J. Symbolic Computation (1995) **11**, 1–000

Comparing Curried and Uncurried Rewriting[†]

RICHARD KENNAWAY[‡], JAN WILLEM KLOP[§], RONAN SLEEP[‡] AND FER-JAN DE VRIES[¶]

[‡]*School of Information Systems, University of East Anglia, Norwich NR4 7TJ, U.K.*
[§]*CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*
[¶]*NTT, Communication Science Laboratories, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan*

(Received 1 August 1994)

Currying is a transformation of term rewrite systems which may contain symbols of arbitrary arity into systems which contain only nullary symbols, together with a single binary symbol called application. We show that for all term rewrite systems (whether orthogonal or not) the following properties are preserved by this transformation: strong normalization, weak normalization, weak Church-Rosser, completeness, semi-completeness, and the non-convertibility of distinct normal forms. Under the condition of left-linearity we show preservation of the properties NF (if a term is reducible to a normal form, then its reducts are all reducible to the same normal form) and $\mathrm{UN}^{\rightarrow}$ (a term is reducible to at most one normal form). We exhibit counterexamples to the preservation of NF and $\mathrm{UN}^{\rightarrow}$ for non-left linear systems. The results extend to partial currying (where some subset of the symbols are curried), and imply some modularity properties for unions of applicative systems.

1. Introduction

Curry and Feys originally introduced term rewriting in 1958 in the "applicative" form (or in their terminology, quasi-applicative). In this form, terms are built from variables, nullary function symbols, and a binary application (often suppressed in notation). An alternative is the "functional" form, where terms are constructed from function symbols of various arities and variables.

The main example of an applicative term rewrite system, Combinatory Logic (CL), was developed by Schönfinkel and rediscovered by Curry (Curry and Feys, 1958). Applicative term rewriting systems related to CL play an important role in the design and implementation of functional programming languages such as Haskell and Miranda[||] (cf.

[†] Dedicated to Dirk van Dalen on the occasion of his 60th anniversary. The authors are partially sponsored by SEMAGRAPH II, ESPRIT working group 6345. Richard Kennaway was also partially supported by a SERC Advanced Fellowship and by SERC grant no. GR/F 91582. From July 1995, F.J. de Vries will be at the Hitachi Advanced Research Laboratory, Hatoyama, Saitama 350-03, Japan. The authors' respective email addresses are `jrk@sys.uea.ac.uk`, `jwk@cwi.nl`, `mrs@sys.uea.ac.uk` AND `ferjan@cwi.nl`.

[||] Miranda is a trademark of Research Software Ltd., Canterbury, U.K.

Field and Harrison, 1958). However, the theory of term rewriting nowadays is more usually studied in the functional form (cf. Dershowitz and Jouannaud, 1988; Klop, 1992). It is therefore interesting to study the relationship between the two styles of presentation.

Any functional term rewrite system can be transformed into an applicative term rewrite system by the well-known method of *currying* credited to Schönfinkel in Curry and Feys (1958). So it is natural to ask which properties of term rewrite systems are preserved by currying.

In this paper we show that strong normalization (SN), weak normalization (WN), the weak Church-Rosser property (WCR), the unique normal form property (UN), completeness, and semi-completeness are preserved by currying. For left-linear term rewrite systems we show that currying also preserves the normal form property (NF) and the $\mathrm{UN}^{\rightarrow}$ property. (All these properties are defined in Section 2.5.) Counterexamples demonstrate that NF and $\mathrm{UN}^{\rightarrow}$ are not always preserved for non-left-linear systems. We also explore connections between currying and modular properties.

Kahrs (1994) has recently shown that the Church-Rosser property (CR) is preserved by currying for all rewrite systems. This corrects an error in an earlier version of this paper, which proved preservation of CR for left-linear systems, but claimed a counterexample for non-left linear systems.

We use two different definitions of currying: Schönfinkel's original definition, and a different definition which is not equivalent, but is technically easier to work with. For every term rewrite system, and for each of the properties mentioned above, we prove that the property is preserved for that system by one form of currying if and only if it is preserved by the other. In addition, we consider partial currying, in which the currying transformation is applied to only some of the symbols of the system. All our results for full currying also apply to partial currying.

Finally, we consider the modularity of these properties with respect to unions of curried rewrite systems.

2. Preliminaries

2.1. TERM REWRITING

Dershowitz and Jouannaud (1989) and Klop (1992) contain ample introductions to term rewriting. A *term rewriting system* (TRS) is a pair (Σ, R) of a signature Σ and a set of rewrite rules R. The signature consists of a set of function symbols, each of which has an arity (a non-negative integer). The set $\mathrm{Ter}(\Sigma)$ of terms over Σ is built in the usual way from the function symbols and a countably infinite set of variables, disjoint from Σ. Every variable is a term, and if F is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then $F(t_1, \ldots, t_n)$ is a term. When $n = 0$, we write just F instead of $F()$.

A *position* or *occurrence* is a finite string of positive integers. The empty string is denoted by ϵ. The set of positions $O(t)$ of a term t is defined inductively by:

(1) If t is a variable then $O(t) = \epsilon$.
(2) If $t = F(t_1, \ldots, t_n)$ then $O(t) = \{\epsilon\} \cup \{i \cdot u \mid 1 \leq i \leq n \text{ and } u \in O(t_i)\}$.

Positions are partially ordered by prefix order: $u \leq v$ if there is a (necessarily unique) w such that $u \cdot w = v$. The subterm of t at position $u \in O(t)$ is denoted by $t_{|u}$ and defined inductively by:

(1) $t_{|\epsilon} = t$,
(2) $t_{|i \cdot u} = (t_i)_{|u}$, if $t = f(t_1, \ldots, t_n)$.

If the principal function symbol of $t_{|u}$ is F, t is said to have an occurrence of F at u.

A *context* is a "term" containing one occurrence of a special symbol "$\Box$", denoting an empty place. A context may be written as C[]. The result of substituting a term t for the hole in $C[]$ is denoted $C[t]$. t is said to be a *subterm* of $C[t]$.

A *substitution* is a function from a set of variables to terms. A substitution σ may be extended to be a function from terms to terms by defining $\sigma(t)$ to be the term resulting from replacing every occurrence in t of a variable x in the domain of σ by $\sigma(x)$. Where σ is defined on variables $x_1, \ldots, x_n$, and maps them to $t_1, \ldots, t_n$, we may write $t[x_1 := t_1, \ldots, x_n := t_n]$ for $\sigma(t)$.

A term s is *contained* in (or encompassed by) a term t (notation $s \trianglelefteq t$) if there is a context $C[]$ and a substitution σ such that $t = C[\sigma(s)]$. (Examples: $F(x) \trianglelefteq G(F(a))$ and $x \trianglelefteq y$.) s is also said to be a *component* of t. s will be called *strict* if the substitution σ maps distinct variables to distinct terms, and *linear* if none of the variables in the domain of σ occurs more than once in s. It is clear that every component can be represented as either a strict or a linear component (though not necessarily both simultaneously). We may write the component s as $P[x_1, \ldots, x_n]$, where $x_1, \ldots, x_n$ consists of all the variables in the domain of σ (perhaps with repetitions), and write the corresponding subterm of t as $P[t_1, \ldots, t_n]$, where $t_i = \sigma(x_i)$ for all i.

A *rewrite rule* is a pair of terms, written $l \rightarrow r$, subject to the following two conditions: l must not be a variable, and every variable occurring in r must also occur in l. l and r are called the left- and right-hand sides of the rule. A rewrite rule $l \rightarrow r$ is called *collapsing* if r is a variable. It is *duplicating* if there is a variable which occurs more often in r than it does in l.

A *rewrite system* R consists of a signature Σ and a set of rewrite rules over Σ. We also write Ter(R) for its set of terms, i.e. Ter(Σ).

A set of rewrite rules determines a *reduction relation* $\rightarrow$ on Ter(Σ). A *redex* of a term t is an occurrence u and a rule $l \rightarrow r$ such that l is a component of t at u. t thus has the form $C[\sigma(l)]$; the term $C[\sigma(r)]$ is the result of *reducing* this redex. We write $t \rightarrow s$ when t reduces to s. $\rightarrow^=$ is the reflexive closure of $\rightarrow$, $\rightarrow^+$ its transitive closure, $\rightarrow^*$ its reflexive transitive closure, and $\overset{*}{\leftrightarrow}$ its reflexive transitive symmetric closure. The last relation is also called *conversion*; when $t \overset{*}{\leftrightarrow} s$ we say that t is *convertible with* s.

A *normal form* is a term which contains no redexes (with respect to some set of rewrite rules). A normal form of a term t is a normal form n such that $t \rightarrow^* n$.

We will often consider pairs of reduction sequences having the same initial or final term. A *fork* is a pair of reduction sequences of the form $t_1 \leftarrow^* t \rightarrow^* t_2$. A *join* is a pair of reduction sequences of the form $t'_1 \rightarrow^* t' \leftarrow^* t'_2$. It is a join of the above fork if $t'_1 = t_1$ and $t'_2 = t_2$, and the four reduction sequences together are then called a *tile*.

If R_1 and R_2 are rewrite systems such that every function symbol which they have in common has the same arity in both systems, then $R_1 + R_2$ denotes the TRS obtained by taking the union of their sets of function symbols and rules. If their signatures are disjoint we write $R_1 \oplus R_2$.

