Confluence of Curried Term-Rewriting Systems

STEFAN KAHRS[†]

Department of Computer Science, University of Edinburgh,

Edinburgh EH9 3JZ, United Kingdom

(Received 31 July 1995)

Term rewriting systems operate on first-order terms. Presenting such terms in curried form is usually regarded as a trivial change of notation. However, in the absence of a type-discipline, or in the presence of a more powerful type-discipline than simply typed λ-calculus, the change is not as trivial as one might first think.

It is shown that currying preserves confluence of *arbitrary* term rewriting systems. The structure of the proof is similar to Toyama's proof that confluence is a modular property of TRS.

1. Introduction

Currying is usually seen as an operation on types with a corresponding operation on terms of those types. But even in the world of untyped Term Rewriting Systems (TRSs), currying has a meaning: it allows the partial application of functions. In Functional Programming, one can distinguish certain programming styles, e.g. in Standard ML:

```
datatype NAT = Z | S of NAT
fun add   Z   x = x
 |  add (S x) y = S(add x y)
fun plus(Z,x)   = x
 |  plus(S x,y) = S(plus(x,y))
```

The functions `add` and `plus` are both addition operations for the datatype `NAT`. The difference is: `add` is curried, `plus` is not. They have different types and `add` can be applied to a single natural number, i.e. `add Z` is a well-formed term (of a function type). As long as the programming style remains first-order, currying is purely a matter of syntax. But in the presence of functionals, the curried version has now a wider range of application, for example:

```
datatype 'a LIST = NIL | CONS of 'a * 'a LIST
fun map f NIL = NIL
 |  map f (CONS(x,xs)) = CONS(f x, map f xs)
```

[†] E-mail: `smk@dcs.ed.ac.uk`

The operation `map` is a functional, its first argument is expected to be a function. Such a function can be obtained by providing `add` with only one argument, e.g. `add(S(S(Z)))` has the required form, and `map(add(S(S(Z))))` is then a function on lists. This is not directly expressible with `plus`, one had to use a λ-abstraction instead, for example `fn x=>plus(S(S(Z)),x)` is the same function as `add(S(S(Z)))`.

Since application can be expressed with a binary function symbol `Apply`, the curried form of a TRS is still a TRS. Therefore, an interesting question is: to what extent is a TRS equivalent to its curried form? What happens to CR, SN and other properties, if partial applications are allowed?

At first, one may think that the question is either meaningless or trivial. Meaningless, because in a *typed* first-order TRS the partial applications have function types and hence cannot be terms. Trivial, because the absence of functionals such as `map` from TRSs seems to make partial applications pointless. Both thoughts miss an important point: the type problem has to do with the fact that the type system one usually considers for TRSs comes from the simply typed λ-calculus (Hindley and Seldin, 1986) and one could well consider more powerful type systems; the absence of functionals is disturbed by collapsing rules (rules with a variable as right-hand side), because they can act as functionals in a curried untyped TRS. We can even observe both aspects in ML:

```
fun head(CONS(x,xs)) = x
```

The type of the function `head` is a polymorphic function type, the result type of which is a bound type variable; it can be instantiated to any type, in particular to a function type. Combining `head` with partial applications, we can form a term like `head(CONS(S,NIL))Z` which is well-formed, well-typed, and which evaluates to `S(Z)`. Notice that the definition of `head` is an ordinary term rewriting rule — in the corresponding curried TRS we indeed have the reduction `head(CONS(S,NIL))Z`$\twoheadrightarrow$`S Z`. An important feature of the example is that the rule for `head` was collapsing — without collapsing rules, currying would indeed be insignificant, see Breazu-Tannen and Gallier (1989).

We are going to prove that *currying preserves confluence* for arbitrary TRSs. The technique to obtain this result is derived from Toyama's proof that confluence is a modular property of TRSs. We split the proof into two parts: in section 4 we give an abstract confluence proof extracting the technique common to both proofs, and in section 5 we instantiate this abstract proof with the data particular to the problem of currying. The general idea behind the technique is that for certain properties of terms there are associated cofinal reduction strategies.

This result first appeared in my thesis (Kahrs, 1991). At the time, I did not bother to publish it separately, because the proof was rather tedious and the result appeared to me as being too predictable to justify such complications.

I rapidly changed my mind when I came across a technical report by Kennaway, Klop, Sleep and de Vries (1993) on the same subject. In this 1993 paper, the authors showed that currying preserves a variety of rewrite properties, most notably termination and completeness; but they also contradicted my thesis in claiming that confluence is in general *not* preserved by currying, and gave the construction of a counter-example. Besides challenging my scientific integrity this incidence showed me that my preconception about the predictability of the result was rather premature. I shall explain in section 6 where the construction of the alleged counter-example goes wrong. A corrected and updated version of their paper (1995) is about to be published in the Journal of Symbolic Computation.

2. Preliminaries

We identify total and partial functions (and relations) with their graphs and operate on them with the usual set-theoretic operations. We write $A \to B$ for the set of functions between A and B, and $A \rightharpoonup B$ for the corresponding set of partial functions. $\mathit{Dom}\, f$ is the domain of a partial function f. We take $\uplus$ for the disjoint union of sets, $\wp A$ for the powerset of the set A.

We write $\mathcal{N}$ for the set of natural numbers, and $\mathcal{N}_+$ for $\mathcal{N} \setminus \{0\}$.

Given a set A, a relation R on A is a subset of $A \times A$. We write xRy for $(x, y) \in R$ and $xRySz$ for $xRy \wedge ySz$. For two relations R, S on A, the expression $R;S$ denotes the composition of the relations R and S, i.e. $x\ R;S\ y \Leftrightarrow \exists z \in A.\ xRz \wedge zSy$.

R^{-1} is the *inverse* relation of R: $xR^{-1}y \Leftrightarrow yRx$. For the inverse of a relation called $\to_a$ (for various a) we write $\leftarrow_a$. A relation R on A is reflexive, iff $\forall x \in A.\ xRx$. R is *symmetric* iff $R = R^{-1}$ and it is *transitive* iff $R;R \subseteq R$. We write $R^=$ for the reflexive, R^+ for the transitive, and R^* for the reflexive and transitive closure of R. Following Klop (1992), we write $\twoheadrightarrow_a$ for the reflexive and transitive closure of a relation called $\to_a$. An equivalence relation is reflexive, symmetric and transitive. We write $\overset{*}{\leftrightarrow}_A$ for the equivalence closure of a relation $\to_A$.

A relation R is antisymmetric if $R \cap R^{-1} \subseteq \emptyset^=$. A relation $\leq$ is a partial order iff it is reflexive, transitive and antisymmetric. A relation $\leq$ on A is a total order iff it is a partial order and if $\leq \cup \geq\ = A \times A$. Given a partial order $\leq$ on A, an antichain is a subset $A' \subseteq A$ such that $\forall x, y \in A'.\ \neg x < y \wedge \neg y < x$ with $< = \leq \setminus \emptyset^=$.

Given any set A, A^* is the set of finite words over A (free monoid); ϵ is the empty word, $v \cdot w$ denotes concatenation of the words v and w. Elements of A are also understood as singleton words. A^* is partially ordered by the prefix ordering $\preceq$:

$$p \preceq q \iff \exists w \in A^*.\ p \cdot w = q$$

If $p \preceq q$ then p is called a *prefix* of q. It is a *proper prefix*, $p \prec q$, if additionally $p \neq q$. Two words p and q are *independent*, written $p \mid q$, iff neither is a prefix of the other, i.e. $p \not\preceq q \wedge q \not\preceq p$.

Definition 2.1. *Given an antichain M of words over A and a word $v \in A$, we write (i) $M \mid v$ iff $\forall w \in M.\ v \mid w$; (ii) $M \preceq v$ iff $\exists w \in M.\ w \preceq v$; (iii) $M \succ v$ iff $\exists w \in M.\ v \succ w$.*

Notice that $M \mid v$, $M \preceq v$ and $M \succ v$ are mutually exclusive (an antichain in the set of predicates on pairs of antichains and elements, ordered by "$\Rightarrow$") and cover all cases.

2.1. Abstract Reduction Systems

Definition 2.2. *An Abstract Reduction System (short: ARS) consists of a set A and a sequence $\to_i$ of binary relations on A, labelled by some set I. We often drop the label if I is a singleton.*

We write $\mathcal{A} \models P$ if the ARS $\mathcal{A} = (A, \to_i, \ldots), i \in I$ has the property P. Further we write $\mathcal{A} \models P \wedge Q$ iff $\mathcal{A} \models P$ and $\mathcal{A} \models Q$.

An ARS $\mathcal{A} = (A, \to)$ has the *diamond property*, $\mathcal{A} \models \Diamond$, iff $\leftarrow;\to\ \subseteq\ \to;\leftarrow$. It has the *Church-Rosser property* (is *confluent*), $\mathcal{A} \models \mathrm{CR}$, iff $(A, \twoheadrightarrow) \models \Diamond$. Given an ARS $\mathcal{A} = (A, \to)$, we write $\mathrm{CR}(t)$ as shorthand for $(\{u \mid t \twoheadrightarrow u\}, \to) \models \mathrm{CR}$.

Under most circumstances, confluence is a useful property of ARSs, mainly because: if $(A, \rightarrow) \models \mathrm{CR}$, and if two elements $x, y \in A$ are equivalent w.r.t. the smallest equivalence containing $\rightarrow$, then there is a $z \in A$ such that $x \twoheadrightarrow z \twoheadleftarrow y$. Roughly: the ARS decides the equivalence.

An ARS $\mathcal{A} = (A, \rightarrow_a, \rightarrow_b)$ *commutes directly*, $\mathcal{A} \models \mathrm{CD}$, iff $\leftarrow_a ; \rightarrow_b \subseteq \rightarrow_b ; \leftarrow_a$.

To prove confluence of an ARS, it is sometimes useful to translate it into another, see for instance Staples (1975). We use here a slightly different proposition:

Proposition 2.1. *Let $\mathcal{A} = (A, \rightarrow_A)$ and $\mathcal{B} = (B, \rightarrow_B)$ be ARSs with $\mathcal{B} \models \mathrm{CR}$. Let $F : A \rightarrow B$, $G : B \rightarrow A$ be functions such that:*

$$\forall x, y \in A.\ x \rightarrow_A y \Rightarrow F(x) \overset{*}{\leftrightarrow}_B F(y)$$
$$\forall x \in A. y' \in B.\ F(x) \twoheadrightarrow_B y' \Rightarrow x \twoheadrightarrow_A G(y')$$

Then $\mathcal{A} \models \mathrm{CR}$.

Proof. If $x \overset{*}{\leftrightarrow}_A y$ then $F(x) \overset{*}{\leftrightarrow}_B F(y)$ by the first condition. We get a $z \in B$ with $F(x) \twoheadrightarrow_B z$ and $F(y) \twoheadrightarrow_B z$ by the confluence of $\mathcal{B}$. Finally, we have $x \twoheadrightarrow_A G(z)$ and $y \twoheadrightarrow_A G(z)$ by the second condition. $\square$

Given an ARS $\mathcal{A} = (A, \rightarrow_A)$, $t \in A$ is a *normal form* if $\neg \exists u \in A.\ t \rightarrow_A u$. We write $\mathrm{NF}_{\mathcal{A}}$ for the set of normal forms of $\mathcal{A}$. t *has* a normal form $u \in \mathrm{NF}_{\mathcal{A}}$ if $t \twoheadrightarrow_A u$.

An ARS $\mathcal{A} = (A, \rightarrow)$ is *strongly normalising*, $\mathcal{A} \models \mathrm{SN}$, iff there is no non-empty relation R on A satisfying the equation $R = \rightarrow ; R$, i.e. iff there are no infinite chains of $\rightarrow$-steps. It is weakly Church-Rosser, $\mathcal{A} \models \mathrm{WCR}$, iff $\leftarrow ; \rightarrow \subseteq \twoheadrightarrow ; \twoheadleftarrow$.

