as is easily seen. However, in contrast with the obstacle of commutativity axioms, this time the problem is surmountable: an extension of the rpo method with a lexicographic component will do the job of proving termination. For details we refer to Dershowitz [87], an extensive survey of termination proof methods.

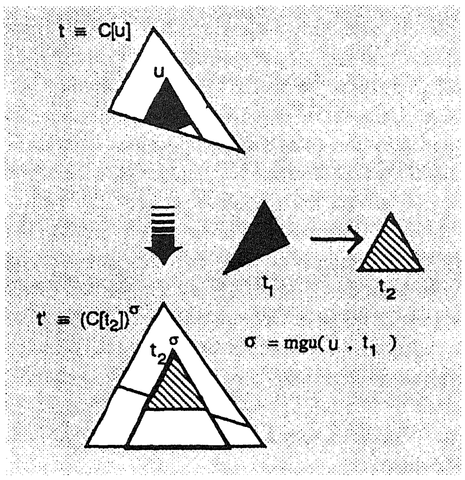
An interesting fact is proved in Kurihara & Ohuchi [89]: 'simple termination' is a modular property (in contrast with the general case, see section 1.4). A TRS is *simply terminating* if the termination can be proved by the rpo method (or by some other termination proof methods, like polynomial orderings, not discussed here).

2.4. Narrowing

After our discussion of one major application of complete TRSs, viz. *deciding validity of equations* in an equational theory, we will now briefly discuss another major application: *solving equations* in an equational theory. 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 $\mathcal{X} \subseteq \mathrm{SUB}$, let $\sigma\mathcal{X}$ denote $\{\sigma\tau \mid \tau \in \mathcal{X}\}$. (Here $\sigma\tau$ is the composition of σ, τ written in the usual logic programming notation; in ordinary mathematical notation it would be $\tau \circ \sigma$.) Now noting that for every substitution 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 kinds 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 $t = s$ at hand. 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 $r = r$ (that is SUB):

$$[\![t = s]\!]_E \supseteq \sigma\,[\![t^\sigma = s^\sigma]\!]_E =$$
$$\sigma\,[\![r = s^\sigma]\!]_E \supseteq \sigma\sigma'\,[\![r^{\sigma'} = s^{\sigma\sigma'}]\!]_E =$$
$$\ldots \supseteq \ldots \supseteq \sigma\sigma'\ldots\sigma^{(n)}\,[\![r = r]\!]_E.$$



narrowing step on terms

Figure 8

The last solution set of this 'narrowing' chain has as a most general element the substitution $\sigma\sigma'\ldots\sigma^{(n)}$. 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 defined on terms, as in the following definition. Suppose E is given as (or equivalent to) a TRS R.

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

(Here 'mgu' abbreviates 'most general unifier'.)

We now extend the narrowing transformation, which was defined on terms, to equations: if $t \twoheadrightarrow_\sigma t'$, then $t{=}s \twoheadrightarrow_\sigma t'{=}s^\sigma$ and likewise $s{=}t \twoheadrightarrow_\sigma s^\sigma{=}t'$ are said to be narrowing steps on equations. As we have seen, the word narrowing actually refers to the solution sets: if $t{=}s \twoheadrightarrow_\sigma t'{=}s^\sigma$ then $[\![t{=}s]\!]_R \supseteq \sigma[\![t'{=}s^\sigma]\!]_R$. 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. For more precise information on the subject of completeness of narrowing, we refer to Hölldobler [89] or one of the extensive surveys mentioned before. We conclude this subsection by drawing attention to the fact that the narrowing relation on terms is actually a *generalization of reduction*: if $t \to s$ then $t \twoheadrightarrow_\sigma s$ for some σ that leaves the variables occurring in t unaffected.

3. BEYOND

3.1. Orthogonality

From the previous chapters it is clear that the two main obstacles to obtain confluence of a TRS are: the presence of repeated variables (non-left-linear reduction rules), and the presence of critical pairs (overlapping reduction rules). Both obstacles need to be fatal for confluence. In the presence of non-left-linear rules we may have confluence, provided the TRS in question is not 'too strong'. For example, $R_1 = CL \cup \{D(x, x) \to T\}$ is confluent, but the stronger system $R_2 = CL \cup \{Dxx \to T\}$ is not. The former TRS can be viewed as a sub-TRS of the latter; namely, by restricting term formation in R_2 such that each D has to have two 'arguments' (i.e. each D appears in a subterm (Dst)) we have a sub-TRS which is 'isomorphic' to R_1. There is a more precise sense in which R_2 can be said to be stronger: R_2 is still combinatory complete but R_1 is not. (It is not hard to show that R_1 does not possess a ground term F, built from S, K, I, D, satisfying Fx → D(x, I).) Also the other potential obstacle, the presence of critical pairs, does not per se prohibit confluence as the Knuth-Bendix theorem 2.2.2 shows. Nevertheless, the presence of one or two of these obstacles requires extra conditions in order to ensure confluence.

