INFORMATIQUE THÉORIQUE ET APPLICATIONS

M. BELLIA

Logic and functional programming by retractions : operational semantics

Informatique théorique et applications, tome 22, n° 4 (1988), p. 395-436

<http://www.numdam.org/item?id=ITA_1988__22_4_395_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 22, n° 4, 1988, p. 395 à 436)

LOGIC AND FUNCTIONAL PROGRAMMING BY RETRACTIONS: OPERATIONAL SEMANTICS (*)

by M. BELLIA (1)

Communicated by G. LONGO

Abstract. – *Retractions are featuring predicates so that, to each predicate defined by a Horn theory we associated a retraction of a set theory built according to the structure of the Herbrand Universe,* HU_c. *The set theory allowed a set theoretic interpretation of the Herbrand terms and supplied them with a combinatory formulation,* constant expression.

In the present paper we discuss normal forms for constant expressions and define a system of rewrite rules which reduces constant expressions to normal forms. This set of rewrite rules toghether with the rules for α, β, *and Y reduction of Lambda terms, forms also an operational semantic of our calculus with retractions. The reduction system is finally considered in order to investigate relations between inference on logic formulas and reduction on this kind of combinatory forms.*

Résumé. – *Les rétractions modélisent les prédicats de la programmation logique de façon que à chaque prédicat défini par une théorie de Horn, on associe une rétraction d'une théorie des ensembles qui est construite en accord avec la structure de l'Univers d'Herbrand,* HU_c. *Cette théorie permet une interprétation théorique des termes d'Herbrand en fournissant une formulation combinatoire,* l'expression constante. *Dans cette article nous traitons les formes normales pour les expressions constantes et nous définissons un système de règles de réécriture qui réduit les expressions constantes aux formes normales. Le système de réécriture muni des règles de réduction* α, β *et Y du Lambda-calcul au premier ordre forme aussi une sémantique opérationnelle de notre calcul avec les rétractions. Ce système est enfin utilisé pour étudier les relations entre l'inférence sur les formules logiques et la réduction de ce type de formes combinatoires.*

1. INTRODUCTION

[Bellia88] considered the definition of a functional paradigm for logic programming. It identified retractions as the most primitive concept which relates predicates and functions. Retractions are featuring predicates so that to each predicate defined by a set of Horn clauses we associate a retraction

(*) Received in 1986
This work was partially supported by the European Community under ESPRIT Project 415.
(1) Dipartimento di Informatica, Universita' di Pisa Corso Italia, 40-56100 Pisa, Italy.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 88/04 395 42/$6.20/

on some cartesian product of the Herbrand Universe, HU_c. The approach has been discussed in a functional language which is essentially Church's Lambda calculus restricted to first order and extended with a symbolic data domain, HU_c^*. The data domain has been completely but in abstract way defined by giving an axiomatization of its operators. In the present paper, we discuss the operational, machine oriented, definitions of the data domain operators.

In our language programs are *closed forms* (i.e. λ-abstractions without occurrences of free variables). For example, in the extended syntax that we introduced in [Bellia88], the following:

$$f1(x)=x+y \qquad \text{where} \quad 1=y;$$
$$f2(x, y)=f1(x)+y$$

is a program which declares $f1$ and $f2$ as the functions denoted by the closed forms $\lambda x.((\lambda y.x+y)\ 1)$ and $\lambda xy.((\lambda x.((\lambda y.x+y)\ 1))\ x)+y)$, respectively. Expressions are applications of λ-abstractions to data, and evaluations are α, β and Y expression reductions. For example, in the extended syntax, the following expression,

$$7\ {}_*\ x \qquad \text{where} \quad f2(2, 3)=x,$$

which corresponds to the application

$$((\lambda x.7\ {}_*\ x)((\lambda xy.((\lambda x.((\lambda y.x+y)\ 1))\ x)+y)\ 2\ 3)),$$

after 4 β-reductions, results into the expression $7\ {}_*((2+1)+3)$. The expression could be further reduced according to the structure of the data domain and the semantics of the symbols occurring in the expression. The semantics of the domain operators can be embodied into an equivalence relation. Therefore, we can define evaluations as α, β and Y reductions modulo the equivalence relation. For example, if the data domain contains 7, 2, 1, 3 as Integers and the equivalence relation interprets ${}_*$ and $+$ as the product and the addition operations on Integers, then the above expression is further reduced to the integer 42. Our language data domain is more complex than the domain of the Integers and our operators compute with data which denote (possibly infinite) sets. Therefore, though α, β and Y reductions are familiar in functional languages and an equivalence relation, $\approx$, has been completely and formally stated for our set operators [Bellia88] the operational semantics of

the operators should be discussed in order to clarify:

– the relations between α, β, Y reductions and evaluations of expressions which contain set operators;

– the operational semantics of set intersection, **Intset**, and;

– the computer architecture of the language machine support.

2. INSTANCES AND EXPRESSION EVALUATION

We remember some well known facts. First, *external evaluation rules* [Stoy77] on λ-expressions have to be used to guarantee a finite sequence of Y-reductions. Second, operators have to be *lazy* [Friedman76, Henderson76] to handle operators on possibly infinite data structures (such as the elements of $HU\,\omega$). These features are both achieved in a *demand-driven* and *call-by-need* [Henderson80] evaluation strategy on the selection of subexpressions and in their evaluations [Bellia84]. By using this strategy, we could compute for instance, the expression

$$\textbf{Intset}(\textbf{In}(2,\ \pi),\ \otimes\pi,\ \pi\otimes) \tag{1}$$

by finitely many approximations as needed by the *main* (computation). For example, according to the following lazy implementation of **Intset**

$$\begin{aligned}\textbf{Intset}(t_1 \bullet t_1',\ t_2 \bullet t_2') = {} & \textbf{if card}(t_1) = 1 \textit{ and } \textbf{card}(t_2) = 1 \text{ and } t_1 = t_2 \\ & \textbf{then } t_1 \bullet \textbf{Intset}(t_1',\ t_2') \\ & \textbf{else Intset}(t_1 \bullet t_1',\ t_2') \bullet \textbf{Intset}(t_1',\ t_2 \bullet t_2') \\ \textbf{Intset}(\varnothing,\ t) = {} & \varnothing\end{aligned} \tag{2}$$

the evaluation of (1) results into the expression:

$$\langle \underline{0},\ \underline{0} \rangle \bullet \textbf{Intset}(\textbf{In}(2,\ \mathbf{S}(\pi)),\ \otimes \underline{0},\ \mathbf{S}(\pi) \otimes \bullet \otimes \mathbf{S}(\pi),\ \underline{0} \otimes \bullet \otimes \mathbf{S}(\pi),\ \mathbf{S}(\pi) \otimes). \tag{3}$$

Note that (2) imposes that both t_1 and t_2 are elements of HU_T. Therefore, the application of **Intset** forces the evaluation of both the expressions $\textbf{In}(2,\ \pi)$ and $\otimes\pi,\ \pi\otimes$.

Implementations like (2) for our operators are operationally realistic and easy to design. They recursively enumerate all the finite approximations we

need. However, they are inadequate. Let us consider the expression

$$\mathbf{Intset}(\mathbf{S}(2, \otimes\pi, \pi\otimes), \mathbf{In}(2, \pi)). \tag{4}$$

If we use (2) to compute (4) we could indefinitely look for some value (different from the empty set, $\varnothing$) to be computed. (2) implements a semi-decision procedure to compute set intersection when applied to expressions which denote infinite sets (i. e. elements of HU_ω) even if they are constant expressions. However, as pointed out in [Bellia88], *constant expressions* are combinatory formulas which denote a special class of (possibly infinite) sets. The class is a subclass of all the recursive subsets of elements of HU_T and, more important, this class is closed under set intersection, that is if E and E' are constant expressions, **Intset** (E, E') could be expressed by some constant expression E''. This is a consequence of the Property 5.1 in [Bellia88] which states that **Intset**, applied to two constant expressions, $E1$ and $E2$, corresponds to the computation of the *most general instance*, Mgi, on the tuples of Herbrand terms $H1$ and $H2$ such that

$$E1 = \eta(H1) \qquad \text{and} \qquad E2 = \eta(H2).$$

Mgi is a byproduct of the unification algorithm (hence the existence of the Mgi of two tuples of Herbrand terms is decidable), while η is a meaning preserving map from Herbrand terms into constant expressions. For example, in the expression **Intset** $(\otimes\pi, \pi\otimes, \mathbf{In}(2, \pi))$, we have

$$E1 = \otimes\pi, \pi\otimes \qquad \text{and} \qquad E2 = \mathbf{In}(2, \pi).$$

By Property 5.1 and because

$$\otimes\pi, \pi\otimes = \eta(\underline{x, y}) \qquad \text{and} \qquad \mathbf{In}(2, \pi) = \eta(\underline{z, z}),$$

Intset $(\otimes\pi, \pi\otimes, \mathbf{In}(2, \pi))$ corresponds to the computation of the Mgi of the tuple $\underline{x, y}$ with $\underline{x, x}$, i. e.

$$\mathbf{Intset}(\otimes\pi, \pi\otimes, \mathbf{In}(2, \pi)) = \eta(\underline{x, x}),$$

and because

$$\mathbf{In}(2, \pi) = \eta(\underline{x, x}),$$

$$\mathbf{Intset}(\otimes\pi, \pi\otimes, \mathbf{In}(2, \pi)) = \mathbf{In}(2, \pi).$$

The expression **Intset** $(\otimes\pi, \pi\otimes, \mathbf{In}(2, \pi))$ should be reduced to the constant expression $\mathbf{In}(2, \pi)$ which, according to $\approx$, could be further reduced to the

constant expression $\langle \underline{0, 0} \rangle \bullet \mathbf{In}(2, \mathbf{S}(\pi))$. It computes as (3) involving only constant expressions.

