Simulating Forward-Branching Systems with Constructor Systems

Bruno Salinier and Robert Strandh

LaBRI, URA 1304, Université Bordeaux I
351, Cours de la Libération, 33405 Talence, France
{salinier,strandh}@labri.u-bordeaux.fr

Abstract. Strongly sequential constructor systems admit a very efficient algorithm to compute normal forms. The class of forward-branching systems contains the class of strongly sequential constructor systems, and admits a similar reduction algorithm, but less efficient on the entire class of forward-branching systems. In this article, we present a new transformation which transforms any forward-branching system into a strongly sequential constructor one. We prove the correctness and completeness of the transformation algorithm, then that the new system is equivalent to the input system, with respect to the behavior and the semantics. As a programming language, it permits us to have a less restrictive syntax without compromise of semantics and efficiency.

1 Introduction

Term rewriting systems (TRS for short) are of a great interest for a number of applications involving computing with equations. Orthogonal TRSs which ensure confluent reductions but not necessarily termination, form a good framework for programming with equations. The evaluation of a term with a TRS consists of repeatedly replacing redexes (a redex is an instance of a left-hand side) of the input term by the corresponding right-hand sides. This process, called *reduction*, stops if a normal form is reached. For a term having a normal form there may be infinite sequences of reductions, thus not leading to the normal form.

The *strongly sequential* TRSs (*SS*) was defined by Huet and Lévy [4]. The class of *forward-branching* systems (*FB*) introduced by Strandh [7] is a subclass of *SS*. He proved that in *FB*, *outermost evaluation* can be preserved while still doing *innermost stabilization* (computing strong head-normal forms), leading to an efficient strategy for sequences of reductions. Furthermore, Durand [1] has proved that the forward-branching property can be decided in quadratic time.

Thatte [8] demonstrated the possibility of simulating an orthogonal TRS with a left-linear constructor system obtained from the original system via a simple syntactic transformation. Unfortunately, it does not preserve strong sequentiality. The *constructor equivalent* systems, for which strong sequentiality is preserved by Thatte's transformation, form a subclass of *FB* [2, 3, 6].

In this paper, we present a new transformation which allows us to simulate any *FB* system with a strongly sequential constructor system. This new constructor system is generated from the *index tree* (automaton driving the reduction)

of the original system. We prove that this algorithm is complete and correct. Moreover, the equivalence between the final and original system is proved.

2 Terminology and Notation

We mainly follow the terminology of [4] and [5]. Let $\mathcal{F}_n$ be a set of *function symbols* of arity n, $\mathcal{F} = \bigcup\{\mathcal{F}_n \mid n \geq 0\}$, and $\mathcal{V}$ a denumerable set of *variables*. Our expression language is the set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of first order *terms* formed from $\mathcal{F}$ and $\mathcal{V}$. When $\mathcal{F}$ and $\mathcal{V}$ are fixed, we denote $\mathcal{T}(\mathcal{F}, \mathcal{V})$ by $\mathcal{T}$. For any term M, we define its set of *occurrences* $\mathcal{O}(M)$ as a set of sequences of integers: $\Lambda \in \mathcal{O}(M)$, and $F \in \mathcal{F}_n$ and $u \in \mathcal{O}(M_i) \Rightarrow iu \in \mathcal{O}(F(M_1, \ldots, M_n))$ for $1 \leq i \leq n$. Intuitively, an occurrence of M names a subterm of M by its access path. The occurrences are partially ordered by the *prefix ordering* $\leq$: $u \leq v$ iff $\exists w$ such that $uw = v$. In this case, we define v/u as w. Finally, $u < v$ iff $u \leq v$ and $u \neq v$.

If $u \in \mathcal{O}(M)$, we define the *subterm of M at u* as the term M/u defined by $M/\Lambda = M$, and $F(M_1, \ldots, M_n)/iu = M_i/u$ for $1 \leq i \leq n$. We write $root(M)$ to denote the *root symbol* of M. We also use $\overline{\mathcal{O}}(M)$ to denote the nonvariable occurrences in M: $\overline{\mathcal{O}}(M) = \{u \in \mathcal{O}(M) \mid M/u \notin \mathcal{V}\}$. If $u \in \mathcal{O}(M)$, we define for a term N the *replacement in M at u by N* as the term $M[u \leftarrow N]$ defined by:

$M[\Lambda \leftarrow N] = N$,

$F(M_1, \ldots, M_i, \ldots, M_n)[iu \leftarrow N] = F(M_1, \ldots, M_i[u \leftarrow N], \ldots, M_n)$.

A *substitution* σ is a map from $\mathcal{T}$ to $\mathcal{T}$ which satisfies $\sigma(F(M_1, \ldots, M_n)) = F(\sigma(M_1), \ldots, \sigma(M_n))$. So, σ is determined by its restriction to $\mathcal{V}$. We use *term rewriting system* for any set Σ of pairs of terms $L \to R$ such that $\mathcal{V}(R) \subseteq \mathcal{V}(L)$ where $\mathcal{V}(M)$ denotes the set of variables appearing in the term M. We write Red_Σ to denote the set of left-hand sides (lhs for short) L of Σ. For any substitution σ and $N \in Red_\Sigma$, $\sigma(N)$ is called a *redex* of Σ. An occurrence u of a term M is called a redex occurrence if M/u is a redex of Σ. A term which does not contain any redex is in *Σ-normal form*. We will drop Σ if it is fixed.