2.2. APPLICATIVE TERM REWRITING

DEFINITION 2.1. *(1) For any signature Σ, an* applicative *term over Σ is a term over* *the signature obtained by making the arity of every member of Σ zero, and adding the binary symbol Ap (assumed not to occur in Σ). The set of applicative terms over Σ is denoted by $ATer(\Sigma)$.*

(2) The function $cur : Ter(\Sigma) \to ATer(\Sigma)$ is defined by induction on the structure of the terms in $Ter(\Sigma)$:
$cur(x) = x$
$cur(F(t_1, \ldots, t_n)) = Ap(\ldots Ap(F, t_1), \ldots, t_n)$

(3) An applicative *TRS (or ATRS) is a TRS whose terms are the applicative terms over some signature, and in which the left-hand side of every rule has the form $cur(t)$ for some (non-applicative) term t.*

Combinatory Logic is a standard example of an ATRS. It has the symbols S, K, and I, with applicative arities 3, 2, and 1 respectively, and the following rewrite rules:

$$\begin{array}{lcl} Ap(Ap(Ap(S,x),y),z) & \to & Ap(Ap(x,z),Ap(y,z)) \\ Ap(Ap(K,x),y) & \to & x \\ Ap(I,x) & \to & x. \end{array}$$

The presence of only one binary operator allows for the usual notational conventions:

(1) use infix notation $t \cdot s$ for $Ap(t,s)$;
(2) as in ordinary algebra, suppress the dot;
(3) associate to the left in order to use as few brackets as possible.

Following these notational conventions the above rewrite rules become:

$$CL = \left\{ \begin{array}{lcl} Sxyz & \to & xz(yz) \\ Kxy & \to & x \\ Ix & \to & x. \end{array} \right.$$

It is a convenient fiction to view the S, K, and I as "operators with variable arity".

2.3. CURRYING

Currying is a well-known construction that given a TRS R produces a corresponding *applicative* TRS Cur(R). It is usually credited to Schönfinkel (cf. Curry and Feys, 1958). Consider the TRS R given by the following rules:

$$\begin{array}{lcl} M(x,x) & \to & 0 \\ M(Succ(x),x) & \to & 1. \end{array}$$

Its currying Cur(R) is the ATRS given by the rules:

$$\begin{array}{lcl} Mxx & \to & 0 \\ M(Succ\, x)x & \to & 1. \end{array}$$

So the curried TRS is constructed with the same set of function symbols, together with application, but we have "forgotten" about their arity and treat the former function symbols as constants.

DEFINITION 2.2. *Let R be a TRS with signature Σ. The* currying Cur(R) *of R is the applicative TRS whose terms are the set $ATer(\Sigma)$ and whose rewrite rules are $\{cur(l) \to cur(r) \mid l \to r \in R\}$.*

The function cur :Ter(Σ) $\rightarrow$ATer(Σ) is not surjective: e.g., in the example above, the terms xx, M, Mx and $Mxyz$ are not in the image of cur: $R \rightarrow \mathrm{Cur}(R)$.

LEMMA 2.1. *(1) cur is one-one.*

(2) $\sigma(l)$ is a redex for R in t, if and only if $(cur \circ \sigma)(cur(l))$ is a redex for Cur(R) *in $cur(t)$.*

(3) $t \rightarrow s$ in R if and only if $cur(t) \rightarrow cur(s)$ in Cur(R).

(4) If $cur(t) \rightarrow s$ in Cur(R) *then there is an s' in R such that $t \rightarrow s'$ and $cur(s') = s$.*

(5) t is a normal form in R if and only if $cur(t)$ is a normal form in Cur(R).

The lemma can be summed up as saying that Cur(R) contains an isomorphic copy of R which is closed under reduction.

2.4. AN ALTERNATIVE DEFINITION OF CURRYING

While Cur(R) is the traditional notion of currying, we find it more convenient to use an alternative notion, in which R is extended to a system called PP(R) by adding an Ap symbol and certain other symbols and rules, but keeping all the symbols and rules of R. PP is Kahrs's notation, and stands for "partial parameterization". The two notions are sufficiently similar that each of the properties we consider is easily proved to hold of Cur(R) if and only if it holds of PP(R).

This version of currying has been known since at least Kennaway and Sleep (1982), and probably since much earlier. Kahrs (1994) uses it in his proof of preservation of CR.

DEFINITION 2.3. *For any TRS R, the TRS* PP(R) *consists of R plus the following function symbols and rules:*

(1) For every symbol F of arity n, new symbols F_i for $i : 0 \ldots n-1$, of arity i. These are called incomplete *function symbols.*

(2) A binary symbol Ap.

(3) For each new symbol F_i, a rule

$$Ap(F_i(x_1, \ldots, x_i), y) \rightarrow F_{i+1}(x_1, \ldots, x_i, y)$$

where if $i+1 = arity(F)$ then F_{i+1} denotes F. These rules will be called the uncurrying *rules.*

The uncurrying rules are orthogonal. Since each uncurrying rewrite reduces the size of the term, they are strongly normalizing. Therefore every term has a unique normal form with respect to them.

DEFINITION 2.4. *For any term t of* PP(R), Uncur(t) *is the unique normal form of t with respect to the uncurrying rules.*

As for Cur(R), we may indicate the Ap operator in PP(R) by juxtaposition when this is not syntactically ambiguous.

There is a very close connection between Cur(R) and PP(R) which the following definition and theorem make precise.

$$\begin{tikzcd}
t \arrow[r, "*"] \arrow[d, "*"'] & t_1 \arrow[d, dotted, "*"] \\
t_2 \arrow[r, dotted, "*"] & s
\end{tikzcd}$$

Figure 1. The Church-Rosser property.

DEFINITION 2.5. *For a term t of* $\mathrm{Cur}(R)$, *let* $\mathrm{PP}(t)$ *be the term obtained by first subscripting every non-nullary function symbol in t by 0, and then reducing the resulting term of* $\mathrm{PP}(R)$ *to normal form with respect to the uncurrying rules. For a term t of* $\mathrm{PP}(R)$, *define* $\mathrm{Cur}(t)$ *to be obtained by replacing every subterm of the form $F_i(x_1, \ldots, x_i)$ by $Ap(\ldots Ap(F, x_1), \ldots, x_i)$.*

THEOREM 2.1. *(1) When restricted to the normal forms of* $\mathrm{Cur}(R)$ *and* $\mathrm{PP}(R)$ *respectively,* PP *and* Cur *are inverse bijections.*

(2) If $t \to s$ in $\mathrm{Cur}(R)$, *then* $\mathrm{PP}(t) \to^{+} \mathrm{PP}(s)$ *in* $\mathrm{PP}(R)$, *by a sequence consisting of a single non-uncurrying reduction followed by a series of uncurrying reductions.*

(3) If $t \to s$ in $\mathrm{PP}(R)$ *by an uncurrying rule, then* $\mathrm{Cur}(t) = \mathrm{Cur}(s)$. *If $t \to s$ in* $\mathrm{PP}(R)$ *by a non-uncurrying rule, then* $\mathrm{Cur}(t) \to \mathrm{Cur}(s)$ *in* $\mathrm{Cur}(R)$.

(4) For t in $\mathrm{Cur}(R)$, $\mathrm{Cur}(\mathrm{PP}(t)) = t$. *For t in* $\mathrm{PP}(R)$, $\mathrm{PP}(\mathrm{Cur}(t)) = \mathrm{Uncur}(t)$.

(5) For every t in $\mathrm{PP}(R)$, *there are only finitely many t' in* $\mathrm{Cur}(R)$ *such that* $\mathrm{PP}(t') = t$.

An example illustrating item (2) is the rule $R = G(x, y) \to y$, and the curried reduction $t = GG(Gx)z \to Gxz = s$. In the system $\mathrm{PP}(R)$ we have $\mathrm{PP}(t) = Ap(G(G_0, G_1(x)), z) \to Ap(G_1(x), z) \to G(x, z) = \mathrm{PP}(s)$.

The usefulness of $\mathrm{PP}(R)$ lies in the fact that it contains not merely an abstractly isomorphic copy of R, as $\mathrm{Cur}(R)$ does, but R itself.

2.5. PROPERTIES OF TERM REWRITE SYSTEMS

DEFINITION 2.6. *A property P of term rewrite systems is* preserved *by currying if $P(R)$ implies $P(\mathrm{Cur}(R))$. It is* reflected *by currying if $P(\mathrm{Cur}(R))$ implies $P(R)$.*

We will be considering the following properties of term rewrite systems, and will demonstrate which of them are preserved or reflected by the currying transformation.

DEFINITION 2.7. *(1) A TRS is* strongly normalizing *(SN) if it contains no infinite reduction sequences.*

(2) A TRS is weakly normalizing *(WN) if every term has a normal form.*

(3) A TRS is confluent *or* Church-Rosser *(CR) if every fork $t_1 \leftarrow^{*} t \to^{*} t_2$ has a join $t_1 \to^{*} s \leftarrow^{*} t_2$.*

(4) A TRS is locally confluent *or* weakly Church-Rosser *(WCR) if every fork of the form $t_1 \leftarrow t \to t_2$ (that is, where the two reduction sequences each contain exactly one step) has a join $t_1 \to^{*} s \leftarrow^{*} t_2$.*

(5) A TRS is complete *if it is both SN and CR.*

(6) A TRS is semi-complete *if it is both WN and CR.*

$$\begin{tikzcd}
t \arrow[r] \arrow[d] & t_1 \arrow[d, dotted, "*"'] \\
t_2 \arrow[r, dotted, "*"] & s
\end{tikzcd}$$

Figure 2. The weak Church-Rosser property.

$$\begin{tikzcd}
t \arrow[rr, leftrightarrow, "*"] \arrow[dr, "*"] & & s \arrow[dl, dotted, "*"] \\
& n &
\end{tikzcd}$$

Figure 3. The NF property.