As another example, consider the expression (4).

$$E1 = \mathbf{S}(2, \otimes\pi, \pi\otimes) \qquad \text{and} \qquad E2 = \mathbf{In}(2, \pi),$$

then by Property 5.1 and because

$$\mathbf{S}(2, \otimes\pi, \pi\otimes) = \eta(\underline{x, S(y)}) \qquad \text{and} \qquad \mathbf{In}(2, \pi) = \eta(\underline{z, z}),$$

$$\mathbf{Intset}(\mathbf{S}(2, \otimes\pi, \pi\otimes), \mathbf{In}(2, \pi)) = \varnothing$$

since the Mgi of $\underline{x, S(y)}$ and $\underline{z, z}$ does not exist.

The problem is that many relevant properties, which could be deduced from the axiomatization of our set operators and which relate **Intset** to the other operators, are lost in the brute force implementation of **Intset** by (2). We would like to have an implementation for **Intset** such that, for each tuples of Herbrand terms,

$$H1 = h_1, \ldots, h_n \qquad \text{and} \qquad H2 = h'_1, \ldots, h'_n,$$

if there exists Mgu ϑ such that:

$$\varphi(h_1, \ldots, h_n).\vartheta = \varphi(h'_1, \ldots, h'_n).\vartheta$$

then

$$\mathbf{Inset}(\eta(h_1, \ldots, h_n), \eta(h'_1, \ldots, h'_n)) = \eta(h_1\vartheta, \ldots, h_n\vartheta)/\approx$$

otherwise, **Inset** finitely computes $\varnothing$. This means: firstly, that evaluations of **Intset** compute finitely when applied to constant expressions, and finally each class of congruent constant expressions has some representantive, i. e. normal form. **Inset** computes these representantives.

We give it in two different ways. The first solution is based on the existence of the function ρ_e. It is a weak form of η^{-1} and associates to each constant expression (including expressions which contain applications of $\bullet$, **Pr** and constructor inverses) a finite structure of Herbrand terms which (according to the set interpretation of Herbrand terms [Bellia88]) denotes the same set.

Moreover, ρ_e is unique, and therefore it maps all the constant expressions which are $\approx$-congruent into the same structure of Herbrand terms. Hence, we can use the Mgi, computed by a straightforward simplification of the unification algorithm, to implement **Intset**. This solution will be considered in Sections 3 and 4.

Moreover, we will see that constant expressions have *normal form*. Thus we can reduce constant expressions to normal forms and, then compare normal forms to decide if two different constant expressions compute the same set. A rather different solution to the implementation of **Inset** exploting normal forms, is proposed in Section 5.

3. HERBRAND TERMS AND CONSTANT EXPRESSIONS

We introduce the function ρ_e. It associates to each constant expression which does not contain applications of $\bullet$, exactly one tuple of Herbrand terms. The tuple denotes the same set which is denoted by the constant expression. For instance, each π which occurs in a constant expression is mapped into a different variable symbol. Moreover, ρ_e maps applications of $\bullet$ into finite sequences of tuples of Herbrand terms. We will denote sequences of tuples of Herbrand terms by the sequence operator $+$.

DEFINITION 3.1 (ρ_e):

1- (constructors of arity 0)

$$\rho_e(c)=c.$$

2- (application of $\langle - \rangle$)

$$\rho_e(\langle c_1, \ldots, c_n \rangle)=\rho_e(c_1), \ldots, \rho_e(c_n).$$

3- (π and $\varnothing$)

$$\rho_e(\pi)=x, \qquad \rho_e(\varnothing)=\varnothing.$$

4- (application of $\bullet$)

$$\rho_e(E_1 \bullet \ldots \bullet E_n)=\rho_e(E_1)+\ldots+\rho_e(E_n).$$

5- (application of $\otimes - \otimes$)

$$\rho_e(\otimes E_1, \ldots, E_n \otimes)=H_1+\ldots+H_k$$

where if

$$\rho_e(E_1)=H\,1_1+\ldots+H\,1_{m\,1}, \ldots, \rho_e(E_n)=H\,n_1+\ldots+H\,n_{mn}$$

then,

$k=m\,1\times\ldots\times mn$ and, for each $w_1, \ldots, w_n$,

such that $w_i=H\,i_j$ for some $j\in[1, mi]$,

$$H_l=w_1, \ldots, w_n \quad \text{for some } l\in[1, k].$$

6- (application of **In**)

$$\rho_e(\mathbf{In}(k, E))=H_1+\ldots+H_n$$

where if

$$\rho_e(E) = H'_1 + \ldots + H'_n$$

then

H_j is the tuple which contains k times the tuple H'_j (i. e. $H_j = H'_j, \ldots, H'_j$).

7- (constructors of arity > 0)

$$\rho_e(\mathbf{c}_k(j, E)) = H_1 + \ldots + H_n$$

where if

$$\rho_e(E) = H'_1 + \ldots + H'_n$$

with H'_i such that

$$H'_i = t_1, \ldots, t_{j-1}, t_j, \ldots, t_{j+k-1}, \ldots, t_m$$

then

$$H_i = t_1, \ldots, t_{j-1}, \underline{C_k}(t_j, \ldots, t_{j+k-1}), \ldots, t_m$$

8- (constructor inverses)

$$\rho_e(\mathbf{c}_k \downarrow (j, E)) = H_1 + \ldots + H_n$$

where if

$$\rho_e(E) = H'_1 + \ldots + H'_n,$$

with H'_i such that:

$$- H'_i = t_1, \ldots, t_{j-1}, \underline{C_k}(t_j, \ldots, t_{j+k-1}), \ldots, t_m$$

then

$$H_i = t_1, \ldots, t_{j-1}, t_j, \ldots, t_{j+k-1}, \ldots, t_m$$

$$- H'_i = t_1, \ldots, t_{j-1}, x, \ldots, t_m$$

then

$$H_i = t'_1, \ldots, t'_{j-1}, x_1, \ldots, x_k, \ldots, t'_m$$

where $x_1, \ldots, x_k$ are k different variable symbols and,

$$t'_r = t_r[x \leftarrow \mathbf{c}_k(x_1, \ldots, x_k)]$$

$- H_i = \varnothing$, otherwise.

($t[x \leftarrow h]$ is the term t where each occurrence of x is replaced by h).