2.2. Terms and Occurrences

Definition 2.3. *A tree domain $\mathcal{D}$ is a non-empty subset of $\mathcal{N}_+^*$ satisfying the following properties:*

$$w \in \mathcal{D},\ v \prec w \ \Rightarrow\ v \in \mathcal{D}$$
$$m, n \in \mathcal{N}_+,\ v \cdot n \in \mathcal{D},\ m < n \Rightarrow v \cdot m \in \mathcal{D}$$
$$v \in \mathcal{D} \Rightarrow \exists n \in \mathcal{N}_+.\ v \cdot n \notin \mathcal{D}$$

Given a tree domain $\mathcal{D}$, last($\mathcal{D}$) is a natural number $n \in \mathcal{N}$, such that $n = 0 \iff \mathcal{D} = \{\epsilon\}$ and otherwise $n \in \mathcal{D}$, $n + 1 \notin \mathcal{D}$. A tree domain is finite iff it is a finite set. For a finite tree domain $\mathcal{D}$, we define depth($\mathcal{D}$) to be the length of the longest word in $\mathcal{D}$.

The last property we required for a tree domain forces trees to be finitely branching. Infinitely branching trees may be interesting for certain applications, but not in this context. We assume the existence of a countably infinite set $\mathcal{V}$ of *variables*.

Definition 2.4. *A ranked alphabet $\Sigma = (A, \#)$ consists of a set A and a function $\# : A \rightarrow \mathcal{N}$. A preterm over Σ is a partial function $t : \mathcal{N}_+^* \rightharpoonup A \uplus \mathcal{V}$, where $\mathit{Dom}\, t = \mathcal{D}$ is a tree domain and for any $p \in \mathcal{D}$ with $t(p) \in \mathcal{V}$ we have $\forall w.\ p \cdot w \in \mathcal{D} \Rightarrow w = \epsilon$. A preterm $t : \mathcal{D} \rightarrow A \uplus \mathcal{V}$ is called a term if it satisfies the following property:*

$$p \in \mathcal{D},\ t(p) \in A,\ n \in \mathcal{N} \quad \Rightarrow \quad p \cdot n \in \mathcal{D} \iff n \leq \#(t(p))$$

The set of preterms (or terms) over Σ is called Pre(Σ) (or Ter(Σ), respectively). If

$\mathcal{D} = Dom\,t$, we say that p is an occurrence in t if $p \in \mathcal{D}$ and write t/p for the preterm $t/p(w) = t(p \cdot w)$ with domain $\mathcal{D}' = \{w \mid p \cdot w \in \mathcal{D}\}$. If $A \subseteq \mathcal{D}$, we write t/A to abbreviate the set $\{t/p \mid p \in A\}$. If p is an occurrence in t and u is a preterm, we write $t[p = u]$ for the preterm determined by: $t[p = u]/p = u \;\wedge\; \forall q \in Dom\,t.\ (p \not\leq q \Rightarrow t(q) = t[p = u](q))$.

The idea of viewing a (pre)term as a function from a tree domain to a ranked alphabet is fairly standard, see for instance Brainerd (1969), Huet (1980), Lloyd (1984), and Rosen (1973). The particular advantage is the simple addressing of a subterm.

Proposition 2.2. *Let t be a preterm and $p, q \in Dom\,t$ with $p \mid q$.
We have $t[p = a][q = b] = t[q = b][p = a]$.*

This observation motivates the following notation: If $\{p_1, \ldots, p_n\} \subseteq Dom\,t$ is an antichain then $t[p_1 = a_1, \ldots, p_n = a_n]$ is shorthand for $t[p_1 = a_1] \cdots [p_n = a_n]$.

It is convenient for technical reasons to allow proper preterms, i.e. "terms" where function symbols are used with the wrong arity. Since currying forgets the arity of a function symbol, uncurrying naturally introduces such "terms".

For matters of presentation, we display preterms over the ranked alphabet $(A, \#)$ as words over the alphabet $A \uplus \mathcal{V} \uplus \{(,,,)\}$, in the following way: let $\hat{t}$ denote the word corresponding to a preterm $t : \mathcal{D} \to A \uplus \mathcal{V}$, then this has to satisfy:

$$\begin{array}{lcll}
\hat{t} & = & t(\epsilon) & \text{if } last(t) = 0 \\
\hat{t} & = & t(\epsilon) \cdot (\cdot \arg(t, n) \cdot) & \text{if } last(\mathcal{D}) = n > 0 \\
\arg(t, 1) & = & \widehat{t/1} & \\
\arg(t, n) & = & \arg(t, n-1) \cdot , \cdot \widehat{t/n} & \text{if } n > 1
\end{array}$$

Notice that the commas and parentheses are necessary to distinguish different preterms, i.e. to make the function $t \mapsto \hat{t}$ injective. They would be redundant if we only considered terms.

In the following we tacitly assume a fixed ranked alphabet $\Sigma = (A, \#)$, unless we explicitly give different ranked alphabets.

Definition 2.5. *A prevaluation σ is a function from variables to preterms. Given a prevaluation σ, a presubstitution $\overline{\sigma}$ is a map $Pre(\Sigma) \to Pre(\Sigma)$ with the properties: let p be an occurrence in t, then*

$$\begin{array}{lcll}
\overline{\sigma}(t)(p) & = & t(p), & \text{if } t(p) \in A \\
\overline{\sigma}(t)/p & = & \sigma(t(p)), & \text{if } t(p) \in \mathcal{V}
\end{array}$$

Let $Dom\,t = \mathcal{D}$, then the tree domain $\mathcal{D}'$ of the preterm $\overline{\sigma}(t)$ is the set $\mathcal{D} \cup \{a \cdot b \mid a \in \mathcal{D},\ t(a) \in \mathcal{V},\ b \in Dom\,\sigma(t(a))\}$. A prevaluation is called a valuation *if its codomain only consists of terms. A substitution is a presubstitution that is a valuation on variables.*

Definition 2.6. *A context $C\langle p\rangle$ consists of a term C and an occurrence $p \in Dom\,C$. We write $C[t]$ for $C[p = t]$.*

Definition 2.7. *A binary relation $\to_R$ on preterms is substitutive, if $t \to_R u \Rightarrow \overline{\sigma}(t) \to_R \overline{\sigma}(u)$ for any prevaluation σ. It is called* compatible, *if for any context $C\langle p\rangle$ we have: $t \to_R u \Rightarrow C[t] \to_R C[u]$. If a relation is both substitutive and compatible, then it is called a* rewrite relation.

DEFINITION 2.8. *A Term Rewriting System (short: TRS) $R = (\Sigma, \rightarrow)$ consists of a ranked alphabet Σ and a binary relation $\rightarrow$ on terms over Σ, provided that:*

$$\begin{array}{lcl} l \rightarrow r & \Rightarrow & l(\epsilon) \notin \mathcal{V} \\ l \rightarrow r & \Rightarrow & \forall p \in Dom\, r.\ r(p) \in \mathcal{V} \Rightarrow \exists q \in Dom\, l.\ l(q) = r(p) \end{array}$$

The relation $\rightarrow_R$ denotes the smallest rewrite relation on terms over Σ such that $\rightarrow \subset \rightarrow_R$.

In other words: we associate with a TRS $R = (\Sigma, \rightarrow)$ an abstract reduction system $\mathcal{R} = (\mathrm{Ter}(\Sigma), \rightarrow_R)$. Occasionally we want to make the redex occurrence of a rewrite step explicit. We write $t \overset{v}{\rightarrow}_R u$ as shorthand for: there exists a valuation σ and a rule $l \rightarrow r$ such that $t/v = \overline{\sigma}(l)$ and $u = t[v = \overline{\sigma}(r)]$.

3. Currying

The definition of currying strongly depends on the notion of signature for terms. The signatures I have chosen ("ranked alphabets") have the particular property of assigning any function symbol its arity. This is not the only possible notion of signature.

Instead of the rank function, one could have a function mapping each natural number to the set of symbols having this number as their arity. This approach seems to be more fashionable in Universal Algebra (for example in Meinke and Tucker (1992)), and within Term Rewriting it does not really make an important difference. In fact, it very much corresponds to the definition of a TRS given above, with the only change that the requirement $\rightarrow\, \subset \mathrm{Ter}(\Sigma) \times \mathrm{Ter}(\Sigma)$ is slightly weakened to $\rightarrow\, \subset \mathrm{Pre}(\Sigma) \times \mathrm{Pre}(\Sigma)$.

The technical difference is that having a family of arity-indexed sets of symbols allows overloading, i.e. function symbols can have more than one arity. Concerning the presentation of terms (not preterms) as words, terms with overloading require parentheses and commas for disambiguation, otherwise e.g. the word `A B C` could correspond to `A(B,C)` and `A(B(C))`, if both `A` and `B` are accordingly overloaded. If we are interested in currying, TRSs with overloading have very bad properties, because currying identifies overloaded symbols. Example:

$$\begin{array}{rcl} F(x) & \rightarrow & G(x) \\ G & \rightarrow & F \\ F & \rightarrow & H \\ G(x) & \rightarrow & I(x) \end{array}$$

The example is a confluent and terminating TRS, but currying loses both properties. But the example also shows the identification of symbols, which is an undesirable side-effect. Thus we stick to ranked alphabets. So far, we have used the word "currying" more or less informally, we have not provided yet a formal definition. We shall consider two ways to define it formally as a map between TRSs; the first one is called "currying" and the second one "partial parameterisation". They are strongly related, but not quite the same.

DEFINITION 3.1. *Let $R = (\Sigma, \rightarrow)$ be a TRS with $\Sigma = (A, \#)$. $\mathrm{Cu}(R)$ is the curried TRS of R with: $\mathrm{Cu}(R) = (\Theta, \rightsquigarrow)$, $\Theta = \mathrm{Cu}(\Sigma) = (A' \uplus \{\mathtt{Apply}\}, \#')$, $A' = \{F_0 \mid F \in A, \#(F) > 0\} \uplus \{F \mid F \in A, \#(F) = 0\}$. The rank function $\#'$ is given by $\#'(\mathtt{Apply}) = 2$ and $\#'(F) = 0$ for $F \neq \mathtt{Apply}$. The relation $\rightsquigarrow$ is the smallest relation satisfying $t \rightarrow u \;\Rightarrow\; \mathrm{Cu}(t) \rightsquigarrow \mathrm{Cu}(u)$, where $\mathrm{Cu} : Pre(\Sigma) \rightarrow Ter(\Theta)$ is determined as follows: Let*

$t : \mathcal{D} \to A \uplus \mathcal{V}$ be a preterm over Σ, then $\mathrm{Cu}(t)$ *is a term over Θ such that:*

$$\begin{array}{lcll}
\mathrm{Cu}(t)(\epsilon) & = & t(\epsilon) & \textit{if } t(\epsilon) \in \mathcal{V} \vee (t(\epsilon) \in A \wedge \#(t(\epsilon)) = 0) \\
\mathrm{Cu}(t)(\epsilon) & = & t(\epsilon)_0 & \textit{if } t(\epsilon) \in A \wedge \textit{last}(\mathcal{D}) = 0 \wedge \#(t(\epsilon)) > 0 \\
\mathrm{Cu}(t)(\epsilon) & = & \mathtt{Apply} & \textit{if last}(\mathcal{D}) > 0 \\
\mathrm{Cu}(t)/2 & = & \mathrm{Cu}(t/n) & \textit{if last}(\mathcal{D}) = n > 0 \\
\mathrm{Cu}(t)/1 & = & \mathrm{Cu}(t') & \textit{if last}(\mathcal{D}) = n > 0 \\
 & & & \textit{where Dom}\, t' = \{w \in \mathcal{D} \mid n \not\preceq w\},\ \forall p \in \textit{Dom}\, t'.\ t'(p) = t(p)
\end{array}$$

Informally, the operation Cu is the identity on variables and constants. It maps a Σ-term $F(M_1, \cdots, M_n)$ with $n > 0$ to a Θ-term $\mathtt{Apply}(\mathrm{Cu}(F(M_1, \cdots, M_{n-1})), \mathrm{Cu}(M_n))$. This informal description uses the same recursive structure as the definition — which explains why Cu is defined for preterms rather than just for terms.