(7) *A term t has the* normal form property *(NF) if when t is reducible to normal form n, and t is convertible with s, then s is reducible to n. A TRS has the normal form property if each of its terms does.*

(8) *A TRS has* unique normal forms *(UN) if convertible normal forms are identical.*

(9) *A term t has* unique normal forms with respect to reduction *($UN^{\rightarrow}(t)$) if it cannot be reduced to two distinct normal forms. A TRS has the $UN^{\rightarrow}$ property if each term in it does.*

Completeness is equivalent to the combined properties SN and WCR by the well-known Newman's Lemma.

Lemma 2.2. *If a TRS is SN and WCR, then it is CR (see, e.g. Klop, 1992).*

Several implications hold among these. For example, CR $\Rightarrow$ NF $\Rightarrow$ UN $\Rightarrow$ $\mathrm{UN}^{\rightarrow}$ (Klop, 1992). However, these implications do not imply corresponding implications among the corresponding preservation properties, each of which must be proved separately. In fact, of the above properties, we will see that CR and UN are preserved by currying, but NF and $\mathrm{UN}^{\rightarrow}$ are not.

As we mentioned above, we find it technically more convenient to study $\mathrm{PP}(R)$ rather than $\mathrm{Cur}(R)$. The following theorem justifies this approach.

Theorem 2.2. *Let P be any of the properties SN, WN, CR, WCR, NF, UN, and $UN^{\rightarrow}$. Then $P(\mathrm{Cur}(R))$ if and only if $P(\mathrm{PP}(R))$.*

Proof. These follow immediately from Theorem 2.1. For example, by Theorem 2.1 item (2), the translation of $\mathrm{Cur}(R)$ to $\mathrm{PP}(R)$ maps infinite reduction sequences to infinite reduction sequences. By Theorem 2.1 item (3) and the strong normalization of the uncurrying rules, the reverse translation also has that property. Hence $\mathrm{Cur}(R)$ is SN if and only if $\mathrm{PP}(R)$ is.

The other properties may be dealt with equally simply. □

Henceforth we will always deal with $\mathrm{PP}(R)$ instead of $\mathrm{Cur}(R)$.

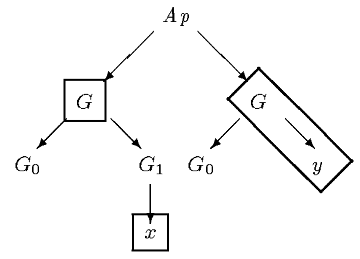

Figure 4. A term with three patches.

$$\begin{tikzcd}
Ap(G(G_0,G_1(x)),G(G_0,y)) \arrow[r] \arrow[d] & Ap(G_1(x),G(G_0,y)) \arrow[r] \arrow[d] & G(x,G(G_0,y)) \arrow[r] \arrow[d] & G(G_0,y) \arrow[d] \\
Ap(G(G_0,G_1(x)),y) \arrow[r] & Ap(G_1(x),y) \arrow[r] & G(x,y) \arrow[r] & y
\end{tikzcd}$$

Figure 5. All reductions of the term in Figure 4.

2.6. PATCHES

Of importance are the fragments of a term in a curried TRS in which all function symbols receive exactly the right number of arguments.

DEFINITION 2.8. *Let t be a term of a TRS* PP(R). *A* patch *of t is a component $s \trianglelefteq t$ such that s is a term of R and there is no s' in R such that $s \triangleleft s' \trianglelefteq t$. In other words, a patch is a component containing neither Ap nor any incomplete function symbol, and subject to that condition, maximal with respect to $\trianglelefteq$.*

Our proofs will often use the following classification of terms of PP(R). Every term of PP(R) is either $F_i(t_1,\ldots,t_i)$ for some incomplete function symbol F_i ($i \geq 0$), or it is $Ap(\ldots Ap(P[t_1,\ldots,t_n],s_1),\ldots s_m)$, where $n \geq 0$, $m \geq 0$, and $P[x_1,\ldots,x_n]$ is a patch of the term.

Patches are important because reductions performed within a patch are reductions of R. The reduction behaviour of a term of PP(R) does not, however, consist of the reduction behaviour of each patch. This is because a patch can "collapse", that is, reduce to one of its arguments. This can create a redex by one of the uncurrying rules, and thus lead to reduction sequences that have no direct counterpart in R. As an example consider the ATRS obtained by currying the TRS consisting of the single rule $G(x,y) \to y$. Consider the term $Ap(G(G_0,G_1(x)),G(G_0,y))$. This term has three patches: $G(p,q)$, x, and $G(p,y)$, whose roots are at positions 1, $1 \cdot 2 \cdot 1$, and 2, respectively. This is pictured in Figure 4. All possible derivations of the term are pictured in Figure 5. Note that the middle step in each of the horizontal sequences causes two patches to be merged into one. In the upper sequence, the two patches x and $G(p,y)$ merge into $G(x,G(p,y))$, and in the lower, x and y merge into $G(x,y)$. Finally, it is useful to observe that the pattern of any rewrite rule in any term t of a curried TRS PP(R) is entirely contained in a patch, since the patterns are terms of R.

2.7. MODULARITY PROPERTIES AND SIGNATURE EXTENSIONS

DEFINITION 2.9. *A property P of term rewrite systems is* modular *if, given two disjoint systems (i.e. systems having no function symbols in common), P is true of their union if and only if it is true of both systems.*

We will use several modularity results, which we collect here. Klop (1992) discusses all of these results except the last.

THEOREM 2.3. *(1) The disjoint sum of two strongly normalizing TRSs is strongly normalizing, if one of them contains neither collapsing rules nor duplicating rules. (Conjectured by Rusinowitch and proved in Middeldorp, 1990.)*
(2) WN is modular (noted by several authors, e.g. Middeldorp, 1990).
(3) CR is modular (Toyama, 1987).
(4) UN is modular (Middeldorp, 1990).
(5) NF is modular for left-linear systems (Middeldorp, 1990).

We do not actually need the full generality of the above result, but only the particular case of signature extensions. A *signature extension* of a TRS is a system obtained by adding to it a set of new function symbols, but no extra rules. Thus it is a special case of a disjoint union, so all the properties listed in the preceding theorem are preserved by signature extensions.

2.8. WELL-ORDERING LEMMAS

We will need the following constructions of well-founded partial orderings, i.e. partial orderings containing no infinite descending chains.

LEMMA 2.3. *(Dershowitz and Manna, 1979) Let $(X, \leq)$ be a well-founded partial order. Define a relation $\leq$ on the set of finite multisets of members of X, by stipulating that (i) $\leq$ is transitive and reflexive, and (ii) for multisets A and B, $A \geq B$ if B can be obtained from A by replacing any element a of A by any finite multiset of elements strictly less than a. Then $\leq$ is a well-founded partial ordering of the set of finite multisets of elements of X.*

LEMMA 2.4. *Let $(X, \leq)$ be a well-founded partial order. Consider the set of finite trees whose nodes are labelled by elements of X. For trees T_1 and T_2, define $T_1 \geq T_2$ if T_2 can be obtained from T_1 by a finite sequence of any of the following operations:*

(1) Delete any proper subtree.
(2) Replace the label of any node by a strictly lesser label, and replace each immediate subtree of that node by any finite number of copies.

Then $\geq$ is a well-founded partial ordering of the set of trees.

PROOF. This follows from theorem 2.3.5 of Klop (1992), since both of the above transformations of trees can be expressed as sequences of the transformations which that theorem deals with. That theorem in turn is proved from Kruskal's Tree Theorem (Kruskal, 1960).

We shall give a more direct proof which uses only the preceding lemma on well-orderings of multisets.

For any tree T, let $P(T)$ mean that it is not possible to apply an infinite sequence of the above operations to T. We prove $P(T)$ for all T by induction. The inductive hypothesis is that P is true for each immediate subtree of T.

Let T have label x at the root, and immediate subtrees $T_1, \ldots, T_n$. (There is no special base case of the induction: the case where $n = 0$ does not require a separate argument.) There are only finitely many immediate subtrees, and by induction, P holds of each of them. An infinite sequence of operations applied to T must therefore eventually apply the second transformation at the root, replacing the root label by a smaller one, and each immediate subtree by a finite number of copies. By the well-foundedness of $(X, \leq)$, this can happen only finitely many times before a tree is obtained whose root label is a minimal member of X. All subsequent operations must be applied to proper subtrees of that tree. But each immediate subtree of that tree is a tree which was obtained from one of $T_1, \ldots, T_n$ by a sequence of operations, and by induction satisfies P. Therefore the sequence of operations applied to T must terminate, i.e. $P(T)$. □

3. Properties reflected by currying arbitrary term rewriting systems

The question, which properties are reflected by currying, can be disposed of immediately.

THEOREM 3.1. *All of the properties SN, WN, CR, WCR, completeness, semi-completeness, NF, UN, and UN$^{\rightarrow}$ are reflected by currying arbitrary term rewrite systems.*

PROOF. This follows immediately from the fact that PP(R) contains an isomorphic reduction-closed copy of R. For example, if every reduction sequence in PP(R) is finite, then every reduction sequence in that copy of R is finite, hence also every reduction sequence of R. Thus SN is reflected by currying.

