Chain Properties of Rule Closures

Miki HERMANN
Centre de Recherche en Informatique de Nancy
CNRS and INRIA-Lorraine
Campus Scientifique, BP 239,
54506 Vandœuvre-les-Nancy, France

e-mail: hermann@loria.crin.fr

Abstract

This article presents an introduction to the generalization of the crossed rule approach to the detection of Knuth–Bendix completion procedure divergence. It introduces the closure chains, which are special rule closures constructed by means of particular substitution operations and operators, as a suitable formalism for a progress in this direction. Supporting substitution algebra is developed first, followed by considerations concerning rule closures in general, and concluded by investigation of closure chain properties.

1 Introduction

In [4] the importance of forward closures and later also backward closures for the definition of *crossed rules* has been discovered. During the attempts to generalize the crossed rule notion it became apparent that their construction is strictly dependent on *special* forward or backward rule closures, called forward/backward *chains*. These closure chains are derived from rule closures by substitution variable constraints, and are constructed by means of special substitution operations and iterative operators. It is for this iterative construction way of the closure chains, that makes them most suitable for the description of the Knuth–Bendix procedure divergence. Therefore the attention is focused on a sufficiently general definition with the potential to describe the largest class of possible cases by their structural entities.

2 Basic notation and definitions

2.1 Term rewriting systems

Let F be a finite or enumerable set of *function symbols* graded by arity (signature). F_0 denotes the set of constants. Let X be an enumerable set of *variables* such that $F \cap X = \emptyset$. Denote $T(F, X)$ the set of all *terms* (free algebra) over variables X and symbols F. The terms containing no variables are called *ground terms*. $Var(t)$ denotes the set of all variables in the term t.

Let N^* be a set of strings of natural numbers with a special symbol $\varepsilon \in N^*$ for the empty string. Suppose the existence of string concatenation operation on N^*. Using the elements of N^* as labels, the terms can be viewed as labeled trees. A term t is a partial function $N^* \to F \cup X$ such that its domain $D(t)$ satisfies the following properties: if $t \in F_0 \cup X$ then $D(t) = \{\varepsilon\}$; if $t = f(t_1, \ldots, t_n)$ then $D(t) = \{\varepsilon\} \cup \{i.a \mid i = 1, \ldots, n \text{ and } a \in D(t_i)\}$. $D(t)$ is the set of *occurrences* of the term t. The subset of non-variable occurrences of t is denoted as $D_s(t)$.

A *subterm* of t at an occurrence $a \in D(t)$ is denoted t/a. If $t = f(t_1, \ldots, t_n)$ then $t/\varepsilon = t$ and $t/ia = t_i/a$ for all $i = 1, \ldots, n$. Denote $s[a \leftarrow t]$ a new term obtained from the term s by replacing its subterm s/a by t. For properties of replacement see the article [8].

the variables x_i and $x_i \neq t_i$, for $i = 1, \ldots, n$. Substitutions have a homomorphic extension on terms. A substitution is *ground* if, and only if, its range is a set of ground terms. A substitution σ, such that $\sigma(x) \in X$ for all $x \in Lvar(\sigma)$, is a *variable renaming*. Substitutions need not to be idempotent in our approach.

Two terms s and t are *unifiable* if, and only if, there is an idempotent substitution σ such that $\sigma s = \sigma t$. The substitution σ is called a *unifier*. The substitution σ is called the *most general unifier* for s and t if there are no substitutions φ and ψ such that $\sigma = \psi.\varphi$ and, $\varphi s = \varphi t$, and ψ is not a variable renaming.

The sets $Lvar(\varphi)$, $Rvar(\varphi)$, $Var(\varphi)$ denote the domain, range, and all variables of the substitution φ, respectively. For iteration reasons, standard unification with variable renaming cannot take place, therefore the substitution in *own variables* must be defined. The substitution σ on term t is a *substitution in own variables* of t if it does not introduce new variables, i.e. $Var(\sigma t) \subseteq Var(t)$, and does not contain a variable renaming. This notion can be enlarged to a set of substitutions.

