Term Rewriting Systems with Priorities

J.C.M.Baeten,
Department of Computer Science, University of Amsterdam.

J.A.Bergstra,
Department of Computer Science, University of Amsterdam,
Department of Philosophy, State University of Utrecht.

J.W.Klop,
Centre for Mathematics and Computer Science, Amsterdam,
Department of Computer Science, Free University, Amsterdam.

Abstract: Term rewriting systems with rules of different priority are introduced. The semantics are explained in detail and several examples are discussed.
Note: Partial support received from the European Communities under Esprit contract no. 432, Meteor (An integrated formal approach to industrial software development).

1. Introduction.

Term rewriting systems are an important tool to analyse the consistency of algebraic specifications, and are also becoming increasingly important for implementation. Some general references for algebraic specifications are Ehrig & Mahr [5], Goguen, Thatcher & Wagner [6], Goguen, Thatcher & Wright [7], Klaeren [10] and Kutzler & Lichtenberger [13]. Some general references for term rewriting systems are Huet & Oppen [8], O'Donnell [14, 15] and Klop [11].

For implementation purposes it is sometimes convenient to write down term rewriting systems (TRS's) where some ambiguities between the rules are present, while adopting some restrictions on the use of these rewrite rules to the effect that the ambiguities are not actually 'used'. The mechanism that we discuss in this paper consists of giving priority to some rules over others in cases of 'conflict'. Another extension of the purely equational formalism, that retains the initial algebra semantics, and increases expressive power, is the introduction of conditional equations, see Pletat, Engels & Ehrich [16], Kaplan [9] or Bergstra & Klop [2].

Here we consider a TRS with priorities, called a **priority rewrite system (PRS)**. We study the effect of such a priority assignment to rules, *without imposing further restrictions* such as choosing a certain reduction strategy in combination with rule priorities. That is, we wish to consider the priority mechanism on itself. As to the *executability* of the specification given by a PRS this is a drawback: in general a PRS without more will not be an executable specification. In fact, it turns out that it is rather problematic whether a 'pure PRS' has a well-defined semantics at all. It may even be the case that a pure PRS does not possess a well-defined semantics (i.e. does not determine an actual rewrite relation). Apart from the fact that pure PRS's have some interesting mathematical properties, we find that it is worth-while to establish some facts about them in order to get a feeling what extra mechanisms (e.g. reduction strategies) should be adopted on top of rule priorities. Moreover, a decent subclass of PRS's can be determined which does possess a well-defined semantics. We will also establish a general theorem ensuring confluency for several of

such PRS's. A typical example will be: TRS's with a so-called *'specificity ordering'*.
The theory of PRS's is also useful in connection with **modularity**: we can break up a specification in a number of (parametrized) smaller specifications in ways that are not expressible by means of equational specifications. Moreover we feel that the priority mechanism is rather appealing from an intuitive point of view (indeed it has been used by many authors, but in a rather implicit way).
This article is a revision of [1], also incorporating some subsequent work of the third author.
In §2, we give the basic definitions and some examples. We define when a priority rewrite system is **well-defined**, that is when there exists a unique sound and complete rewrite set associated with it. In §3, we look at bounded PRS's, and show that bounded PRS's are always well-defined. In §4, we prove a stabilization lemma, which gives a different way of showing well-definedness. Finally, in §5, we prove a confluency result for left-linear PRS's.

2. Basic definitions.

We start out with some examples, to give the reader an intuitive idea of a PRS.

2.1 Example: Consider the signature for the natural numbers with predecessor, successor, sum and zero, and the rewrite rules in table 1.

	r1:	$P(0) \rightarrow 0$
	r2:	$P(S(x)) \rightarrow x$
↓	r3:	$x + 0 \rightarrow x$
↓	r4:	$x + y \rightarrow S(x + P(y))$

Table 1.

Without the arrow this set of rewrite rules is ambiguous (i.e. more than one rule can be applied to a certain redex), and does not implement our intention (to specify predecessor and sum on the natural numbers). The arrow now means that the third rule (r3) has priority over the fourth (r4). However, there is a caveat: the term $x + P(S(0))$ does not match the left-hand side of r3; but this does not mean that r3 may be 'by-passed' in favour of applying r4 on this term. We may only by-pass r3 if in no subsequent reduction of $y = P(S(0))$ we will get a match with the left-hand side of r3. So, in this case, we are not allowed to by-pass r3 and the correct reduction is:

$$x + P(S(0)) \xrightarrow{r2} x + 0 \xrightarrow{r3} x.$$

2.2 Example: Finite sets of natural numbers with insertion and deletion. The signature consists of

sorts	NAT, SET
functions	S: NAT → NAT
	ins: NAT x SET → SET
	del: NAT x SET → SET
constants	$0 \in$ NAT
	$\emptyset \in$ SET
variables	$x,y,.... \in$ NAT
	$X,Y,.... \in$ SET.