A particularly attractive class of TRSs arises if we forbid both obstacles, repeated variables as well as critical pairs. This is the class of *orthogonal* TRSs, characterized by the definition that all reduction rules are left-linear and non-overlapping. A comment on the terminology is in order. Instead of orthogonal, in the literature also the phrases 'consistent', 'left-linear and non-ambiguous' and 'regular' are used. The last term was introduced in Klop [80] and subsequently adopted by some authors, but is deemed objectionable by other authors. Dershowitz proposed in private communication to employ the term 'orthogonal' and since this term has in the present context exactly the right intuitive connotations, we adopt this term.

The orthogonality requirement allows the development of a sizeable amount of theory, of which the basic fact is the following.

3.1.1. THEOREM. *All orthogonal term rewriting systems are confluent.*

Various proofs have been given of this theorem, e.g. the one in Rosen [73]. Note that the theorem is also an immediate consequence of Huet's theorem 2.2.3: since there are no critical pairs, the condition of that theorem is trivially satisfied. Intuitively, the theorem can be understood very well by realizing that in an orthogonal TRS the reduction steps in a reduction diagram, constructed in order to find a common reduct, move in an orthogonal way 'through each other' (see Figure 9), thereby retaining their identity. The orthogonality, in the sense of independence, of reductions in an orthogonal TRS can be understood by the following reformulation of the absence of critical pairs. Let a *redex pattern* be a left-hand side of a reduction rule where the variables have been replaced by □, the symbol denoting an empty place. As an example, consider the three redex patterns of the TRS CL, in the notation with explicit application operator:

Ap(Ap(Ap(S, □), □), □),   Ap(Ap(K, □), □),   Ap(I, □).

Now in a CL-term, even though it may contain nested redexes, it is easily seen not to be possible that the patterns of these redexes overlap (example: Figure 10(b)). So, reduction of some redex R does not disturb (the pattern of) a super-redex S containing R, nor does it disturb a sub-redex S' contained by R (though the reduction of R may multiply S' into a number $n \geq 0$ of copies). In general, in an orthogonal TRS, redex patterns do not overlap, which is just a rephrasing of the statement that there are no critical pairs. (Note, for contrast, the heavy overlapping into a term from the TRS R in section 2.2 as in Figure 10(a).)

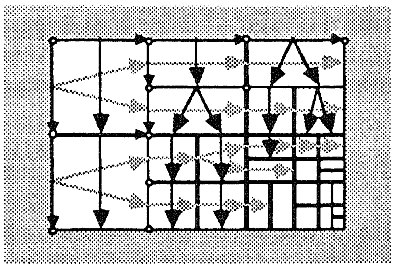


Figure 9

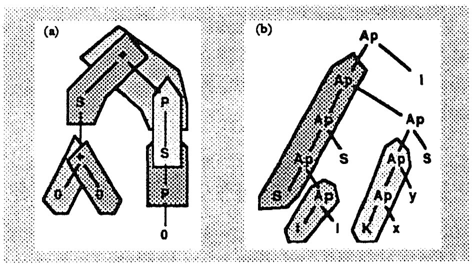


Figure 10

So CL is an orthogonal TRS. Also its extension SKIM is orthogonal, hence confluent.

Many interesting theorems can be proven for orthogonal TRSs. We mention two fundamental ones. Let WN (*Weak Normalization*) be the property of a TRS that every term has a normal form (though infinite reductions may exist). Let NE (*non-erasing*) be the syntactical property of a TRS which holds if in every rewrite rule t → s, both sides t, s contain the same variables (so CL is not NE). Let WIN (*Weak Innermost Normalization*) be the property of a TRS stating that every term has a normal form which can be reached by an innermost reduction, i.e. a reduction in which only redexes are contracted that do not properly contain other redexes. The properties SN, WN, WIN can also be specialized to individual terms:

SN(t) states that t has no infinite reductions, WN(t) means that t has a normal form, WIN(t) that t has a normal form reachable by innermost reduction. Now we have the following facts.