In this particular definition of currying, I chose to label constants originating from non-nullary symbols with 0 and to leave variables and other symbols unchanged. This is not significant for any of the results, it is just technically convenient within the proofs.

The given definition of currying resembles the idea of a different programming style. However, it has the disadvantage of "changing notation" and therefore arguing about rewrite steps has to be done modulo the operation Cu and its inverse. An alternative way of expressing Currying is to enrich a TRS. We call this: "partial parameterisation".

Definition 3.2. *Let $R = (\Sigma, \to)$ be a TRS with $\Sigma = (A, \#)$.* $\mathrm{PP}(R)$ *is the partially parameterised TRS of R with:* $\mathrm{PP}(R) = (\mathrm{PP}(\Sigma), \to \uplus \to')$*, where* $\mathrm{PP}(\Sigma) = (A \uplus \{\mathtt{Apply}\} \uplus A', \# \uplus \#')$*. A' is the set $\{F_n \mid F \in A, \#(F) > n \geq 0\}$. $\#'$ is defined as $\#'(\mathtt{Apply}) = 2$ and $\#'(F_n) = n$. The set $\to'$ consists of the following rules:*

$$\begin{array}{lcl}
l \to' r & \Longleftrightarrow & r \textit{ is injective} \wedge l(\epsilon) = \mathtt{Apply} \wedge \\
 & & \exists n \in \mathcal{N}, F \in A.\ n = \textit{last}(\textit{Dom}\, r) \wedge l(1) = F_{n-1} \wedge \\
 & & (r(\epsilon) = F_n \ \vee\ r(\epsilon) = F) \wedge \forall k < n.\, l/1 \cdot k = r/k \wedge \\
 & & l/2 = r/n \wedge \forall k \leq n.\, r(k) \in \mathcal{V}
\end{array}$$

The last part of the definition is perhaps a bit cryptic. Informally, the rules in $\to'$ have all the following form:

$$\begin{array}{ll}
\mathtt{Apply}(F_n(x_1, \ldots, x_n), x_{n+1}) \to' F_{n+1}(x_1, \ldots, x_n, x_{n+1}) & \text{if } \#(F) > n+1 \\
\mathtt{Apply}(F_n(x_1, \ldots, x_n), x_{n+1}) \to' F(x_1, \ldots, x_n, x_{n+1}) & \text{if } \#(F) = n+1
\end{array}$$

for arbitrary symbols $F \in A$ and distinct variables x_1 to x_{n+1}.

Notice that $\mathrm{PP}(R)$ contains R as a proper subsystem. Notice also that $\mathrm{PP}(\Sigma)$ contains $\mathrm{Cu}(\Sigma)$ as a subsignature. This makes it easier to reason about properties preserved by currying. It is easy to check that the ARSs $\mathcal{U} = (U, \to_U)$ and $\mathcal{C} = (C, \to_C)$ associated with the TRSs $(\mathrm{PP}(\Sigma), \to')$ and $(\mathrm{PP}(\Sigma), \leftarrow')$, respectively, are both CR and SN.

It is easy to see that $\mathrm{Cu}(R)$ and $\mathrm{PP}(R)$ are closely related, e.g. $\mathrm{Cu}(R)$ can be seen as the same as $\mathrm{PP}(R)$ quotiented by the congruence relation induced by $\to'$. For the purposes of a confluence proof, we can instantiate proposition 2.1 and reduce the problem of showing confluence for $\mathrm{Cu}(R)$ to the problem of showing it for $\mathrm{PP}(R)$. Notice that $\mathrm{Cu}(t) \twoheadrightarrow_U t$ for any term $t \in \mathrm{Ter}(\Sigma)$.

Proposition 3.1. *Let R be a TRS. Let $\mathcal{A} = (A, \to_A)$ and $\mathcal{B} = (B, \to_B)$ be the ARSs associated with* $\mathrm{Cu}(R)$ *and* $\mathrm{PP}(R)$*, respectively. Then $\mathcal{B} \models$ CR implies $\mathcal{A} \models$ CR.*

PROOF. We instantiate proposition 2.1 by giving maps $F : \mathrm{Ter}(\mathrm{Cu}(\Sigma)) \to \mathrm{Ter}(\mathrm{PP}(\Sigma))$ with $F(x) = x$ and $G : \mathrm{Ter}(\mathrm{PP}(\Sigma)) \to \mathrm{Ter}(\mathrm{Cu}(\Sigma))$ with $x \twoheadrightarrow_C G(x)$, $G(x) \in \mathrm{NF}_{\mathcal{C}}$. G is well-defined since $\mathcal{C} \models \mathrm{CR} \wedge \mathrm{SN}$.

We have to show that F and G satisfy the two properties required in proposition 2.1.

(i) suppose $a \to_A b$. We can assume $a = D[\overline{\sigma}(\mathrm{Cu}(l))]$ and $b = D[\overline{\sigma}(\mathrm{Cu}(r))]$ for some valuation σ and some context $D\langle p\rangle$. We have $F(a) = a = D[\overline{\sigma}(\mathrm{Cu}(l))] \twoheadrightarrow_B D[\overline{\sigma}(l)] \to_B D[\overline{\sigma}(r)] \twoheadleftarrow_B D[\overline{\sigma}(\mathrm{Cu}(r))] = b = F(b)$.

(ii) suppose $a \to_B b$. Since G distributes over the applications of contexts and substitutions, we have either $G(a) \to_A G(b)$ (if the applied rule is in R) or $G(a) = G(b)$ (if the applied rule is in $\to'$). Hence $a \twoheadrightarrow_B b$ implies $G(a) \twoheadrightarrow_B G(b)$. For each term $x \in \mathrm{Ter}(\mathrm{Cu}(\Sigma))$ we have $G(x) = x$. Now assume $F(x) \twoheadrightarrow_B y$: we get $x = G(F(x)) \twoheadrightarrow_A G(y)$.□

4. A modular proof of CR

The proof that confluence is preserved by currying follows very much the lines of Toyama's proof (Toyama, 1987; Klop, Middeldorp, Toyama, and de Vrijer, 1991) for the confluence of the disjoint union of confluent TRSs. To make this proof more easily re-usable, we abstract in this section from its common structure.

To some extent, this is in the spirit of Hindley's abstract CR proof for λ-calculus and Combinatory Logic (Hindley, 1969), although one could say that Hindley abstracts different aspects of the proof. I always felt that Hindley abstracted *too much*, making it too difficult to instantiate the proof for other applications — and I have not seen any application of his proof since. On the other hand, Hindley's abstract proof seems to have limited use for TRSs, as it (unsurprisingly) does not address the problems implied by non-left-linear rewrite rules. Another related paper is by Jean Gallier (1994), in which he abstracts common structure for proofs of confluence or termination for various enriched typed λ-calculi.

Throughout this section we assume a fixed rewrite system with ranked alphabet Σ and rewrite relation $\to_R$.

DEFINITION 4.1. *Let P be predicate on terms. It is called subterm-closed if*

$$\forall t \in \mathit{Ter}(\Sigma).\, \forall v \in \mathit{Dom}(t).\ P(t) \;\Rightarrow\; P(t/v).$$

P is called reduction-closed if

$$\forall t, u \in \mathit{Ter}(\Sigma).\ P(t) \wedge t \twoheadrightarrow_R u \;\Rightarrow\; P(u).$$

P is called a rewrite predicate if it is subterm-closed and reduction-closed.

DEFINITION 4.2. *A rewrite predicate P is called a pre-confluence if it satisfies:*

$$\forall t \in \mathit{Ter}(\Sigma).\ P(t) \;\Rightarrow\; \mathrm{CR}(t)$$

At this point I should perhaps warn the reader that the technique used here has not much in common with Girard's reducibility candidates (Girard, Lafont, and Taylor, 1989). In particular, I am *not* going to define some pre-confluence P which holds for all terms. The proof idea is to show that (under certain conditions) (i) every term reduces to a term satisfying P and that (ii) this reduction does not "disturb" other reductions.

4.1. Induction via contexts

Definition 4.3. *A function $s : Ter(\Sigma) \to \wp\mathcal{N}_+^*$ is called a context selector, if $s(t) \subseteq Dom\, t$ is an antichain. A context selector is called proper if $\epsilon \notin s(t)$ for any t.*

Justification: since the occurrences in $s(t)$ are independent, we can stratify t into an n-ary context plus subterms, i.e. $t = C[t/p_1, \ldots, t/p_n]$ for some context $C[\ ]$, where $\{p_1, \ldots, p_n\} = s(t)$. This gives rise to some induction principle if s is proper.

Definition 4.4. *Given a predicate P, we write $\overline{P}$ for the predicate*

$$\overline{P}(t) \overset{def}{\Longleftrightarrow} \forall v \in Dom(t).\ (\neg(P(t/v)) \iff v = \epsilon).$$

We also write $\widehat{P}$ for the function $\widehat{P} : Ter(\Sigma) \to \wp\mathcal{N}_+^*$ with $\widehat{P}(t) = \{v \in Dom\, t \mid \overline{P}(t/v)\}$.

Lemma 4.1. *Let P be a predicate on terms. Then for any finite term t:*

1 $\overline{P}(t) \Rightarrow \neg P(t)$
2 $\forall w \in Dom\, t.(\forall v \in \widehat{P}(t).\ v \mid w \ \Rightarrow\ P(t/w))$.

Proof. First property:
$\overline{P}(t) \ \Rightarrow (\forall v \in Dom\, t.\ \neg P(t/v) \Leftrightarrow v = \epsilon) \ \Rightarrow (\neg P(t/\epsilon) \Leftrightarrow \epsilon = \epsilon) \ \Rightarrow \neg P(t)$.
Second property: $\forall v \in \widehat{P}(t).\ v \mid w \quad \Rightarrow \quad \forall v \in Dom\, t.\ \overline{P}(t/v) \Rightarrow v \mid w \quad \Rightarrow$
$\forall v \in Dom\, t.\ (\forall v' \in Dom(t/v).\ \neg P(t/v/v') \Leftrightarrow v' = \epsilon) \Rightarrow v \mid w \quad \Rightarrow$
$\forall v \in Dom\, t.(\neg P(t/v) \wedge \forall v' \in Dom(t/v) \setminus \{\epsilon\}.\ P(t/v/v')) \Rightarrow v \mid w \quad \Rightarrow$
$\forall v \in Dom\, t.\ P(t/v) \vee (v \mid w) \vee \exists v' \in Dom(t/v) \setminus \{\epsilon\}.\ \neg P(t/v/v'))$
Taking $v = w$, the last formula implies $P(t/w) \vee \exists v' \in Dom(t/w) \setminus \{\epsilon\}.\ \neg P(t/w/v'))$. Assume there is such a $v' \in Dom(t/w)$ with $\neg P(t/w/v')$. We obviously have $(w \cdot v') \in Dom\, t$. Because $t/w/v' = t/(w \cdot v')$ and because $\neg(w \mid w \cdot v')$, specialisation of the formula with $w \cdot v'$ gives us $\exists v'' \in Dom(t/(w \cdot v')) \setminus \{\epsilon\}.\ \neg P(t/(w \cdot v')/v''))$. We can repeat the argument with $w \cdot v' \cdot v''$, generating words of arbitrary length in $Dom\, t$. But $Dom\, t$ has finite depth — contradiction. Thus the assumption was wrong, i.e. $P(t/w)$ holds. $\square$

Throughout this paper, finiteness of terms is tacitly assumed. I made the finiteness assumption explicit in lemma 4.1, because its proof makes use of this assumption and indeed the lemma does not generalise to infinite terms. Other proofs in this paper do not directly depend on the assumption, but they may indirectly via lemma 4.1.