The term M *reduces to* N, written $M \to N$, at occurrence u using rule $L \to R$ iff there exists a substitution σ such that $M/u = \sigma(L)$ and $N = M[u \leftarrow \sigma(R)]$. We use $\stackrel{*}{\to}$ to denote the reflexive and transitive closure of $\to$.

A TRS Σ is *orthogonal* iff it is *left-linear* (for every L in Red, every variable of L occurs only once), and *non ambiguous* (if $L_i, L_j \in Red$, for every $u \in \overline{\mathcal{O}}(L_i)$ there are no substitutions σ, σ', such that $\sigma(L_i/u) = \sigma'(L_j)$, except in the trivial case $i = j$ and $u = \Lambda$). The second condition is also called non-overlapping condition. It is well known that for orthogonal TRSs, the relation $\to$ is confluent. In this article, we restrict ourselves to the class of orthogonal TRSs.

To represent a lack of knowledge in a term, we use *Ω-terms*, i.e. terms where the new nullary function symbol Ω can occur. Let $\mathcal{T}_\Omega$ be the set of these Ω-terms. Let us consider the *prefix ordering* $\preceq$ on $\mathcal{T}_\Omega$ defined by $\Omega \preceq M$ for all $M \in \mathcal{T}_\Omega$, and $F(M_1, \ldots, M_n) \preceq F(N_1, \ldots, N_n)$ iff $M_i \preceq N_i$ for each $i, 1 \leq i \leq n$.

All the previous operations are obviously extended to Ω-terms. Furthermore, two Ω-terms M and N are *compatible*, written $M \uparrow N$ iff $M \preceq P$ and $N \preceq P$ for some P. If $F \in \mathcal{F}_n$, we write $F(\vec{\Omega})$ to denote $F(\Omega, \ldots, \Omega)$. If $M \in \mathcal{T}_\Omega$ then we

write $\mathcal{O}_\Omega(M)$ for the Ω-occurrences of M: $\mathcal{O}_\Omega(M) = \{u \in \mathcal{O}(M) \mid M/u = \Omega\}$. The set $\mathcal{O}(M) \setminus \mathcal{O}_\Omega(M)$ is denoted by $\overline{\mathcal{O}}_\Omega(M)$. An *$\Omega$-normal form* is an Ω-term N without redex and containing at least one occurrence of Ω.

We write M_Ω for M where all variables x of M are replaced by Ω. If L is a lhs then L_Ω is a *redex scheme*. We denote the set of redex schemes by Red_Ω. A *preredex* M is an Ω-term such that $M \preceq L$ where $L \in Red_\Omega$. It is *proper* if it is neither Ω nor a redex scheme. A *partial redex* is a proper preredex or Ω.

A *constructor* symbol is a symbol of $\mathcal{F}$ that does not appear at the root of any redex scheme. We denote the subset of constructor symbols by $\mathcal{C}$ and the subset of nonconstructor (or *defined*) symbols by $\mathcal{D}$. A TRS is a *constructor system* iff for every L in Red_Ω, all $u \in \overline{\mathcal{O}}_\Omega(L)$, $u \neq \Lambda$ are such that $root(L/u) \in \mathcal{C}$.

The constructor class is denoted by C. We write Red'_Ω to denote the set of all subterms of redex schemes having a nonconstructor symbol at their root:

$Red'_\Omega = \{M \mid \exists L \in Red_\Omega, \exists u \in \overline{\mathcal{O}}_\Omega(L), L/u = M \text{ and } root(M) \in \mathcal{D}\}$.

It is clear from the definition of Red'_Ω that $Red_\Omega \subseteq Red'_\Omega$. An element of Red'_Ω is a *subscheme*. It is *strict* if it is not a scheme.

Lemma 1. *Let Σ be an orthogonal system. Σ is constructor iff $Red_\Omega = Red'_\Omega$.*

3 Forward-Branching Systems

Before presenting the forward-branching class. We need first to recall the definition of strongly sequential systems of Huet and Lévy [4]. A predicate P on $\mathcal{T}_\Omega$ is *monotonic* if $P(M)$ implies $P(M')$ whenever $M \preceq M'$.

Let P be a monotonic predicate on $\mathcal{T}_\Omega$. An Ω-occurrence u of an Ω-term M is said to be an *index* of P in M iff $\forall N$ s.t. $M \preceq N$, $P(N) = true$ implies $N/u \neq \Omega$. Then P is *sequential at* M iff whenever $P(M) = false$, and $\exists N \succeq M$ s.t. $P(N) = true$, it follows that there exists an index of P in M.

Let M and N be in $\mathcal{T}_\Omega$. We write $M \rightarrow_? N$ iff $N = M[u \leftarrow T]$ for some redex occurrence u and some Ω-term T. It corresponds to reduction with arbitrary right-hand sides and is called *arbitrary reduction*. The predicate $nf_?$ is defined as: $nf_?(M) = true$ iff $\exists N$ in normal form such that $M \xrightarrow{*}_? N$.