Proofs for the other properties are equally simple. □

We sketch how the proofs of reflection can be cast into a more systematic form. Each of the properties we consider can be expressed in a first-order language containing the predicates $x \rightarrow y$, $x \rightarrow^* y$, $x \leftrightarrow^* y$, $\mathrm{Seq}(x, s, y)$ (s is a reduction sequence from x to y), and $\mathrm{Inf}(x, s)$ (s is an infinite reduction sequence starting from x). x and y range over terms, s ranges over reduction sequences. For example, the UN property is $\forall xy.(x \leftrightarrow^* y \wedge \mathrm{nf}(x) \wedge \mathrm{nf}(y)) \Rightarrow x = y$, where $\mathrm{nf}(x)$ abbreviates $\neg\exists z.x \rightarrow z$. Reflection by currying means that the statement in which the variables range over terms and sequences of PP(R) implies the statement where they are restricted to R. That R is a reduction-closed subset of PP(R) implies that the above predicates have the following properties.

$$x \rightarrow y \wedge x \in R \;\Rightarrow\; x \rightarrow_R y \wedge y \in R$$

$$x \rightarrow^* y \wedge x \in R \;\Rightarrow\; x \rightarrow^*_R y \wedge y \in R$$

$$x \leftrightarrow^* y \wedge x \in R \;\Rightarrow\; x \leftrightarrow^*_R y \wedge y \in R$$

$$\mathrm{Seq}(x, s, y) \wedge x \in R \;\Rightarrow\; \mathrm{Seq}_R(x, s, y)$$

$$\mathrm{Inf}(x, s) \wedge x \in R \;\Rightarrow\; \mathrm{Inf}_R(x, s).$$

$\rightarrow_R$ is the reduction relation of R, and similarly for Seq_R and Inf_R. The notation $x \in R$ means that x is a term of R. From these properties, it is easy to prove that a large class of statements formed from these predicates, including those of the theorem, are reflected by currying. But we shall not attempt a formal description of such a class.

4. Properties preserved by currying arbitrary term rewriting systems

In this section we will show that currying preserves the following properties of term rewrite systems: weak normalization, strong normalization, the weak Church-Rosser property, and completeness.

4.1. PRESERVATION OF WN BY CURRYING

THEOREM 4.1. *If R is WN, then* PP(R) *is WN.*

PROOF. Let the TRS R be WN. By induction on the structure of terms we will prove that PP(R) is WN.

If t is a variable, it is already in normal form.

If $t = F(t_1, \ldots, t_n)$, where F is any function symbol of PP(R), then by induction, the terms $t_1, \ldots, t_n$ have normal forms $t'_1, \ldots, t'_n$. Either $F(t'_1, \ldots, t'_n)$ is a normal form of t, or it contains a single redex, at its root. If that redex is an uncurrying redex, reducing it gives another term containing no redex other than at its root (and that redex, if present, cannot now be an uncurrying redex). If the redex is by a rule of R, then the term has the form $P[s_1, \ldots, s_m]$, where $P[x_1, \ldots, x_m]$ is a patch (which we shall choose to be strict) and each of $s_1, \ldots, s_m$ are normal forms. $P[x_1, \ldots, x_m]$ is a term of R, so has a normal form P'.

Now consider the term $P'[x'_1 := s_1, \ldots, x'_m := s_m]$. A redex of this term must be either a redex of some s_i, a redex of P', a redex whose pattern is partly in P' and partly in some s_i, or a non-left-linear redex whose pattern lies in P' which tests the equality of some subterms of $P'[x'_1 := s_1, \ldots, x'_m := s_m]$ including at least one s_i, and which is not a redex of P'.

The first two cases are impossible, since P' and all the s_i are normal forms.

Every symbol in P' is a symbol of R, but the root symbol of each s_i is not in R. Since no left-hand side of PP(R) contains both symbols of R and symbols not in R, the third case is impossible.

The strictness of $P[x_1, \ldots, x_m]$, and the fact that no s_i can be unifiable with any subterm of $P[x_1, \ldots, x_m]$ other than variables rules out the fourth case.

Therefore $P'[x'_1 := s_1, \ldots, x'_m := s_m]$ is a normal form of t. □

4.2. PRESERVATION OF SN BY CURRYING

This is the most complicated of our proofs. We begin by outlining the method. First, we extend R to R^+, which consists of R and all the incomplete symbols of PP(R), but without Ap or any of the uncurrying rules. This is a signature extension of R, therefore it is SN. It is a subsystem of PP(R).

Our first idea to prove that PP(R) is SN is to take any term t of PP(R), perform in advance all the possible uncurrying reductions that it might undergo in PP(R), and thus eliminate all occurrences of Ap. Call the resulting term $\alpha(t)$. $\alpha(t)$ will contain no occurrences of Ap, and hence is a term of R^+, and its reduction behaviour must be identical in both R^+ and PP(R). If we can show that $t \rightarrow s$ in PP(R) implies $\alpha(t) \rightarrow^+ \alpha(s)$, then SN($R^+$) implies SN(PP($R$)).

This does not quite work. The best we can do is to construct a transformation α such that if $t \rightarrow s$ in PP(R) then $\alpha(t) \rightarrow^* \alpha(s)$. The possibility that the latter reduction is empty prevents the desired conclusion that SN(R^+) implies SN(PP(R)). The possible emptiness of the reduction results from the fact that our α transformation may discard some parts of the term. However, we can use α to construct a more complicated transformation β of terms in PP(R) which never discards subterms. β will map a term t to a tree, each node of which is labelled with a term of the form $\alpha(t')$ for some subterm t' of t. The root will be labelled with $\alpha(t)$, and the other nodes will be labelled with $\alpha(t')$ for subterms t' which α might erase from t. This construction ensures that no part of t is discarded. We define a reduction relation on the set R^{TLT} of these term-labelled trees, such that $t \rightarrow s$ in PP(R) implies $\beta(t) \rightarrow^+ \beta(s)$. A general theorem about well-orderings of labelled trees implies that if R is SN, then so is R^{TLT}. From this it follows that PP(R) is SN.

We now carry out this programme.

LEMMA 4.1. *If $t \rightarrow s$ in* PP(R) *then* Uncur(t) $\rightarrow^=$ Uncur(s).

PROOF. If $t \rightarrow s$ by an uncurrying rule, then completeness of the uncurrying rules implies that Uncur(t) = Uncur(s).

If $t \rightarrow s$ by some other rule, then since there are no critical pairs involving uncurrying rules, and Uncur treats identical subterms identically, Uncur(t) $\rightarrow$ Uncur(s) by the same rule that reduces t to s. $\square$

Let R^+ be the TRS consisting of R together with all the incomplete function symbols of PP(R), but without the Ap symbol or the uncurrying rules. R^+ is thus a subsystem of PP(R), and furthermore is closed under the reduction relation of PP(R), since none of its terms contains any redexes by the uncurrying rules. We shall define a transformation α of terms of PP(R) to R^+, by means of a reduction relation $\rightarrow_\alpha$.

DEFINITION 4.1. *A* candidate for collapsing *is a term t of the form $Ap(\ldots Ap(P[t_1,\ldots, t_n], s_1),\ldots s_k)$, where P is a patch, $n \geq 0$, and $m > 0$. For such a term, $\Delta(t)$ denotes the term $P[Ap(\ldots Ap(t_1,s_1)\ldots s_m),\ldots,Ap(\ldots Ap(t_n,s_1)\ldots s_m)]$, and $\overline{\Delta}(t)$ denotes the set $\{s_1,\ldots,s_m\}$.*

DEFINITION 4.2. *Let t be a term of* PP(R). *Let t' be any outermost candidate for collapsing of* Uncur(t). *Replace t' by $\Delta(t')$ in* Uncur(t), *to obtain a term t''. If* Uncur(t'') *is different from t, then $t \rightarrow_\alpha$* Uncur(t'').

$\alpha(t)$ will be defined below to be the unique normal form of t with respect to $\rightarrow_\alpha$ (which we will prove exists). The idea behind this transformation is that the patch $P[x_1,\ldots,x_n]$ might reduce to any of its arguments x_i, and thus cause its ith argument to be applied to the arguments $s_1,\ldots,s_m$, possibly creating uncurrying redexes. We take a pessimistic view and apply every one of the arguments of $P[\ldots]$ to $s_1,\ldots,s_m$. The resulting term may be able to reduce in ways which the original term could not (because $P[\ldots]$ may in fact be incapable of collapsing to some of its arguments). But this does not matter: we only want that reductions of the original term t can be mapped to reductions of $\alpha(t)$.

LEMMA 4.2. *$\rightarrow_\alpha$ is confluent and strongly normalizing.*

PROOF. If Uncur(t) contains two distinct candidates for $\rightarrow_\alpha$, and if either subterm is replaced as described, the other subterm will still be a candidate for a subsequent $\rightarrow_\alpha$ reduction. Thus if $t \rightarrow_\alpha u_0$ and $t \rightarrow_\alpha u_1$, then for some s, $u_0 \rightarrow_\alpha s$ and $u_1 \rightarrow_\alpha s$. Confluence follows.

Strong normalization is proved by induction, based on the way that the Ap symbol occurs in t and s when $t \rightarrow_\alpha s$. For any term t, for each occurrence of Ap in t, consider the size of its first argument. Define $a(t)$ to be the multiset of all these sizes.

Each application of an uncurrying rule removes one occurrence of Ap, and hence one element from $a(t)$, and every other element of $a(t)$ is either unchanged or reduced by 1.