Lemma 4.2. *Let P be a predicate on terms. Then $\widehat{P}$ is a context selector.*

Proof. Let t be a term and $p, q \in Dom\, t$ with $p \prec q$. We have to show: $\neg(\overline{P}(t/p) \wedge \overline{P}(t/q))$. Assume $\overline{P}(t/p)$. We have $p \cdot p' = q$ for some $p' \neq \epsilon$ (because $p \prec q$) and so $p' \in Dom(t/p)$. By definition of $\overline{P}$ we have then $P(t/p/p')$, thus $P(t/q)$ and also $P(t/q/\epsilon)$. Thus $\neg\overline{P}(t/q)$. $\square$

Definition 4.5. *A pre-confluence P is called strong iff $P \vee \overline{P}$ is a pre-confluence and if the following property holds:*

$$\forall t \in Ter(\Sigma).\exists u \in Ter(\Sigma).\ \overline{P}(t) \ \Rightarrow\ (t \twoheadrightarrow_R u \wedge P(u))$$

Intuitively, a pre-confluence is some (preferably simple) property on terms which implies confluence and which is closed under reduction and taking subterms. The rôle of the predicate $\overline{P}$ is to serve as an induction step.

Definition 4.6. *Let P be subterm-closed. We define the ARS $\mathcal{P} = (\mathit{Ter}(\Sigma), \rhd_P)$ as follows: $t \rhd_P u$ iff $\widehat{P}(t) \neq \emptyset$ and if there is a function $F : t/\widehat{P}(t) \to \mathit{Ter}(\Sigma)$ such that the following properties hold:*

$$\begin{array}{l} \forall p \in \mathit{Dom}\, t.\ \widehat{P}(t) \mid p \ \Rightarrow\ u/p = t/p \\ \forall p \in \mathit{Dom}\, t.\ \widehat{P}(t) \succ p \ \Rightarrow\ u(p) = t(p) \\ \forall p \in \widehat{P}(t).\ t/p \twoheadrightarrow_R u/p = F(t/p) \wedge P(F(t/p)) \end{array}$$

We say "$t \rhd_P u$ with F" when we want to make F explicit.

Obviously, we have $\rhd_P\ \subseteq \twoheadrightarrow_R$ for any subterm-closed predicate P on terms. Without the condition $\widehat{P}(t) \neq \emptyset$, the relation $\rhd_P$ would be reflexive on terms satisfying P. In the given form, the normal forms of $\rhd_P$ are exactly the terms satisfying P, provided P is a strong pre-confluence (see below). The domain of F is $t/\widehat{P}(t)$ rather than $\widehat{P}(t)$ itself. This makes sure that $t/v = t/w$ implies $F(t/v) = F(t/w)$; for readers familiar with Toyama's proof: this condition has the same purpose as the $\propto$ relation in Toyama (1987).

Proposition 4.1. *Let P be subterm-closed. Then $\mathcal{P} \models$ SN.*

Proof. Consider a sequence $a_1 \rhd_P a_2 \rhd_P a_3 \cdots$. We assign to each a_i the set $V_i = \widehat{P}(a_i)$. Assume w.l.o.g. $v \in V_{i+1}$ (if V_{i+1} is empty, then $a_{i+1} \in \mathrm{NF}_{\mathcal{P}}$). We either have: (i) $V_i \mid v$ or (ii) $V_i \preceq v$ or (iii) $V_i \succ v$.

In case (i), it is clear that $a_i/v = a_{i+1}/v$. Lemma 4.1 gives us $P(a_i/v)$, thus $P(a_{i+1}/v)$. But we have also $\overline{P}(a_{i+1}/v)$ because $v \in V_{i+1}$, hence $\neg P(a_{i+1}/v)$. Contradiction.

Case (ii): let $y \in V_i$ with $y \preceq v$. We know $\overline{P}(a_i/y)$, hence $\neg P(a_i/y)$ by lemma 4.1. By definition of $a_i \rhd_P a_{i+1}$ it follows that $P(a_{i+1}/y)$. Because P is subterm-closed, we also have $P(a_{i+1}/v)$. But this means $\neg\overline{P}(a_{i+1}/v)$, i.e. $v \notin V_{i+1}$. Contradiction.

Thus only the third case applies. Because each V_i is finite, we can assign each a_i a natural number n_i, the sum of the lengths of all $v \in V_i$. The corresponding sequence of numbers is strictly decreasing. □

It is interesting that it was not even necessary to require that P is reduction-closed to obtain strong normalisation of $\rhd_P$. However, in some applications we need an even stronger property of $\rhd_P$ — it has to be "hyper-normalising", i.e. its termination should not be disturbed by intermediate $\to_R$ steps. To obtain this property, we need further conditions on $\to_R$ and P; we shall come to that later.

Proposition 4.2. *Let P be a strong pre-confluence. Then $\lhd_P ; \rhd_P \subseteq \twoheadrightarrow_R ; \twoheadleftarrow_R$.*

Proof. Let $s \rhd_P t$ with F_t and $s \rhd_P u$ with F_u. By definition of $\rhd_P$ we have:

$$\forall v \in \widehat{P}(s).\ F_t(s/v) \twoheadleftarrow_R s/v \twoheadrightarrow_R F_u(s/v).$$

Because P is a strong pre-confluence, $P \vee \overline{P}$ is a pre-confluence. For any $v \in \widehat{P}(s)$ we have $\overline{P}(s/v)$ and thus $\mathrm{CR}(s/v)$, i.e. $\forall v \in \widehat{P}(s). \exists g_v \in \mathrm{Ter}(\Sigma).\ F_t(s/v) \twoheadrightarrow_R g_v \twoheadleftarrow_R F_u(s/v)$

We can now define a term s' as follows:

$$\begin{aligned}&\forall v \in Dom\, s.\ \widehat{P}(s) \mid v \ \Rightarrow\ s'/v = s/v = t/v = u/v\\ &\forall v \in Dom\, s.\ \widehat{P}(s) \succ v \ \Rightarrow\ s'(v) = s(v) = t(v) = u(v)\\ &\forall v \in \widehat{P}(s).\ s'/v = g_v\end{aligned}$$

By compatibility we get $t \twoheadrightarrow_R s' \twoheadleftarrow_R u$.□

LEMMA 4.3. *Let P be a strong pre-confluence. Then: $\forall t \in Ter(\Sigma).\ P(t) \iff t \in \mathrm{NF}_{\mathcal{P}}$.*

PROOF. Let $\{p_1, \ldots, p_n\} = \widehat{P}(t)$.

"$\Rightarrow$" Assume $P(t)$. This implies $P(t/v)$ for all occurrences v in t, because P is subterm-closed. Consequently $\neg\overline{P}(t/v)$ by lemma 4.1, i.e. $\widehat{P}(t) = \emptyset$, violating one of the requirements for t to be in the domain of $\rhd_P$.

"$\Leftarrow$" Assume $\neg P(t)$. Thus $\widehat{P}(t) \neq \emptyset$. For any $t_i = t/p_i, 1 \leq i \leq n$, there is a u_i with $t_i \twoheadrightarrow_R u_i$ and $P(u_i)$ (follows immediately from the definition of strong pre-confluence). We can simply take $F(t_i) = u_i$. □

Notice that neither the proof of lemma 4.3 nor the proof of proposition 4.1 depend on the property that P implies confluence. One can use the same construction to prove other properties of rewriting systems.

Lemma 4.3 tells us that any term either satisfies P or is in the domain of $\rhd_P$. Proposition 4.1 also tells us that $\rhd_P$ is strongly normalising. Together this means that for every term t there is a term u which satisfies P and $t \twoheadrightarrow_R u$. This is almost but not quite enough for a confluence proof — theorem 4.1 makes this explicit.

4.2. A First Confluence Result

THEOREM 4.1. *Let P be a strong pre-confluence.*
If $\twoheadleftarrow_R; \rhd_P\ \subseteq \twoheadrightarrow_R; \twoheadleftarrow_R$ then $(Ter(\Sigma), \rightarrow_R) \models \mathrm{CR}$.

PROOF. Suppose $t \twoheadleftarrow_R s \twoheadrightarrow_R u$. If $s \in \mathrm{NF}_{\mathcal{P}}$ then $P(s)$ by lemma 4.3, i.e. CR(s). Otherwise there is an s' with $s \rhd_P s'$. Since $\rhd_P$ is SN (by proposition 4.1), we can assume CR(s') as induction hypothesis. The rest follows easily, see figure 1. □

Theorem 4.1 has a snag: it still requires to prove the property $\twoheadleftarrow_R; \rhd_P\ \subseteq \twoheadrightarrow_R; \twoheadleftarrow_R$ and the proofs of such properties are in general very similar to (that is: just as difficult as) confluence proofs. Typically one would attempt to prove the somewhat stronger property that $\rightarrow_R$ and $\rhd_P$ commute, which is sufficient because $\rhd_P\ \subset \twoheadrightarrow_R$. However, such an attempt is destined to fail as an $\rightarrow_R$ step can destroy the property of a term of being in the domain of $\rhd_P$. Instead we shall do the following: embed $\rhd_P$ in another relation $\blacktriangleright_P$, which still is contained in $\twoheadrightarrow_R$, and then show that this relation (almost) commutes with $\rightarrow_R$.

DEFINITION 4.7. *Let P be subterm-closed. We define two relations $\rightarrow_i$ and $\blacktriangleright_P$ on terms as follows:*

$$t \rightarrow_i u \overset{def}{\iff} P(t) \wedge t \twoheadrightarrow_R u \qquad \blacktriangleright_P \overset{def}{=} \rightarrow_i \cup \rhd_P$$

$$\begin{tikzcd}
 & u \arrow[rr, two heads, "R"] & & u' \arrow[dr, two heads, "R"] & \\
s \arrow[ur, two heads, "R"] \arrow[rr, "\rhd_P"] \arrow[dr, two heads, "R"'] & & s' \arrow[ur, two heads, "R"] \arrow[dr, two heads, "R"'] & \text{CR} & s'' \\
 & t \arrow[rr, two heads, "R"'] & & t' \arrow[ur, two heads, "R"'] &
\end{tikzcd}$$

Figure 1. Induction step of theorem 4.1

The goal is now to prove that $\blacktriangleright_P$ and $\twoheadrightarrow_R$ commute, preferably directly. For the $\rightarrow_i$ part, this is quite simple:

LEMMA 4.4. *Let P be a pre-confluence.* $(\mathit{Ter}(\Sigma), \rightarrow_i, \twoheadrightarrow_R) \models \text{CD}$.

PROOF. $u \twoheadleftarrow_R s \rightarrow_i t$ implies $P(s)$ by definition of $\rightarrow_i$. Since P is a pre-confluence, we have $\text{CR}(s)$, i.e. $u \twoheadrightarrow_R s' \twoheadleftarrow_R t$ for some s'. Because P is reduction-closed, we also have $P(u)$ and thus by definition 4.7 $u \rightarrow_i s'$. □

4.3. A SECOND CONFLUENCE RESULT

To eliminate the difficult condition $\twoheadleftarrow_R ; \rhd_P \subseteq \twoheadrightarrow_R ; \twoheadleftarrow_R$ from the confluence theorem and to replace it by properties that are easier to prove (if things go well), we need to consider the relative position of redexes in a span of the form $u \leftarrow_R s \rhd_P t$.

If $s \overset{v}{\rightarrow}_R u$, then v is either independent from all occurrences in $\widehat{P}(s)$, or one of them is a prefix of v, or v is a prefix of at least one of them. For each of these three cases we have a lemma, see below.

LEMMA 4.5. *Let P be a rewrite predicate and let s, t, u be terms with $u \overset{v}{\leftarrow}_R s \rhd_P t$ for some $v \in \mathit{Dom}\, s$. If $\forall w \in \widehat{P}(s).\ w \mid v$ then $\exists s' \in \mathit{Ter}(\Sigma).\ u \rhd_P s' \leftarrow_R t$.*

PROOF. We can choose s' to be $t[v = u/v]$. Since v is independent from $\widehat{P}(s)$, we have $u/v \leftarrow_R s/v = t/v$, i.e. $t \rightarrow_R s'$. Since for all $w \in \widehat{P}(s)$ we have $u/w = s/w$ and thus $\widehat{P}(s) \subseteq \widehat{P}(u)$. From lemma 4.2 we know that $\widehat{P}$ is a context selector, i.e. every $p' \in \widehat{P}(u) \setminus \widehat{P}(s)$ must be independent from $\widehat{P}(s)$. Assume there is such a p'.