Definition 2. An orthogonal system Σ is *strongly sequential* iff the predicate $nf_?$ is sequential at any M in Ω-normal form.

The strongly sequential class is denoted by SS. Deciding that an occurrence is an index (of $nf_?$) is easy, but deciding whether a TRS is strongly sequential is not trivial [4] and is conjectured to be NP-complete [5].

3.1 Index Trees

Huet and Lévy also defined strongly sequential systems in terms of the existence of a matching DAG [4]. Durand [1] proved that the *index tree* of Strandh [7] is equivalent to the matching DAG. We now recall the definition of an index tree.

An Ω-term M is a *potential redex* iff $\exists N, T$ s.t. $M \preceq N$, $N \xrightarrow{*}_? T$ and T is a redex. $u \in \mathcal{O}(M)$ is a *potential redex occurrence* iff M/u is a potential redex. An Ω-term is *in strong head normal form* iff it is not a potential redex.

An Ω-term is a potential redex if there is a way to refine it, and then arbitrarily reduce it so it becomes a redex. The root symbol of an Ω-term M in strong head normal form cannot change even if M is refined and arbitrarily reduced.

Let M be an Ω-term. An occurrence u of M is a *strongly stable* occurrence of M iff M/u is in strong head normal form. Let M be an Ω-term. M is a *firm* Ω-term iff $\exists u \in \mathcal{O}_\Omega(M)$ such that $\forall v \in \overline{\mathcal{O}}_\Omega(M)$, either v is strongly stable or $v < u$. We call such an occurrence u a *firm extension occurrence of* M.

Definition 3. An *index point* is a pair (M, u) where M is a firm Ω-term and a partial redex, u is a firm extension occurrence of M and u is an index in M.

Definition 4. Let $s = (M, w)$ and $t = (N, v)$ be two index points s.t. $M \neq \Omega$. t is a *failure point of* s iff $\exists u \neq \Lambda$ s.t. $w = uv$ and $N = M/u$. t is the *immediate* failure point of s iff every other failure point of s is a failure point of t.

Definition 5. An *index tree* $\mathcal{I}$ for a set of lhs *Red*, is a *finite state automaton* which also has a *failure function*. The set of final states is Red_Ω, nonfinal states are index points, the initial state is (Ω, Λ). Given $s = (M, u)$ and F, the transfer function, written $\delta(s, F)$, is constructed s.t. $\delta((M, u), F) = (M', u')$ (or $\delta((M, u), F) = M'$ if $M' \in Red_\Omega$) if $M' = M[u \leftarrow F(\vec{\Omega})]$. It is undefined for a final state. The *failure function* ϕ is defined by $\phi(s) = t$ iff t is the immediate failure point of s. It is undefined for both the initial and the final states.

An index tree is shown in Fig. 1. Only failure transitions leading to a state different from the initial state are shown. The transfer function δ is deterministic, thus not all index points are accessibles from the initial state $s_0 = (\Omega, \Lambda)$ via transfer transitions only.

Theorem 6. *[1] Let Σ be an orthogonal system. Σ is strongly sequential iff there exists an index tree for Red_Σ.*

Lemma 7. *Let Σ be a strongly sequential system, and let $\mathcal{I}$ be an index tree for Red. $\Sigma \in C$ if and only if $\forall s \in \mathcal{I}$ such that ϕ is defined, we have $\phi(s) = (\Omega, \Lambda)$.*

Proof. ($\Leftarrow$) Let $s \in \mathcal{I}$ s.t. ϕ is defined, and let $t \in \mathcal{I}$ s.t. $\exists F \in \mathcal{F}$ s.t. $\delta(t, F) = s$. As $\phi(s) = (\Omega, \Lambda)$, it follows from Def. 4 that $F \in \mathcal{C}$. So, $\Sigma \in C$.

($\Rightarrow$) Let $s = (M, w) \in \mathcal{I}$ and $(N, v) = \phi(s)$. From Def. 4, $\exists u \neq \Lambda$ s.t. $w = uv$ and $N = M/u$. But M is a partial redex and $\Sigma \in C$, then $\forall u' \neq \Lambda \in \mathcal{O}(M)$, $root(M/u') \in \mathcal{C}$. So, $u = w$ and $M/u = \Omega$, then $(N, v) = s_0$. □

3.2 Forward-Branching Systems

In an index tree, some states may not be reachable from the initial state (Ω, Λ) via transfer transitions only, because the index tree is deterministic. This led Strandh [7] to define the class of forward-branching systems.

Definition 8. An index tree is said to be *forward-branching* iff every state of the index tree can be reached via transfer transitions only from the initial state.

The index tree of Fig. 1 is forward-branching. We have the following property in a forward-branching index tree.

Lemma 9. *[7] In a forward-branching index tree, two index points (M, u) and (M, v) where $u \neq v$ cannot exist.*

Definition 10. A system Σ is *forward-branching* iff there exists a forward-branching index tree for Red_Σ.