9- (application of **Pr**)

$$\rho_e(\mathbf{Pr}(j, k, E)) = H_1 + \ldots + H_n$$

where if

$$\rho_e(E) = H'_1 + \ldots + H'_n \qquad \text{with} \quad H'_i$$

such that:

$-H'_i = t_1, \ldots, t_{j-1}, t_j, \ldots, t_{j+k-1}, \ldots, t_m$ and for no $r \in [1, m]$, $t_r = \varnothing$

then

$$H_i = t_j, \ldots, t_{j+k-1}$$

$-H'_i = t_1, \ldots, t_{j-1}, t_j, \ldots, t_{j+k-1}, \ldots, t_m$ and for some $r \in [1, m]$, $t_r = \varnothing$

then

$$H_i = \varnothing.$$

10- (application of **Pe**)

$$\rho_e(\mathbf{Pe}(n_1 \ldots n_k, E)) = H_1 + \ldots + H_n$$

where if

$$\rho_e(E) = H'_1 + \ldots + H'_n, \qquad \text{with} \quad H'_i = t_1, \ldots, t_k$$

then

$$H_i = tn_1, \ldots, tn_k.$$

Example 3.1: Let $C = \{\underline{\text{NIL}_0}, \underline{S_1}, \underline{CONS_2}\}$, the computation of

$$\rho_e(\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1 \downarrow (4, \mathbf{Pe}(1\ 3\ 4\ 2\ 5, \otimes \mathbf{In}(2, \pi), \pi, \pi, \pi \otimes)))))$$

proceeds as follows:

- $\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1 \downarrow (4, \mathbf{Pe}(1\ 3\ 4\ 2\ 5, \otimes \mathbf{In}(2, \underline{x}), \underline{y}, \underline{z}, \underline{w} \otimes))))$;
- $\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1 \downarrow (4, \mathbf{Pe}(1\ 3\ 4\ 2\ 5, \underline{x}, \underline{x}, \underline{y}, \underline{z}, \underline{w}))))$;
- $\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1 \downarrow (4, \underline{x}, \underline{y}, \underline{z}, \underline{x}, \underline{w})))$;
- $\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1(\underline{x}\,1), \underline{y}, \underline{z}, \underline{x}\,1, \underline{w}))$;
- $\mathbf{cons}_2(1, \underline{S_1}(\underline{x}\,1), \underline{y}, \underline{z}, \underline{CONS_2}(\underline{x}\,1, \underline{w}))$;
- $\underline{CONS_2}(\underline{S_1}(\underline{x}\,1), \underline{y}), \underline{z}, \underline{CONS_2}(\underline{x}\,1, \underline{w}))$;

Note that the result is a triple of terms.

PROPOSITION 3.1: *Under the interpretation of the sequence operator, +, as set union,* ρ_e *is a* meaning preserving *map, that is* $\rho_e(E) = H_1 + \ldots + H_n$ *implies that for each* $u \in \mathrm{HU}_T$, *u is a member of E if and only if u is a ground instance of* H_i *for some* $i \in [1, n]$, *i.e.* $\rho_e(E)$ *and* $H_1 + \ldots + H_n$ *denote the same set.*

Proof: In the case of constant expressions which are $\varnothing$ or π or 0-arity constructors, or $\langle - \rangle$ applied to 0-arity constructors, it is immediate to prove that each ground Herbrand term is ground instance of $\rho_e(E)$ if and only if it is a member of E.

To prove the Proposition in the other cases, we can use induction on the structure of constant expressions. Moreover, note that $+$ comes from the presence of applications of $\bullet$, and $\bullet$ is an *endomorphic* operator on HU_c^*. Therefore each E containing k applications of $\bullet$ can be expressed by $E1 \bullet \ldots \bullet Ek$ such that each Ei does not contain applications of $\bullet$.

Assume $E_1, \ldots, E_n$ be constant expressions which do not contain applications of $\bullet$, i.e. $\rho_e(E_i) = H_i'$, and such that E_i and H_i' denote the same set. Then

4– $$\rho_e(E_1 \bullet \ldots \bullet E_n) = H_1' + \ldots + H_n'$$

and,

$$E_1 \bullet \ldots \bullet E_n \qquad \text{and} \qquad H_1' + \ldots + H_n'$$

denote the same set because of the definition of $\bullet$, the interpretation on $+$ and the assumptions about the components E_i and H_i'.

5– $\rho_e(\otimes E_1, \ldots, E_n \otimes) = H_1', \ldots, H_k'$, where $H_1', \ldots, H_k'$ is a tuple of Herbrand terms, and from the definition of $\otimes - \otimes$ immediately follows that $\otimes E_1, \ldots, E_n \otimes$ and $H_1', \ldots, H_k'$ denote the same set.

In case of **In**, $\mathbf{c}_k$ and **Pe**, cases 6, 7 and 10, the proof proceeds exactly as in $\otimes - \otimes$.

We show now the proposition in the case of constructor inverses. The proof is similar in the case of **Pr**.

8– $\rho_e(\mathbf{c}_k \downarrow (j, E_i)) = H_i$ where H_i is such that:

– if $H_i' = t_1, \ldots, t_{j-1}, \underline{C_k}(t_j, \ldots, t_{j+k-1}), \ldots, t_m$

then $H_i = t_1, \ldots, t_{j-1}, t_j, \ldots, t_{j+k-1}, \ldots, t_m$, and by definition of $\mathbf{c}_k \downarrow$, u is a member of $\mathbf{c}_k \downarrow (j, E_i)$ if and only if u has the form $\langle u_1, \ldots, u_{j-1}, u_j, \ldots, u_{j+k-1}, \ldots, u_m \rangle$ and for each $i \notin [j, j+k-1]$, u_i is member of the projection of $\mathbf{c}_k \downarrow (j, E_i)$ on i, i.e. u_i is a ground instance of the i-th component of the tuple H_i', and hence of H_i. Furthermore, for each $i \in [j, j+k-1]$, u_i is the $(i-j+1)$-th argument of an application of $\mathbf{c}_k$, hence u_i is a ground instance of t_{i-j+1}.

– if $H_i' = t_1, \ldots, t_{j-1}, x, \ldots, t_m$ then by definition of $\mathbf{c}_k \downarrow$, u is a member of $\mathbf{c}_k \downarrow (j, E_i)$, if and only if u has the form $\langle u_1, \ldots, u_{j-1}, u_j, \ldots, u_{j+k-1}, \ldots, u_m \rangle$ and there exists a ground instance $v = v_1, \ldots, v_j, v_{j+k}, \ldots, v_m$ of H_i' such that $u_i = v_i$ for each $i \notin [j, j+k-1]$ and,

$v_j = \underline{C}_k(u_j, \ldots, u_{j+k-1})$. But if such an instance exists, let ϑ be the ground instantiation function.

Then the function ϑ' such that $\vartheta'(\underline{x}_i) = u_{i-j+1}$ for each variable in $\{\underline{x}_1, \ldots, \underline{x}_k\}$, and $\vartheta'(x) = \vartheta(x)$ otherwise, is a ground instantiation function of H and is such that $H.\vartheta' = u$.

Note that, in constrast to η, ρ_e is unique. This is due to the following property on Herbrand terms.

PROPERTY 3.1: If h and h' are two Herbrand terms, $h.\vartheta = h'.\vartheta$ for each (ground) instantiation function ϑ, if and only if h is a renaming of h'.

Therefore the following proposition holds.

PROPOSITION 3.2: *For each pair of constant expressions E and E':*

$$E \approx E' \text{ iff } \rho_e(E) \text{ equals } \rho_e(E') \text{ modulo a variable renaming.}$$

Proof: Follows immediately from Proposition 3.1 and Property 3.1.

PROPOSITION 3.3: *For each pair of constant expressions, E_1 and E_2, such that*

$$\rho_e(E_1) = H_1 + \ldots + H_{n1} \qquad \text{and} \qquad \rho_e(E_2) = H'_1 + \ldots + H'_{n2},$$

then

$$\textbf{intset}(E_1, E_2) = \eta(H''_1) + \ldots + \eta(H''_{n3})$$

where

$$H''_1 + \ldots + H''_{n3}$$

is such that:

$$\forall i \in [1, n1], \quad \forall j \in [1, n2],$$

$$\exists \text{Mgu } \vartheta i_j \text{ such that } H_i.\vartheta i_j = H'_j.\vartheta i_j \text{ iff } H''_k = H_i.\vartheta i_j \text{ for some } k \in [1, n3].$$

Proof: Follows immediately from Property 5.1 on **Intset** [Bellia88] and from Proposition 3.1.

4. THE MOST GENERAL INSTANCE OF HERBRAND TERMS

Proposition 3.3 states that, in order to implement **intset** on constant expressions, we can map the expressions into the corresponding (finite) sequences of tuples of Herbrand terms. Then, we compute the most general instance, if any, of each pair of tuples respectively in the first and in the second sequence. Finally, we apply the function η to each instance and collect

them through applications of •. Note that, the specific function chosen for η is unessential to the approach.

We reformulate the Robinson's Unification Algorithm in order to compute only the most general instance of two tuples of Herbrand terms.

DEFINITION 4.1 (Most general instance of tuples of Herbrand terms, *Mgi*):

1. Let $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ be two tuples with no colliding variables [that is, intersection of $\text{var}(p_1) \cup \ldots \cup \text{var}(p_n)$ with $\text{var}(q_1) \cup \ldots \cup \text{var}(q_n)$ is empty], otherwise a renaming is provided.