Three cases: (i) If $p' \mid v$ then $s/p' = u/p'$ and lemma 4.1 gives us $P(s/p')$ and thus $\neg\overline{P}(u/p')$. Contradiction. (ii) If $p' \preceq v$ then $P(s/p')$ and $s/p' \rightarrow_R u/p'$. Because P is reduction-closed, we conclude $P(u/p')$. Contradiction. (iii) If $p' \succ v$ then $P(s/v)$; $P(u/v)$, because P is reduction-closed, and $P(u/p')$, because P is subterm-closed. Contradiction.

Therefore, the assumption was wrong and $\widehat{P}(s) = \widehat{P}(u)$. We conclude: $u \rhd_P s'$. □

LEMMA 4.6. *Let P be a strong pre-confluence and let s, t, u be terms with $u \overset{v}{\leftarrow}_R s \rhd_P t$ for some $v \in Dom\, s$. If $\exists w \in \widehat{P}(s).\ w \preceq v$ then $\exists s' \in Ter(\Sigma).\ u \blacktriangleright_P ; \twoheadrightarrow_R s' \twoheadleftarrow_R t$.*

PROOF. Because $\widehat{P}$ is a context selector (lemma 4.2), w is unique and $v \mid v'$ for all $v' \in \widehat{P}(s) \setminus \{w\}$. We have $\overline{P}(s/w)$ and $P(t/w)$ by definition of $\rhd_P$. We also have $s/w \rightarrow_R u/w$ by compatibility of $\rightarrow_R$. Because P is a strong pre-confluence, $P \vee \overline{P}$ is a pre-confluence, i.e. we know CR(s/w) and there has to be a term r with $t/w \twoheadrightarrow_R r \twoheadleftarrow_R u/w$. Because P is reduction-closed, we also have $P(r)$. $P \vee \overline{P}$ is also reduction-closed, i.e. $P(u/w) \vee \overline{P}(u/w)$.

Consider $U = \widehat{P}(u)$. We obviously have $\widehat{P}(s) \setminus \{w\} \subseteq U$. There are three cases: (i) $U = (\widehat{P}(s) \setminus \{w\}) \cup \{p\}$ for some $p \preceq w$ and (ii') $U = \widehat{P}(s) \setminus \{w\}$. In case (ii') we know $P(u/w)$ from lemma 4.1. We further split case (ii') into (ii) $U \neq \emptyset$ and (iii) $U = \emptyset$.

Case (i): we have $u \rhd_P u'$ for some u'. For all $q \in \widehat{P}(s) \setminus \{w\}$ there are common reducts c_q of t/q and u'/q by the pre-confluence property of $P \vee \overline{P}$. Similarly, we have $\overline{P}(u/p)$ and thus a common reduct r' of u'/p and $u/p[p' = r]$, where $w = p \cdot p'$. From this, we conclude $u \rhd_P u' \twoheadrightarrow_R s' = u'[p_1 = c_{p_1}, \ldots, p_n = c_{p_n}, p = r']$ and $t \twoheadrightarrow_R s'$.

Case(ii): since $U \neq \emptyset$ we have $\neg P(u)$ and by lemma 4.3 there is a u' with $u \rhd_P u'$. We also have $U \mid w$, hence $u/w = u'/w$. By compatibility we get $u' \twoheadrightarrow_R u'[w = r]$. Eventually, we make the same construction as in case (i), defining $s' = u'[p_1 = c_{p_1}, \ldots, p_n = c_{p_n}, w = r]$ with $U = \{p_1, \ldots, p_n\}$.

Case (iii): if $U = \emptyset$ then $\widehat{P}(s) = \{w\}$ and $t = s[w = t']$ for some term t'. As $v = w \cdot w'$ for some w', we also have $u = s[v = u'] = s[w = (s/v)[w' = u']]$ for some u'. Let $u'' = (s/v)[w' = u']$. We have now $t[w = r] = s[w = t'][w = r] = s[w = r] = s[w = u''][w = r] = s[w = r]$. We know $t \twoheadrightarrow_R t[w = r]$ and $u \twoheadrightarrow_R u[w = r]$ by compatibility. From $U = \widehat{P}(u) = \emptyset$ we also know $P(u)$. Thus $u \rightarrow_i u[w = r]$ and we have $u \blacktriangleright_P u[w = r] \twoheadrightarrow_R u[w = r] = t[w = r] \twoheadleftarrow_R t$. $\square$

For the remaining case, i.e. when the redex occurrence v of $s \overset{v}{\rightarrow}_R u$ is a proper prefix of at least one of the occurrences in $\widehat{P}(s)$, we cannot establish such a general result. For this, we need that $\rhd_P$ does not "destroy" outermost redexes.

DEFINITION 4.8. *Let P be a rewrite predicate. We call P redex-compatible, iff for all valuations σ and all rules $t \rightarrow r$ or $l \rightarrow t$ we have:*

$$\forall w \in \widehat{P}(\overline{\sigma}(t)).\ w = \epsilon \vee w \notin Dom\, t \vee t(w) \in \mathcal{V}$$

The important property of redex-compatibility is that rewrite rules do not "overlap" with $\rhd_P$ except at the root. Rewrite systems which only overlap at the root are called "overlay systems". Gramlich (1992) shows that a locally confluent (WCR) overlay system is complete (CR$\wedge$SN) iff it is innermost normalisable. The technique used here is related, as $\rhd_P$ can be seen as an innermost strategy to eliminate subterms that do not satisfy P; similarly, the purpose of many of the little lemmas in this section is to establish a WCR-like property for the combination of $\rightarrow_R$ and $\rhd_P$.

LEMMA 4.7. *Let P be a strong pre-confluence and redex-compatible. Let $\overline{\sigma}(l) = s \rhd_P t$ with $\widehat{P}(s) \neq \{\epsilon\}$ and let there be a rule $l \rightarrow r$. Then there is a substitution τ with $t = \overline{\tau}(l)$ and $\sigma(x) \twoheadrightarrow_R \tau(x)$ for all x in $\{l(v) \mid v \in Dom\, l \wedge l(v) \in \mathcal{V}\}$.*

PROOF. Let $s \rhd_P t$ with F. We define τ as follows:

$$\tau(l(v)) = s/v[w_1 = F(s/v \cdot w_1), \ldots, w_n = F(s/v \cdot w_n)] \qquad \text{if } v \in \mathit{Dom}\, l \wedge l(v) \in \mathcal{V}$$

where $\{w_1, \ldots, w_n\} = \{w \mid v \cdot w \in \widehat{P}(s)\}$.

We have to show that τ is well-defined, i.e. if $l(v) = l(v') \in \mathcal{V}$, then $\tau(l(v)) = \tau(l(v'))$. It is clear from the definition of $\rhd_P$ that $t = \overline{\tau}(l)$ if τ is well-defined (induction on occurrences of l) and that σ pointwise rewrites to τ.

If $l(v) = l(v') \in \mathcal{V}$ then $s/v = s/v'$ because $\overline{\sigma}(l) = s$. Since $\{w_1, \ldots, w_n\} = \widehat{P}(s/v)$, it is the same for v and v'. Similarly $F(s/v \cdot w_i) = F(s/v' \cdot w_i)$, because $s/v \cdot w_i = s/v' \cdot w_i$.□

LEMMA 4.8. *Let P be a strong pre-confluence and redex-compatible, and let s, t, u be terms with $u \overset{v}{\leftarrow}_R s \rhd_P t$ for some $v \in \mathit{Dom}\, s$. If $\exists z \in \widehat{P}(s).\ z \succ v$ then $\exists s' \in \mathit{Ter}(\Sigma).\ u \blacktriangleright_P ; \twoheadrightarrow_R s' \twoheadleftarrow_R t$.*

PROOF. We can show that $u/v \overset{\epsilon}{\leftarrow}_R s/v \rhd_P t/v$, because the premise of the lemma ensures $\neg P(s/v)$ and $\neg\overline{P}(s/v)$. From lemma 4.7 we know that $t/v = \overline{\tau}(l)$ and $s/v = \overline{\sigma}(l)$ and $u/v = \overline{\sigma}(r)$ for some valuation τ and some rule $l \rightarrow r$. Thus we have $t \rightarrow_R t' = t[v = \overline{\tau}(r)]$. We also have $u \twoheadrightarrow_R t'$.

If $u \in \mathrm{NF}_{\mathcal{P}}$ then $P(u)$ by lemma 4.3 and thus $u \rightarrow_i t'$ and $u \blacktriangleright_P t' = s'$.

Otherwise there is a u' with $u \rhd_P u'$. The terms u' and t' can only differ at occurrences $\widehat{P}(u')$ (and "underneath"). Consider $w \in \widehat{P}(u)$.

Case (i) $w \mid v$. We have $(P \vee \overline{P})(s/w)$ and the pre-confluence property gives us common reducts c_w for u'/w and $t'/w = t/w$.

Case (ii) $w \preceq v$ and $w \cdot w' = v$. We have $\overline{P}(u/w)$ and $u/w \twoheadrightarrow_R u'/w$. From $u/v \twoheadrightarrow_R \overline{\tau}(r)$ we get $u/w \twoheadrightarrow_R u/w[w' = \overline{\tau}(r)] = t/w[w' = \overline{\tau}(r)] = t'/w$. Because $\mathrm{CR}(u/w)$ we have a common reduct c_w of u'/w and t'/w.

Case (iii) $w \succ v$. We have $\overline{P}(u/w)$ and because P is subterm-closed $\neg(P \vee \overline{P})(u/v)$. Since $u/v = \overline{\sigma}(r)$ and because P is redex-compatible there has to be a variable occurrence $x \in \mathit{Dom}\, r$ with $\neg P(u/v \cdot x)$ and $v \cdot x \preceq w$; assume $v \cdot x \cdot w' = w$. Notice $u/v \cdot x = \sigma(r(x))$. There has to a variable occurrence $x' \in \mathit{Dom}\, l$ with $l(x') = r(x)$, because all variables of the right-hand side of a rewrite rule occur on the left as well. Then we have $u/v \cdot x = \sigma(r(x)) = \sigma(l(x')) = s/v \cdot x'$ and we know $\neg P(s/v \cdot x')$. Hence $v \cdot x' \cdot w' \in \widehat{P}(s)$. From $s/v \cdot x' = u/v \cdot x$ we get $u/w = s/v \cdot x' \cdot w'$ and thus $u/w \twoheadrightarrow_R u'/w$ and $u/w \twoheadrightarrow_R t/v \cdot x' \cdot w' = \overline{\tau}(l)/x' \cdot w' = \tau(l(x'))/w' = \tau(r(x))/w' = \overline{\tau}(r)/x \cdot w' = t'/v \cdot x \cdot w' = t'/w$. By $\mathrm{CR}(u/w)$ we have again a common reduct c_w of u'/w and t'/w. □

LEMMA 4.9. *Let P be a strong pre-confluence and redex-compatible. Let $t \leftarrow_R s \rhd_P u$. Then there is an s' such that $t \blacktriangleright_P ; \twoheadrightarrow_R s' \twoheadleftarrow_R u$.*

PROOF. Lemmas 4.5, 4.6 and 4.8 cover all cases of redex positions v in $s \overset{v}{\rightarrow}_R t$.□

DEFINITION 4.9. *Given an ordinal α, an α-weight (or just weight, suppressing the upper bound ordinal) is a function from $\mathit{Ter}(\Sigma)$ to $\{\beta \mid \beta < \alpha\}$.*

The purpose of weight functions is to provide an induction principle. For that matter, it is nice if they have the following property.