A *rewrite rule* is an ordered pair of terms $r = (s, t)$ (written $s \to t$) such that $Var(t) \subseteq Var(s)$. A *term rewriting system* is a finite set of rules R. A term rewriting system R is said to be *variable preserving*, if for all rules $(s \to t) \in R$, $Var(s) = Var(t)$. The term rewriting system constructed from a variable preserving rewrite system R by turning the rules in the opposite direction is the set of rules $R^{op} = \{t \to s \mid s \to t \in R\}$, called the *opposite* term rewriting system to R.

2.2 Rule closures

The following constructions come from Lankford and Musser [6] and also from Guttag, Kapur and Musser [2]. They are mentioned also by Dershowitz [1].

Definition 2.1 *Let R be an arbitrary set of rules.*

The set of **forward closures** *$FC(R)$ of R is inductively defined as follows:*

1. *Every rule $s \to t$ from R is a forward closure $s \rightarrowtriangle t$.*
2. *Let $s_1 \rightarrowtriangle t_1$, $s_2 \rightarrowtriangle t_2$ be two forward closures and $\sigma t_1/a = \sigma s_2$ holds for a most general unifier σ, and occurrence $a \in D_s(t_1)$, then $\sigma s_1 \rightarrowtriangle \sigma t_1[a \leftarrow \sigma t_2]$ is a forward closure.*

The set of **backward closures** *$BC(R)$ of R is inductively defined as follows:*

1. *Every rule $s \to t$ from R is a backward closure $s \triangleleftarrow t$.*
2. *Let $s_1 \triangleleftarrow t_1$, $s_2 \triangleleftarrow t_2$ be two backward closures and $\sigma t_1 = \sigma s_2/a$ holds for a most general unifier σ and occurrence $a \in D_s(s_2)$, then $\sigma s_2[a \leftarrow \sigma s_1] \triangleleftarrow \sigma t_2$ is a backward closure.*

The set of **overlap closures** *$OC(R)$ of R is inductively defined as follows:*

1. *Every rule $s \to t$ from R is an overlap closure $s \mapsto t$.*
2. *Let $s_1 \mapsto t_1$, $s_2 \mapsto t_2$ be two overlap closures and $\sigma t_1/a = \sigma s_2$ holds for a most general unifier σ, and occurrence $a \in D_s(t_1)$, then $\sigma s_1 \mapsto \sigma t_1[a \leftarrow \sigma t_2]$ is an overlap closure.*
3. *Let $s_1 \mapsto t_1$, $s_2 \mapsto t_2$ be two overlap closures and $\sigma t_1 = \sigma s_2/a$ holds for a most general unifier σ and occurrence $a \in D_s(s_2)$, then $\sigma s_2[a \leftarrow \sigma s_1] \mapsto \sigma t_2$ is an overlap closure.*

3 Substitution operations and operators

In the first part, two substitution operations — the sum and product — are defined, and successively their properties are proved. Upon the sum operation three other substitution operators (the exponent, W– and T–operator) have been built. These three operators are defined as iterations of the substitution sum on different basic level substitutions.

- *The* **sum** *of φ and ψ is the substitution $\psi \triangle \varphi = [x \leftarrow \psi(\varphi x) \mid x \in Lvar(\varphi), \psi(\varphi x) \neq x]$.*
- *The* **product** *of φ and ψ is the substitution $\psi \circ \varphi = (\psi \triangle \varphi) \cup [x \leftarrow \psi(x) \mid x \in Lvar(\psi) - Lvar(\varphi)]$.*

The substitution sum and product are well–defined (i.e. they yield a substitution). For the substitution sum this can be easily deduced from the definition. The substitution product is the well-known substitution composition. This fact automatically implies the substitution product properties as they are known for composition. The sum is a composition restricted to the variables of the first substitution.

The substitution product is an associative operation. The substitution sum is in general neither commutative nor associative. Other properties are cumulated into the following lemma.