3.1.2. THEOREM (O'Donnell [77]). (i) *For orthogonal TRSs:* NE $\Rightarrow$ (WN $\Leftrightarrow$ SN).
(ii) *For orthogonal TRSs:* WIN $\Leftrightarrow$ SN.
(iii) *For every term* t *in an orthogonal TRS:* WIN(t) $\Leftrightarrow$ SN(t).

3.2. Strategies and sequentiality

Since terms may have a normal form as well as an infinite reduction, one is interested in formulating 'strategies' that as much as possible avoid the infinite reductions and are heading for the normal form. Formally, we define:

3.2.1. DEFINITION. (i) Let R be a TRS. A *one step* or *sequential reduction strategy* $\mathbb{F}$ for R is a map from Ter(R) to Ter(R) such that t $\to$ $\mathbb{F}$(t), if t is not in normal form, and t $\equiv$ $\mathbb{F}$(t) if t is a normal form.
(ii) Likewise, $\mathbb{F}$ is a *many step* or *parallel reduction strategy* for R, if t $\to^+$ $\mathbb{F}$(t) if t is not in normal form, and t $\equiv$ $\mathbb{F}$(t) if t is a normal form.
(iii) A reduction strategy is *normalizing* when it finds the normal form whenever it exists, i.e. if t has a normal form t', then $\exists$n $\mathbb{F}^n$(t) $\equiv$ t'.

We consider five of the main strategies. First, two sequential strategies: (1) the *leftmost outermost* (or *normal order*) strategy; (2) the *leftmost innermost* strategy. In the first, each time the leftmost outermost redex is contracted; in the second, the leftmost of the innermost redexes. Next, there are these three parallel strategies: (3) the *parallel outermost* strategy; (4) the *parallel innermost* strategy; and (5) the *'full substitution'* (or Gross-Knuth) strategy. In the third, all outermost redexes are contracted in one parallel 'step'; in the fourth, all innermost redexes. Of course, outermost redexes are pairwise disjoint, so performing this parallel step is unproblematic, and likewise for the innermost redexes. Actually, the definition of strategies (1-4) makes sense for arbitrary TRSs; this is not so for the full substitution strategy, where each time a 'parallel' step is performed consisting of *contracting all redexes present at that time.* For general TRSs, this notion is not well-defined, but for orthogonal TRSs there can be shown to be an unequivocal result of contracting all redexes that are already present at once. We restrict our consideration of strategies in the sequel to orthogonal TRSs; not much can be said for the general case.

The two innermost strategies are not interesting from the point of view of normalization; in fact, they are 'anti-normalizing', finding infinite reductions whenever possible.

Although finding normal forms is important, we are often interested in terms that do not have a normal form, but rather an 'infinite normal form'. (We will not attempt a formal definition of infinite normal form here.) For example, in SKIM one can define, using combinatory completeness and the fixed point combinator, terms φ, A* such that:

$$\varphi \twoheadrightarrow P\underline{1}(A^*\varphi(P\underline{0}\varphi))$$
$$A^*xy \twoheadrightarrow P(A(P_0x)(P_0y))(A^*(P_1x)(P_1y)).$$

Here P, P_0, P_1 are the pairing, respectively unpairing constants from SKIM, and A is addition (see Table 3). Using pairing, we may have infinite sequences of natural numbers, or rather potentially infinite sequences, i.e. terms t such that t $\equiv$ t_0, $t_k \twoheadrightarrow$ P $\underline{n}_k$ t_{k+1} (so t represents the sequence $\underline{n}_0$, $\underline{n}_1$, $\underline{n}_2$, ...). Now A* represents pointwise addition of such infinite sequences, and as one may check, φ represents the sequence of Fibonacci numbers. A closer analysis of φ will reveal that this term has no normal form. Yet we need a strategy to compute the 'infinite normal form' of φ, since there are also reductions of φ that make no progress towards the infinite normal form. A normalizing strategy will not do now. Here we need a stronger version: a 'cofinal' reduction strategy.

3.2.2. DEFINITION. $\mathbb{F}$ is a *cofinal* reduction strategy if for every reduction t $\twoheadrightarrow$ s there is an n such that s $\twoheadrightarrow$ $\mathbb{F}^n$(t).