2. If for each $i \in [1, n]$, $p_i = q_i$ then **stop** with $\eta(p_1, \ldots, p_n)$.

3. Otherwise, let t_p and t_q be the first two sub-terms (in left-to-right and top down visit of lists) which are different:

3.1. if one of them is a variable, let t_p be the variable of name x:

(*a*) if t_q is a term different from a variable and is not containing x, then

$$\text{replace each } p_i \text{ with } p_i[x \leftarrow t_q] \text{ and}$$
$$\text{each } q_i \text{ with } q_i[x \leftarrow t_q]$$

(*b*) if t_q is a term different from a variable and is containing x, then **stop** with $\varnothing$

(*c*) if t_q is the variable of name y, let w be a new variable name, then

$$\text{replace each } p_i \text{ with } p_i[x, y \leftarrow w] \text{ and}$$
$$\text{each } q_i \text{ with } q_i[x, y \leftarrow w].$$

3.2. otherwise: **stop** with $\varnothing$

4. **Repeat** steps **2.** and **3.**

PROPOSITION 4.1: *Let*

$$H1 = h1_1, \ldots, h1_k \quad \textit{and} \quad H2 = h2_1, \ldots, h2_k$$

be two **k***-tuples of Herbrand terms and,*

$$\mathbf{f} = \max\{v(h1_i), v(h2_j) \mid i, j \in [1, k]\},$$

where $v(h)$ *is the number of variable occurrences in the term* h*;*

$$\mathbf{c} = \max\{c(h1_i), c(h2_j) \mid i, j \in [1, k]\},$$

where $c(h)$ *is the number of constructors in the term* h*;*

$$\mathbf{x} = \max\{V_D(H1), V_D(H2)\},$$

where $V_D(H)$ is the number of different variables symbols occurring in the tuple H, then if there exists $\mathrm{Mgi}(H1, H2)$, *it is such that:*

$$C_D(\mathrm{Mgi}(H1, H2)) \leqq \mathbf{k}_*((\mathbf{f}+1)^{\mathbf{x}-1}{}_*\mathbf{c})$$

where $C_D(H)$ is the number of constructors occurring in the tuple H.

Proof: It is a consequence of the *occur-check* given in steps 3.1 *a* and 3.1 *b* of the Mgi algorithm. At each step 3, either the tuples structure are left unchanged by 3.1 *c* (only a variable renaming is produced) or some variable y is replaced by a term t_p such that $c(t_p) \leqq (\mathbf{f}+1)_*\mathbf{c}$ and $v(t_p) \leqq (\mathbf{x}-1)-r$, if y has been produced in the current term after r applications of step 3 to that term.

Example 4.1: Let us consider the computation of:

$$\mathbf{Intset}(\mathbf{In}(2, \pi), \otimes\pi, \pi\otimes)$$

by ρ_e:

$$\rho_e(\mathbf{In}(2, \pi)) = \underline{x, y}, \qquad \rho_e(\otimes\pi, \pi\otimes) = \underline{x, x}$$

by Mgi step 1 (renaming):

$$p_1, p_2 = \underline{x1, y}, \qquad q_1, q_2 = \underline{x2, x2}$$

step 3.1 *c*:

$$p_1, p_2 = \underline{w1, y}, \qquad q_1, q_2 = \underline{w1, w1};$$

step 3.1 *c*:

$$p_1, p_2 = \underline{w2, w2}, \qquad q_1, q_2 = \underline{w2, w2}.$$

step 2:

$$\mathbf{stop} \text{ with } \eta(\underline{w2, w2}),$$

which (according the definition of η) results $\mathbf{In}(2, \pi)$.

4.1. An example of evaluation based on Mgi

Defined ρ_e, η, and the Mgi algorithm, we can now complete the language semantics with the following operational semantics for our operators.

In order to evaluate the expression $E = \mathrm{op}(E1, \ldots, En)$, where op is the primitive operator:

•: We compute $E1$. If $E1$ results different from $\varnothing$, then we return it and the computation of $\mathrm{op}(E2, \ldots, En)$ is *suspended* until a further evaluation of E is required. Otherwise, the evaluation proceeds with $\mathrm{op}(E2, \ldots, En)$ if $n > 1$, or returns $\varnothing$.

$\langle - \rangle$, $\otimes - \otimes$, $\mathbf{c}_k$, $\mathbf{c}_k\downarrow$, **Pe**, **Pr**, **In**: We compute each Ei in order to obtain a constant expression.

Intset: If $n > 2$, we arbitrarily select a pair of arguments, Ei and Ej, and replace it with the constant expression resulting from the computation of **Intset** (Ei, Ej).

If $n = 2$, we compute both $E1$ and $E2$, which, due to the implementation of •, could result in a constant expression (which does not contain •) and in a *suspension.* If a suspension is returned, a corresponding suspension is generated for **Intset**. In any case, ρ_e is applied to the constant expressions resulting from the evaluation of $E1$ and $E2$, and then the Mgi algorithm is applied to the resulting tuples of Herbrand terms.

We return the result of Mgi (and, possibly a suspension).

Note that in this semantics all the primitive operators are lazy. Moreover, computations are driven by the request for a value which possibly results in a data and a suspension. The suspension is activated if the computed data is not sufficient for the request.

As an example, consider the following program

$$F_{\mathrm{ADD}}(w) = u \bullet v$$

where

$$\mathbf{Intset}(w, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes)) = u,$$

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w', \qquad \mathbf{Intset}(w, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z.$$

It defines a function from HU_3^* *to* HU_3^*, *which is the retraction* [Bellia88] *for the predicate* ADD, *defined by the following PROLOG-like program* [Kowalski74]

$$\mathrm{ADD}(\underline{x, 0, x}) \leftarrow ., \qquad \mathrm{ADD}(\underline{x, S(y), S(z)}) \leftarrow \mathrm{ADD}(\underline{x, y}). \tag{5}$$

In SuperLOGLISP [Berkling82], ADD *can be defined by the following program*

$$\mathrm{ADD} = \{\langle x, 0, z\rangle \mid x = z\} \cup \{\langle x, S(y), S(z)\rangle \mid \mathrm{ADD}(x, y, z)\}. \quad (6)$$

The expression $F_{\mathrm{ADD}}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes)$ *corresponds to the query* ADD $(\underline{S(x)}, \underline{S(0)}, z)$, *and its evaluation proceeds as follows.*

1. *by Y and* β *reductions:*

$u \bullet v$ *where*

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes) = u,$$

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w', \mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z.$$

To compute $u \bullet v$, *we compute u which is bound to:*

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes),$$

which contains only constant expressions. Then

2. by **Intset:**

$$\rho_e(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes) = \underline{S(x), S(0), z},$$

$$\rho_e(\mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes) = \underline{x, 0, x}$$

$$\mathrm{Mgi}(\underline{S(x), S(0), z}, \underline{x, 0, x}) = \varnothing$$

then,

$\varnothing \bullet v$ *where*

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w', \mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z.$$

$\varnothing \bullet v$ *is reduced to v, and the computation proceeds through the computation of v. Variable v is not bound to a constant expression. Therefore*

3. *Y and* β *reductions:*

$\mathbf{S}(2, \mathbf{S}(3, x))$ *where*

$u \bullet v = x$ *where*

$$\mathbf{Intset}(w', \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes) = u,$$

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w'',$$

$$\mathbf{Intset}(w', \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = \mathrm{w}',$$

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z.$$

To compute u we have to compute

$$\mathbf{Intset}(w', \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), 0 \otimes),$$

but w′ is not bound to a constant expression. Therefore

4. by **intset**:

$$\rho_{\mathbf{e}}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes) = \underline{S(x), S(0), z},$$

$$\rho_{\mathbf{e}}(\mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = \underline{x, S(y), S(z)}$$

$$\mathrm{Mgi}(\underline{S(x), S(0), z, x, S(y), S(z)}) = \mathbf{S}(1, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \underline{0}, \pi \otimes))$$

then,

$\mathbf{S}(2, \mathbf{S}(3, x))$

where

$$u \bullet v = x$$

where

$$\mathbf{Intset}(w', \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes) = u,$$