Lemma 3.2 *Let φ, ψ and σ be substitutions.*

1. *The identity $\psi \circ \varphi = \psi \triangle \varphi$ holds if, and only if, $Lvar(\psi) \subseteq Lvar(\varphi)$.*
2. *The identity $\psi \triangle \varphi = \varphi$ holds if, and only if, $Lvar(\psi) \cap Rvar(\varphi) = \emptyset$.*
3. *The identity $\sigma \triangle (\varphi \triangle \psi) = (\sigma \triangle \varphi) \triangle \psi$ holds if $Lvar(\sigma) \subseteq Lvar(\varphi)$ and σ, φ, ψ does not contain variable renamings.*

The following lemma is a backbone for proving the theorem 4.10.

Lemma 3.3 *Let φ and ψ be substitutions. The identity $(\psi \triangle \varphi) \circ \psi = \psi \circ \varphi$ holds if $Lvar(\varphi) \cap Var(\psi) = \emptyset$.*

The following property presents only a shorthand for a long notation.

Definition 3.4 *The substitutions φ and ψ are* **coherent** *(denote it by $\varphi - \psi$) if $Lvar(\varphi) \cap Var(\psi) = \emptyset$ or $Var(\varphi) \cap Lvar(\psi) = \emptyset$.*

The coherence relation is symmetric.

3.2 Substitution operators

On the basis of substitution sum we define three iterative substitution operators — the exponent, W–operator and T–operator. For convenience and to remember their names, we can call the last two *Whale*, and *Turtle*. Applying these iterative operators on basic level substitution yields a graded sequence of new derived substitutions. The first two operators are necessary for closure chain definitions, the third one will be used in the crossed rule generalization.

Definition 3.5 *Let φ and ψ be substitutions. We define the iterative* **exponent** *operator on a substitution φ inductively:*

1. $\varphi^0 = [\,]$
2. $\varphi^{n+1} = \varphi \circ \varphi^n$

There is a generalization of the lemma 3.3 for the exponent operator.

Lemma 3.6 *Let φ and ψ be substitutions. If $Lvar(\varphi) \cap Var(\psi) = \emptyset$ then $(\psi^n \triangle \varphi) \circ \psi^n = \psi^n \circ \varphi$ holds for all n.*

and ψ inductively:

1. $W_0(\psi, \varphi) = [\,]$
2. $W_{n+1}(\psi, \varphi) = \psi \triangle (W_n(\psi, \varphi) \triangle \varphi)$

Because of their close relation, it would be interesting to know when the exponent and the W–operator coincide.

Lemma 3.8 *Let φ and ψ be substitutions. If $Lvar(\psi) \cap Rvar(\varphi) = \emptyset$ then $W_n(\psi, \varphi) = \varphi^n$ holds for all n.*

Last but not least, we introduce the T–operator. This operator will not be used in closure chains but in crossed systems definition [3]. We present here only its definition and determine its connection to the previous operators.

Definition 3.9 *Let φ, ψ and σ be substitutions. We define the iterative operator T* **(Turtle)** *on them inductively:*

1. $T_0(\varphi, \psi, \sigma) = \varphi \triangle \sigma$
2. $T_{n+1}(\varphi, \psi, \sigma) = \varphi \triangle (T_n(\varphi, \psi, \sigma) \triangle \psi)$

The T–operator can be regarded as a slight extension of the W–operator. Because of their close relation it would be interesting to know when do they coincide.

Fact 3.10 *The identity $T_n(\varphi, \psi, \psi) = W_{n+1}(\varphi, \psi)$ holds for all n.*

The following proposition determines the connection between the T– and exponent operators when the substitution σ sifts upward.

Lemma 3.11 *Let φ, ψ and σ be substitutions. If $Lvar(\varphi) \cap (Rvar(\psi) \cup Rvar(\sigma)) = \emptyset$ and $Lvar(\sigma) \subseteq Lvar(\psi)$, then $T_n(\varphi, \psi, \sigma) = \sigma \triangle \psi^n$ holds for all n.*

4 Properties of rule closures

In the first part we investigate the structure and construction of rule closures in general. The second part introduces special rule closures, called closure chains, together with the investigation of their properties.