Clearly, a cofinal strategy is normalizing; but the reverse need not be the case.

3.2.3. THEOREM. *For all orthogonal TRSs:*

(i) *The parallel outermost reduction strategy is normalizing, although not necessarily cofinal.*

(ii) *The 'full substitution' reduction strategy is cofinal, hence normalizing.*

(iii) *The leftmost outermost reduction strategy is normalizing for 'left-normal' TRSs.*

In (iii), a TRS is *left-normal* if for every reduction rule $t \to s$, in t the variables are to the right of the function and constant symbols (in the linear term notation). E.g. CL is left-normal, SKIM is not due to the rule for U. Proofs of (i-iii) can be found e.g. in O'Donnell [77] or Klop [80].

A paradigm example in considerations of sequential versus parallel reduction is *Berry's TRS*, with the three rules:

$$F(A, B, x) \to 0, \qquad F(x, A, B) \to 1, \qquad F(B, x, A) \to 2.$$

When added to, say, CL, the resulting TRS seems to require a parallel reduction strategy for normalization. For, in a term F(p, q, r) there seems at first sight to be no computable way of detecting the 'right' redex; selecting a redex in say r, we might find ourselves in the case that p, q, reduce to A, B respectively, and then our selection of a redex in r was useless. Likewise, by symmetry, for selections of a redex in one of the other two arguments. Note that it is undecidable whether indeed p, q reduce to A, B respectively. (The same problem occurs with the rules {$\underline{\text{or}}(x, \underline{\text{true}}) \to \underline{\text{true}}$, $\underline{\text{or}}(\underline{\text{true}}, x) \to \underline{\text{true}}$}; however, in contrast with the present example, these are not orthogonal .) Actually, it has been claimed in some papers that CL together with Berry's TRS does not admit a computable sequential normalizing strategy. However, there is the following remarkable fact.

3.2.4. THEOREM (Kennaway [89]). *Every orthogonal TRS possesses a computable, sequential, normalizing reduction strategy.*

In general, the algorithm involved in Kennaway's theorem is, however, too complicated to be of more than theoretical interest. So, the task remains to isolate a (decidable) class of orthogonal TRSs for which 'feasibly computable' sequential normalizing strategies exist. This complicated problem was successfully tackled by Huet & Lévy [79]. Moreover, the sequential strategy that they established has the virtue of not only being normalizing, but also of being efficient: no wasteful reductions are performed.
They defined a property of orthogonal TRSs called 'strong sequentiality'; a strongly sequential TRS admits the selection (by a simple algorithm) of a so-called 'needed' redex, needed for reaching the normal form, in the sense that every reduction to normal form must contract one of the descendants of that redex. Since, as proved in Huet & Lévy [79], repeated contraction of a needed redex inevitably finds the normal form if it exists, we have therefore a normalizing sequential strategy that is computable by a simple algorithm.

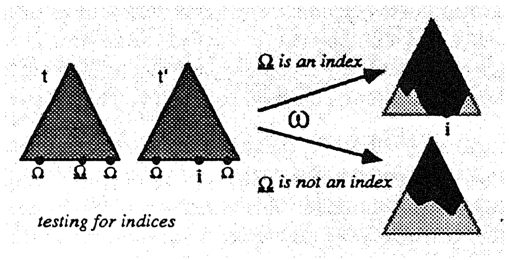


Figure 11

One may ask why we are interested in sequential strategies when parallel strategies (parallel outermost and full substitution) are available that are normalizing for all orthogonal TRSs. The reason is that we do not want to be forced to evaluate in parallel; maybe we have only a reduction machine at our disposal that operates in a sequential way.

Let us describe now the algorithm in Huet & Lévy [79] detecting needed redexes. In general, it is undecidable whether a redex in a term is needed. However, for strongly sequential TRSs the algorithm will always point to at least one needed redex.

First, the signature Σ is extended with a constant Ω (playing the same role as the symbol $\Box$). Terms M, N, ... are henceforth over the alphabet $\Sigma \cup \{\Omega\}$ and are also called Ω-terms. M is a *prefix* of N if M results from N by replacing some subterms of N by Ω. A *redex compatible* term is a prefix of a redex. A new reduction $\rightarrow_\omega$ is defined: if P is a redex compatible term, $P \not\equiv \Omega$, then $C[P] \rightarrow_\omega C[\Omega]$ for arbitrary Ω-contexts C[ ]. The reduction $\rightarrow_\omega$ is confluent and terminating; the $\rightarrow_\omega$-normal form of M, notation $\omega(M)$, is called the *fixed part* of M.