DEFINITION 4.10. *A weight s is called stable if $t \rightarrow_R u$ implies $s(t) \geq s(u)$.*

$$\begin{tikzcd}
s' \arrow[r, "R"] \arrow[dd, "\rhd_P"'] & s'' \arrow[r, two heads, "R"] \arrow[d, "\blacktriangleright_P"'] \arrow[dr, phantom, "\text{I.H. of lemma 4.10}"] & t' \arrow[d, two heads, "R"] \\
 & x \arrow[l, phantom, "\text{lemma 4.9}"] \arrow[r, two heads] \arrow[d, two heads, "R"'] \arrow[dr, phantom, "\text{I.H. of theorem 4.2}"] & \cdot \arrow[d, two heads, "R"] \\
u' \arrow[r, two heads, "R"'] & \cdot \arrow[r, two heads, "R"'] & \cdot
\end{tikzcd}$$

Figure 2. Lemma 4.10

Each proper context selector gives rise to a weight function, the maximal nesting depth (the "rank") of context layers. Toyama's proof for the confluence of the disjoint union of two confluent TRS is based on a context selector which always picks the next symbol of the other system. The rank function of this context selector is stable.

DEFINITION 4.11. *Let s be a weight and let P be subterm-closed. We say that s is innermost reduced by P iff $t \rhd_P u$ implies $s(t) > s(u)$.*

PROPOSITION 4.3. *Let P be subterm-closed, let s be a stable weight that is innermost reduced by P. Then $(Ter(\Sigma), \twoheadrightarrow_R; \rhd_P; \twoheadrightarrow_R) \models \mathrm{SN}$.*

PROOF. Trivial. □

THEOREM 4.2. *Let P be a strong pre-confluence and redex-compatible.*
Let $(Ter(\Sigma), \twoheadrightarrow_R; \rhd_P; \twoheadrightarrow_R) \models \mathrm{SN}$. Then $(Ter(\Sigma), \twoheadrightarrow_R) \models \mathrm{CR}$.

PROOF. We have to show that for any term s with $t \twoheadleftarrow_R s \twoheadrightarrow_R u$ we have $t \twoheadrightarrow_R; \twoheadleftarrow_R u$. Proof by induction on $\twoheadrightarrow_R; \rhd_P; \twoheadrightarrow_R$, i.e. we assume CR($s'$) for any s' with $s \twoheadrightarrow_R; \rhd_P; \twoheadrightarrow_R s'$. We extend this to $\twoheadrightarrow_R; \blacktriangleright_P; \twoheadrightarrow_R$, because anything in domain or codomain of $\rightarrow_i$ satisfies P and because P is reduction-closed and implies confluence. The base case (normal forms) is given by lemma 4.3 and the pre-confluence property of P.

We first prove the following auxiliary lemma, in which we use the induction hypothesis of the theorem; notice that the metavariable s occurs free in the lemma, i.e. it comes from the proof of theorem 4.2.

LEMMA 4.10. *For all s' with $s \twoheadrightarrow_R s'$ we have:*

$$\forall t', u' \in Ter(\Sigma).\ t' \twoheadleftarrow_R s' \blacktriangleright_P u' \ \Rightarrow t' \twoheadrightarrow_R; \twoheadleftarrow_R u'$$

PROOF. The case $s' \rightarrow_i u'$ is trivial (lemma 4.4). Proof by induction on the number n of $\rightarrow_R$ steps in $s' \twoheadrightarrow_R t'$. Base case ($n = 0$) is trivial, because $\rhd_P \subseteq \twoheadrightarrow_R$. Otherwise we have $s' \rightarrow_R s'' \twoheadrightarrow_R t'$.
We either have $s'' \rhd_P x$ which implies $s' \twoheadrightarrow_R; \rhd_P \twoheadrightarrow_R x$ and allows us to apply

the induction hypothesis of theorem 4.2 to x, as indicated in figure 2; or $s'' \rightarrow_i x$ which implies $P(x)$ and hence CR(x) by the pre-confluence property of P.
End of proof of lemma 4.10.$\Box$

Since $s \twoheadrightarrow_R s$, lemma 4.10 is applicable to s. We can use exactly the same argument as in the proof of theorem 4.1, see figure 1.
End of proof of theorem 4.2 $\Box$

Corollary 4.1. *Let P be a strong and redex-compatible pre-confluence. Let s be a stable weight which is innermost reduced by P. Then $(Ter(\Sigma), \rightarrow_R) \models \mathrm{CR}$.*

Proof. Follows immediately from theorem 4.2 and proposition 4.3.$\Box$

5. Instantiating the abstract proof

To exploit theorem 4.2 (or its corollary 4.1) for our problem that currying preserves confluence of TRSs, we have to find a predicate on terms that satisfies all the nice properties that make it a strong pre-confluence, etc. Therefore, this section consists mainly of (a few definitions and) a string of lemmas establishing these properties.

We set out to prove that the rewrite relation of the partial parameterised TRS PP(R) is confluent, provided $\rightarrow_R$ is confluent — a property which we will assume throughout this section. We will write $\rightarrow_{R'}$ for $\rightarrow_{PP(R)}$ and Σ' for PP(Σ). We interpret the definitions of section 4 using the rewrite relation $\rightarrow_{R'}$ and the signature Σ', e.g. "reduction-closed" means "reduction-closed w.r.t. $\rightarrow_{R'}$".

Recall the uncurrying system $\mathcal{U}$, i.e. the ARS associated with the subsystem of PP(R) containing only the rules $l \rightarrow' r$ with $l(\epsilon) = \texttt{Apply}$.

Definition 5.1. *We define a predicate* Noapp *on terms as follows:*

$$\mathrm{Noapp}(t) \overset{def}{\Longleftrightarrow} \forall u \in \mathrm{Ter}(\Sigma').\ t \twoheadrightarrow_{R'} u \Rightarrow u \in \mathrm{NF}_{\mathcal{U}}$$

The partial parameterised system PP(R) can behave like Cu(R) using the uncurrying rules. If these rules are not applicable for a term then reduction exclusively operates with the original rewrite system $\mathcal{R}$ (which is confluent) and all remaining occurrences of **`Apply`** are idle. To prove confluence, we have to show that the strategy of making all **`Apply`** occurrences idle succeeds and is cofinal in the reduction graph of a term.

Lemma 5.1. Noapp *is a pre-confluence.*

Proof. Noapp is subterm-closed: if $t/v \twoheadrightarrow_{R'} u' \notin \mathrm{NF}_{\mathcal{U}}$ for some $v \in Dom\, t$ then by compatibility $t \twoheadrightarrow_{R'} t[v = u'] \notin \mathrm{NF}_{\mathcal{U}}$.

Noapp is reduction-closed: if $t \twoheadrightarrow_{R'} u$ and $u \twoheadrightarrow_{R'} u' \notin \mathrm{NF}_{\mathcal{U}}$ then $t \twoheadrightarrow_{R'} u' \notin \mathrm{NF}_{\mathcal{U}}$.

Noapp implies confluence: if Noapp(s) and $t \twoheadleftarrow_{R'} s \twoheadrightarrow_{R'} u$ then $t \twoheadleftarrow_R s \twoheadrightarrow_R u$ by definition of Noapp. We have assumed that $\rightarrow_R$ is confluent on Ter(Σ); it remains confluent on

Ter(Σ'), because signature extensions preserve confluence[†]. Hence $t \twoheadrightarrow_R ; \twoheadleftarrow_R u$ and so $t \twoheadrightarrow_{R'} ; \twoheadleftarrow_{R'} u$.□

Lemma 5.2. *Let t be a term with* Noapp(t/i) *for all i with $1 \leq i \leq last(Dom\, t)$ and $t(\epsilon) \neq$* `Apply`*. Then* Noapp(t).

Proof. No rule has an occurrence of `Apply` on the right-hand side. Thus any potential $\rightarrow'$-redex must originate from an occurrence $v \in Dom\, t$ with $t(v) =$ `Apply`. But we assumed Noapp(t/v) for $v \neq \epsilon$ and $t(v) \neq$ `Apply` for $v = \epsilon$.□

Notice that lemma 5.2 does not generalise to Higher-Order Rewrite Systems (Nipkow, 1991). For instance, λ`x.Apply(x,y)` is in normal form, but by context application and rewriting in the context we may instantiate `x` to something that "activates" the `Apply` symbol.

Corollary 5.1. *Let* $\overline{\text{Noapp}}(t)$*. Then $t(\epsilon) =$* `Apply`*.*

Proof. Immediate consequence of lemma 5.2.□

Lemma 5.2 and its corollary may look rather simple, but their rôle in the entire proof is essential: if we apply one of the `Apply` rules to a term satisfying $\overline{\text{Noapp}}$ then the reduct must satisfy Noapp, because all its proper subterms are proper subterms of the redex (and therefore satisfy Noapp) and its root symbol is not `Apply`, so the lemma applies.

Definition 5.2. *We define a relation $\overset{\bullet}{\rightarrow}$ on terms as follows:*

$$t \overset{\bullet}{\rightarrow} u \overset{def}{\Longleftrightarrow} \exists v \in Dom\, t.\ t \overset{v}{\rightarrow}_{R'} u \wedge v \neq \epsilon$$

Lemma 5.3. *Let* $\overline{\text{Noapp}}(s)$ *and $t \overset{\bullet}{\twoheadleftarrow} s \overset{\bullet}{\twoheadrightarrow} u$. Then $\exists s'.\ t \overset{\bullet}{\twoheadrightarrow} s' \overset{\bullet}{\twoheadleftarrow} u$.*

Proof. For all i with $1 \leq i \leq last(Dom\, t)$ we have: $s/i \twoheadrightarrow_{R'} t/i$ and $s/i \twoheadrightarrow_{R'} u/i$. Since Noapp($s/i$) and Noapp is a pre-confluence (lemma 5.1), we have terms s'_i with $t/i \twoheadrightarrow_{R'} s'_i \twoheadleftarrow_{R'} u/i$ and can construct s' as $s'(\epsilon) = t(\epsilon) = u(\epsilon) = s(\epsilon)$ and $s'/i = s'_i$.□

Within this section we also abbreviate $\overset{\epsilon}{\rightarrow}_{R'}$ as $\overset{\epsilon}{\rightarrow}$.

Lemma 5.4. *Let* $\overline{\text{Noapp}}(t)$ *and $t \overset{\epsilon}{\rightarrow} u$. Then* Noapp($u$).

Proof. By corollary 5.1 we have $t(\epsilon) =$ `Apply`. Because $t \overset{\epsilon}{\rightarrow} u$, the applied rule $l \rightarrow r$ also has $l(\epsilon) =$ `Apply`, i.e. we have $l \rightarrow' r$. From this we know that all proper subterms of $u = \overline{\sigma}(r)$ are proper subterms of $t = \overline{\sigma}(l)$, i.e. Noapp(u/v) for all $v \in Dom\, u \setminus \{\epsilon\}$. Since $u(\epsilon) = r(\epsilon) \neq$ `Apply` we can apply lemma 5.2 and conclude Noapp(u).□

Lemma 5.5. *Let* $\overline{\text{Noapp}}(t)$ *and $t \overset{\bullet}{\rightarrow} u$. Then* $\overline{\text{Noapp}}(u)$*.*

[†] This is a special case of Toyama's theorem and a rather trivial property for TRSs. But see Plump (1993) and van Oostrom and van Raamsdonk (1993) for other rewrite formalisms for which this is not true.