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w'',$$

$$\mathbf{Intset}(w', \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, \mathbf{S}(1, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \underline{0}, \pi \otimes))))) = w',$$

then

5. by **intset**:

$$\rho_e(\mathbf{S}\downarrow(2,\ \mathbf{S}\downarrow(3,\ \mathbf{S}(1,\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \underline{0},\ \pi\otimes)))))=\underline{S(x),\ 0,\ z}$$

$$\rho_e(\mathbf{Pe}(1\,3\,2,\ \otimes\mathbf{In}(2,\ \pi),\ \underline{0}\otimes)=\underline{x,\ 0,\ x}$$

$$\mathbf{Mgi}(\underline{S(x),\ 0,\ z},\ \underline{x,\ 0,\ x})=\mathbf{S}(1,\ \mathbf{S}(3,\ \mathbf{Pe}(1\,3\,2,\ \otimes\mathbf{In}(2,\ \pi),\ \underline{0}\otimes$$

then,

$$\mathbf{S}(2,\ \mathbf{S}(3,\ x))$$

where

$$\mathbf{S}(1,\ \mathbf{S}(3,\ \mathbf{Pe}(1\,3\,2,\ \otimes\mathbf{In}(2,\ \pi),\ \underline{0}\otimes\ \bullet\ v=x$$

where

$$\mathbf{S}(2,\ \mathbf{S}(3,\ \mathbf{F}_{\mathbf{ADD}}(w'')))=v$$

where

$$\mathbf{S}\downarrow(2,\ \mathbf{S}\downarrow(3,\ z))=w'',$$

$$\mathbf{Intset}(w',\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \pi,\ \pi\otimes)))=z$$

where

$$\mathbf{S}\downarrow(2,\ \mathbf{S}\downarrow(3,\ \mathbf{S}(1,\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \underline{0},\ \pi\otimes)))))=w'$$

which results into the value:

$$\mathbf{S}(1,\ \mathbf{S}(2,\ \mathbf{S}(3,\ \mathbf{S}(3,\ \mathbf{Pe}(1\,3\,2,\ \otimes\mathbf{In}(2,\ \pi),\ \underline{0}\otimes))))),$$

and into the suspension:

$$(2,\mathbf{S}(3,\ \mathrm{x}))$$

where

$$\mathbf{S}(2,\ \mathbf{S}(3,\ \mathbf{F}_{\mathbf{ADD}}(w'')))=x$$

where

$$\mathbf{S}\downarrow(2,\ \mathbf{S}\downarrow(3,\ z))=w'',$$

$$\mathbf{Intset}(w',\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \pi,\ \pi\otimes)))=z$$

where

$$\mathbf{S}\downarrow(2,\ \mathbf{S}\downarrow(3,\ \mathbf{S}(1,\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \underline{0},\ \pi\otimes)))))=w'.$$

Note that in step 4 **Intset** reduces z to the expression $\mathbf{S}(1,\ \mathbf{S}(2,\ \mathbf{S}(3,\ \otimes\pi,\ \underline{0},\ \pi\otimes))$ which is just the set of all the triples in $\underline{S(x),\ S(0),\ z}$ which are also in $\underline{x,\ S(y),\ S(z)}$. In a *SLD resolution* [Apt82] of $\underline{\mathrm{ADD}(S(x),\ S(0),\ z)}$ with (5), step 4 corresponds to resolve $\underline{\mathrm{ADD}(S(x),\ S(0),\ z)}$ with the second clause in (5) and to unify the terms: $\underline{S(x)}$ with $\underline{x}$, and $\underline{S(0)}$ with $\underline{S(y)}$, and $\underline{z}$ with $\underline{S(z)}$. In *ε-reduction* [Berkling85] of $\underline{\mathrm{ADD}(S(x),\ S(0),\ z)}$ with

ADD defined as in (6), step 4 corresponds to the insertion of the equations $S(x)=x'$, $0=y'$ and $z=S(z')$ in the redex:

$$\{\langle x', S(y'), S(z')\rangle \mid S(x)=x', 0=y', z=S(z'), \mathrm{ADD}(x', y', z')\}.$$

5. A NORMAL FORM BASED OPERATIONAL SEMANTICS

In [Bellia88], we shown that constant expressions are combinatory forms for the tuples of Herbrand terms and, that tuples of Herbrand terms are less than constant expressions, i. e. there is a one-to-many correspondence between tuples of Herbrand terms and constant expressions. For instance, we saw that to each tuple of Herbrand terms we can associate infinite different but equivalent constant expressions. However, in Section 3 the uniqueness of ρ_e and Proposition 3.3 show that to each constant expression we can uniquely associate a structure of Herbrand terms. This structure is a (finite) sequence of tuples of Herbrand terms and, more important, is unique (modulo variable renamings) for all the infinite $\approx$ – congruent constant expressions. In some way, ρ_e induces a *normal form* on constant expressions, and, therefore, the Mgi algorithm correctly implemented **Intset**.

5.1. Normal forms and the reduction system, R

We now show that constant expressions have normal form and therefore, the operational semantics of **Intset** can be reformulated in terms of comparisons of normal forms and of (sub)expression reductions. The result will be a reduction system for the set operators of our language.

Notational remark ($E[i]$): If E is a constant expression of the form $\otimes E'_1, \ldots, E'_m \otimes$, in the following we will use $E[i]$ to denote the subexpression E'_r such that $\sum_{p=1, r-1} \# E'_p < i \leqq \sum_{p=1, r} \# E'_p$.

Analogously, if $E=\mathbf{Pe}(n!, \otimes E'_1, \ldots, E'_m \otimes)$, then $E[i]$ denotes $\otimes E'_1, \ldots, E'_m \otimes[n!\,(i)]$, and if $E=\langle c_1, \ldots, c_m\rangle$, then $E[i]$ denotes c_i.

DEFINITION 5.1 (constant normal form expressions): *Constant normal form* expressions have the following general structure:

$$E_1 \bullet \ldots \bullet E_n \quad \text{or simply } E_i$$

where each E_i has form:

$$\mathbf{c}_{k\,1}(j_1, (\ldots, \mathbf{c}_{kn}(j_n, E)\ldots)) \quad \text{or simply } E$$

where $j_i \leqq j_{i+1}$, and E has the form:

$$\langle c_1, \ldots, c_n \rangle \text{ or } \mathbf{Pe}(n!, \mathrm{E}') \text{ or simply } E'$$

where c_i is a 0-arity constructor, $n!$ is a permutation on the first n naturals, and E' has form:

$$\otimes E''_1, \ldots, E''_m \otimes$$

where

$$\forall i, j \in [1, n],\ i < j \qquad \text{and} \qquad n!(i) > n!(j)$$

iff

$-E''[n!(i)]$ and $E''[n!(j)]$ are not the same E''_r for some $r \in [1, m]$,

and

$-\exists k \in [1, n]$ such that $k < i$ and $n!(k) < n!(j)$ and

$E''[n!(k)]$ and $E''[n!(j)]$ are the same E''_r for some $r \in [1, m]$

(i. e. terms in a product are ordered so that for each i and j, $n!(i) > n!(j)$ if and only if when the $n!(i)$-th and the $n!(j)$-th projection of $\otimes E''_1, \ldots, E''_m \otimes$ are projections of different terms E''_{ki} and E''_{kj}, then there exists some $k \in [1, n]$ such that $k < i$ and $n!(k) < n!(j)$ and the $n!(k)$-th projection of $\otimes \mathbf{E''_1}, \ldots, E''_m \otimes$ is projection of E''_{kj}.)

Finally, E''_i is:

$$0\text{-arity constructor, or } \pi, \text{ or } \mathbf{In}(k, \pi) \text{ with } k > 1$$

and for at least one $j \in [1, m]$, E''_j is π, or is $\mathbf{In}(k, \pi)$ with $k > 1$.

Example 5.1: Consider the expression

$$\mathrm{E} = \langle \underline{S(0), 0, 0} \rangle \bullet \mathbf{Pe}(1\ 3\ 2, \otimes \mathbf{In}(2, \pi), \pi \otimes).$$

E is a constant normal form expression. In contrast, both

$$E' = \langle \underline{S(0), 0, 0} \rangle \bullet \mathbf{Pe}(2\ 1\ 3, \otimes \pi, \mathbf{In}(2, \pi) \otimes)$$

and,

$$E'' = \langle \underline{S(0), 0, 0} \rangle \bullet \mathbf{Pe}(2\ 3\ 1, \otimes \mathbf{In}(2, \pi), \pi \otimes),$$

are not constant normal form expressions because of the condition on **Pe**. Note that E, E' and E'' are $\approx -$congruent.