The forward-branching class is denoted by *FB*. We deduce from the lemma 9 that the extension occurrence part of an index point is fully determined by its Ω-term part. So, by abuse of notation, we will consider only the Ω-term part of the index point. We can define an obvious partial order on index points.

Definition 11. Let S and T be two index points. $S \sqsubset T$ if and only if there exists a non-empty sequence of index points $(S = P_1, \ldots, P_n = T)$ such that $\forall i, 1 \leq i < n, \exists F \in \mathcal{F}$ such that $\delta(P_i, F) = P_{i+1}$.

Lemma 12. *Let S and T be two index points. If $S \sqsubset T$ then $S \prec T$.*

Lemma 13. *Let (M, u) be an index point of a forward-branching index tree $\mathcal{I}$. $\forall N \in Red'_\Omega$, if $M \uparrow N$ then $M \prec N$.*

Proof. Suppose not: $\exists N \in Red'_\Omega$ such that $M \uparrow N$ but $M \not\prec N$. Obviously, $N \not\preceq M$ since otherwise the system would have an overlap.

Let $L \in Red_\Omega$ and $w \in \overline{\mathcal{O}}_\Omega(L)$ such that $L/w = N$ (possibly $w = \Lambda$), and let (Q, w) be the index point corresponding to w. Let P be the largest Ω-term such that $P \sqsubset M$ (M belongs to $\mathcal{I}$ by hypothesis), and let (P, v) be its corresponding index point. Finally, let R be the largest index point such that $R \sqsubset Q[w \leftarrow N]$.

From lemma 12, it follows that $P \prec M$ and $R/w \prec N$. But $M \uparrow N$, so either $root(N/v) = root(M/v) = K$, or $N/v = \Omega$ or $M/v = \Omega$. In the first case, $M \preceq N$ which contradicts $M \not\prec N$ and $N \not\preceq M$. In the second case, (R, wv) can not be an index point because wv is not an index of R. In the last case, (P, v) can not be an index point because v could not be the next element in $\mathcal{I}$. □

Durand found an characterization of *FB*, which shows a close connection between redex schemes and subschemes.

Property 14. *[1] $\forall N \in Red_\Omega$, $\forall M \prec N$, $\exists u \in \mathcal{O}_\Omega(M)$, $\forall N' \in Red'_\Omega$ with $M \prec N'$, $N'/u \neq \Omega$.*

Proposition 15. *[1] An orthogonal system Σ is* forward-branching *if and only if it verifies property 14.*

Lemma 16. *[1] $FB \subset SS$.*

Lemma 17. *$FB \cap C = SS \cap C$.*

Proof. ($SS \cap C \subseteq FB \cap C$): Let $\Sigma \in SS \cap C$, and let $\mathcal{I}$ be an index tree for *Red*. From lemma 7, all the failure transitions are to s_0. It follows that all index points are reachable from s_0 via transfer transitions only. Then $\Sigma \in FB$.

($FB \cap C \subseteq SS \cap C$): Trivial since $FB \subset SS$ by lemma 16. □

4 Transforming Forward-Branching Systems

We now present an algorithm for transforming a *FB* system into a *SS* constructor system. We illustrate the algorithm with an example. We prove its correctness and completeness and the equivalence between the input and output systems.

The bulk of the transformation work is done by three procedures. *Forward-Branching* builds an index tree. *FindDT* uses it to find a *differentiating* Ω*-term* T (see below). *Transform* replaces all the instances of T in *Red*, which suppresses some nonconstructor symbols within the lhs; finally, it adds a new rule to collapse the terms which were recognized in the original system but are not recognized anymore. This process is repeated until the system is constructor.

4.1 An Example

Let $\Sigma = \{H(G(A,A,x),A) \to A, H(G(A,x,A),B) \to B, G(B,B,B) \to C\}$. Given Σ, the *Forward-Branching* procedure builds an index tree (see Fig. 1).

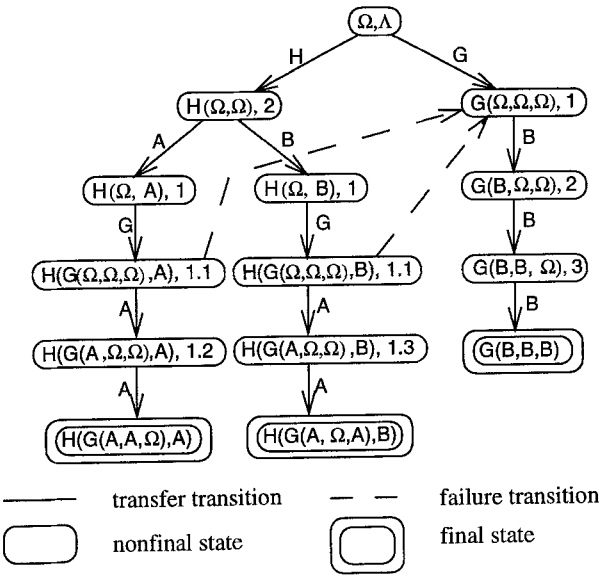

Fig. 1. A forward-branching index tree for Red_Σ

Analyzing this, *FindDT* finds that $T = G(A,\Omega,\Omega)$ is a differentiating Ω-term. *Transform* creates a symbol G_1 of same arity as G, and $R = G_1(A,\Omega,\Omega)$. It finds instances of T in $H(G(A,A,x),A)$ and $H(G(A,x,A),B)$ of *Red*, and then replaces G by G_1. We now need to rewrite all subterms containing T to a term that the new system can match. So *Transform* adds a rule $G(A,x_1,x_2) \to G_1(A,x_1,x_2)$. We finally obtain the *FB* system $\Sigma^1 = \{H(G_1(A,A,x),A) \to$