Proof. From corollary 5.1 we have $t(\epsilon) = \texttt{Apply}$. Since $t \overset{\bullet}{\rightarrow} u$, we have $u(\epsilon) = \texttt{Apply}$ and $t/i \rightarrow^{=}_{R'} u/i$ with $\text{Noapp}(u/i)$ for $i \in \{1, 2\}$ as Noapp is reduction-closed. From $\overline{\text{Noapp}}(t)$ we know $\neg\text{Noapp}(t)$ (lemma 4.1), i.e. there exist terms t', t'' with $t \twoheadrightarrow_{R'} t' \rightarrow_U t''$. From lemma 5.4 we also know $t \overset{\bullet}{\twoheadrightarrow} t'$, as otherwise $\text{Noapp}(t')$ and $t' \in \text{NF}_{\mathcal{U}}$. From lemma 5.3 and the premise we find then a u' with $t' \overset{\bullet}{\twoheadrightarrow} u' \overset{\bullet}{\twoheadleftarrow} u$. From $t \overset{\bullet}{\twoheadrightarrow} t'$ we know $t'(\epsilon) = \texttt{Apply}$ and (for $i \in \{1, 2\}$) $t/i \twoheadrightarrow_{R'} t'/i$ and $\text{Noapp}(t'/i)$, because Noapp is reduction-closed. Since $t' \rightarrow_U t''$ and all proper subterms of t' are in $\text{NF}_{\mathcal{U}}$, we must have $t'(1) = F_k$ for some symbol $F_k \in A'$ (see definition 3.2). From $t' \overset{\bullet}{\twoheadrightarrow} u'$ we know $t'/1 \twoheadrightarrow_{R'} u'/1$. Since $F_k \neq l(\epsilon)$ for any rule $l \rightarrow r$ or $l \rightarrow' r$, we also have $t'/1 \overset{\bullet}{\twoheadrightarrow} u'/1$. Thus u' has the form $\texttt{Apply(}F_k\texttt{(}b_1\texttt{,}\ldots\texttt{,}b_k\texttt{),}b_{k+1}\texttt{)}$ which is not in $\text{NF}_{\mathcal{U}}$.□

The last two lemmas show that $\text{Noapp} \vee \overline{\text{Noapp}}$ is reduction-closed, which should not be too surprising. They also show something stronger, e.g.: if $t \twoheadrightarrow_{R'} u$ and both t and u satisfy $\overline{\text{Noapp}}$, then $t \overset{\bullet}{\twoheadrightarrow} u$. This means: it is not possible to destroy an (innermost) `Apply`-redex by inner reduction.

Lemma 5.6. *Let* $\overline{\text{Noapp}}(s)$ *and* $t \overset{\bullet}{\twoheadleftarrow} s \xrightarrow{\epsilon} u$. *Then there is an* s' *such that* $t \xrightarrow{\epsilon} s' \overset{\bullet}{\twoheadleftarrow} u$.

Proof. Corollary 5.1 gives us $s(\epsilon) = \texttt{Apply}$ and $\text{Noapp}(s/i), i \in \{1, 2\}$. Since $s \xrightarrow{\epsilon} u$ we have $s(1) = F_k$ for some $F_k \in A'$. From $s \overset{\bullet}{\twoheadrightarrow} t$ we have $s/1 \twoheadrightarrow_{R'} t/1$ and since $s(1) = F_k \neq l(\epsilon)$ for any left-hand side l of a rule, we also have $s/1 \overset{\bullet}{\twoheadrightarrow} t/1$; thus t has the form $\texttt{Apply(}F_k\texttt{(}t/1\cdot 1\texttt{,}\ldots\texttt{,}t/1\cdot k\texttt{),}t/2\texttt{)}$. Let $G = u(\epsilon)$ (either $G = F_{k+1}$ or $G = F$), then $t \xrightarrow{\epsilon} s' = G(t/1\cdot 1\texttt{,}\ldots\texttt{,}t/1\cdot k\texttt{,}t/2)$. Similarly we have $u = G(s/1\cdot 1\texttt{,}\ldots\texttt{,}s/1\cdot k\texttt{,}s/2)$. From $s/1 \overset{\bullet}{\twoheadrightarrow} t/1$ and $s \overset{\bullet}{\twoheadrightarrow} t$ we conclude $u \overset{\bullet}{\twoheadrightarrow} s'$.□

Lemma 5.7. *Let* $\overline{\text{Noapp}}(t)$ *and* $t \xrightarrow{\epsilon} u$ *and* $t \xrightarrow{\epsilon} u'$. *Then* $u = u'$.

Proof. From corollary 5.1 we know $t(\epsilon) = \texttt{Apply}$. Hence we have applied rules from $\rightarrow'$. Since the left-hand sides of different $\rightarrow'$ rules are not unifiable, we have applied the same rule. □

Lemma 5.8. *Let* $\overline{\text{Noapp}}(t)$. *There is a term* u *with* $t \overset{\bullet}{\twoheadrightarrow}; \xrightarrow{\epsilon} u$ *and* $\text{Noapp}(u)$.

Proof. From $\overline{\text{Noapp}}(t)$ we know that there exists a t' with $t \twoheadrightarrow_{R'} t' \rightarrow_U t''$. From lemma 5.4 we know $t \overset{\bullet}{\twoheadrightarrow} t'$ and from lemma 5.5 $\overline{\text{Noapp}}(t')$. Thus all proper subterms of t' are in $\text{NF}_{\mathcal{U}}$ and we must have $t' \xrightarrow{\epsilon} t''$. By lemma 5.4 $\text{Noapp}(t'')$ and we can choose $u = t''$.□

Lemma 5.9. *Let* $\overline{\text{Noapp}}(s)$. *Then* $\text{CR}(s)$.

Proof. Suppose $t \twoheadleftarrow_{R'} s \twoheadrightarrow_{R'} u$. Because of lemmas 5.5 and 5.8 it is sufficient to consider the case $t \twoheadleftarrow_{R'}; \xleftarrow{\epsilon}; \overset{\bullet}{\twoheadleftarrow} s \overset{\bullet}{\twoheadrightarrow}; \xrightarrow{\epsilon}; \twoheadrightarrow_{R'} u$ (otherwise $s \overset{\bullet}{\twoheadrightarrow} t$ (or u) and we can extend the span by lemma 5.8 to $s \overset{\bullet}{\twoheadrightarrow} t \overset{\bullet}{\twoheadrightarrow}; \xrightarrow{\epsilon} t'$ and reduce it to the mentioned case). See Figure 3.

The argument for each of the single rectangles is given as follows: (i) is lemma 5.3. (ii) follows from lemmas 5.5 and 5.6. (iii) follows from lemmas 5.5 and 5.7. Finally, (iv) follows from lemmas 5.5, 5.4 and the pre-confluence property of Noapp, i.e. lemma 5.1.□

Lemma 5.10. Noapp *is a strong pre-confluence.*

$$\begin{tikzcd}
 & & & s \arrow[dl, "\bullet"'] \arrow[dr, "\bullet"] & & & \\
 & & \cdot \arrow[dl, "\epsilon"'] \arrow[dr, "\bullet"] & \text{i} & \cdot \arrow[dl, "\bullet"'] \arrow[dr, "\epsilon"] & & \\
 & \cdot \arrow[dl, "R'"'] \arrow[dr, "\bullet"] & \text{ii} & \cdot \arrow[dl, "\epsilon"'] \arrow[dr, "\epsilon"] & \text{ii} & \cdot \arrow[dl, "\bullet"'] \arrow[dr, "R'"] & \\
t \arrow[ddrr, "R'"'] & & \cdot \arrow[dr, "="] & \text{iii} & \cdot \arrow[dl, "="'] & & u \arrow[ddll, "R'"] \\
 & \text{iv} & & \cdot \arrow[dl, "R'"'] \arrow[dr, "R'"] & & \text{iv} & \\
 & & \cdot \arrow[dr, "R'"'] & \text{iv} & \cdot \arrow[dl, "R'"] & & \\
 & & & \cdot & & &
\end{tikzcd}$$

Figure 3. Main case of lemma 5.9

PROOF. $\underline{\text{Noapp}}$ is a pre-confluence (lemma 5.1) and since $\xrightarrow{\epsilon} \cup \xrightarrow{\bullet} = \longrightarrow_{R'}$, we have that $\underline{\text{Noapp}} \vee \overline{\text{Noapp}}$ is reduction-closed from lemmas 5.4 and 5.5. It is subterm-closed, because $\underline{P} \vee \overline{P}$ is subterm-closed for any subterm-closed P. That $\underline{\text{Noapp}} \vee \overline{\text{Noapp}}$ implies confluence has been shown in lemmas 5.1 and 5.9. Lemma 5.8 gives us the last required property.$\square$

LEMMA 5.11. Noapp *is redex-compatible.*

PROOF. Suppose t is left-hand or right-hand side of a rule in $\text{PP}(R)$. Let σ be a valuation. Consider an arbitrary $w \in \widehat{Noapp}(\overline{\sigma}(t))$. From definition of $\widehat{P}$ we know $\overline{\text{Noapp}}(t/w)$. But then we have by corollary 5.1 $\overline{\sigma}(t)(w) = \texttt{Apply}$. Since `Apply` only occurs as outermost symbol of rules (in left-hand sides of $\longrightarrow'$) it can only be an occurrence in t if either $w = \epsilon$ or if $t(w) \in \mathcal{V}$ and $\sigma(t(w))(\epsilon) = \texttt{Apply}$.$\square$

DEFINITION 5.3. *Let t be a term and $v \in Dom\, t$. We write $|t|_v$ for* $\mathrm{card}\{w \mid w \preceq v, t(w) = \mathtt{Apply}\}$. *We define* $|\cdot| : \mathrm{Ter}(\Sigma') \rightarrow \mathcal{N}$ *as follows:*

$$|t| \overset{def}{=} \max\{|t|_w \mid w \in \widehat{\mathrm{Noapp}}(t)\}$$

In the above definition, card is the function assigning a set its cardinality. Thus $|\cdot|$ assigns a term the maximal number of occurrences of `Apply` which prefix its subterms satisfying $\overline{\mathrm{Noapp}}$.

It is worth noticing that the above construction (and the forthcoming lemmas) can be made subject to further generalisations. For instance, one can think of the above $|t|$ as a modification of a simpler weight function, which only requires $w \in Dom\, t$ rather than $w \in \widehat{\mathrm{Noapp}}(t)$. It it not difficult to see that such a modification preserves in general the stability of a weight, provided the modifying predicate is subterm-closed.

LEMMA 5.12. *Let $l \rightarrow r$ or $l \rightarrow' r$ be a rule and σ a valuation. Then* $|\overline{\sigma}(l)| \geq |\overline{\sigma}(r)|$.

PROOF. For all rules $l \rightarrow r$ we have $|\overline{\sigma}(l)| = \max\{|\sigma(l(x))| \mid x \in Dom\, l, l(x) \in \mathcal{V}\}$, analogously for r. Since all variables occurring in r also occur in l, the result follows.

Let $l \rightarrow' r$. Clearly $\neg\mathrm{Noapp}(\overline{\sigma}(l))$. If $\overline{\mathrm{Noapp}}(\overline{\sigma}(l))$ then $\mathrm{Noapp}(\overline{\sigma}(r))$ by lemma 5.4, and $|\overline{\sigma}(l)| = 1 > 0 = |\overline{\sigma}(r)|$. Otherwise $|\overline{\sigma}(l)| = |\overline{\sigma}(r)| + 1$, because $\overline{\sigma}(l)(\epsilon) = l(\epsilon) = \mathtt{Apply}$ and ϵ is in the prefix of any $v \in \widehat{\mathrm{Noapp}}(\overline{\sigma}(l))$.□

LEMMA 5.13. $|\cdot|$ *is a stable ω-weight.*

PROOF. Suppose $t \xrightarrow{v}_{R'} u$. We know from lemma 5.12 that $|t/v| \geq |u/v|$. If $\mathrm{Noapp}(t/v)$ then $\mathrm{Noapp}(u/v)$ and $|t| = |u|$. Otherwise v is a prefix of an occurrence $w \in \widehat{\mathrm{Noapp}}(t)$. We have $|t| = \max\{|t|_w \mid w \in \widehat{\mathrm{Noapp}}(t)\} = \max(\{|t|_v + |t/v|\} \cup \{|t|_w \mid w \in \widehat{\mathrm{Noapp}}(t), w \mid v\}) = \max(\{|u|_v + |t/v|\} \cup \{|u|_w \mid w \in \widehat{\mathrm{Noapp}}(u), w \mid v\}) \geq \max(\{|u|_v + |u/v|\} \cup \{|u|_w \mid w \in \widehat{\mathrm{Noapp}}(u), w \mid v\}) \geq |u|$□

LEMMA 5.14. $|\cdot|$ *is innermost reduced by* Noapp.