Next to rewrite rules for the natural numbers, there are rewrite rules for insertion and deletion as in table 2.

	r1:	ins(x, ins(x, X)) → ins(x, X)
	r2:	ins(x, ins(y, X)) → ins(y, ins(x, X))
↓	r3:	del(x, ins(x, X)) → del(x, X)
	r4:	del(x, X) → X

Table 2.

Again, r3 has priority over r4. That r4 is 'correct', is because if one is allowed to use it, then del(x, X) does not match the left-hand side of r3, so X is not of the form ins(x, Y), in other words "$x \notin X$", hence $X - \{x\} = X$.

2.3 Example: The factorial function. Add to the rules of table 1 rules for multiplication. Then factorial can be specified as in table 3.

	Fac(0) → S(0)
↓	Fac(x) → x·Fac(P(x))

Table 3.

2.4 Example: In a signature containing booleans, one may encounter rules for equality as in table 4.

	eq(x,x) → T
↓	eq(x,y) → F

Table 4.

Thus, for any specification, containing booleans, adding these equations describes the equality function on a certain sort. We claim that, without using rewrite rules with priority, such a parametrized specification *cannot be found!* Even when one uses auxiliary sorts and functions, or even conditional equations, can such a specification not be found. The proof of this fact rests on the fact, that otherwise each initial algebra would be decidable, and requires a very systematic analysis of initial algebra semantics in the light of computability theory. In essence, this work has been carried out in Bergstra & Klop [3, 4].
Our conclusion is, that equational specifications do not support proper modularisation (in unexpected cases). We claim that priority rewrite systems support modularity much better.

2.5 Definition: A **term rewriting system with priorities** or **PRS (priority rewrite system)** is a pair $(\mathbb{R},<)$, where $\mathbb{R}$ is a term rewriting system and $<$ is a partial order on the set of rules of $\mathbb{R}$.

2.6 Notation: In a listing of rewrite rules we write | r1 when r1 > r2. ↓ r2
Likewise | r1 means r1 > r2 > r3, etc.
r2 Rules not in the scope of the same arrow are incomparable.
↓ r3

2.7 Definition: Let r be a reduction rule of the PRS $\mathbb{R}$.
i. An instantiation (possibly containing variables) of the left-hand side of r is called an **r-redex**.

Note that this is regardless of whether the r-redex is, in view of the priority restrictions, actually 'enabled', i.e. is allowed to be rewritten according to rule r.

ii. A closed instantiation (closed instance) $t \rightarrow s$ of the rewrite rule r is called a **rewrite**. We will write $t \overset{r}{\rightarrow} s$ or $r: t \rightarrow s$.

iii. The closure of the relation $\rightarrow$ under contexts is **one-step reduction**, and denoted by $\rightarrow$.

iv. The transitive and reflexive closure of the relation $\rightarrow$ is **(more-step) reduction**, denoted $\twoheadrightarrow$.

2.8 Definition: Let $F(t_1,....,t_n)$ be some term in a TRS (or $Ft_1.....t_n$ in the case of an applicative TRS). A reduction of F(t) (resp. Ft) is called **internal** if it proceeds entirely in the arguments t (so the head-symbol F is 'unaffected').

2.9 Now we can formulate, in a first intuitive approximation, what reduction relation a PRS is meant to describe:

Let r be a rule of the PRS $\mathbb{R}$ and let t be an r-redex. Then t may be rewritten according to r if *for no rule r'>r it is possible to rewrite t, by means of an internal reduction, to an r'-redex t'*. See figure 1.

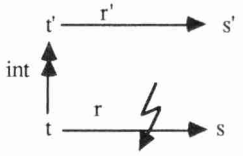

Fig. 1.

To see why the reduction to a 'higher' redex scheme, blocking the 'lower' reduction of t, must be *internal,* one should consider that only internal reductions preserve the 'identity' of the term-to-be-reduced, in casu t. The following example may clarify this:

Consider the PRS in table 2, and consider the r4-rewrite

$$del(0, \underline{del(0, ins(0,\emptyset))}) \overset{r4}{\rightarrow} \underline{del(0, ins(0,\emptyset))}.$$

Intuitively, this application of r4 is correct since the underlined part in the left-hand side denotes a set not containing 0. But if we had stipulated above that the internal reduction could be *any* reduction, the present application of r4 would be illegal since the right-hand side is also a r3-redex and r3 > r4. The point is that the right-hand side is not the 'same' redex as the left-hand side (i.e. the right-hand side is not a descendant of the left-hand side). Indeed, application of r4 on a term del(t, T) is only allowed if it is not the case that $t \twoheadrightarrow s$ and $T \twoheadrightarrow ins(s, S)$ for some s,S. That is, if there is an internal reduction $del(t, T) \overset{int}{\twoheadrightarrow} del(s, ins(s, S))$.