Now suppose that a term M, without Ω's and not in normal form, is given and that we want to determine among the outermost redexes of M one that is needed. To that end, all outermost redexes of M are replaced by Ω, result $C[\Omega, ..., \Omega]$ where all Ω's are displayed. An Ω-occurrence in $C[\Omega, ..., \Omega]$ is called an *index*, when the following "i-test" is positive. Replace the Ω-occurrence under investigation by a fresh symbol i (see Figure 11). Now determine the fixed part. If and only if the test-symbol i is preserved in the fixed part, it indicates an index-Ω. The redex present in the original M at the place of this Ω, is a needed one. As remarked earlier, repeated contraction of a needed redex is a normalizing sequential strategy.

Not for all orthogonal TRSs we will find an index-Ω when running the i-test on all Ω-occurrences in $C[\Omega, ..., \Omega]$. But for *strongly sequential* orthogonal TRSs the i-test will find at least one index-Ω in every $C[\Omega, ..., \Omega]$ as defined above. This fact can even be taken as a definition of strongly sequential TRSs. Moreover, strong sequentiality is a decidable property of TRSs; the proof is rather intricate. It is important to realize that all concepts involved in the definition of index and of strong sequentiality are independent of the right-hand sides of the reduction rules; this is what makes strong sequentiality decidable.

3.3. Conditional rewriting

A very prominent place in investigations of term rewriting in the last years is taken by the theme of *conditional* rewriting. Conditional Equational Logic originated in Universal Algebra, from the need to deal with conditional equations for algebraic structures, as for instance a left-cancellation law $xy = xz \rightarrow y = z$. Conditional equations were also studied in the field of Abstract Data Types, not only because they provide easier specifications, but also because they can be shown to have a greater expressive power. Furthermore, conditional equations turn out to be an essential ingredient in Equational Logic Programming (Hölldobler [89]), an attempt to integrate logic programming and functional programming.

We will write conditional equations in the form $e \Leftarrow e_1, ..., e_n$ where $e, e_1, ..., e_n$ are equations between first order terms. The reversed implication is used in accordance with the usual logic programming notation; its right-hand side $e_1, ..., e_n$ is the conjunction of these equations. Much of Equational Logic (section 2.1) generalizes easily to Conditional Equational Logic, such as initial algebra semantics and a completeness theorem analogous to Birkhoff's completeness theorem 2.1.1; a simple version of a complete inference system can be found in Klop [90].

After orientation of the head equation e, we have a *conditional rewrite rule*:

$$t \rightarrow s \Leftarrow t_1 = s_1, \dots, t_n = s_n. \qquad (*)$$

Here we require that $t \rightarrow s$ is an ordinary rewrite rule, i.e. t must not be a variable and s contains no more variables than t (although this condition is less natural now). A Conditional Term Rewriting System (CTRS) consists of (a signature Σ and) a set R of conditional rewrite rules. Actually, there is some ambiguity in the definition of a CTRS, due to the possible interpretations of the equality signs in the conditional part $t_1 = s_1, \dots, t_n = s_n$. We will consider the three main interpretations. The terminology stems from Dershowitz, Okada & Sivakumar [87, 88].

(1) *Semi-equational* CTRSs. Here the '=' in (*) is *convertibility* of the rewrite relation $\rightarrow$ (the definition of '=' and '$\rightarrow$' thus depend on each other, but this is no problem since the conditions are positive).
(2) *Join* CTRSs. Here '=' is interpreted as *joinability* or convergence($\downarrow$, see 2.2.1), i.e. having a common reduct.
(3) *Normal* CTRSs. Here '=' is interpreted as $\twoheadrightarrow_!$, defined as: $t \twoheadrightarrow_! s$ if $t \twoheadrightarrow s$ and s is a normal form.