4.1 Structure and construction of rule closures

As it can be easily seen from the definition 2.1, a duality principle determines the construction of forward and backward closures.

Proposition 4.1 (First duality principle) *Let R be a variable preserving rewrite system. Then $s \rightarrowtriangle t$ is a forward closure in $FC(R)$ if, and only if, $t \trianglerightarrow s$ is a backward closure in $BC(R^{op})$.*

either forward or backward ones without closer distinction. A *rule closure* without distinction will be denoted $s \rhd\!\!\rightarrow\!\!\rhd t$ which means either $s \rhd\!\!\rightarrow t$ or $t \leftarrow\!\!\lhd s$. A *closure set* of R without distinction will be denoted $RC(R)$ which means either $FC(R)$ or $BC(R^{op})$.

Let us investigate the construction of rule closures in particular. First, we introduce the closure operations according to the definition 2.1.

Definition 4.2 *Let R be an arbitrary set of rules. Let $p_1 = s_1 \rhd\!\!\rightarrow\!\!\rhd t_1$, $p_2 = s_2 \rhd\!\!\rightarrow\!\!\rhd t_2$ be two rule closures in $RC(R)$ such that $\sigma t_1/a = \sigma s_2$ holds with a most general unifier σ and a nonvariable occurrence $a \in D_s(t_1)$. Then the rule closure $\sigma s_1 \rhd\!\!\rightarrow\!\!\rhd \sigma t_1[a \leftarrow \sigma t_2]$ is* **constructible** *and the operation producing it is denoted $p_1 \rightsquigarrow_a p_2$.*

The definition 2.1 may imply a brute force procedure for producing the closure set $FC(R)$ or $BC(R)$ from R based on the superposition of all newly produced rule closures to the existing ones. This can result in a rather inefficient way of producing the closure set (if it is finite) or deciding that the produced closure set is infinite. The following lemma implies a smarter procedure with considerably better performance.

Lemma 4.3 *Let p_1, p_2 and p_3 be rule closures of uniform kind. If the rule closure $q = p_1 \rightsquigarrow_a (p_2 \rightsquigarrow_b p_3)$ is constructible then $q = (p_1 \rightsquigarrow_a p_2) \rightsquigarrow_{ab} p_3$.*

The previous lemma proves a semi–associativity of the closure operations. Forward closures are left–, backward ones are right–associative.

Corollary 4.4 *Let R be a variable preserving rewrite system. If p is a rule closure from $RC(R) - R$ then it can be written as a composition $p = q_1 \rightsquigarrow q_2$ where $q_1 \in RC(R)$ and q_2 is a rewrite rule in R, resp. R^{op}.*

From this corollary follows that a new rule closure can be produced from an already existing one and a basic rewrite rule. This is the principle of the closure producing procedure 4.5. This procedure is the same for both forward and backward closure sets, except of the closure producing operation.

Procedure 4.5 (Closure set generation)

Input: *R, resp. R^{op}*
Output: *$RC(R)$ if it halts*
Method:

```
var A, B, C: set;
begin
  B := R; A := ∅; C := ∅;
  repeat
    for (all p ∈ B) and (all q ∈ R) and (all a ∈ D_s(p)) do
      if p ⇝_a q is constructible for the occurrence a
      then C := C ∪ {p ⇝_a q}
      fi
    od;
    A := A ∪ B; B := C; C := ∅
  until B = ∅;
  RC(R) := A
end
```

closure set $RC(R)$ is (in)finite is undecidable for an arbitrary rewrite system R.

Theorem 4.6 *It is undecidable if the closure set $RC(R)$ of a rewrite system R is finite, even if R has only two rules, one of them a ground rule containing no variables.*

4.2 Closure chains

We define a special kind of rule closures called *closure chains*. As in the case of rule closures, two types of closure chains may basically appear — forward and backward chains. Actually, our interest is focused on the closure chains, because, according to their construction, they are responsible for the divergence of Knuth–Bendix completion procedure [3].