DEFINITION 5.2 (term substitution, $E[E_1, \ldots, E_j \leftarrow E']$): Let $\mathrm{E} = \otimes E'_1, \ldots, E'_m \otimes$ be in constant normal form, $E_1, \ldots, E_j$ be j different expressions

in $\{E'_1, \ldots, E'_m\}$, and E' be an expression such that $\sum_{p=1,j} \# E_p = \# E'$. Then we call *substitution* and we denote it by $E[E_1, \ldots, E_j \leftarrow E']$, the expression $\mathbf{Pe}(n!, E'')$ such that:

$$\forall i \in [1, n], \quad E''[i] = \begin{cases} E[i] & \text{if } E[i] \notin \{E_1, \ldots, E_j\} \\ E' \text{ otherwise} \end{cases}$$

Analogously, for $E = \mathbf{Pe}(n'!, \otimes E'_1, \ldots, E'_m \otimes)$, $E[E_1, \ldots, E_j \leftarrow E']$ is the expression $\mathbf{Pe}(n''!, E'')$ such that:

$$\forall i \in [1, n], \quad E''[n''!(i)] = \begin{cases} E[n'!(i)] & \text{if } E[n'!(i)] \notin \{E_1, \ldots, E_j\} \\ E' \text{ otherwise.} \end{cases}$$

Appendix I defines a system R of rewrite rules which reduce constant expressions into constant normal form expressions. It includes some examples. Rules are grouped according to the top-most operator which occurrs in the (sub)expression which has to be reduced to normal form. Consider the (sub)expression

$$E = \text{op1}(E1_1, \ldots, \text{op2}(E2_1, \ldots, E2_{n1}), \ldots, E1_{n2}).$$

Rougthly speacking, if E is not in normal form because of the presence of the operator op 2 (i. e. applications of op 2 cannot occur as argument of op 1) then R provides for a rule which removes $\text{op2}(E2_1, \ldots, E2_{n1})$ from E and reduces E into an equivalent expression E'. Note that, families of operators are treated as a single operator (with the indexes as additional operator parameters).

In figure 1 we define a table which summarizes the rules of R and shows how to appropriately apply them in order to efficiently make reductions.

Example 5.3: Consider the following constant expression

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \pi, \underline{0}\otimes))))$$

according to the table in figure 2

– $R(\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \pi, \underline{0}\otimes)))))$ is reduced by row $\mathbf{c_k}\downarrow$, column $\mathbf{c_k}\downarrow$, i. e. $\downarrow$;

Figure 1

	$\langle - \rangle$	•	c	$\varnothing$	π	**In**	$\otimes - \otimes$	$\mathbf{c_k}$	$\mathbf{c_k}\downarrow$	**Pr**	**Pe**
$\langle - \rangle$ . . .	$R1a$	$R1b$	$R1c$	$R1d$		$\downarrow$	$\downarrow$	$R1e$	$\downarrow$	$\downarrow$	$\downarrow$
•	**ok**	**ok**	**ok**	$R2a$	$R2b$	**ok**	**ok**	**ok**	**ok**	**ok**	**ok**
In	$\downarrow$	$R3a$	$R3b$	$R3c$	$R3d$	$R3d$	$\downarrow$	$R3e$	$\downarrow$	$\downarrow$	$\downarrow$
$\otimes - \otimes$ . .	$R4a$	$R4b$	$R4c$	$R4d$	**ok**	**ok**	$R4e$	$R4f$	$\downarrow$	$\downarrow$	$R4g$
$\mathbf{c_k}$	**ok**	$R5a$	**ok**	$R5b$	**ok**	**ok**	**ok**	$R5c$	$\downarrow$	$\downarrow$	$\downarrow$
$\mathbf{c_k}\downarrow$	$\downarrow$	$R6a$	$R6b$	$R6b$	$R6c$	$R6d$	$R6e$	$R6f$	$\downarrow$	$\downarrow$	$R6g$
Pr	$R7a$	$R7b$	$R7c$	$R7c$	$R7c$	$R7d$	$R7e$	$R7f$	$\downarrow$	$R7g$	$R7h$
Pe	$R8a$	$R8b$	$R8c$	$R8c$	$R8c$	**ok**	$R8c$	$R8d$	$\downarrow$	$\downarrow$	$R8e$

Row op 1 and column op 2 select the rules to be applied in order to reduce expressions of the form op 1 (op 2 (E)) [or, according to the arity of op 1, op 1 (E_1, . . . , E_n) where for some E_i, E_i = op 2 (E)].

ok means that the expression is either in normal form, or the sub-expression E has to be reduced to its normal form. Note if op 2 has arity > 1, E stands for E'_1, . . . , E'_n. In this case, to reduce E, we have to reduce each E'_i.

$\downarrow$ means that sub-expression op 2 (E) has to be reduced to its normal form before applying a rule to the expression op 1 (op 2 (E)).

Ri specifies the rule to be applied to reduce the expression op 1 (op 2 (E)).

to:

– $R(\mathbf{S}\downarrow(2, R(\mathbf{S}\downarrow(3, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \pi, \underline{0}\otimes))))))$ is reduced by row $\mathbf{c_k}\downarrow$, column $\mathbf{c_k}$, i. e. $R6f.3$, to:

– $R(\mathbf{S}\downarrow(2, \mathbf{S}(2, \mathbf{S}\downarrow(3, \mathbf{S}(3, \otimes\pi, \pi, \underline{0}\otimes)))))$ is reduced by row $\mathbf{c_k}\downarrow$, column $\mathbf{c_k}$, i. e. $R6f.1$, to:

– $R(\mathbf{S}\downarrow(3, \mathbf{S}(3, \otimes\pi, \pi, \underline{0}\otimes)))$ is reduced by row $\mathbf{c_k}\downarrow$, column $\mathbf{c_k}$, i. e. $R6f.1$, to:

– $R(\otimes\pi, \pi, \underline{0}\otimes)$ is reduced by row $\mathbf{c_k}\downarrow$, column $\mathbf{c_k}$, i. e. **ok**, to: $\otimes\pi, \pi, \underline{0}\otimes$.

PROPOSITION 5.1: *R is terminating.*

Proof: There exists a *simplification ordering* [Dershowitz82], $\gg^*$, which satisfies all the rules of R, i. e. $\forall l \to r \in R$, $l \gg^* r$.

Let $\gg$ be the following partial ordering on the set of operators which can occur in a constant expression:

$$\langle - \rangle \gg \mathbf{c_k} \gg \bullet \quad \text{and,} \quad \mathbf{In} \gg \otimes - \otimes \gg \mathbf{Pe} \gg \mathbf{c_k}$$

$$\text{and,} \quad \mathbf{Pr}, \mathbf{c_k}\downarrow \gg \otimes - \otimes \quad \text{and,} \quad \mathbf{c_k} \gg \varnothing.$$

Then, $\gg$ can be extended to a *recursive path ordering* [Dershowitz82], $\gg^*$, on constant expressions, i. e.

$$f(s1, \ldots, sn) \gg^* g(t1, \ldots, tm)$$

iff:

$$- f \gg g \quad \text{and} \quad \{f(s1, \ldots, sn)\} \gg^* \{t1, \ldots, tm\},$$

or

$$- \{s1, \ldots, sn\} \gg^* \{t1, \ldots, tm\}$$

where

$$\{s1, \ldots, sn\} \gg^* \{t1, \ldots, tm\} \quad \text{iff} \quad \forall tj, \quad \exists si \text{ such that } si \gg^* tj.$$

Notational remark ($R(E)$): Given a constant expression E, we indicate by $R(E)$ the expression such that: $E \rightarrow^*_R R(E)$, where $\rightarrow^*_R$ is the transitive closure of $\rightarrow_R$.

PROPOSITION 5.2: *For each constant expression E, $R(E)$ is a constant normal form expression.*

Proof: If $R(E)$ is irreducible by R, then $R(E)$ contains the operators $\bullet$, $\mathbf{c}_k$, $\langle - \rangle$, **Pe**, c_0, $\otimes - \otimes$, π, **In**, $\varnothing$, only. Moreover, by contradiction we can see that the structure of $R(E)$ satisfies Definition 5.1.

Finally we show that R is a system of rules which satisfy the axiomatization given for our operators in [Bellia88]. We show a somewhat more general fact. We recall that Proposition 4.1 states that ρ_e is a meaning preserving function.

PROPOSITION 5.3: *For each rule $l \rightarrow r \in R$, $\rho_e(l)$ is a variable renaming of $\rho_e(r)$.*

Proof: Proof tedious but easy to give.

Proposition 5.3 states also that R behaves like ρ_e in giving an operational semantics to our set operators. In particular, in the comparison of two constant expressions E and E', by Propositions 5.2 and 5.3, $E \approx E'$ if and only if $R(E)$ and $R(E')$ are identical modulo commutativity of $\bullet$.

We conclude with two remarks on the system R. First, the system R contains a great number of rules. This is due to the number of our set operators (which includes $\varnothing$ and π). The existence of alternative set models which, on one hand, are equivalent to our data domain HU^*_c, and, on the other hand, require a small number of (possibly, more general) operators, is a relevant question. However, techniques [O'Donnell77, Huet80] to efficiently handle reduction rules and recent progress in the design of computer architecture [Treleaven82] makes realistic an efficient implementation of R.

A second remark concerns efficiency. For the most part, the rules of R are term rewritings and their application is immediate. In contrast, rules $R3e$, $R7h$ and $R8d$ are more complex reductions, hence their application could be expensive to make. The complexity of these rules could be inherent

to our definition of expression normal form, and we can question if a different choice in the structure of normal forms yelds a simplification of the rules of R. Our opinion is that different choices only result into marginal simplifications of the rules of R. Moreover, the three mentioned rules heavily involve (cartesian) product permutations. An efficient (machine) realization of **Pe** should drastically improve the efficiency of R.