The definition of critical pairs in a CTRS (of one of the above types) is slightly more complicated than for the unconditional case. It will now have a conditional form too. If $t \rightarrow s \Leftarrow E$ and $t' \rightarrow s' \Leftarrow E'$ are conditional rewrite rules, then by a definition analogous to 2.2.1 we find in case of overlap of t and t' the critical pair $\langle C^\sigma[s'^\sigma], s^\sigma\rangle \Leftarrow E^\sigma, E'^\sigma$, also written as a conditional equation $C^\sigma[s'^\sigma] = s^\sigma \Leftarrow E^\sigma, E'^\sigma$. The critical pair $t = s \Leftarrow E$ is called *feasible* in Dershowitz, Okada & Sivakumar [87, 88], if there is a substitution σ such that E^σ is true. Critical pair $t = s \Leftarrow E$ is called *joinable* if for all σ such that E^σ is true, $s^\sigma \downarrow t^\sigma$. The following theorem is the conditional analogue of the Knuth-Bendix theorem 2.2.2.

3.3.1. THEOREM (Dershowitz, Okada & Sivakumar [87, 88]). (i) *Let* R *be a semi-equational CTRS. Then: if* R *is terminating and all critical pairs are joinable, R is confluent.*
(ii) *Let* R *be a join CTRS. Then: if* R *is decreasing and all critical pairs are joinable,* R *is confluent.*

Here R is *decreasing* (a property stronger than termination) if it has a decreasing ordering, i.e. an ordering $>$ on Ter(R) satisfying: (1) $>$ is well-founded; (2) $t \subset s \Rightarrow t < s$ ($\subset$ is the proper subterm ordering); (3) $t \to s \Rightarrow t > s$; (4) for each rewrite rule $t \to s \Leftarrow t_1 \square s_1, \ldots, t_n \square s_n$ and each σ we have $t^\sigma > t_i^\sigma, s_i^\sigma$ ($i = 1,\ldots,n$). Here $\square$ is $=, \downarrow, \twoheadrightarrow_!$.

In general, the property of being a normal form is not decidable in a CTRS. (This is not surprising, since being a normal form depends on some conditions that refer to convertibility or reduction and hence may be undecidable.) However, for decreasing CTRSs being a normal form is decidable.

For more about conditional rewriting, we refer to Kaplan [84, 85], Dershowitz & Plaisted [87]. For conditional critical pair completion see Ganzinger [87]. Recently, Middeldorp [89c] generalized Toyama's theorem 1.4.1 to the case of CTRSs, showing that also in the conditional case (for all three types above mentioned) confluence is a modular property. We conclude this section by mentioning a confluence theorem for orthogonal CTRSs; a CTRS is called orthogonal if its unconditional part, that is the TRS arising after removing all conditions from the conditional rewrite rules, is an orthogonal TRS as defined in 3.1.

3.3.2. THEOREM (Bergstra & Klop [86]). *Orthogonal semi-equational and orthogonal normal CTRSs are confluent. Orthogonal join CTRSs are in general not even weakly confluent.*

3.4. Other rewrite formats

We conclude by mentioning some important other rewriting formats that could not be discussed in this paper. An useful extension of the 'pure' rewrite format is *equational term rewriting*, where one rewrites not terms but equivalence classes of terms. For instance, it is sometimes convenient to work modulo commutativity and/or associativity of some operators; recall the problems, discussed in section 2 above, that commutativity and associativity axioms present to pure term rewriting. For a treatment of this subject we refer to Dershowitz & Jouannaud [90]. Another rewrite format is that of *graph rewriting*, introduced to avoid duplications of subterms in reductions. For implementations this is of crucial importance. For an introduction and further references we refer to Barendregt et. al. [87]. Also not covered here are term rewriting systems with *bound variables*, or Combinatory Reduction Systems as they are called in Klop [80, 90]. For instance, λ-calculus is a TRS with bound variables.

As said in the Introduction, the design of functional programming languages poses many stimulating questions to the study of TRSs. E.g. Peyton Jones [87] introduces *λ-calculus with patterns*, with (instead of the usual β-reduction rule of the ordinary λ-calculus) the reduction rule $(\lambda P.\ M)N \to M^\sigma$, if $\sigma = \mathrm{mgu}(P, N)$ exists. Here P is a linear term built from 'constructors' and free variables (a 'pattern'). (Without the linearity, i.e. if P may contain repeated variables, the system is not Church-Rosser.) With $P \equiv x$ we get the β-reduction rule. (We have not defined the concept of 'constructors'. An interesting calculus, without referring to constructors, and in the syntax of pure λ-calculus, arises if P above is taken to be a linear λ-calculus term in normal form.) How much of the well-known theory of the λ-calculus can be generalized to this extension?