$A, H(G_1(A, x, A), B) \rightarrow B, G(B, B, B) \rightarrow C, G(A, x_1, x_2) \rightarrow G_1(A, x_1, x_2)\}$, which is "more constructor" because a defined symbol in some lhs is replaced by a constructor one. We then restart the process with Σ^1 by computing the index tree of Fig. 2. All the failure transitions of this index tree lead to s_0, so *Transform* returns Σ^1, since it is a forward-branching constructor system (lemma 7).

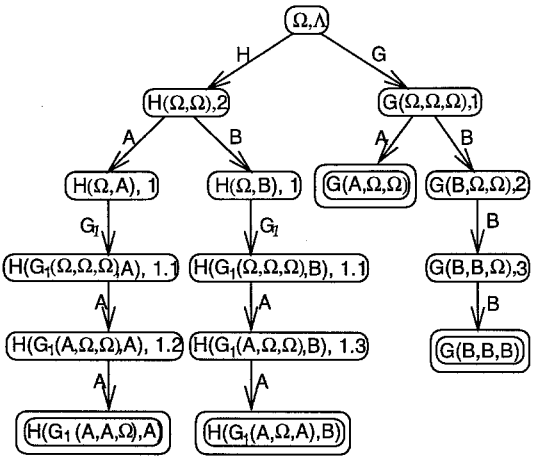

Fig. 2. An index tree after one step of transformation

4.2 Algorithm

We now provide the algorithmic description. We skip *Forward-Branching* which is described in [1]. We just point out that this procedure fails if the input system is not *FB*. It runs in quadratic time w.r.t. the number of symbols of the lhs.

Finding a Differentiating Ω-Term. In the following we search for a differentiating Ω-term T. This Ω-term will become a scheme in the new system. Therefore, it should not create an overlap: T must not be compatible with other redex schemes. Moreover, T must be sufficiently "small" to ensure that T collapse strict subschemes that otherwise would overlap with T. The *FindDT* algorithm shown in Fig. 3 returns such a *differentiating Ω-term T*. It is clear that the strict subscheme N chosen on line 1 of the *FindDT* algorithm always exists because the system is orthogonal. We now demonstrate that T has some nice properties. In the following lemmas, T is a differentiating Ω-term.

Lemma 18. *$\exists N \in Red'_\Omega \setminus Red_\Omega$ such that $T \preceq N$.*

Lemma 19. *In T, all inner symbols are constructor.*

Lemma 20. *$\forall L \in Red_\Omega$, T and L are not compatible.*

```
function FindDT(I);    /* I is a forward-branching index tree */
begin
1  choose N ∈ Red'_Ω \ Red_Ω such that
          ∀u ∈ O_Ω(N) with u ≠ Λ, root(N/u) ∈ C;
2  K ← root(N);
3  t ← s_0;
4  while δ(t, K) is defined do
     begin
5       t ← δ(t, K);
6       let t = (P, w);
7       K ← root(N/w);
     end;
8  T ← P[w ← K(Ω→)];
9  return T;
end;
```

Fig. 3. The *FindDT* algorithm

Proof. Let T, $t = (P, w)$ and K defined as in line 8 of *FindDT*. Suppose not, $\exists L \in Red_\Omega$ such that $T \uparrow L$. Since $P \prec T$, $P \uparrow L$. But (P, w) is an index point, so by lemma 13, $P \prec L$. Then $\delta((P, w), root(L/w))$ is defined.

Since $T \uparrow L$, either $root(L/w) = K$, a contradiction with line 4 of *FindDT*, or $root(L/w) = \Omega$ which is impossible in a forward-branching index tree. □

Lemma 21. *$\forall N \in Red'_\Omega \setminus Red_\Omega$ such that $T \uparrow N, T \preceq N$.*

Proof. Let T, $t = (P, w)$ and K defined as in line 8 of *FindDT*. As $T \uparrow N$ and $P \prec T$, we have $P \uparrow N$. By lemma 13, it follows $P \prec N$, and as the system is *FB*, by proposition 15, we obtain $N/w \neq \Omega$. Moreover, from the construction of T, $root(T/w) = K$. Because $T \uparrow N$ and $N/w \neq \Omega$, it implies that $root(N/w) = K$. As $P \prec N$, and from the construction of T, it follows that $T \preceq N$. □

Intuitively, lemma 20 means that we can put T as a redex scheme without creating an overlap at the root (possibly T might overlap with some strict subschemes), whereas lemma 21 says that T is a lower bound of all strict subschemes compatible with T. In other words, T only overlaps with strict subschemes greater than T. This lemma ensures that we collapse all strict subschemes which would create overlaps if we add T to the set of redex schemes.