5.2. A reduction system for Intset

As a first application of R, we use it to make decidable set-inclusion on constant expressions, i. e., given E and E', to decide $E \subseteqq E'$ (*see* Definition 3.4 in [Bellia88] for a formal definition of $\subseteqq$ on the elements of HU_c^*).

PROPOSITION 5.4: *Let* $E1$ *and* $E2$ *be constant normal forms (whithout occurrences of* $\bullet$*), then:*

(*a*) *if* $E1 = \mathbf{c}_{k1}(h1, E1')$ *and* $E2 = \mathbf{c}_{k2}(h2, E2')$, *then*

$E1 \subseteqq E2$ *iff* $1-\ h1 = h2,\ k1 = k2$ *and* $E1' \subseteqq E2'$, *or*

$2-\ h1 < h2$ *and* $E1' \subseteqq R(\mathbf{c}_{k1}\downarrow(h1, \mathbf{c}_{k2}(h2, E2')))$

(*b*) *if* $E1 = \mathbf{c}_{k1}(h1, E1')$ *and* $E2 = \mathbf{Pe}(n2!, E2')$, *then*

$E1 \subseteqq E2$ *iff* $E1' \subseteqq R(\mathbf{c}_{k1}\downarrow(h1, E2))$.

(*c*) *if* $E1 = \mathbf{Pe}(n1!, E1')$ *and* $E2 = \mathbf{Pe}(n2!, E2')$, *then*

$E1 \subseteqq E2$ *iff* $\forall i \in [1, n]$, *one of the following holds:*

1. $E1[i] = E2[i]$, *or*
2. $E1[i]$ *is a* 0*-arity constructor and* $E2[i] = \pi$, *or*
3. $E1[i]$ *is the* 0*-arity constructor* c_0 *and* $E2[i] = \mathbf{In}(k2, \pi)$ *and*

$$E1 \subseteqq R(E2[i] \leftarrow \mathbf{In}(k2, c_0)].$$

4. $E1[i] = \mathbf{In}(k1, \pi)$ *and* $E2[i] = \mathbf{In}(k2, \pi)$ *and* $k1 > k2$.

(*d*) *if* $E1 = \varnothing$, *then* $E1 \subseteqq E2$

(*rule c is also applied for* $E1$ *and/or* $E2$ *of the form:* $\otimes E_1, \ldots, E_m \otimes$ *and* $\langle c_1, \ldots, c_n \rangle$, *and again* 0*-arity constructors,* π, $\mathbf{In}(k2, \pi)$, *for which* $n = 1$ *and* $E[1] = E$.)

Proof: In cases (*b*), (*c*) and (*d*) the proof is trivial. We prove it in case (*a*).

($\Rightarrow$)

a.1 Assuming $\mathbf{c}_k(h, E1') \subseteqq \mathbf{c}_k(h, E2')$, since $\mathbf{c}_k\downarrow$ is a (weak) inverse of $\mathbf{c}_k$ (*see* Property 3.1), then $\mathbf{c}_k\downarrow(h, \mathbf{c}_k(h, E1')) = E1'$.

a.2 Assuming $\mathbf{c}_{k1}(h1, E1') \subseteqq \mathbf{c}_{k2}(h2, E2')$, since $\mathbf{c}_k\downarrow$ is a (weak) inverse of $\mathbf{c}_k$ and all the constructor functions are monotonic under $\subseteqq$, then $E1' \subseteqq \mathbf{c}_{k1}\downarrow(h1, \mathbf{c}_{k2}(h2, E2'))$.

($\Leftarrow$)

We show that in order to have $E1 \subseteqq E2$ no cases are possible other than *a*.1 and *a*.2. Assuming $E1 = \mathbf{c}_{k1}(h1, E1')$ and $E2 = \mathbf{c}_{k2}(h2, E2')$ to be in constant normal form, if $h1 > h2$ then $E1$ and $E2$ are not comparables or $E2 \subseteqq E1$. As a matter of fact, consider $\mathbf{Pr}(h2, 1, E1)$. It could result into a 0-arity constructor or π. In contrast, $\mathbf{Pr}(h2, 1, E2)$ is $\mathbf{c}_{k2}(1, E2'')$ for some $E2''$, hence $E1$ denotes a set of tuples which cannot (*see* Property 3.3) be contained in the set denoted by $E2$. The case $h1 = h2$ and $\mathbf{c}_{k1} \neq \mathbf{c}_{k2}$ can be proved analogously.

Example 5.3: Let $\underline{C}$, $\underline{S}$, $\underline{0}$ be 2-1-0-arity constructors of HU_c, and:

$$E1 = R(\eta(\underline{S(x), C(S(y), x)})) = \mathbf{S}(1, \mathbf{C}(2, (\mathbf{S}(2, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \pi \otimes))))$$

$$E2 = R(\eta(\underline{x, C(S(z), y)})) = \mathbf{C}(2, (\mathbf{S}(2, \otimes\pi, \pi, \pi \otimes))).$$

In order to decide $E2 \subseteqq E1$, we derive:

by *a*.2

$$\mathbf{C}(2, (\mathbf{S}(2, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \pi \otimes))) \subseteqq \mathbf{R}(\mathbf{S}\downarrow(1, \mathbf{C}(2, (\mathbf{S}(2, \otimes\pi, \pi, \pi \otimes)))))$$

where, by $R6f$ and $R6c$:

$$R(\mathbf{S}\downarrow(1, \mathbf{C}(2, (\mathbf{S}(2, \otimes\pi, \pi, \pi \otimes))))) = \mathbf{C}(2, (\mathbf{S}(2, \otimes\pi, \pi, \pi \otimes))).$$

Then

$$\mathbf{C}(2, (\mathbf{S}(2, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \pi \otimes))) \subseteqq \mathbf{C}(2, (\mathbf{S}(2, \otimes\pi, \pi, \pi \otimes)))$$

by $a.1$

$$\mathbf{S}(2, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \pi \otimes)) \subseteqq \mathbf{S}(2, \otimes \pi, \pi, \pi \otimes)$$

by $a.1$

$$\mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \pi \otimes) \subseteqq \otimes \pi, \pi, \pi \otimes$$

by $c.4$

$$E1[1] = \mathbf{In}(2, \pi) \qquad \text{and} \qquad E2[1] = \pi.$$

Hence, $E2 \subseteqq E1$.

Note that Proposition 5.5 makes set-inclusion decidable by means of a set of reduction rules. The set of rules is confluent and terminating. As a matter of fact, given E 1 and E 2 if E 1 contains m constructor functions, then rule a is applied no more than m times, each one reducing a constructor function application in E 1. Furthermore, when rule c is applied to $E1 \subseteqq E2$, it reduces $E2$ to an expression $E2'$ such that $E1 \subseteqq E2' \subseteqq E2$. Roughly speaking, we decide set-inclusion by means of a descending chain. Therefore, we can reformulate the implementation of **intset** in terms of a system of reduction rules.

DEFINITION 5.3 (index set, $\Im(E, i)$): Let $E = \otimes E'_1, \ldots, E'_m \otimes$ be in constant normal form, and i be an index in $[1, m]$. We call *index set* of E with i the set of indexes $\Im(E, i) = \{i_1, \ldots, i_k\}$ such that for each $j \in \Im(E, i)$, $E[i]$ and $E[j]$ are the same E'_r for some $r \in [1, m]$.