In such an internal reduction, the right-hand side is the 'same' redex as the left-hand side.

Of course, it is not at all clear what reduction relation actually is 'defined' by a PRS. To see the problem, consider the following PRS in table 5 (on the following page). Here 0,1,2,S,T are constants and A,B are unary functions.

Intuitively, every closed term should eventually be rewritten to 0, 1 or 2. In particular, we expect either $T \twoheadrightarrow 1$ or $T \twoheadrightarrow 2$, and $S \twoheadrightarrow 0$ or $S \twoheadrightarrow 2$. There are the following equivalences:

$T \twoheadrightarrow 1 \Leftrightarrow B(S) \twoheadrightarrow 1 \Leftrightarrow S \not\twoheadrightarrow 0 \Leftrightarrow A(T) \not\twoheadrightarrow 0 \Leftrightarrow T \twoheadrightarrow 1;$

$T \twoheadrightarrow 2 \Leftrightarrow B(S) \twoheadrightarrow 2 \Leftrightarrow S \twoheadrightarrow 0 \Leftrightarrow A(T) \twoheadrightarrow 0 \Leftrightarrow T \not\twoheadrightarrow 1;$

$S \twoheadrightarrow 0 \Leftrightarrow A(T) \twoheadrightarrow 0 \Leftrightarrow T \not\twoheadrightarrow 1 \Leftrightarrow B(S) \not\twoheadrightarrow 1 \Leftrightarrow S \twoheadrightarrow 0;$

$S \twoheadrightarrow 2 \Leftrightarrow A(T) \twoheadrightarrow 2 \Leftrightarrow T \twoheadrightarrow 1 \Leftrightarrow B(S) \twoheadrightarrow 1 \Leftrightarrow S \not\twoheadrightarrow 0.$

Hence it is consistent to assume either $T \twoheadrightarrow 1, T \not\twoheadrightarrow 2$ and $S \twoheadrightarrow 2, S \not\twoheadrightarrow 0$,

or $T \twoheadrightarrow 2, T \not\twoheadrightarrow 1$ and $S \twoheadrightarrow 0, S \not\twoheadrightarrow 2$.

In other words, there is no well-defined reduction relation corresponding to the PRS in table 5. We will now proceed to give precise definitions of what is meant by a 'well-defined' reduction relation.

↓	$A(1) \to 2$
	$A(x) \to 0$
↓	$B(0) \to 2$
	$B(y) \to 1$
	$T \to B(S)$
	$S \to A(T)$

Table 5.

2.10 Definition: Let $\mathbb{R}$ be a PRS. Let R be an arbitrary set of rewrites (i.e. closed instantiations of the rules of $\mathbb{R}$). R is called a **sound rewrite set** for $\mathbb{R}$ if:

whenever $t \overset{r}{\to} s \in R$ then there is no internal reduction, using the rewrites of R, of t to an r'-redex with r'>r and such that $t' \overset{r'}{\to} s' \in R$.

Notation: $\overset{R}{\to}$ denotes a reduction using only rewrites from R.

So, in the situation of fig. 2, R is *not* sound.

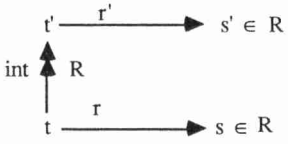

Fig. 2.

Clearly, if a PRS $\mathbb{R}$ determines a reduction relation R as its semantics, we will require that R is sound. Now it might be thought that all we have to do is to look for a *maximal* sound rewrite set for $\mathbb{R}$. However, such a maximal sound rewrite set will not be unique in general, and therefore does not qualify as the semantics of $\mathbb{R}$; another reason is that we will certainly require of the semantics of $\mathbb{R}$ that it contains all r-rewrites for rules r which have maximal priority, and a maximal sound rewrite set need not obey this requirement, as the following example shows.

2.11 Example: Let $\mathbb{R}$ be the PRS with rules and priorities in table 6.

	$A \to B$
↓	$P(B) \to 0$
	$P(x) \to 1$

Table 6.