PROOF. Suppose $t \rhd_{\mathrm{Noapp}} u$. From corollary 5.1 we know $t(v) = \mathtt{Apply}$ for all $v \in \widehat{\mathrm{Noapp}}(t)$. Moreover, it is easy to check that $|t/v| = 1$ and $|u/v| = 0$. Thus $|t| = \max\{|t|_v \mid v \in \widehat{\mathrm{Noapp}}(t)\} = \max\{|t|_{v'} + 1 \mid v \in \widehat{\mathrm{Noapp}}(t), v' \prec v\} > \max\{|t|_{v'} \mid v \in \widehat{\mathrm{Noapp}}(t), v' \prec v\} = \max\{|u|_{v'} \mid v \in \widehat{\mathrm{Noapp}}(t), v' \prec v\} \geq \max\{|u|_v \mid v \in \widehat{\mathrm{Noapp}}(u)\} = |u|$.□

We have now instantiated all preconditions we need to be able to apply theorem 4.2 or its corollary 4.1.

THEOREM 5.1. *Let* $(Ter(\Sigma), \rightarrow_R) \models \mathrm{CR}$. *Then* $(\mathrm{Ter}(\Sigma'), \rightarrow_{R'}) \models \mathrm{CR}$.

PROOF. Follows directly from corollary 4.1 and lemmas 5.10, 5.11, 5.13 and 5.14.□

THEOREM 5.2. *Let R be a confluent TRS. Then* $\mathrm{Cu}(R)$ *is confluent.*

PROOF. Follows directly from theorem 5.1 and proposition 3.1.□

6. Related work

Breazu-Tannen and Gallier (1989) show how several properties (including CR) are preserved by currying, provided the TRS in question is non-collapsing. Kennaway, Klop, Sleep, and de Vries (1993) go a step further: they show that currying preserves SN and SN $\wedge$ CR for arbitrary TRSs.

However, the latter paper also shows that currying does in general not preserve confluence — clearly the opposite of what is stated in theorem 5.2. They even construct a counter-example, an infinitary TRS which is CR, but for which the curried system does not even satisfy the $\mathrm{UN}^{\rightarrow}$ property (uniqueness of normal forms). "Allegedly" I hasten to add, because the proof of lemma 6.9 in their paper has a major (unrepairable) flaw. I briefly repeat their construction here to show what went wrong. Apart from underlining the correctness of the proof for theorem 5.2, this may give some useful insights into the nature of currying.

The disjoint union of applicative TRSs is known not to preserve confluence. Applicative TRSs have a binary symbol `Apply` (shared among all ATRSs) and all other symbols have arity 0. The symbol `Apply` is notationally suppressed using the conventions of λ-calculus and Combinatory Logic (Hindley and Seldin, 1986). Consider the following ATRS M:

$$\begin{aligned} \mathtt{M\ x\ x} &\rightarrow \mathtt{0} \\ \mathtt{M\ (Succ\ x)\ x} &\rightarrow \mathtt{1} \end{aligned}$$

M is a confluent ATRS, because (as a TRS) it has no critical pairs and it is terminating. If we combine it with the rewrite system of Combinatory Logic (which is confluent in itself), we lose confluence. The ATRS of combinatory logic is the following system CL:

$$\begin{aligned} \mathtt{I\ x} &\rightarrow \mathtt{x} \\ \mathtt{K\ x\ y} &\rightarrow \mathtt{x} \\ \mathtt{S\ x\ y\ z} &\rightarrow \mathtt{(x\ z)\ (y\ z)} \end{aligned}$$

Within CL, one can express a fixpoint combinator, i.e. a term `Y` that has the property $\mathtt{Y\ x} \twoheadrightarrow \mathtt{x\ (Y\ x)}$. In the combined system, we can rewrite the term `M (Y Succ) (Y Succ)` to the normal form `0` but also to the normal form `1`.

Apart from the (implicit) symbol `Apply`, M and CL do not share any symbols. M is in the image of Cu, but CL is not, because of the rule for the `S` combinator. The paper defines a TRS CL^{fun} whose image under Cu is very similar to CL. This system has an infinite signature and infinitely many rules of the following forms:

$$\begin{aligned} &\mathtt{I}_{n+1}(\alpha_k(\mathtt{x}_1,\ldots,\mathtt{x}_k),\mathtt{y}_1,\ldots,\mathtt{y}_n) \rightarrow \alpha_{k+n}(\mathtt{x}_1,\ldots,\mathtt{x}_k,\mathtt{y}_1,\ldots,\mathtt{y}_n) \\ &\mathtt{K}_{n+2}(\alpha_k(\mathtt{x}_1,\ldots,\mathtt{x}_k),\mathtt{z},\mathtt{y}_1,\ldots,\mathtt{y}_n) \rightarrow \alpha_{k+n}(\mathtt{x}_1,\ldots,\mathtt{x}_k,\mathtt{y}_1,\ldots,\mathtt{y}_n) \\ &\mathtt{S}_{n+3}(\alpha_k(\mathtt{x}_1,\ldots,\mathtt{x}_k),\beta_m(\mathtt{v}_1,\ldots,\mathtt{v}_m),\mathtt{z},\mathtt{y}_1,\ldots,\mathtt{y}_n) \rightarrow \\ &\qquad \alpha_{k+n+2}(\mathtt{x}_1,\ldots,\mathtt{x}_k,\mathtt{z},\beta_{m+1}(\mathtt{v}_1,\ldots,\mathtt{v}_m,\mathtt{z}),\mathtt{y}_1,\ldots,\mathtt{y}_n) \end{aligned}$$

The metavariables α and β range over $\{\mathtt{S},\mathtt{K},\mathtt{I}\}$ and some infinite set of other symbols which are in one-to-one correspondence to the set of variables $\mathcal{V}$. The system CL^{fun} performs partial application explicitly for all symbols in its signature. CL^{fun} can be shown to be confluent. The alleged counter-example is then constructed by disjointely adding an uncurried version of M and CL^{fun} (which is confluent by Toyama's theorem) and then currying the result. This result appears to be more or less the same as $M+\mathrm{CL}$, which is not confluent.

But this does not work. The giveaway was the remark that CL^{fun} "performs partial application for all symbols *in its signature*". This excludes the signature of M, in particular it excludes partial parameterisation of `Succ`. For instance, the signature of $M + \mathrm{Cu}(\mathrm{CL}^{fun})$ allows the term `(`$\mathtt{I}_2$ `Succ x)` which is in normal form, while the corresponding term `(I Succ x)` in $M + \mathrm{CL}$ rewrites to `(Succ x)`. Another clue why this construction cannot work is the syntactic shape of M and CL^{fun}: neither the uncurried version of M nor CL^{fun} has any collapsing rules — but in the absence of collapsing rules there are *much* simpler proofs for the preservation of confluence by currying. One could argue that there are simply collapsing rules missing for $\mathtt{K}_2$ and $\mathtt{I}_1$, but instead of using Combinatory Logic, it would have been sufficient for the argument in the "counter-example" to start from the (confluent) ATRS

$$\mathtt{Y\ x} \quad \rightarrow \quad \mathtt{x\ (Y\ x)}$$

and construct analogously to CL^{fun} a TRS $\mathtt{Y}^{fun}$ which is clearly non-collapsing anyway.

The mentioned paper used this "proof" not only for the wrong result about the non-preservation of confluence, but also to claim the non-preservation of NF, UN, and $\mathrm{UN}^{\rightarrow}$, i.e. these claims become invalid as well. In their updated version, Kennaway et.al. (1995) show that NF and $\mathrm{UN}^{\rightarrow}$ are indeed not preserved by giving proper counter-examples; on the other hand, they show that currying does preserve UN by reducing the problem to the preservation of confluence. Interestingly, all properties that have been investigated so far are preserved by currying if and only if they are preserved by signature extensions.

7. Conclusion

We have shown (theorem 5.2) that currying preserves confluence for arbitrary TRSs. The method we used should carry over to other confluence proofs for TRSs — we stated a confluence theorem (4.2) independent from the particular systems we were interested in; Toyama's proof (1987) is expressible as an application of this abstract theorem.

A remaining open problem is whether currying preserves confluence of HRSs (Nipkow, 1991; van Oostrom and van Raamsdonk, 1993), since at least the very useful lemma 5.2 does not hold in this generalised setting. Similar problems exist with the method employed by Kennaway et.al. (1993) to prove the preservation of SN. One may think that HRSs support currying anyway, but this is only true for the simply-typed variety. It is possible to express (polymorphic) currying uniformly in HRS by adding a rule

$$\mathtt{Apply}(\lambda x.t\ x, u) \rightarrow t\ u$$

but this is nothing else but β-reduction of untyped λ-calculus, which (even if disjointly added) can destroy confluence of an HRS and does destroy its termination. Thus, general positive results for currying of HRSs have to rely on a non-uniform method of currying, in the style of explicit partial parameterisation.

Acknowledgements

Thanks to an observation by Richard Kennaway the proof of lemma 4.10 is significantly simpler than in an earlier version of this paper. Thanks also to the referee for spotting some (fortunately minor) mistakes.

The research reported here was partially supported by SERC grants GR/E 78463 and GR/J 07303.

References

Brainerd, W.S. (1969). Tree generating regular systems. *Information and Control* **14**, 217–231.

Breazu-Tannen V., Gallier, J. (1989). Polymorphic rewriting conserves algebraic strong normalization and confluence. In *Proceedings 16th International Colloquium on Automata, Languages and Programming.* Springer LNCS 372, 137–150.

Gallier, J. (1994). On the correspondence between proofs and λ-terms. In de Groote, P., editor, *The Curry-Howard Isomorphism*, Cahiers du Centre de Logique. Université Catholique de Louvain, 1994. To appear.

Girard, J.-Y., Lafont, Y., Taylor, P. (1989). *Proofs and Types.* Cambridge University Press.

Gramlich, B. (1992). Relating innermost, weak, uniform and modular termination of rewriting. In *Proceedings Logic Programming and Automated Reasoning*, Springer LNCS 624, 285–296.

Hindley, J.R. (1969). An Abstract Form of the Church-Rosser Theorem I. *Journal of Symbolic Logic* **34**, 545–560.

Hindley, J.R., Seldin, J.P. (1986). *Introduction to Combinators and λ-Calculus.* Cambridge University Press.

Huet, G. (1980). Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM* **27**, 797–821.

Kahrs, S. (1991). *λ-rewriting.* PhD thesis, Universität Bremen. In German.

Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.-J. (1993). Comparing curried and uncurried rewriting. Technical Report CS-R9350, Centrum voor Wiskunde en Informatica.

Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.-J. (1995). Comparing curried and uncurried rewriting. To appear in: Journal of Symbolic Computation.

Klop, J.W. (1992). Term rewriting systems. In Abramsky, S., Gabbai, D.M., Maibaum, T.S.E., editors, *Handbook of Logic in Computer Science, Volume 2*, 1–116. Oxford University Press.

Klop, J.W., Middeldorp, A., Toyama, Y., de Vrijer, R. (1991). A simplified proof of Toyama's theorem. Technical Report CS-R9156, Centrum voor Wiskunde en Informatica.

Lloyd, J.W.. (1984). *Foundations of Logic Programming.* Springer.

Meinke, K., Tucker, J.V. (1992). Universal algebra. In Abramsky, S., Gabbay, D.M., Maibaum, T.S.E., editors, *Handbook of Logic in Computer Science, Volume 1*, 189–411. Oxford University Press.

Nipkow, T. (1991). Higher order critical pairs. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, 342–349.

van Oostrom, V., van Raamsdonk, F. (1993). Comparing combinatory reduction systems and higher-order rewrite systems. In *Proceedings Higher-Order Algebra, Logic and Term Rewriting.* Springer LNCS 816, 276–304.

Plump, D. (1993). *Evaluation of Functional Expressions by Hypergraph Rewriting.* PhD thesis, Universität Bremen.

Rosen, B.K. (1973). Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM* **20**, 160–187.

Staples, J. (1975). Church-Rosser theories for replacement systems. In Crossley, J.N., editor, *Algebra and Logic*, 291–307. Springer LNM 450.

Toyama, Y. (1987). On the Chuch-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM* **34**, 128–143.