Let us consider a rule closure

$$s \rhd\!\!\rightarrow t \tag{1}$$

with the requirement that it should be possible to chain it with itself. This means that the term s must be unifiable with a nonvariable subterm t/b. Unification is possible only if the unified terms have disjoint variables, which is not the case here, therefore we need to perform variable renaming. If we want to control the chaining iteration, variable renaming must be performed explicitly. These considerations leed to the split of a unifier into variable renamings and substitutions in own variables.

First, we need a variable renaming $\rho_0 = [x \leftarrow x_0 \mid x \in Var(s)]$ to rename the variables in (1) by introducing indexes to them. Now, we can consider the chaining of the rule closures $\rho_0 s \rhd\!\!\rightarrow \rho_0 t = s_0 \rhd\!\!\rightarrow t_0$ and $s \rhd\!\!\rightarrow t$. The actual unifier in the superposition of previous rule closures will be split into the substitutions φ_1 and φ_2 in own variables of s, such that

$$\varphi_1 t/b = \varphi_2 s \tag{2}$$

holds, and the variable renaming $\pi_0 = [x_0 \leftarrow x \mid x_0 \in Var(t_0/b)]$ canceling some of the indexes introduced by ρ_0, to allow the application of the substitution φ_1 on the subterm t/b. To assure the idempotence of the unifier, we apply the variable renaming $\rho_1 = [x \leftarrow x_1 \mid x \in Var(s)]$ introducing new indexes, at the end. Thus, the actual unifier will be $\sigma = (\rho_1 \triangle (\varphi_1 \triangle \pi_0)) \cup (\rho_1 \triangle \varphi_2)$ and the rule closure yielded in the first iteration step is

$$(\rho_1 \triangle (\varphi_1 \triangle \pi_0))s_0 \rhd\!\!\rightarrow (\rho_1 \triangle (\varphi_1 \triangle \pi_0))t_0[b \leftarrow (\rho_1 \triangle \varphi_2)t] \tag{3}$$

Up to this step there is nothing interesting in this process, except that variable renaming is performed explicitly. Let us suppose further that the newly created rule closure (3) and the previous rule closure (1) build another rule closure, and that this process may continue up to infinity: the rule closure, produced during the n–th iteration step, and the rule closure (1) always build a new rule closure.

Let us think of the rule closure (3) in the form $s_1 \rhd\!\!\rightarrow t_1$. According to the previous assumption, in the second iteration there must be the substitutions φ_1', φ_2' in own variables of s, the occurrence $b' \in D_s(t_1)$, and the variable renaming $\pi_1 = [x_1 \leftarrow x \mid x_1 \in Var(t_1/b')]$, such that

$$(\varphi_1' \triangle \pi_1)t_1/b' = \varphi_2' s \tag{4}$$

holds. Of course, this is only a recall of the condition (2) from the first iteration. It is obvious to think of b' as an iteration of b, i.e. $b' = b.b$, because this occurrence exists in t_1 in each particular case. This means that (4) can be transformed to

$$(\varphi_1' \circ \varphi_2)t/b = \varphi_2' s \tag{5}$$

substitution function, in this case

$$\varphi_1' = f_1(\varphi_1, \varphi_2) \quad (6)$$
$$\varphi_2' = f_2(\varphi_1, \varphi_2) \quad (7)$$

Let us now compare the identities (2) and (5). Of course the identity (2) can be extended to

$$(\psi' \circ \varphi_1)t/b = (\psi' \circ \varphi_2)s$$

for all substitutions ψ' according to the composition properties, but such kind of extension does not give appropriate results and do not suit our goals. We must proceed smarter and extend the identity (2) to a more general one

$$(\psi' \triangle \varphi_1)t/b = (\psi' \triangle \varphi_2)s \quad (8)$$

It is also obvious to consider the new extending substitution ψ' in terms of φ_1 and φ_2, introducing a third substitution function

$$\psi' = f_3(\varphi_1, \varphi_2) \quad (9)$$

Comparing the identities (5) and (8) in terms of the defined substitution functions f_1, f_2 and f_3 we get the following identities

$$f_1(\varphi_1, \varphi_2) \circ \varphi_2 = f_3(\varphi_1, \varphi_2) \triangle \varphi_1 \quad (10)$$
$$f_2(\varphi_1, \varphi_2) = f_3(\varphi_1, \varphi_2) \triangle \varphi_2 \quad (11)$$

which express the conditions that must be satisfied in the second iteration step.