Then $R_1 = \{A \to B, P(B) \to 0\} \cup \{P(t) \to 1 : t \not\equiv A\}$ is a maximal sound rewrite set (the intended semantics!), but also $R_2 = \{P(B) \to 0\} \cup \{P(t) \to 1 :$ all closed t$\}$ is a maximal sound

rewrite set. As a candidate for the semantics of $\mathbb{R}$, R_2 is unsatisfactory as it does not contain the maximum priority rule instance $A \rightarrow B$.
Note that (in contrast maybe with the PRS in table 5) this PRS is entirely 'plausible'; it resembles the factorial example in table 3 (instead of $A \rightarrow B$ we could have e.g. $0 + 0 \rightarrow 0$).
Now that we have seen that soundness is a 'must' for the PRS-semantics, but that 'maximal sound' is not good enough, we will formulate the second requirement for the semantics of a PRS.

2.12 Definition: Let R be a rewrite set for the PRS $\mathbb{R}$.
i. The rewrite $t \overset{r}{\rightarrow} s$ is **correct w.r.t. R** if there is no internal R-reduction $t \overset{R}{\rightarrow} t'$ to an r'-rewrite $t' \overset{r'}{\rightarrow} s' \in R$ with $r' > r$.
So in the situation of fig. 3, the rewrite $t \overset{r}{\rightarrow} s$ is *not* correct w.r.t. R.

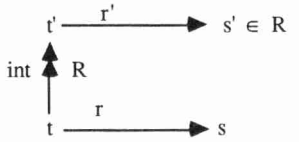

Fig. 3.

ii. R is **complete** if R contains all rewrites which are correct w.r.t. R.

2.13 Notes: i. If R is sound and $t \overset{r}{\rightarrow} s$ is correct w.r.t. R, then $R' = R \cup \{t \overset{r}{\rightarrow} s\}$ need not be sound, since $t \overset{r}{\rightarrow} s$ may be used in an internal R'-reduction making some other rewrite $t^* \rightarrow s^*$ illegal.
ii. This concept of completeness has nothing to do with the notion 'complete' for TRS's, defined as meaning 'confluent and terminating' (see e.g. Klop [12]).
Clearly, we require that the semantics R of a PRS $\mathbb{R}$ is complete: there is no reason to exclude from R a rewrite $t \rightarrow s$ which cannot be shown by R to be illegal. Note that the rewrite set R_2 in example 2.11 is not complete (as $A \rightarrow B$ is correct w.r.t. R_2), but that R_1 is complete.

2.14 Definition: Let the PRS $\mathbb{R}$ have a *unique sound and complete* rewrite set R. Then R is called the **semantics** of $\mathbb{R}$; we also call $\mathbb{R}$ **well-defined**.

2.15 Note: The PRS in table 5 is not well-defined: it has two sound and complete rewrite sets, as indicated there. There is also an example of a PRS with *no* sound and complete rewrite set.

2.16 Note: The direct sum of well-defined PRS's need not be well-defined: G.-J. Akkerman (Free University, Amsterdam) has given an example of a TRS and a well-defined PRS, whose direct sum is not well-defined.

3. Bounded priority rewrite systems.

We will now try to find a reasonable condition guaranteeing the well-definedness of a PRS. Such a

reasonable condition is e.g. that the PRS, after deleting the priority ordering, is a strongly normalising TRS (i.e. there are no infinite reduction sequences). In fact, we can do slightly better: we need only require that the resulting TRS is bounded.

3.1 Definitions: i. Let $\mathbb{R}$ be a TRS, and $R = t_0 \to t_1 \to \ldots$. a possibly infinite reduction sequence in $\mathbb{R}$. Then the reduction R is **bounded** if

$$\exists n \ \forall t_i \in R \quad |t_i| \leq n \qquad (|t_i| \text{ is the length in symbols of } t_i).$$

ii. Let $\mathbb{R}$ be a TRS. Then $\mathbb{R}$ is bounded if all its reduction sequences are.

iii. Let $\mathbb{R}$ be a PRS. Then $\mathbb{R}$ is bounded if the TRS obtained by forgetting the priority ordering is bounded.

3.2 Proposition: i. If the underlying TRS of a PRS is strongly normalising, then it is bounded.

ii. Convertibility of terms in a bounded and confluent TRS is a decidable property (two terms are *convertible* if there are related by the symmetric closure of $\twoheadrightarrow$).

iii. The direct sum of bounded TRS's need not be bounded.

iv. Consider in a TRS $\mathbb{R}$ the following alternating sequence of reductions and *proper* inclusions:

$$t_0 \twoheadrightarrow t_1 \supset t_2 \twoheadrightarrow t_3 \supset t_4 \twoheadrightarrow \ldots.$$

Suppose that this sequence is infinite. Then $\mathbb{R}$ is not bounded.