When t is in normal form with respect to the uncurrying rules, replacing a candidate $t' = Ap(\ldots Ap(P[t_1,\ldots,t_n],s_1),\ldots s_m)$ by $\Delta(t')$ transforms $a(t)$ thus: The element k corresponding to the occurrence of Ap at the root of this subterm is removed (as well as another $m-1$ occurrences of Ap). $n \times m$ new occurrences of Ap are created to attach each t_i to each s_j. Each subterm s_i is replaced by n copies. All the new and copied occurrences of Ap are at subterms strictly smaller than k. Hence the effect is to remove k and some other elements, and to add a finite number of elements smaller than k. This transformation strictly reduces the size of the multiset with respect to the partial ordering of Lemma 2.3. It follows that $\rightarrow_\alpha$ is strongly normalizing. $\Box$

DEFINITION 4.3. *$\alpha(t)$ is the unique normal form of t with respect to $\rightarrow_\alpha$.*

LEMMA 4.3. *α maps terms of* PP(R) *to terms of R^+.*

PROOF. Any term which contains Ap must contain a subterm of the form $Ap(F(\ldots),t)$, where F is either an incomplete symbol or a symbol of R. In the former case there is an uncurrying redex; in the latter case there is a candidate for $\rightarrow_\alpha$, so in both cases, the term can be transformed to a different term by $\rightarrow_\alpha$.

A normal form with respect to $\rightarrow_\alpha$ therefore cannot contain Ap. $\Box$

LEMMA 4.4. *If $t \rightarrow s$ in* PP(R)*, then $\alpha(t) \rightarrow^* \alpha(s)$.*

PROOF. Suppose $t \rightarrow s$ by reduction of an uncurrying redex. Then Uncur(t) = Uncur(s), and $\alpha(t) = \alpha(s)$.

Suppose $t \rightarrow s$ by reduction of a redex by a rule of R, at a position p. Because uncurrying rules do not conflict with rules of R, and do not duplicate or erase subterms, Uncur$(t) \rightarrow$ Uncur(s). So we may assume without loss of generality that t is already in normal form with respect to the uncurrying rules, i.e. t = Uncur(t).

If t contains no candidates for collapsing, then t is already a term of R, and therefore so is s. Therefore $\alpha(t) = t \rightarrow^* s = \alpha(s)$.

Otherwise, it is sufficient to prove that if $t \rightarrow s$, then there are u and v such that $t \rightarrow^*_\alpha u$, $s \rightarrow^*_\alpha v$, and $u \rightarrow^* v$. The lemma then follows from completeness of $\rightarrow_\alpha$.

Either the redex at p contains one or more candidates for $\rightarrow_\alpha$, or it is contained in a candidate, or neither of these holds.

Suppose it contains one or more candidates. Let u be the result of applying $\rightarrow_\alpha$ to all of them. The corresponding subterms of s will also be candidates for $\rightarrow_\alpha$. Let v be the result of applying $\rightarrow_\alpha$ to all of those subterms of s. Then $u \rightarrow v$ by reduction at p. (Note that it is essential to reduce all the candidates for $\rightarrow_\alpha$ contained in the redex, since otherwise, if the redex is by a non-left-linear rule, there might not be a redex at the same position in u.)

Suppose p is contained in a candidate for $\rightarrow_\alpha$, of the form $Ap(\dots Ap(P[t_1, \dots, t_n], s_1), \dots s_m)$, at position q. Then q is the position of a candidate for $\rightarrow_\alpha$ in s. If p is contained in $t_1, \dots, t_n$ or $s_1 \dots s_m$, or if p is in the patch P and reduction at p does not reduce P to one of its arguments, then it is clear that when $\rightarrow_\alpha$ is applied to the candidate at q in both t and s, resulting in terms u and v, u reduced to v by reducing all the resulting copies of the redex. If reduction at p reduces $P[x_1, \dots, x_n]$ to x_i, then $p = q$ and reduction of the redex at p gives the same result as reduction of the candidate at p by $\rightarrow_\alpha$. We can therefore take $u = v = s$.

Finally, if p neither contains nor is contained in any candidate, take u to be the result of reduction by $\rightarrow_\alpha$ at any candidate q. The reductions at p and q clearly do not interfere with each other; v can be taken to be the common result of reduction by R at p in u and reduction by $\rightarrow_\alpha$ at q in s. □

If in the preceding lemma, we were able to show that $\alpha(t) \rightarrow^+ \alpha(s)$, at least when $t \rightarrow s$ by a rule of R, then we could immediately conclude preservation of SN. Any infinite reduction starting from a term t of PP(R) must contain infinitely many steps of R, which would give rise to an infinite reduction starting from $\alpha(t)$. But this would be a reduction in R^+, which must be SN if R is.

However, in general $\alpha(t)$ and $\alpha(s)$ may be identical. This is because if $n = 0$, then the replacement of $t' = Ap(\dots Ap(P[t_1, \dots, t_n], s_1), \dots s_m)$, by $\Delta(t')$ has the effect of discarding all the subterms $\overline{\Delta}(t') = \{s_1, \dots, s_m\}$. To obtain preservation of SN by currying, we need a transformation which will eliminate Ap symbols without discarding any subterms.

Definition 4.4. *R^{TLT} is an abstract rewrite system whose objects are finite trees with nodes labelled by terms of R^+. For such term-labelled trees t and s, we define $t \rightarrow s$ in either of the following cases:*

(1) s is obtained from t by discarding any proper subtree.
(2) s is obtained from t by performing a reduction step of R on a term labelling some node n of t, and then replacing each of the immediate subtrees of n by any finite number of copies of themselves.

Lemma 4.5. *If R is SN, then R^+ is SN.*

Proof. R^+ is a signature extension of R; the result follows from Theorem 2.3 item (1). □

Lemma 4.6. *If R^+ is SN, then R^{TLT} is SN.*

Proof. On R^{TLT}, define $t \geq s$ if $t \rightarrow^* s$. That R^{TLT} is SN is equivalent to $\geq$ being a

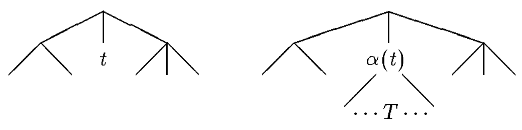

Figure 6. One step in the β transformation.

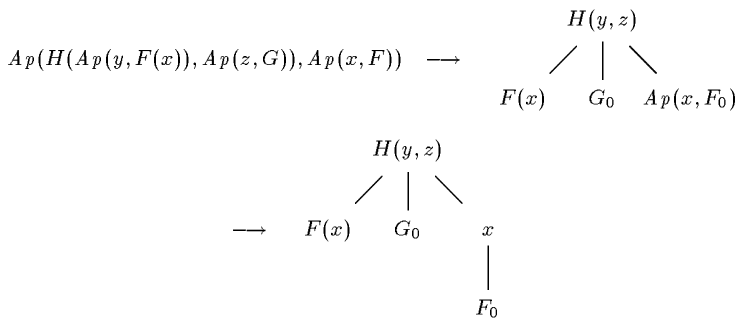

Figure 7. The β transformation.

well-founded partial ordering of the members of R^{TLT}. That it is well-founded follows from Lemma 2.4. □

We now define a transformation β from PP(R) to R^{TLT}.

Definition 4.5. *Let t be a term of* PP(R). *Construct a member of R^{TLT} by the following iterative method.*

Begin with a tree containing a single node, labelled by t.

If none of the labels of the tree contains the symbol Ap, then the tree is a member of R^{TLT}. Stop.

Otherwise, choose any leaf n of the tree whose label t' contains an occurrence of Ap. Reduce t' to $\alpha(t')$ by applying $\longrightarrow_\alpha$ enough times. Each $\longrightarrow_\alpha$-step will replace a candidate $t'' = Ap(\ldots Ap(P[t_1,\ldots,t_n],s_1),\ldots s_m)$ by $\Delta(t'')$. Form the union T of the sets $\overline{\Delta}(t'')$ for each $\longrightarrow_\alpha$-step.

Replace the label of n by $\alpha(t)$, and for each member of T, add a new descendant node of n, labelled with that element of T. See Figure 6.

Since each member of T is smaller than t, by induction on the size of terms this process terminates.

The nodes of the resulting tree will all be labelled by terms of R^+. Denote this tree by $\beta(t)$.

Example 4.1. *Let R have the function symbols F, G, and H, with arities of 1, 2, and 2, respectively. Consider the term $t = Ap(H(Ap(y,F(x)),Ap(z,G_0)),Ap(x,F_0))$. Figure 7 indicates the construction of $\beta(t)$.*

LEMMA 4.7. *If $t \rightarrow s$ in* PP(R), *then $\beta(t) \rightarrow^+ \beta(s)$.*

PROOF. If the reduction of t to s is by an uncurrying rule, then $\beta(t) = \beta(s)$.

Otherwise, let $t \rightarrow s$ by reduction of a redex at u. If u is not contained in any candidate for $\rightarrow_\alpha$ in t, then it is clear that u is the position of a redex in the label of the root of $\beta(t)$, and that reduction at u in that label, together with the replacement of some of the immediate subtrees by some number of copies of themselves yields $\beta(s)$.

If u is contained in a candidate $Ap(\ldots Ap(P[t_1,\ldots,t_n],s_1),\ldots s_m)$ for $\rightarrow_\alpha$ in t, then by arguments similar to those in the proof of Lemma 4.4, we can find a reduction of $\beta(t)$ to $\beta(s)$. However, in the case where u is contained in s_i and $n = 0$, the reduction sequence must be nonempty, since $\beta(t)$ contains $\beta(s_i)$ as a subtree even although $\alpha(t)$ may not contain $\alpha(s_i)$ as a subterm. □

THEOREM 4.2. *If R is SN, then* PP(R) *is SN.*

PROOF. Immediate from Lemmas 4.5, 4.6 and 4.7. □

Zantema (pers. comm.) has given an alternative proof, based on the technique of semantic labelling (Zantema, 1994).