Another feature, common in the practice of functional programming, is the use of *reduction rules with priorities* assigned to them:

$$\underline{\mathrm{fac}}\ \underline{0} \to \underline{1}$$
$$\underline{\mathrm{fac}}\ x \to \underline{\mathrm{mult}}\ x\ (\underline{\mathrm{fac}}\ (\underline{\mathrm{pred}}\ x))$$

Here the first rule should be 'tried' first; only if it is not applicable, the second one may be used. Without this priority assignment, the specification would of course be erroneous. The mechanism of rule priorities is not without some nasty problems; it is studied in Baeten et.al. [88].

As a third example, we mention the question *how to translate (or interpret) rewrite systems into each other*. Peyton Jones [87] contains a translation of a subset of Miranda into the pure λ-calculus. In general, it is known that not every TRS can, in a direct sense, be translated into λ-calculus, not even every orthogonal TRS with the constructor discipline (an example is Berry's TRS). As far as we know, at present it is not rigorously established what subclasses of TRSs (say of orthogonal constructor TRSs) can be 'directly defined' into λ-calculus, by finding λ-terms for the operators of the TRS such that reductions are respected. (This is a vague definition, but part of the question is to find the right notions here. For some proposals in this direction see O'Donnell [85].)

Acknowledgements

I am much obliged to Roel de Vrijer for many detailed criticisms and in general for his help in getting my efforts to write this paper *confluent* and *terminating*. (Yet, the paper is not *complete*, but that is solely due to the author.)

Thanks are also due to the Computer Science Department of the Università degli Studi di Milano, where a part of this paper was written in a stimulating and hospitable environment created by, among others, Giovanni Degli Antoni and Nicoletta Sabadini.

References

BACHMAIR, L. (1988). *Proof by consistency in equational theories*. In: Proc. 3rd Symp. on Logic in Computer Science, Edinburgh. 228-233.

BACHMAIR, L., DERSHOWITZ, N. & HSIANG, J. (1986). *Orderings for equational proofs*. In: Proc. of the IEEE Symp. on Logic in Computer Science, Cambridge, Massachusetts, 346-357.

BAETEN, J.C.M., BERGSTRA, J.A., KLOP, J.W. & W.P. Weijland. (1988). *Term rewriting systems with rule priorities*. TCS 67 (1989) 283-301.

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

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: Proc. PARLE Conf., Springer LNCS 259, 141-158.

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

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

DE BRUIJN, N.G. (1978). *A note on weak diamond properties*. Memorandum 78-08, Eindhoven Univ. of Technology, 1978.

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

DAUCHET, M., TISON, S., HEUILLARD, T. & LESCANNE, P. (1987). *Decidability of the confluence of ground term rewriting systems*. In: Proc. of the 2nd Symp. on Logic in Computer Science, Ithaca, NY, 1987, 353-359.

DERSHOWITZ, N. (1987). *Termination of rewriting*. J. of Symbolic Computation 3 (1&2), 69-115, 1987. Corrigendum: Vol.4, No.3, 409-410.

DERSHOWITZ, N. & JOUANNAUD, J.-P. (1989). *Rewrite systems*. To appear as Chapter 15 of Vol.B of "Handbook of Theoretical Computer Science", North-Holland, 1989. (Rapport de Récherche no.478, Unité Associée au CNRS UA 410: AL KHOWARIZMI, Avril 1989.)

DERSHOWITZ, N., OKADA, M. & SIVAKUMAR, G. (1987). *Confluence of Conditional Rewrite Systems*. In: Proc. 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: Proc. of 9th Conf. on Automated Deduction, Argonne, Springer LNCS 310, 538-549.

DERSHOWITZ, N. & PLAISTED, D.A. (1987). *Equational Programming*. In: Machine Intelligence 11 (eds. J.E. Hayes, D. Michie and J. Richards), Oxford University Press, 21-56.

GANZINGER, H. (1987). *A completion procedure for conditional equations*. In: Proc of the 1st Intern. Workshop on Conditional Term Rewriting Systems, Orsay 1987, Springer LNCS 308, 1988, 62-83.

GOGUEN, J.A. & MESEGUER, J. (1985). *Initiality, induction, and computability*. In: Algebraic methods in semantics (eds. M. Nivat and J.C. Reynolds), Cambridge University Press 1985, 459-542.

HARDIN, T. (1989). *Confluence results for the pure Strong Categorical Logic CCL. λ-calculi as subsystems of CCL*. Theor. Comput Sci. Fundamental Studies, Vol.65, Nr.3, 1989, 291-342.