Proof: i and iv are left to the reader. For ii and iii, see Klop [12] (iii uses a counterexample, similar to one given by Toyama).

3.3 Notation: Let $\mathbb{R}$ be a PRS.

i. R_{max} is the set of *all* rewrites of $\mathbb{R}$ (i.e. all closed instances of all the rules of $\mathbb{R}$).

ii. $Pow_{fin}(R_{max})$ is the collection of finite subsets of R_{max}.

iii. $\alpha, \alpha_0,\ldots,\beta,\ldots$ will vary over R_{max}; $A,A_0,\ldots,B,\ldots$ vary over $Pow_{fin}(R_{max})$.

iv. $LHS(\alpha)$, $RHS(\alpha)$ are given by: $\alpha = LHS(\alpha) \twoheadrightarrow RHS(\alpha)$.

3.4 Definition: $\alpha \lhd A$ (A **obstructs** α) if there is an internal reduction of $LHS(\alpha)$ (say this is an r-redex) to a 'higher' redex (i.e. an r'-redex with r'>r), such that the internal reduction uses precisely all rewrites in A. In figure 4, $t \twoheadrightarrow t'$ is an internal reduction using the rewrites $\beta_1,\ldots,\beta_n$. In this case we have $\alpha \lhd \{\beta_1,\ldots,\beta_n\}$.

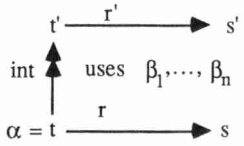

Fig. 4.

We may have that n=0, so $\alpha \lhd \emptyset$. An element (α,A) of $\lhd$ will be called an **obstruction**; A will also be called an obstruction of α.

3.5 Definition: We will now construct the **tree of all obstructions**, with root α. The first level of this tree is shown in fig. 5, where all obstructions of α are displayed.

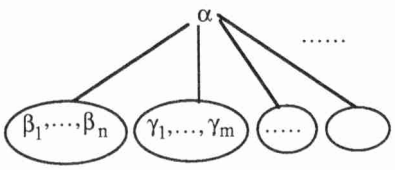

Fig. 5.

The sequence of elements may of course contain repetitions (e.g. β_2 may be the same rewrite as γ_3). The second level of the tree is obtained by appending all the obstructions of the elements $\beta_1,...,\beta_n,\gamma_1,...,\gamma_m,....$ The result of continuing this procedure ad infinitum is called the **obstruction tree** of α, or the tree of obstructions with root α.

3.6 Theorem: Let $\mathbb{R}$ be a PRS such that the obstruction tree of α is well-founded, for every rewrite $\alpha \in R_{max}$. Then $\mathbb{R}$ is well-defined.

Proof: We will envisage the obstruction trees of all $\alpha_0,\alpha_1,.... \in R_{max}$ standing next to one another, as in fig. 6.

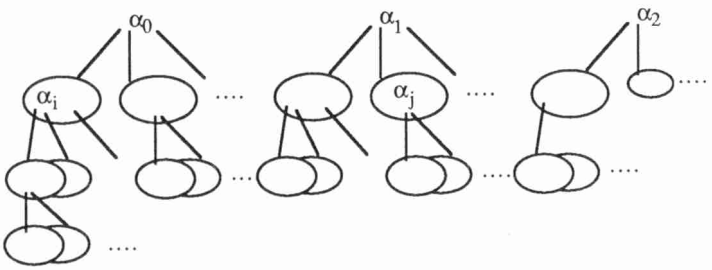

Fig. 6.

Further, to each α_i appearing somewhere in some of these trees, one of the colors red, white and blue will be assigned, in a monotonous procedure. Initially, all α_i are *white* (in fact, white means not colored). At the end of the (transfinitely long) construction, each α_i will be either blue or red (not both). The set of all blue α_i will be the semantics of the PRS $\mathbb{R}$, i.e. the unique sound and complete rewrite set for $\mathbb{R}$. Let us now describe the coloring procedure, which has 5 stages.

Stage 1: All α_i are white.

Stage 2: Since all obstruction trees of $\mathbb{R}$ are well-founded, there are *terminal* α_i (i.e. without successor obstructions). These occurrences are made blue.

Stage 3: Occurrences of α_i having now some obstruction $\{\beta_1,...\beta_n\}$ of which all elements β_j are blue, are made red. Likewise all α_i with an obstruction $\emptyset$ are made red.

Stage 4: Now consider all white α_i lowest in the tree (i.e. such that the whole subtree of that occurrence contains no white element). Clearly, such white α_i are not obstructed by a blue set, since otherwise they would have been red. So every obstruction set of such a white α_i contains a

red element. Since the red elements are definitely not going to be in our semantics to be constructed (the set of blue elements), it is safe to make such white α_i blue. (Moreover, we *must* do so, by the definition of 'complete').