On calling *FindDT* on the index tree of Fig. 1, N can be chosen among the two Ω-subterms of redex schemes $G(A, A, \Omega)$ and $G(A, \Omega, A)$. Whichever one is chosen, we get on line 8 the index point $t = (P, w) = (G(\Omega, \Omega, \Omega), 1)$. So, $K = root(N/1) = A$. Then we return $T = G(A, \Omega, \Omega)$ as the differentiating Ω-term. Observe that both $G(A, A, \Omega)$ and $G(A, \Omega, A)$ are compatible with T.

Transforming Forward-Branching Systems. The *Transform* procedure of Fig. 4 builds a *FB* constructor system. If $M \in \mathcal{T}_\Omega$ then $M_\mathbf{A}$ (read alpha) is a term obtained by replacing from left to right *each* Ω by a *new* variable x_i.

```
procedure Transform(Σ);
begin
1  I ← Forward-Branching(Σ);
2  let Φ = {φ(s)|s is an index point of I};
3  if Φ = {s0}
       then
4         return Σ;
       else
          begin
5            T ← FindDT(I);
6            let T = F(T1, ..., Tn);
7            let Fk be a new symbol of arity n;
8            for each L ∈ Red do
9               for each u ∈ Ō(L) such that T ⪯ L/u do
                   begin
10                    let L/u = F(L1, ..., Ln);
11                    L ← L[u ← Fk(L1, ..., Ln)];
                   end;
12           Σ ← Σ ∪ {T_A → (Fk(T1, ..., Tn))_A};
13           Transform(Σ);
          end;
end;
```

Fig. 4. The *Transform* algorithm

Consider the k^{th} recursive invocation of *Transform*. It first builds a forward-branching index tree $\mathcal{I}$ by calling *Forward-Branching*. If the input system is not forward-branching, *Forward-Branching* fails and exits. *Transform* then constructs the set Φ of all immediate failure points of $\mathcal{I}$. If Φ contains only s_0 then Σ is constructor (lemma 7), and *Transform* returns Σ. If $\Phi \neq \{s_0\}$, *Transform* finds a differentiating Ω-term $T = F(T_1, \ldots, T_n)$ by calling *FindDT*, and creates a *new* symbol F_k of same arity as F. It then replaces the root symbol F by F_k in all instances of T of all lhs. Moreover, it adds a new rule $\{T_{\mathbf{A}} \to (F_k(T_1, \ldots, T_n))_{\mathbf{A}}\}$ to Σ. Finally, it proceeds recursively on the new system.

We now prove the completeness and correctness of our algorithm. Consider an execution of *Transform* on a nonconstructor *FB* system Σ. Let $\mathcal{T}^0 = \mathcal{T}$, $\Sigma^0 = \Sigma$, $Red^0 = Red_\Sigma$ and $Red'^0 = Red'_\Sigma$, and let $\mathcal{T}^k$, Σ^k, Red^k and Red'^k be $\mathcal{T}$, Σ, *Red* and *Red'* after the line 12 of *Transform* while its k^{th} invocation.

Let $M \in \mathcal{T}_\Omega$. We write $|M|$ to denote the number of *inner* defined symbols (i.e. without the root symbol) of M; $|Red|$ stands for $\sum_{L_i \in Red} |L_i|$. It is easy to show that $|Red| > 0$. We can now easily prove the completeness of our algorithm.

Proposition 22. *The algorithm Transform is complete.*

Proof. Consider the k^{th} invocation of *Transform* on a system Σ^{k-1}. If Σ^{k-1} is not forward-branching, *Forward-Branching* at line 1 fails and exits. Otherwise, if Σ^{k-1} is constructor (line 3), the algorithm ends, returning Σ^{k-1} (line 4).

Let $T = F(T_1, \ldots, T_n)$. Let $L \in Red^{k-1}$ s.t. $\exists u \in \overline{\mathcal{O}}(L)$ s.t. $T \preceq L/u$ where $L/u = F(L_1, \ldots, L_n)$ (L exists by lemma 18). Clearly, we have $|T| = 0$ (consequence of lemma 19), and $|L[u \leftarrow F_k(L_1, \ldots, L_n)]| < |L|$. Finally, we have:

$$|Red^k| \leq |T| + |L[u \leftarrow F_k(L_1, \ldots, L_n)]| + \sum_{N_i \in Red^{k-1} \setminus \{L\}} |N_i|$$

$$< 0 + |L| + \sum_{N_i \in Red^{k-1} \setminus \{L\}} |N_i| = \sum_{N_i \in Red^{k-1}} |N_i| = |Red^{k-1}|.$$

As $|Red|$ is a positive integer, the algorithm *Transform* necessarily stops. □

Now, we prove that our algorithm transforms any *FB* system into a forward-branching constructor system. The map $h_k : \mathcal{T}^k \to \mathcal{T}^{k-1}$ is defined as $h_k(N) = M$ where M is obtained by replacing every occurrence of F_k in N by F.