4.3. PRESERVATION OF WCR BY CURRYING

THEOREM 4.3. *If R is WCR, then* PP(R) *is WCR.*

PROOF. (Suggested by Kahrs, pers. comm.) Since Ap and the incomplete function symbols do not occur in rules of R, and there are no conflicts between uncurrying rules, every critical pair of PP(R) must be a critical pair of R. Therefore if R is WCR, so is PP(R). □

4.4. PRESERVATION OF UN BY CURRYING

THEOREM 4.4. *If R is UN, then* PP(R) *is UN.*

PROOF. Lemma 5.1.19 of Middeldorp (1990) shows that if R satisfies UN, then R can be extended by adding extra rules and symbols, to a system R^{CR} such that:

(1) R^{CR} is CR.
(2) $t \overset{*}{\leftrightarrow} s$ in $R \Rightarrow t \overset{*}{\leftrightarrow} s$ in R^{CR}.
(3) If t is a normal form of R, then it is a normal form of R^{CR}.

(His lemma 5.1.19 is actually stronger, but the above is all that we require.) Now let R be a TRS satisfying UN. Kahrs (1994) has proved preservation of CR, and R^{CR} is CR, therefore PP(R^{CR}) is CR. We will prove that PP(R) and PP(R^{CR}) stand in the same relation as do R and R^{CR}.

Ad (2): R^{CR} is R plus extra rules and symbols, therefore PP(R^{CR}) is PP(R) plus extra rules and symbols. Therefore convertibility in PP(R) implies convertibility in PP(R^{CR}).

Ad (3): Let t be a term of PP(R) which is not a normal form of PP(R^{CR}). If t contains an uncurrying redex, that redex is a redex of PP(R), since the two systems have the same set of uncurrying rules. If t contains a redex by a rule of R^{CR}, then that redex is in some patch $P[x_1,\ldots,x_n]$ of t. $P[x_1,\ldots,x_n]$ is a non-normal form of R^{CR}, therefore it is a non-normal form of R. Hence t is a non-normal form of $\mathrm{PP}(R)$.

Now suppose t and s are normal forms of $\mathrm{PP}(R)$, and $t \overset{*}{\leftrightarrow} s$ in $\mathrm{PP}(R)$. By (2), $t \overset{*}{\leftrightarrow} s$ in $\mathrm{PP}(R^{\mathrm{CR}})$, and by (3), t and s are normal forms of $\mathrm{PP}(R^{\mathrm{CR}})$. Since $\mathrm{PP}(R^{\mathrm{CR}})$ is CR, t and s must be identical. □

4.5. PRESERVATION OF (SEMI-)COMPLETENESS BY CURRYING

THEOREM 4.5. *If R is (semi-)complete, then so is* $\mathrm{PP}(R)$.

PROOF. A TRS is complete if and only if it is SN and CR, and semi-complete if and only if it is WN and CR. Theorems 4.1 and 4.2 establish preservation of WN and SN, and Kahrs (1994) has proved preservation of CR. □

5. Counterexamples

5.1. NF IS NOT PRESERVED BY CURRYING

Let R_{NF} be the system having the symbols A, B, C, F, and G, and the following rules:

$$\begin{array}{rcll}
F(x,x) & \rightarrow & G & \\
A & \rightarrow & B & \\
A & \rightarrow & C & \\
B & \rightarrow & B & \\
C & \rightarrow & C & \\
F(Z,x) & \rightarrow & G, & \textit{where } Z \textit{ is any of } A, B, \textit{ or } C \\
F(x,Z) & \rightarrow & G, & \textit{where } Z \textit{ is any of } A, B, \textit{ or } C.
\end{array}$$

REMARK 5.1. *The first rule and the next four rules are the two systems Middeldorp used to refute modularity of NF (Middeldorp, 1990). The last six rules restore NF, but allow it to be broken when a new symbol is added.*

LEMMA 5.1. R_{NF} *is NF.*

PROOF. Each of the terms A, B, and C has the NF property.

Call a reduction sequence an *ABC-elimination* if it uses only the last six rules, and results in a term containing no occurrence of A, B, or C. Every term other than A, B, or C can be so reduced: the effect is to replace every occurrence of F, either of whose arguments is A, B, or C, by G. We indicate such a reduction sequence by $\rightarrow_{ABC}$. It is clear that ABC-elimination commutes with every reduction step.

Let t be any term other than A, B, or C. Let $t \rightarrow^* s$ be a reduction of a term t to a normal form s, and let $t \rightarrow^* r$ be any reduction sequence. Then by the preceding remarks, we have reduction sequences $t \rightarrow_{ABC} t' \rightarrow^* s'$, $s \rightarrow_{ABC} s'$, $t' \rightarrow^* r'$, and $r \rightarrow_{ABC} r'$. Since s is a normal form, $s = s'$. Since t' contains no A, B, or C, only the first rule of the system can be used in the reductions of t' to s and r'. But this rule on its own is

Church-Rosser. Therefore r' and s have a common reduct, which must be s. Therefore r reduces to s. $\Box$

If we add a new unary function symbol H, the resulting system is not NF. A counterexample is the term $F(H(A), H(A))$, which reduces to both the normal form G and the term $F(H(B), H(C))$, which reduces only to itself.

From this we obtain a counterexample to the preservation of NF by currying. $Ap(Ap(F, Ap(A, G)), Ap(A, G))$ has normal form G. It also reduces to $Ap(Ap(F, Ap(B, G)), Ap(C, G))$, which reduces only to itself.

5.2. UN$^{\rightarrow}$ IS NOT PRESERVED BY CURRYING

Let $R_{\mathrm{UN}^{\rightarrow}}$ be the system having the symbols A, B, C, D, E, F, and G, and the following rules:

$$\begin{array}{rcll}
F(x,x) & \rightarrow & G & \\
A & \rightarrow & B & \\
A & \rightarrow & C & \\
D & \rightarrow & C & \\
D & \rightarrow & E & \\
C & \rightarrow & C & \\
F(Z,x) & \rightarrow & G, & \textit{where } Z \textit{ is any of } A, B, C, D, \textit{ or } E \\
F(x,Z) & \rightarrow & G, & \textit{where } Z \textit{ is any of } A, B, C, D, \textit{ or } E.
\end{array}$$

REMARK 5.2. *This system bears the same relation to Middeldorp's counterexample for UN$^{\rightarrow}$ as the example of the previous section does to his counterexample for NF.*

LEMMA 5.2. *$R_{\mathrm{UN}^{\rightarrow}}$ is UN$^{\rightarrow}$.*

PROOF. None of the terms A, B, C, D, or E is reducible to two distinct normal forms.

Define, as for the NF counterexample, an *ABCDE-elimination* (notation $\rightarrow_{ABCDE}$) to be a reduction sequence using only the last ten rules, ending with a term not containing any of the symbols A to E. Every term other than A, B, C, D, or E can be so reduced. $\rightarrow_{ABCDE}$ clearly commutes with the other rules.

Let t be any term other than A, B, C, D, or E. Suppose we have reductions of t to normal forms r and s. Applying $ABCDE$-elimination to every term in both sequences gives a reduction of a term not containing any of A, B, C, D, or E to r and s. Such reduction sequences can only use the first rule of the system. That rule on its own is confluent, hence r and s must be identical. $\Box$

If we add a new unary function symbol H, the resulting system is not UN$^{\rightarrow}$. A counterexample is the term $F(H(A), H(D))$, which reduces to both the normal forms G and $F(H(B), H(E))$.

From this we obtain a counterexample to the preservation of UN$^{\rightarrow}$ by currying. $Ap(Ap(F, Ap(A, G)), Ap(D, G))$ has normal forms G and $Ap(Ap(F, Ap(B, G)), Ap(E, G))$.

6. Properties preserved by currying left-linear term rewriting systems

In this section we will prove that the currying of a left-linear TRS preserves the NF and $\mathrm{UN}^{\rightarrow}$ properties.

6.1. PRESERVATION OF NF BY CURRYING FOR LEFT-LINEAR TRSs

DEFINITION 6.1. *A patch $P[x_1, \ldots, x_n]$ of a term t of a TRS R is* collapsible *if $P[x_1, \ldots, x_n]$ can be reduced in R to some x_i.*

We write $t \rightarrow_C s$ if $t \rightarrow^* s$ by a reduction sequence which takes place in some collapsible patch of t, and reduces it to one of its arguments.

LEMMA 6.1. *$\rightarrow_C$ is strongly normalizing.*

PROOF. Every application of $\rightarrow_C$ reduces the size of the term. □

LEMMA 6.2. *If a left-linear TRS R is NF, then every term of* $\mathrm{PP}(R)$ *containing no collapsible patches is NF.*

PROOF. By induction on the structure of terms.

If t has the form $F_n(t_1, \ldots, t_n)$, where F_n is an incomplete function symbol, then a reduction of t is the same thing as an interleaving of reductions of the subterms $t_1, \ldots, t_n$. Since each of those subterms is smaller than t, they each satisfy NF. Therefore so does t.

Otherwise, t has the form $Ap(\ldots Ap(P[t_1, \ldots, t_n], s_1), \ldots s_k)$ where $P[x_1, \ldots, x_n]$ is a patch of t, $n \geq 0$, and $k \geq 0$. If $n = 0$ and $k = 0$, then t is a single patch, hence is a term of R, hence is NF.

Otherwise, since P is not collapsible, a normal form of t must have the form $s = Ap(\ldots Ap(t', s'_1) \ldots s'_k)$, where each s'_i is a normal form of s_i, and t' is obtained from $P[x_1, \ldots, x_n]$ by first reducing it to normal form, then substituting for each occurrence of each x_i some normal form of t_i. Since $P[x_1, \ldots, x_n]$ and each t_i are smaller than t, by induction they each have at most one normal form, so t' in fact has the form $P'[x_1 := t'_1, \ldots, x_n := t'_n]$, where P' is a normal form of $P[x_1, \ldots, x_n]$, and for each i such that x_i occurs in P', t'_i is a normal form of t_i.