Stage 5: Go to stage 3.

We claim that this coloring procedure will terminate, after possibly transfinitely many steps, and will yield a red/blue partition of all elements $\alpha_i \in R_{max}$. To prove this claim rigorously, we will need some form of bar-induction; here we will be satisfied with a more informal treatment.

It is easy to see that the coloring is not only a coloring of the *occurrences* of the α_i, but also of the elements α_i themselves. To see this, we must show that if one α_i occurs twice or more in the forest of obstruction trees, all these occurrences will get the same color, red or blue. The argument is this: a repetition of one α_i along the same branch is not possible since that branch would then be infinite, in contradiction with the well-foundedness condition. So the only repetitions of α_i are 'incomparable'. Each occurrence of α_i trivially has the same subtree, hence also the same color.

We have to check that the set of blue rewrites is: i. sound; ii. complete; iii. unique.

Ad i: soundness follows immediately from the fact that a blue rewrite α_i cannot be obstructed by a blue set A. Ad ii: a rewrite not in the 'blue semantics', i.e. red, must be obstructed by a blue set. This gives completeness. Ad iii: each coloring is forced.

3.7 Corollary: Bounded PRS's are well-defined.

Proof: Let $\mathbb{R}$ be a bounded PRS, and consider the rewrite $\alpha \in R_{max}$. Then, we claim, the obstruction tree of α is well-founded. Proving the claim will prove the corollary by theorem 3.6.

Suppose the obstruction tree of α is not well-founded. Then there is an infinite sequence

$$\alpha = \alpha_0 \lhd A_0 \ni \alpha_1 \lhd A_1 \ni \alpha_2 \lhd A_2 \ni \ldots\ldots$$

Now $\alpha_0 \lhd A_0$ and $\alpha_1 \in A_0$ imply that $LHS(\alpha_0) \twoheadrightarrow t_0$ and $LHS(\alpha_1) \subset t_0$ $(\neq)$ for some t_0, since there must be an *internal* reduction of $LHS(\alpha_0)$ to a higher redex using the rewrite $\alpha_1 = LHS(\alpha_1) \rightarrow RHS(\alpha_1)$. So we find a sequence

$$LHS(\alpha_0) \twoheadrightarrow t_0 \supset LHS(\alpha_1) \twoheadrightarrow t_1 \supset LHS(\alpha_2) \twoheadrightarrow \ldots\ldots$$

and by proposition 3.2.iv this implies that $\mathbb{R}$ is not bounded.

3.8 Notes: Let $\mathbb{R}$ be a bounded PRS.

i. The obstruction trees of $\mathbb{R}$ are not only well-founded, but even finite.

ii. The semantics of $\mathbb{R}$ (i.e. the unique sound and complete rewrite set R for $\mathbb{R}$) is decidable. (This does not mean that the convertibility relation determined by R is decidable, but that "$\alpha \in R$" is decidable.)

3.9 Example: The PRS $\mathbb{R}$ given in table 7 has semantics $R_{max} - \{P(A) \rightarrow 1, P(B) \rightarrow 1\}$.

$A \rightarrow B$
$P(B) \rightarrow 0$
$P(x) \rightarrow 1$

Table 7.

3.10 Example: Suppose we have a bounded PRS over a sort NAT. Adding the rules in table 2 to this PRS will give a PRS which specifies finite sets of elements of NAT, and which is still bounded (since the rules in table 2 are all length nonincreasing). Thus, by 3.7, this is a well-defined PRS.

3.11 Note: In 3.10, we give a parametrized specification for finite sets over a given sort. In [1], well-defined PRS's are also presented for stacks, queues and trees over a given sort.

4. The stabilization lemma.

The PRS in table 1 (in 2.1) is not bounded. Therefore, we need a criterion for well-definedness of a PRS without the boundedness condition. This is provided in the following stabilization lemma.

4.1 Definition: Let R be a set of rewrites for the PRS $\mathbb{R}$. Then the **closure** of R, written $\overline{R}$, is the set of rewrites that are correct w.r.t. R. Further, we define $\overline{R}^0 = R$, $\overline{R}^{n+1} = \overline{\overline{R}^n}$.