Lemma 23. *Let M be an Ω-term of $\mathcal{T}^k$. We have $\mathcal{O}_\Omega(M) = \mathcal{O}_\Omega(h_k(M))$ and $\overline{\mathcal{O}}_\Omega(M) = \overline{\mathcal{O}}_\Omega(h_k(M))$.*

Lemma 24. *h_k is a strictly increasing map.*

Proof. Let M and M' be two Ω-terms of $\mathcal{T}^k$ such that $M \prec M'$. So, $\forall u \in \overline{\mathcal{O}}_\Omega(M)$, $root(M/u) = root(M'/u) = G$. If $G \neq F_k$ (line 7) then h_k leaves G unchanged. Otherwise, h_k replaces F_k by F. Finally, $h_k(M) \prec h_k(M')$. □

From now on, in all following statements and proofs, T, P and w will always refer to T, P and w as defined on line 8 of the $(k-1)^{\text{th}}$ invocation of *FindDT*.

Lemma 25. *h_k is a bijection from $Red'^k_\Omega \setminus \{T\}$ to Red'^{k-1}_Ω.*

Proof. Let $N_{k-1} \in Red'^{k-1}_\Omega$. We have two cases:
(1) N_{k-1} does not contain T. By construction of Σ^k, $N_{k-1} \in Red'^k_\Omega \setminus \{T\}$. It follows that $N_k = N_{k-1}$ does not contain symbol F_k s.t. $F = root(T)$. So, N_k is the only redex scheme or subscheme of $Red'^k_\Omega \setminus \{T\}$ s.t. $h_k(N_k) = N_{k-1}$.
(2) N_{k-1} contains T: $\exists u \in \overline{\mathcal{O}}_\Omega(N_{k-1})$ s.t. $T \preceq N_{k-1}/u$. *Transform* only replaces $F = root(T)$ with F_k. But h_k does the opposite operation. So, it exists a redex scheme or subscheme $N_k \in Red'^k_\Omega \setminus \{T\}$ s.t. $h_k(N_k) = N_{k-1}$. □

Lemma 26. *h_k is a bijection from $Red^k_\Omega \setminus \{T\}$ to Red^{k-1}_Ω.*

Proof. Similar to the proof of lemma 25. □

The two following lemmas will be used to show that *Transform* preserves the forward-branching property.

Lemma 27. *If Σ^{k-1} is a nonconstructor FB system then $\forall M \prec T, \exists u \in \mathcal{O}_\Omega(M)$ such that $\forall N' \in Red'^k_\Omega$ with $M \prec N', N'/u \neq \Omega$.*

Proof. Let $M \prec T$, $u \in \mathcal{O}_\Omega(M)$ and Q such that (Q,u) is the greatest index point such that $Q \preceq M$. We have $h_k(M) = M$ and $h_k(Q) = Q$ because $h_k(T) = T$ (by construction of T) and $Q \preceq M \prec T$. As Σ^{k-1} is forward-branching, property 14 holds for Σ^{k-1}. In particular, Q is a partial redex of Σ^{k-1}. So we have $\forall N' \in Red'^{k-1}_\Omega$ with $Q \prec N'$, $N'/u \neq \Omega$. From the lemmas 23, 24 and 25, we obtain $\forall N' \in Red'^{k}_\Omega \setminus \{T\}$ with $Q \prec N'$, $N'/u \neq \Omega$.

Moreover, as $Q \prec T$, we have $Q \sqsubseteq P$. So, it follows $T/u \neq \Omega$. We have then $\forall N' \in Red'^{k}_\Omega$ with $Q \prec N'$, $N'/u \neq \Omega$. As $Q \preceq M$, conclusion follows. □

Lemma 28. *If Σ^{k-1} is a nonconstructor FB system then $\forall N \in Red^k_\Omega \setminus \{T\}$, $\forall M \prec N, \exists u \in \mathcal{O}_\Omega(M)$ such that $\forall N' \in Red'^{k}_\Omega$ with $M \prec N', N'/u \neq \Omega$.*

Proof. As Σ^{k-1} is a forward-branching system, property 14 holds for Σ^{k-1}. From lemma 24, h_k preserves the partial order on Ω-terms. From lemma 23, h_k preserves the Ω-occurrences. From lemmas 26 and 25, h_k is a bijection between Red^{k-1}_Ω and $Red^k_\Omega \setminus \{T\}$, and between Red'^{k-1}_Ω and $Red'^{k}_\Omega \setminus \{T\}$. So, we obtain

$\forall N \in Red^k_\Omega \setminus \{T\}, \forall M \prec N, \exists u \in \mathcal{O}_\Omega(M)$ such that $\forall N' \in Red'^{k}_\Omega \setminus \{T\}$ with $M \prec N'$, $N'/u \neq \Omega$. (1)

Suppose that $M \prec T = P[w \leftarrow K(\vec{\Omega})] \in Red'^{k}$. Let (Q,u) be the greatest index point such that $Q \preceq M$. Necessarily, $Q \sqsubseteq P$. By construction of T, $T/u \neq \Omega$. From (1), conclusion follows immediately. □

We now give the main result. Let κ be k in the last invocation of *Transform*.