As it was described earlier, we require that it must be possible to iterate this process up to infinity. This means that an iterated infinite rule closure sequence $s_n \rhd\!\!\rightarrow t_n$ must be constructible from $s \rhd\!\!\rightarrow t$, together with the iterated sequences of substitutions

$$\varphi_1^{(n)} = f_1^{(n)}(\varphi_1, \varphi_2)$$
$$\varphi_2^{(n)} = f_2^{(n)}(\varphi_1, \varphi_2)$$
$$\psi^{(n)} = f_3^{(n)}(\varphi_1, \varphi_2)$$

taking advantage of the explicit variable renaming by a pair of fold/unfold substitutions

$$\pi_n = [x_n \leftarrow x \mid x_n \in Var(t_n/b^{n+1})]$$
$$\rho_n = [x \leftarrow x_n \mid x \in Var(s)]$$

for each iteration step n, which cancel and introduce indexes. Note that

$$Lvar(\pi_n \triangle \rho_n) \cap Var(t_n/b^{n+1}) = \emptyset$$

The actual unifier in the iteration step n will be constructed from $\varphi_1^{(n)}$, $\varphi_2^{(n)}$, π_n, and ρ_{n+1} as

$$\sigma_n = (\rho_{n+1} \triangle (\varphi_1^{(n)} \triangle \pi_n)) \cup (\rho_{n+1} \triangle \varphi_2^{(n)})$$

The iterated substitution sequences $\varphi_1^{(n)}$, $\varphi_2^{(n)}$ and $\psi^{(n)}$ are derived from the notion of substitution function, introduced by the identities (6), (7) and (9). Therefore the new substitution functions $f_1^{(n)}$, $f_2^{(n)}$ and $f_3^{(n)}$ are built up as iterations of the basic ones f_1, f_2 and f_3. Now, the conditions (10) and (11) are to be extended in a similar manner.

As it is to be shown in the following propositions, the previously described iterations are granted by the substitution algebra constructions, especially by the iterative substitution operators *exponent* and *Whale*, developed in Section 3. Hence the closure chain definitions inherit the conditions of their supporting substitution algebra constructions. Particularly, these are the conditions from lemma 3.3, propagated through lemma 3.6. Last but not least, the rule closure (1) can be enlarged to a mixed structure — to an overlap closure.

1. $\varphi_1 t/b = \varphi_2 s$

2. $\varphi_1 - \varphi_2$

Someone may argue that there exist overlap closures which do not satisfy the coherence condition and, nevertheless, build an infinite iteration sequence. It can be proved that the coherence relation becomes satisfied after the first iteration step.

Backward chains represent the counterpart to the forward chains. They are constructed from backward closures in a similar way. Also the same kind of duality according to forward and backward chains can be observed.

Definition 4.8 *An overlap closure $s \rightarrowtail t$ is a* **backward chain** *if there are substitutions φ_1, φ_2 in own variables of s and occurrence $b \in D_s(s)$ such that*

1. $\varphi_1 t = \varphi_2 s/b$

2. $\varphi_1 - \varphi_2$

The same discussion as for the forward chain, concerning the coherence conditions and chaining property, appears also for the backward chain. Of course, not every rule closure builds a closure chain.

As with the closure definitions, also in this case there can be observed a very close duality between both definitions of closure chains.