4.2 Lemma: Let R, S be sets of rewrites for the PRS $\mathbb{R}$.
i. R is sound $\Leftrightarrow R \subseteq \overline{R}$
ii. R is complete $\Leftrightarrow R \supseteq \overline{R}$
iii. R is sound and complete $\Leftrightarrow R = \overline{R}$
iv. $R \subseteq S \Rightarrow \overline{R} \supseteq \overline{S}$
v. $R \supseteq S$, S sound and complete $\Rightarrow$ R complete
vi. $R \subseteq S$, S sound and complete $\Rightarrow$ R sound
Proof: i and ii follow immediately from the definitions; iii follows from i and ii, and iv follows from the definition of closure. v: If $R \supseteq S$ then by iv $\overline{R} \subseteq \overline{S} = S$, whence $R \supseteq \overline{R}$. Now use ii. vi: Likewise.

4.3 Corollary: Suppose $R = R_{max}$ and S is sound and complete. Then we have inclusions as in the following diagram:

$$\begin{array}{ccccccc} R = & \overline{R}^0 & \supseteq & \overline{R}^2 & \supseteq & \overline{R}^4 & \supseteq \ .... \\ & \cup| & & \cup| & & \cup| & \\ & S = & & S = & & S = & .... \\ & \cup| & & \cup| & & \cup| & \\ & \overline{R}^1 & \subseteq & \overline{R}^3 & \subseteq & \overline{R}^5 & \subseteq \ .... \end{array}$$

and moreover that $\cap_{n\geq 0} \overline{R}^{2n}$ is complete and $\cup_{n\geq 0} \overline{R}^{2n+1}$ is sound.

4.4 Stabilization lemma: Let $R = R_{max}$, for a PRS $\mathbb{R}$.
If for some n, $\overline{R}^n = \overline{R}^{n+1}$, then $\mathbb{R}$ is well-defined.
Proof: Immediate from 4.3.

4.5 Next, we will describe an application of the stabilization lemma, in proving that the PRS in 2.1 is well-defined. First, define the interpretation $[\cdot]$ from closed terms to natural numbers by

$$[0] = 0 \qquad [S(t)] = succ([t])$$
$$[P(t)] = pred([t]) \qquad [t + s] = [t] + [s],$$

where t,s are closed. Then, define $R = \{P(0) \to 0, P(S(t)) \to t, t + 0 \to t : t \text{ closed}\} \cup$
$\{t + s \to S(t + P(s)) : t,s \text{ closed}, [s] \neq 0\}$.

Claim 1: If $s \xrightarrow{R} 0$, then $[s] = 0$. Proof: Use induction on the formation of s.

Claim 2: If $[s] = 0$, then $s \xrightarrow{R} 0$.

Proof: First prove with induction on n that $\forall\, m,n \;\; S^m(0) + S^n(0) \xrightarrow{R} S^{m+n}(0)$.
Then use this fact to show with induction on t that $\forall$ closed $t \;\exists\, n \;\; t \xrightarrow{R} S^n(0)$.
Then the claim follows from this and claim 1.

Claim 3: $\overline{R_{max}} = R$. Proof: From claims 1 and 2.

We conclude from claim 3, using the stabilization lemma, that the semantics is well-defined.

5. Left-linear priority rewrite systems.

Up to this point, no requirement was made as to the left-linearity of the rules in a PRS. In this section, we will restrict our attention to PRS's which have left-linear rules (i.e. no left-hand side has a multiple occurrence of the same variable), in order to prove (under certain circumstances) a confluency result for them.

We expect that some confluency results also can be obtained for suitably restricted PRS's with non-left-linear rules, as in examples 2.2 and 2.4, but we will not attempt to do so here.

First we will prove a 'general' theorem, namely confluency for essentially regular TRS's.

Ambiguities in the rewrite rules of a TRS may be an obstacle to confluency (see e.g. Klop [11, 12]). Yet, we may allow the presence of ambiguities if there is some additional mechanism (such as rule priorities) which prevents the ambiguities to be actually 'used'. We will conceive such a 'desambiguating' mechanism as a restriction of the sets R_i of rewrites $r_i: t_{i,k} \to s_{i,k}$.

5.1 Definition: Let $r: t(x_1,...,x_n) \to s(x_1,...,x_n)$ be a rewrite rule, and let $t(\rho(x_1),....,\rho(x_n))$ be an r-redex for some substitution ρ. Let t' be another redex occurring in some $\rho(x_i)$. Then this redex occurrence is called a **small** redex occurrence of t' in t. Notation: $t' << t(\rho(x_1),....,\rho(x_n))$.