Theorem 29. *If Σ is a FB system then Σ^κ is a FB constructor system.*

Proof. By induction on κ. If $\kappa = 1$, then Σ is already a constructor system. If $\kappa > 1$, consider the first invocation of *Transform*. As Σ^0 is a nonconstructor *FB* system, and from lemmas 27 and 28, it is clear that the *FB* property 14 holds for Σ^1. By induction hypothesis, Σ^κ is a forward-branching constructor system. □

4.3 Behavior Equivalence

In this section, we show that, for every *FB* system Σ, the behavior of the *FB* (or *SS* by lemma 17) constructor system Σ^κ parallels that of Σ within the domain $\mathcal{T}$. Σ^κ is expected to deal with terms in $\mathcal{T}^\kappa$ which contains $\mathcal{T}$ as a subset.

The map $h : \mathcal{T}^\kappa \to \mathcal{T}$ is defined as $h = h_1 \circ h_2 \circ \cdots \circ h_\kappa$. If L is a lhs of Σ and L' is the corresponding lhs of Σ^κ, then clearly $h(L') = L$. We now demonstrate the equivalence of behavior between Σ and Σ^κ.

Lemma 30. *Let $M, N \in \mathcal{T}^\kappa$. If $M \xrightarrow{*} N$ in Σ^κ, then $h(M) \xrightarrow{*} h(N)$ in Σ.*

Proof. By induction on the length of the reduction sequence $M \xrightarrow{*} N$ in Σ^κ. For a zero length sequence, $M = N$ and $h(M) = h(N)$ and the lemma thus holds.

Suppose $M \xrightarrow{*} M'$ in i steps, and $M' \to N$ using a rule $L \to R$. If $R = F_k(T_1, \ldots, T_n)$ where F_k as been introduced in the k^{th} iteration of *Transform*, then $h(M') = h(N)$. Otherwise, $h(M') \to h(N)$ in Σ by the rule $h(L) \to R$. □

Lemma 31. *Let M and N be in $\mathcal{T}$. If $M \xrightarrow{*} N$ in Σ then $M \xrightarrow{*} N$ in Σ^κ.*

Proof. By induction on the length of the reduction sequence $M \xrightarrow{*} N$ in Σ. For a sequence of length zero, the lemma trivially holds.

Now, suppose that (in Σ) $M \xrightarrow{*} M'$ in i steps, and $M' \to N$ using a rule $L \to R$. If L only contains constructor symbol (except at the root) then $L \to R$ is also a rule of Σ^κ, so $M' \to N$ in Σ^κ. Otherwise, let $v \in \mathcal{O}(M')$ be a redex occurrence: there exists a substitution σ such that $M'/v = \sigma(L)$. For each strict subscheme $T = F(T_1, \ldots, T_n)$ of L, the corresponding subterm of M'/v can be reduced using the rule of Σ^κ $F(T_1, \ldots, T_n) \to F_k(T_1, \ldots, T_n)$.

So, this subterm of M'/v is an instance of the lhs of the rule $L' \to R$ in Σ^κ corresponding to $L \to R$ in Σ. Finally, $L' \to R$ is used to obtain N. □

Theorem 32. *Σ^κ is equivalent to Σ.*

5 Conclusion

We have demonstrated the possibility of simulating any *FB* system with a *SS* constructor one. The construction is useful in many practical situations where only a small number of lhs sides contain a few nonconstructor symbols, and hence the size of the resulting system increases only modestly over that of the original one. In the worst case, if all the original lhs are made up almost entirely of defined symbols, the size of the new system could be quadratically larger than that of input system, w.r.t. the number of symbols in the lhs. The equivalence between *FB* and strongly sequential constructor systems was suspected for a long time because the reduction algorithms were essentially identical. Hence, as a programming language, forward-branching systems admit a less restrictive syntax as constructor systems, and enhance the interest of working with *FB*.

References

1. Irène Durand. Bounded, strongly sequential and forward-branching term rewriting systems. *Journal of Symbolic Computation*, 18(4):319–352, October 1994.
2. Irène Durand and Bruno Salinier. Constructor equivalent term rewriting systems. *Inform. Processing Letters*, 47:131–137, 1993.
3. Irène Durand and Bruno Salinier. Constructor equivalent term rewriting systems are strongly sequential: a direct proof. *Inform. Processing Letters*, 52:137–145, 1994.
4. Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal term rewriting systems. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, chapter 11–12, pages 397–443. MIT Press, 1991.
5. Jan Willem Klop and Aart Middeldorp. Sequentiality in orthogonal term rewriting systems. *Journal of Symbolic Computation*, 12:161–195, 1991.
6. Bruno Salinier. *Simulation de systèmes de réécriture de termes par des systèmes constructeurs.* Thèse de doctorat, Université Bordeaux I, 351 cours de la Libération, 33405 Talence, France, December 1995.
7. Robert I. Strandh. Classes of equational programs that compile into efficient machine code. In *RTA 89, Rewriting Techniques and Applications*, volume 355 of *LNCS*, pages 449–461. Springer-Verlag, 1989.
8. Satish Thatte. On the correspondence between two classes of reduction systems. *Inform. Processing Letters*, 20:83–85, 1985.