All of $P, t_1, \ldots, t_n, s_1 \ldots s_k$ are smaller than t, hence by induction on the size of terms, they also satisfy NF. Hence if a redex is reduced in t to give a term $r = Ap(\ldots Ap(P''[x_1 := t''_1, \ldots, x_n := t''_n], s''_1) \ldots s''_k)$ we can reduce P'' to P', each s''_i to s'_i, and for each i such that x_i occurs in P' each t''_i to t'_i. This reduces r to s. Therefore t satisfies NF. □

LEMMA 6.3. *If R is left-linear, then* $\mathrm{PP}(R)$ *satisfies the following:*

(1) If $s \leftarrow t \rightarrow_C r$, then there is a u such that $s \rightarrow^*_C u \leftarrow^* r$.
(2) If $s \leftarrow^* t \rightarrow_C r$, then there is a u such that $s \rightarrow^*_C u \leftarrow^* r$.

PROOF. (1) Let $s \leftarrow t \rightarrow_C r$. Suppose the redex q which reduces t to s is in a patch of t other than the patch P which collapses in reducing t to r. Then after collapsing P, either the part of the term containing q has been discarded, or the redex q is still present. (Left-linearity is essential here.) Similarly, if we first reduce q, then

$$\begin{tikzcd}
& r \arrow[rr, "*", "C"'] \arrow[dr, phantom, "6.3"] & & r \\
t \arrow[ur, "*"] \arrow[rr, "C"'] \arrow[dr] & & s \arrow[ur, "*"] \arrow[dr, "*"] \arrow[r, phantom, "\text{Ind.}"] & \\
& q \arrow[rr, "*", "C"'] \arrow[ur, phantom, "6.3"] & & v \arrow[uu, "*"']
\end{tikzcd}$$

Figure 8. Proof of preservation of NF.

there will be some number of copies (perhaps zero) of the patch P, all of which can still be collapsed in the same way. It is clear that if we collapse P, then reduce q, we obtain the same result as if we first reduce q, then collapse the copies of P.
Suppose the reduction of t to s is in the patch P which collapses in reducing t to r. The reduction of t to r reduces a subterm $P[t_1, \ldots, t_n]$ to its i'th argument, by performing reductions only within P. By left-linearity, $P[x_1, \ldots, x_n]$ must be reducible to x_i, which is a normal form. Since $P[x_1, \ldots, x_n]$ is a term of R, it satisfies NF. By left-linearity again, the redex which reduces t to s is present in $P[x_1, \ldots, x_n]$. Suppose reducing it gives a term P'. Then by the NF property of R, P' reduces to x_i. Thus $P'[x_1 := t_1, \ldots, x_n := t_n]$ reduces to t_i, and s reduces to r.

(2) The second part of the lemma follows from the first and Lemma 6.1.
□

THEOREM 6.1. *If R is left-linear and NF, then* PP(R) *is NF.*

PROOF. We proceed by induction on the strongly normalizing reduction relation $\rightarrow_C$. See Figure 8.

If t contains no collapsible patches, then this is Lemma 6.2.

Suppose t contains a collapsible patch. Let $t \rightarrow_C s$, and let $t \rightarrow^* r$ be a reduction of t to normal form. Let $t \rightarrow q$ be any single step. We must show that q is reducible to r. By Lemma 6.3, there are u and v such that $r \rightarrow_C^* u \leftarrow^* s$ and $q \rightarrow_C^* v \leftarrow^* s$. But r is a normal form, therefore $u = r$ and $s \rightarrow_C^* r$. By induction on $\rightarrow_C$, s satisfies NF. Therefore $v \rightarrow^* r$. Composing reduction sequences, $s \rightarrow^* r$. □

6.2. PRESERVATION OF UN$^\rightarrow$ BY CURRYING FOR LEFT-LINEAR TRSs

LEMMA 6.4. *If R satisfies UN$^\rightarrow$ and t is a term of* PP(R) *containing no collapsible patches, then t satisfies UN$^\rightarrow$.*

PROOF. The proof is similar to that of Lemma 6.2, so we only outline it.

If t has the form $F_n(t_1, \ldots, t_n)$, where F_n is an incomplete function symbol, then the lemma follows by exactly the same argument as in Lemma 6.2.

Let $t = Ap(\ldots Ap(P[t_1, \ldots, t_n], s_1), \ldots s_k)$, where $P[x_1, \ldots, x_n]$ is a patch of t. Then a reduction of t to normal form reduces $P[x_1, \ldots, x_n]$ and each of $t_1, \ldots, t_n, s_1 \ldots s_k$ to normal form. $P[x_1, \ldots, x_n]$ is a term of R, hence satisfies UN$^\rightarrow$. Each of $t_1, \ldots, t_n, s_1 \ldots s_k$ is smaller than t, hence satisfies UN$^\rightarrow$, Therefore t satisfies UN$^\rightarrow$. □

$$\begin{tikzcd}
 & r \arrow[rr, "*", "C"'] \arrow[dr, phantom, "6.3"] & & r \arrow[dd, equal] \\
t \arrow[ur, "*"] \arrow[rr, "C"'] \arrow[dr, "*"'] & & s \arrow[ur, "*"] \arrow[dr, "*"] \arrow[r, phantom, "\text{Ind.}"] & {} \\
 & q \arrow[rr, "*", "C"'] \arrow[ur, phantom, "6.3"] & & q
\end{tikzcd}$$

Figure 9. Proof of preservation of UN$^{\rightarrow}$.

THEOREM 6.2. *If R is left-linear and UN$^{\rightarrow}$, then* PP(R) *is UN$^{\rightarrow}$.*

PROOF. See Figure 9. Let $t \rightarrow^* q$ and $t \rightarrow^* r$ be two reductions of t to normal form. Let $t \rightarrow^*_C s$ be a reduction to $\rightarrow^*_C$-normal form. By Lemma 6.3, s is reducible to both q and r. By Lemma 6.4, $q = r$. □

7. Partial currying

We can generalize PP(R) to represent partial currying, where some symbols are curried and others are not.

DEFINITION 7.1. *Let R be a term rewrite system, and $\mathcal{F}$ a subset of its function symbols. The system* PP$_{\mathcal{F}}(R)$ *is defined identically to* PP(R) *in Definition 2.3, expect that in part (1), F ranges only over $\mathcal{F}$, instead of over all the symbols of R.*

PP$_{\mathcal{F}}(R)$ *is called a* partial currying *of R.*

We can similarly define Cur$_{\mathcal{F}}(R)$, and prove a version of Theorem 2.1 relating the two versions of partial currying.

THEOREM 7.1. *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be sets of symbols of a term rewrite system R, such that $\mathcal{F}_1 \supseteq \mathcal{F}_2$. Let P be any of the properties SN, WN, CR, WCR, completeness, semi-completeness, UN, NF, UN$^{\rightarrow}$. Then $P(\mathrm{PP}_{\mathcal{F}_1}(R)) \Rightarrow P(\mathrm{PP}_{\mathcal{F}_2}(R))$. In particular, for any set $\mathcal{F}$ of symbols of R, $P(\mathrm{PP}(R)) \Rightarrow P(\mathrm{PP}_{\mathcal{F}}(R))$ and $P(\mathrm{PP}_{\mathcal{F}}(R)) \Rightarrow P(R)$.*

PROOF. The proofs are virtually the same as for Theorem 3.1. □

Note that the last of the above implications says that these properties are all reflected by partial currying.

COROLLARY 7.1. *All the properties for which we have proved preservation by currying are preserved by every partial currying.*

8. Modularity properties for applicative term rewrite systems

The notion of modularity was introduced in Toyama (1987), although first given that name in Middeldorp (1989). A property of rewrite systems is modular if, whenever it is true of two disjoint systems (i.e. systems sharing no function symbols) it is true of their union. The notion is important for proving properties of large systems, such as large functional programs, which are built up by combining smaller systems. If a property is modular, it can be proved of the whole system by dividing it into disjoint subsystems and establishing it for each of these subsystems.

Much research has been done on modular properties of term rewrite systems, some of which is summarized in Theorem 2.3. Unfortunately, applicative systems are never disjoint, as they all contain the symbol Ap. Thus none of these modularity theorems apply to unions of applicative systems. Some authors (e.g. Middeldorp and Toyama, 1991; Krishna Rao, 1993) have extended the notion to allow certain types of shared symbols, but unions of applicative systems still fall outside the scope of known results.

Definition 8.1. *Two applicative systems are* applicatively disjoint *if their only common function symbol is Ap.*

A property P of applicative systems is applicatively modular *if whenever R_1 and R_2 are applicatively disjoint systems both satisfying P, then their union also satisfies P.*

Our results do not allow us to prove applicative modularity results, but we can prove a weaker form of modularity.

Definition 8.2. *A property P of applicative systems is* weakly applicatively modular *if whenever R_1 and R_2 are disjoint systems, $\mathcal{F}_1$ and $\mathcal{F}_2$ are subsets of the symbols of R_1 and R_2 respectively, and P is true of both* $\mathrm{PP}_{\mathcal{F}_1}(R_1)$ *and* $\mathrm{PP}_{\mathcal{F}_2}(R_2)$*, then P is true of* $\mathrm{PP}_{\mathcal{F}_1}(R_1) + \mathrm{PP}_{\mathcal{F}_2}(R_2)$.