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

HÖLLDOBLER, S. (1989). *Foundations of Equational Logic Programming*. Springer Lecture Notes in A.I. 353.

HSIANG, J. (1985). *Refutational Theorem Proving using Term-Rewriting Systems*. Artificial Intelligence, 25, Vol.3, 1985.

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

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

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

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

KAPLAN, S. (1984). *Conditional Rewrite Rules.* TCS 33(2,3), 1984.

KAPLAN, S. (1985). *Fair conditional term rewriting systems: Unification, termination and confluence,* Recent Trends in Data Type Specification (ed. H.-J. Kreowski), Informatik-Fachberichte 116, Springer-Verlag, Berlin, 1985.

KENNAWAY, J.R. (1989). *Sequential evaluation strategies for parallel-or and related reduction systems.* Annals of Pure and Applied Logic 43 (1989) 31-56.

KLOP, J.W. (1980). *Combinatory Reduction Systems.* Mathematical Centre Tracts Nr.127, CWI, Amsterdam.

KLOP, J.W. (1990). *Term Rewriting Systems,* in: Handbook of Logic in Computer Science (eds. S. Abramsky, D. Gabbay and T. Maibaum) Vol.1, Chapter 6, Oxford University Press, to appear.

KLOP, J.W. & DE VRIJER, R.C. (1989). *Unique Normal Forms for Lambda Calculus with Surjective Pairing.* Information and Computation 80 (2), 97-113.

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

KRUSKAL, J.B. (1960). *Well-Quasi-Ordering, the Tree Theorem, and Vazsonyi's Conjecture.* Trans. AMS 95, 210-225.

KURIHARA, M. & KAJI, I. (1988). *Modular Term Rewriting Systems: Termination, Confluence and Strategies.* Report, Hokkaido University.

KURIHARA, M. & OHUCHI, A. (1989). *Modularity of Simple Termination of Term Rewriting Systems.*Report 89-SF-31, Hokkaido University, Sapporo, 1989.

MIDDELDORP, A. (1989a). *Modular aspects of properties of term rewriting systems related to normal forms.* In: Proc. of 3rd Intern. Conf. 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: Proc. of 4th IEEE Symposium on Logic in Computer Science, Pacific Grove, 396-401.

MIDDELDORP, A. (1989c). *Confluence of the Disjoint Union of Conditional Term Rewriting Systems.* Report CS-R8944, Centre for Mathematics and Computer Science, Amsterdam 1989.

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 MA, 1985.

OYAMAGUCHI, M. (1987). *The Church-Rosser property for ground term rewriting systems is decidable.* Theoretical Computer Science 49 (1).

PEYTON JONES, S.L. (1987). *The Implementation of Functional Programming Languages.* Prentice-Hall 1987.

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

RUSINOWITCH, M. (1987). *On termination of the direct sum of term rewriting systems* IPL 26, 65-70.

SCHÖNFINKEL, M. (1924). *Über die Bausteine der mathematischen Logik.* Math. Annalen 92, 305-316. Translated as: *On the building blocks of mathematical logic,* in From Frege to Gödel, ed. J. van Heyenoort, Harvard Un. Press, 1967, 355-366.

SCOTT, D.S. (1975). *Some philosophical issues concerning theories of combinators.* In: Proc. of Symposium 'λ-calculus and Computer Science Theory', (ed. C. Böhm), Rome 1975, Springer LNCS 37, 346-366.

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

TOYAMA, Y. (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.* JACM, Vol.34, No.1, 1987, 128-143.

TOYAMA, Y., KLOP, J.W. & BARENDREGT, H.P. (1989). *Termination for the direct sum of left-linear term rewriting systems.* In: Proc. of 3rd Intern. Conf. on Rewriting Techniques and Applications, Springer LNCS 355, 477-491.

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

TURNER, D.A.. (1985). *Miranda: a non-strict functional language with polymorphic types.* In: Proc. of the IFIP Intern. Conf. on Functional Programming Languages and Computer Architecture, Nancy. Springer LNCS 201, 1985.

WINKLER, F. & BUCHBERGER, B. (1983). *A criterion for eliminating unnecessary reductions in the Knuth-Bendix algorithm.* In: Proc. of the Coll. on Algebra, Combinatorics and Logic in Computer Science, Györ, Hungary, Sept. 1983.