Proposition 4.9 (Second duality principle) *The closure $s \rightarrowtail t$ is a forward chain if, and only if, $t \rightarrowtail s$ is a backward chain.*

According to this duality principle, we do not need to distinguish between forward and backward chains in general. If a result is provable for one kind of closure chains then it is provable also for the other kind according to this duality principle. Therefore if we speak about closure chains we mean either forward or backward ones. Let us describe the rule closures $s \rhd\!\!\rightarrow s$ with the same terms on both sides as **reflexive**.

The next theorem describes closures sets by means of closure chains. Actually, there are two theorems, one for the forward chain, the other for the backward one, but their proofs are completely corresponding to each other. Therefore we may take advantage of the duality principle 4.9.

Theorem 4.10 *If $RC(R)$ contains a nonreflexive closure chain then $RC(R)$ is infinite.*

Proof: (Sketch)
Let $s \rhd\!\!\rightarrow t$ be a closure chain as in definition 4.7 or 4.8. Let

$$\begin{aligned} s_0 &= \rho_0 s \\ s_{n+1} &= (\rho_{n+1} \triangle ((W_n(\varphi_1, \varphi_2) \triangle \varphi_1) \triangle \pi_n)) s_n \\ t_0 &= \rho_0 t \\ t_{n+1} &= (\rho_{n+1} \triangle ((W_n(\varphi_1, \varphi_2) \triangle \varphi_1) \triangle \pi_n)) t_n [b^{n+1} \leftarrow (\rho_{n+1} \triangle (W_n(\varphi_1, \varphi_2) \triangle \varphi_2)) t] \end{aligned}$$

define a sequence of terms s_n and t_n. It can be proved that $s_n \rhd\!\!\rightarrow t_n$ is in $RC(\{s \rightarrow t\})$ for all n by induction. □

Theorem 4.11 *It is undecidable in general whether $RC(R)$ contains a closure chain.*

The proof consists of a simple modification of the undecidability proof of Narendran and Stillman [7].

First, we introduced a special substitution algebra with a connection to the divergence problem of completion. After that we studied some properties of rule closures in general, pointing out the undecidability of the closure set finiteness. Last but not least, we defined a special class of rule closures, called closure chains, identified them formally as the core reason of divergence, and proved their iterated generation by means of the previously introduced substitution algebra.

Because the closure chains were identified as the underlying notion for divergence, the determination of a system as divergent degrades to a search for closure chains. Unfortunately, it is undecidable in general to determine if eventually a closure chain will be generated.

Acknowledgements

I would like to thank Hélène Kirchner, Leo Bachmair, Laurent Fribourg, Pierre Lescanne, Igor Prívara, and Pierre Réty for comments and suggestions on the early versions of this paper, and an anonimous referee for pointing out a flaw in part 3 in lemma 3.2.

References

[1] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987. Special issue on Rewriting Techniques and Applications.

[2] J.V. Guttag, D. Kapur, and D.R. Musser. On proving uniform termination and restricted termination of rewrite systems. *SIAM Journal on Computing*, 12(1):189–214, February 1983.

[3] M. Hermann. Crossed term rewriting systems. Research report 89-R-003, Centre de Recherche en Informatique de Nancy, 1989. Included in [?].

[4] M. Hermann and I. Prívara. On nontermination of Knuth-Bendix algorithm. In L. Kott, editor, *Proceedings 13th ICALP Conference, Rennes (France)*, volume 226 of *Lecture Notes in Computer Science*, pages 146–156. Springer-Verlag, July 1986.

[5] G. Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computer and System Science*, 23(1):11–21, August 1981. Also as: Rapport 25, INRIA, 1980.

[6] D.S. Lankford and D.R. Musser. A finite termination criterion. Unpublished draft, Information Sciences Institute, University of Southern California, Marina-del-Rey, CA, 1978.

[7] P. Narendran and J. Stillman. It is undecidable whether the Knuth-Bendix completion procedure generates a crossed pair. In B. Monien and R. Cori, editors, *Proceedings 6th Symposium on Theoretical Aspects of Computer Science (STACS'89), Paderborn (Germany)*, volume 349 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, February 1989.

[8] B. K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the Association for Computing Machinery*, 20(1):160–187, January 1973.