Applicative modularity implies weak applicative modularity, but in general modularity is independent of each of these. In the case where P is both preserved and reflected by partial currying, however, we can say more.

Theorem 8.1. *If a property is preserved and reflected by partial currying, then it is modular if and only if it is weakly applicatively modular.*

Proof. Let P be preserved and reflected by partial currying. Let R_1 and R_2 be disjoint rewrite systems, and $\mathcal{F}_1$ and $\mathcal{F}_2$ be subsets of their respective sets of symbols. Define $R'_1 = \mathrm{PP}_{\mathcal{F}_1}(R_1)$, $R'_2 = \mathrm{PP}_{\mathcal{F}_2}(R_2)$, and $R'_{12} = \mathrm{PP}_{\mathcal{F}_1 \cup \mathcal{F}_2}(R_1 + R_2)$.

If P is modular, then:

$$\begin{array}{rcll} P(R'_1) \wedge P(R'_2) & \Rightarrow & P(R_1) \wedge P(R_2) & (P \text{ is reflected}) \\ & \Rightarrow & P(R_1 + R_2) & (\text{modularity}) \\ & \Rightarrow & P(R'_1 + R'_2) & (P \text{ is preserved}) \\ & \Rightarrow & P(R'_{12}) & (R'_1 + R'_2 = R'_{12}). \end{array}$$

Hence P is weakly applicatively modular.

If P is weakly applicatively modular, then:

$$\begin{array}{rcll} P(R_1) \wedge P(R_2) & \Rightarrow & P(R'_1) \wedge P(R'_2) & (P \text{ is preserved}) \\ & \Rightarrow & P(R'_1 + R'_2) & (\text{weak applicative modularity}) \\ & \Rightarrow & P(R'_{12}) & (R'_1 + R'_2 = R'_{12}) \\ & \Rightarrow & P(R_1 + R_2) & (P \text{ is reflected}). \end{array}$$

Hence P is modular. □

COROLLARY 8.1. *WN, CR, WCR, and semi-completeness, are weakly applicatively modular. Completeness, NF, and UN are weakly applicatively modular for left-linear systems. $UN^{\rightarrow}$ is weakly applicatively modular for left-linear systems without collapsing rules (i.e. rules whose right-hand side is a variable). SN is weakly applicatively modular, for applicatively disjoint unions where one system contains neither collapsing rules nor duplicating rules (i.e. rules whose right-hand side contains any repeated variable).*

PROOF. All of these properties are preserved and reflected by currying for the relevant systems, and all of the corresponding modularity properties hold. For SN, WN, CR, UN, and NF, the modularity properties are stated in Theorem 2.3, and that for semi-completeness (i.e. WN and CR) follows immediately. For completeness, see Toyama et al. (199–). For $UN^{\rightarrow}$, see Middeldorp (1990). □

Weak applicative modularity is, as the name suggests, a rather weak property. Although CR is weakly applicatively modular, examples in Klop (1980) show that it is not applicatively modular.

Let CL be the system with symbols S, K, and I, with applicative arities 3, 2, and 1, with rules:

$$\begin{aligned} Sxyz &\rightarrow (xz)(yz) \\ Kxy &\rightarrow y \\ Ix &\rightarrow x. \end{aligned}$$

Let D be the system with a binary symbol D and a nullary symbol E, and the rule $D(x,x) \rightarrow E$. Then CL and Cur(D) are both CR, but CL+Cur(D) is not.

Several other properties fail to be applicatively modular.

THEOREM 8.2. *The following properties are not applicatively modular: SN, completeness, semi-completeness, CR, NF, UN, and $UN^{\rightarrow}$.*

PROOF. For SN, completeness, and semi-completeness, take R_1 to be the system $S_1wxyz \rightarrow w(xz)(yx)$, and R_2 to consist of the rules for K and I in CL. Then both R_1 and R_2 are complete, hence SN and semi-complete, but their union is equivalent to CL (define $S = S_1I$). CL has none of these properties.

For the other properties, take R_1 to be CL, and R_2 to be the system:

$$\begin{aligned} Exx &\rightarrow T \\ E(Sx)x &\rightarrow F. \end{aligned}$$

Both systems are CR, and hence also NF, UN, and $UN^{\rightarrow}$. However, their union contains the following reductions. Define $M = NN$ where $N = B(SI)(SII)$ and $B = S(KS)I$. Then M has the property that $Mx \rightarrow^* x(Mx)$. So we have:

$$EMM \rightarrow T$$

$$EMM \rightarrow^* E(SM)M \rightarrow F.$$

Therefore it is not $UN^{\rightarrow}$, hence not UN, NF, or CR. □

The second counterexample is non-left-linear; we do not know if CR, NF, UN, or $UN^{\rightarrow}$

are applicatively modular for left-linear systems. Likewise, the counterexample for SN contains both duplicating and collapsing rules, for which SN is known not to be modular; we do not know if it is applicatively modular under conditions similar to those of Theorem 2.3.

THEOREM 8.3. *WCR and WN are applicatively modular.*

PROOF. For WCR, the result follows from the absence of critical pairs between applicatively disjoint systems.

For WN, the proof is essentially the same as proof of modularity of WN in Middeldorp (1990). Given a term t of the combined system not in normal form, consider an innermost redex of t. We can find a unique subterm of t of the form $P[t_1, \ldots, t_n]$ having the properties that $P[x_1, \ldots, x_n]$ is a maximal strict component of t consisting entirely of variables and function symbols from one of the two systems R, each of $t_1, \ldots, t_n$ is a normal form, and $P[x_1, \ldots, x_n]$ contains the redex. (We must choose P to be strict, so that every redex of $P[t_1, \ldots, t_n]$ is a redex of $P[x_1, \ldots, x_n]$, even if the system is non-left-linear.)

By the WN property of R_i, $P[x_1, \ldots, x_n]$ is reducible to a normal form P'. Replacing $x_1, \ldots, x_n$ in P' by the normal forms $t_1, \ldots, t_n$, giving a term P'', cannot create any new redexes, since the principal symbol of each t_i is not in R. Therefore P'' is in normal form. Replacing $P[t_1, \ldots, t_n]$ by P'' in t can only create new redexes in t at positions which are proper prefixes of the position of the originally chosen redex. Therefore if we consider the multiset of depths of all redexes in t, this reduction removes one or more members from that set (those corresponding to all the redexes in $P[x_1, \ldots, x_n]$) and any new members which are added are smaller than those which were removed. Therefore by Lemma 2.3, the process terminates, giving a normal form of t. □

Acknowledgements

An earlier version of this paper appeared in Kennaway et al. (1993). We would like to acknowledge the help and criticism of Vincent van Oostrom, Aart Middeldorp, Femke van Raamsdonk, Yoshihito Toyama, Hans Zantema, and especially Stefan Kahrs, whose work inspired us to improve our proofs and extend our results. Some of the figures were drawn with Paul Taylor's diagram package.

References

Curry, H.B., Feys, R. (1958). *Combinatory Logic*, volume I. Amsterdam: North-Holland.

Dershowitz, N., Jouannaud, J.-P. (1989). Rewrite systems. In (van Leeuwen, J., ed.), *Handbook of Theoretical Computer Science*, volume B, chapter 15. Amsterdam: Elsevier.

Dershowitz, N., Manna, Z. (1979). Proving termination with multiset orderings. *Comm.ACM*, 22(8):465–476.

Field, A.J., Harrison, P.G. (1988). *Functional Programming*. Wokingham, England: Addison-Wesley.

Kahrs, S. (1994). Confluence of curried term rewriting systems. Submitted to *J. Symbolic Computation*.

Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.J. (1993). Comparing curried and uncurried rewriting. In (Barendregt, H., Bezem, M., Klop, J., eds.) *Dirk van Dalen Festschrift*, in series *Quaestiones Infinitae*, volume 5, pp. 57–78. Department of Philosophy, University of Utrecht.

Kennaway, J.R., Sleep, M.R. (1982). Expressions as processes. In *Proc. ACM Symposium on LISP and Functional Programming*, pp. 21–27. New York: ACM.

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

Klop, J.W. (1992). Term rewriting systems. In (Abramsky, S., Gabbay, D., Maibaum, T., eds.) *Handbook of Logic in Computer Science, Volume II.* Oxford: Oxford University Press.

Krishna Rao, M.R.K. (1993). Completeness of hierarchical combinations of term rewriting systems. In *Proc. 13th Conference on Foundations of Software Technology and Theoretical Computer Science.* Lecture Notes In Computer Science, vol.761. Berlin: Springer-Verlag.

Kruskal, J.B. (1960). Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Trans. AMS*, 95:210–225.

Middeldorp, A. (1989). Modular aspects of properties of term rewriting systems related to normal forms. In *Proc. 3rd International Conference on Rewriting Techniques and Applications*, pages 263–277. Lecture Notes In Computer Science, vol.355. Berlin: Springer-Verlag.

Middeldorp, A. (1990). *Modular Properties of Term Rewriting Systems.* PhD thesis, Vrije Universiteit, Amsterdam.

Middeldorp, A. and Toyama, Y. (1991). Completeness of combinations of constructor systems. *J. Symbolic Computation*, 15:331–348.

Toyama, Y. (1987). On the Church-Rosser property for the direct sum of term rewriting systems. *JACM*, 34(1):128–143.

Toyama, Y., Klop, J.W., and Barendregt, H.P. (199–). Termination for direct sums of left-linear complete term rewriting systems. *J. Assoc. Comp. Mach.*, to appear.

Zantema, H. (1994). Termination of term rewriting by semantic labelling. Technical report RUU-CS-92-38, University of Utrecht.