5.2 Definitions: i. Let $\mathbb{R}$ be a left-linear TRS (possibly ambiguous). Let $R_1,...,R_n$ be the sets of all rewrites of the rules $r_1,....,r_n$ of $\mathbb{R}$, respectively. So $R_{max} = R_1 \cup .... \cup R_n$. Suppose that R_{max} is partitioned into **enabled** rewrites (E) and **disabled** rewrites (D): $R_{max} = D \cup E$.
Then $(\mathbb{R},E)$ is called a **restricted** TRS.

ii. The restricted TRS $\mathbb{R}$ is **essentially non-ambiguous** if

1. E contains no "critical pair of rewrites" $\{t \to s, t' \to s'\}$ as subset. (Here a "critical pair of rewrites" is a pair of rewrites giving rise to what is known as a critical pair of terms.)

2. E is closed under small redex contractions, i.e. if $r: t \equiv C[t'] \to s \in E, r': t' \to s' \in E$

and t' << t, then r: C[s'] → ... ∈ E. (Here the RHS ... is uniquely determined by r and the LHS.)
iii. (ℝ,E) is **essentially regular** if it is essentially non-ambiguous and the rules are left-linear.

5.3 Theorem: Let the restricted TRS (ℝ,E) be essentially regular. Then it is confluent (i.e. the reduction relation generated by E is confluent).
Proof: Entirely similar to the regular case (see e.g. Klop [12]). The weak confluency, necessary to obtain confluency for developments, follows from requirements 1, 2 in definition 5.2.ii.

5.4 Definition: Let ℝ be a PRS. We say that the ordering < of the rules in ℝ is a **specificity ordering** if: i. $r < s \Leftrightarrow$ the LHS of s is a substitution instance of the LHS of r;
ii. the p.o. < consists of linear fragments, i.e. if $r_1, r_2 > r_3$ then either $r_1 > r_2$ or $r_2 > r_1$.
iii. no ambiguities occur between incomparable rules.

5.5 Corollary: Well-defined left-linear PRS's with specificity ordering are confluent.
Proof: It is easily checked that a left-linear PRS (which of course has to be well-defined) with a specificity ordering, is essentially regular.

5.6 Notes: Note that the PRS in 2.1 (table 1) is confluent. Likewise for factorial in 2.3 (table 3).

References.

[1] J.C.M.Baeten, J.A.Bergstra & J.W.Klop, *Priority rewrite systems,* Report CS-R8407, Centre for Mathematics and Computer Science, Amsterdam 1984.
[2] J.A.Bergstra & J.W.Klop, *Conditional rewrite rules: confluency and termination,* JCSS 32, pp. 323-362, 1986.
[3] J.A.Bergstra & J.W.Klop, *Algebraic specifications for parametrized data types with minimal parameter and target algebras,* Proc. ICALP 1982, Springer LNCS 140, 1982.
[4] J.A.Bergstra & J.W.Klop, *Initial algebra specifications for parametrized data types,* EIK 19, pp. 17-31, 1983.
[5] H.Ehrig & B.Mahr, *Fundamentals of algebraic specification 1,* Springer EATCS Monographs on Theor. Comp. Sci., Springer-Verlag 1985.
[6] J.A.Goguen, J.W.Thatcher & E.G.Wagner, *An initial algebra approach to the specification, correctness and implementation of abstract datatypes,* Current trends in Progr. Meth. IV, Data structuring (R.T.Yeh, ed.), Prentice Hall, New Jersey, 1978.
[7] J.A.Goguen, J.W.Thatcher & J.B.Wright, *Abstract datatypes as initial algebras and correctness of datatype representations,* Proc. ACM Conf. on Comp. Graphics, Pattern Recognition and Data Structure, ACM, New York 1975.
[8] G.Huet & D.C.Oppen, *Equations and rewrite rules, a survey,* Formal Lang., Perspectives and Open Problems, Academic Press, 1980.
[9] S.Kaplan, *Conditional rewrite rules,* Theor. Comp. Sci. 33 (2/3), pp. 175 - 193, 1984.
[10] H.A.Klaeren, *Algebraische Spezification, eine Einführung,* Springer Lehrbuchreihe Informatik, 1983.
[11] J.W.Klop, *Combinatory reduction systems,* Math. Centre Tract 127, Amsterdam 1980.
[12] J.W.Klop, *Term rewriting systems,* Notes for Seminar on Reduction Machines, Ustica 1985, to appear.
[13] B.Kutzler & F.Lichtenberger, *Bibliography on abstract datatypes,* Informatische Fachberichte 68, Springer 1983.
[14] M.J.O'Donnell, *Computing in systems described by equations,* Springer LNCS 58, 1977.
[15] M.J.O'Donnell, *Equational logic as a programming language,* MIT Press, 1985.
[16] U.Pletat, G.Engels & H.D.Ehrich, *Operational semantics of algebraic specifications with conditional equations,* Forschungsbericht 118/81, Abteilung Informatik, Univ. Dortmund, 1981.