DEFINITION 5.4 (Ri): *Let E 1, E 2 be in constant normal form*

---

a) $$\textbf{Intset}\ (E1,E2) = \begin{cases} E1, \text{ if } E1 \text{ equals } E2 \text{ modulo commutativity of } \bullet \\ \varnothing, \text{ if } E1 \text{ or } E2 \text{ are } \varnothing \end{cases}$$

b) $$\textbf{Intset}\ (E1_1 \bullet \ldots \bullet E1_n, E2_1 \bullet \ldots \bullet E2_m) =$$
$$\textbf{Intset}\ (E1_1, E2_1 \bullet \ldots \bullet E2_m) \bullet \textbf{Intset}\ (E1_2 \bullet \ldots \bullet E1_n, E2_1 \bullet \ldots \bullet E2_m)$$
$$\textbf{Intset}\ (E1_1, E2_1 \bullet \ldots \bullet E2_m) = \textbf{Intset}\ (E1_1, E2_1) \bullet \textbf{Intset}\ (E1_1, E2_2 \bullet \ldots \bullet E2_m)$$

c) $$\textbf{Intset}\ (\mathbf{c}_{k1}(h1,E1), \mathbf{c}_{k2}(h2,E2)) = \begin{cases} \mathbf{c}_k(h, \textbf{Intset}\ (E1,E2)) & \text{if } k1=k2=k \text{ and } h1=h2=h \\ \mathbf{c}_{k1}(h1, \textbf{Intset}\ (E1, R(\mathbf{c}_{k1}\downarrow(h1, \mathbf{c}_{k2}(h2,E2))))) & \text{if } h1<h2 \\ \mathbf{c}_{k2}(h2, \textbf{Intset}\ (R(\mathbf{c}_{k2}\downarrow(h2, \mathbf{c}_{k1}(h1,E1))), E2)) & \text{if } h1>h2 \\ \varnothing \text{ otherwise (i.e. } h1=h2 \text{ but } \mathbf{c}_{k1} \neq \mathbf{c}_{k2}) \end{cases}$$

d) $$\textbf{Intset}\ (\mathbf{c}_k(h,E1), \mathbf{Pe}(n!,E2)) = \mathbf{c}_k(h, \textbf{Intset}\ (E1, R(\mathbf{c}_k\downarrow(h, \mathbf{Pe}(n!,E2)))))$$
$$\textbf{Intset}\ (\mathbf{Pe}(n!,E1), \mathbf{c}_k(h,E2)) = \mathbf{c}_k(h, \textbf{Intset}\ (R(\mathbf{c}_k\downarrow(h, \mathbf{Pe}(n!,E1))), E2))$$

e) $$\textbf{Intset}\ (\mathbf{Pe}(n1!,E1), \mathbf{Pe}(n2!,E2)) = \textbf{Intset}\ (R(\mathbf{Pe}(n1'!,E1')), R(\mathbf{Pe}(n2'!,E2')))$$

---

where, if $i \in [1, n]$ is the first index such that $E1\ [n1!(i)] \neq E2\ [n2!(i)]$, then $n1!$, $E1'$, $n2!$, $E2'$ are such that:

– if $E1\ [n1!(i)] = \underline{C_0}$ for some 0-arity constructor $\underline{C_0}$, one of the following cases holds:

- if E2[n2!(i)]= $\underline{C'_0}$ then E1'=E2'= ∅
- if E2[n2!(i)]=**In**(k2,π) then E1'=E1, E2'=E2 [E2[n2!(i)]← **In**(k2, $\underline{C_0}$)]

- if E1[n1!(i)]=**In**(k1,π) , one of the following cases holds
  - if E2[n2!(i)]= $\underline{C'_0}$ then E1'=E1 [E1[n1!(i)]← **In**(k1, $\underline{C'_0}$)] , E2'=E2
  - if E2[n2!(i)]=**In**(k2,π) , then

    Let k1<k2, ℑ(E2,i) be the index-set of E2[n2!(i)] and $E_{\Im}$={$E1_1$,...,E1p} be the set of all the expressions E1[n1!(j)] of E1 such that j∈ ℑ(E2,i). Then one of the following cases holds
    - if ∀ i∈[1,p], $E1_i$ is not a 0-arity constructor
      then E1'=E1 [$E1_1$,...,E1p ← **In**(k2,π)] , E2'=E2
    - if all the $E1_i$'s which are 0-arity constructors are equal to some 0-arity constructor $\underline{C_0}$ then
      E1'=R(E1 [$E1_1$,...,E1p ← **In**(k2, $\underline{C_0}$)])
      E2'=R(E2 [E2[n2!(i)] ← **In**(k2, $\underline{C'_0}$)])
    - otherwise, E1'=E2'= ∅

    Symmetrical in the case k1>k2.

[as in Proposition 5.5, rule *c* is also applied for expressions like $\otimes E_1, \ldots, E_m \otimes$, $\langle c_1, \ldots, c_n \rangle$, π, or **In** $(k\,2, \pi)$].

PROPOSITION 5.5: *The system* R*i is confluent.*

Proof. – Assuming $E1$ and $E2$ be in constant normal form, only one rule of *Ri* is applicable. Moreover, when applied, each rule reduces **Intset** ($E1$, $E2$) to some expression which contains at most one application of **Intset** to expressions which are again in constant normal form.

The system *Ri* is non-terminating due to the presence of the rules (*c*) and (*d*). For all the other rules, the termination could easily be proved by extending the ordering relation ≫, defined in Proposition 5.1, to include **Intset** as an operator which preceeds all the others [in this case, note that $l \gg^* r$ for rules (*a*), (*b*) and (*d*), even if in case of rule (*d*) is not immediate to see that, due to the use of substitutions]. In case of rules (*c*) and (*d*) the following property holds.

PROPOSITION 5.6: *Let* $E1$ *and* $E2$ *be in constant normal form (whithout occurrences of* $\bullet$*). If there exists a constant expression* E *such that* $R(E) \neq \varnothing$ *and* $E \subseteqq E1$, $E \subseteqq E2$, *then*

$$\mathbf{Intset}(E1, E2) \rightarrow^*_{Ri} E' \quad \textit{for some constant expression } E'.$$

Proof: The proof core is to show that if E contains k constructor functions (of arity greater than 0), then $Ri(\mathbf{Intset}(E1, E2))$ terminates with no more than k reductions with rules (c) and (d). That is after no more than k reductions with rules (c) and (d), $E1$ and $E2$ are reduced to some constant expressions, $E1'$ and $E2'$, which do not contain constructor functions.

Let $k1$ be the number of constructor functions contained in $E1$. Since $E \subseteqq E1$, by Proposition 5.4 $k1 \leqq k$ and $E1$ in no more than $k1$ reductions with rule (b) of Proposition 5.4, is reduced to some $E1'$ which does not contain constructor functions. The same holds for $E2$ in $k2$, $k2 \leqq k$, reductions. Let k' be the number of constructor functions which occur in both $E1$ and $E2$. Then after k' reductions with rules $c.1$, and $(k1-k')+(k2-k')$ reductions with the rules $c.2$ and $c.3$, $\mathbf{Intset}(E1, E2)$ is reduced to $\mathbf{Intset}(E1', E2')$, where both $E1'$ and $E2'$ do not contain constructor functions.

Notational remark $(R(E))$: Given $E1$ and $E2$ in constant normal form, we denote by $Ri(\mathbf{Intset}(E1, E2))$ the expression such that

$$\mathbf{Intset}(E1, E2) \rightarrow^*_{Ri} Ri(\mathbf{Intset}(E1, E2)) \qquad \text{if} \quad Ri \text{ terminates,}$$

where $\rightarrow^*_{Ri}$ is the transitive closure of $\rightarrow_{Ri}$.

Example 5.4: Let $\underline{C}$, $\underline{S}$, $\underline{0}$ be those defined in Example 5.3, and consider the following two expressions

$$E1 = R(\eta(\underline{C(y, x), C(x, S(y))}))$$

$$= \mathbf{C}(1, \mathbf{C}(3, \mathbf{S}(4, \mathbf{Pe}(1342, \otimes \mathbf{In}(2, \pi), \mathbf{In}(2, \pi) \otimes))))$$

$$E2 = R(\eta(\underline{C(0, y), z})) = \mathbf{C}(1, \otimes \underline{0}, \pi, \pi \otimes).$$

$Ri(\mathbf{Intset}(E1, E2))$ results [in square brackets, at each step of the reduction process, we give, whenever possible, the corresponding transformation on the tuples of Herbrand terms]:

– by Ri-$c.1$ (Ri-a is not applicable):

$$\mathbf{C}(1, \mathbf{Intset}(\mathbf{C}(3, \mathbf{S}(4, \mathbf{Pe}(1342, \otimes \mathbf{In}(2, \pi), \mathbf{In}(2, \pi) \otimes))), \otimes \underline{0}, \pi, \pi \otimes))$$

[tuples are left unchanged, but a constructor is marked as ok].

– by *Ri-d.*2 (*Ri-a* is not applicable):

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{Intset}(\mathbf{S}(4,\ \mathbf{Pe}(1\,3\,4\,2,\ \otimes\,\mathbf{In}(2,\ \pi),\ \mathbf{In}(2,\ \pi)\otimes)),\ R(\mathbf{C}\downarrow(3,\ \otimes\,\underline{0},\ \pi,\ \pi\otimes)))))$$

where:

$$R(\mathbf{C}\downarrow(3,\ \otimes\,\underline{0},\ \pi,\ \pi\otimes)) = \otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes$$

i. e.

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{Intset}(\mathbf{S}(4,\ \mathbf{Pe}(1\,3\,4\,2,\ \otimes\,\mathbf{In}(2,\ \pi),\ \mathbf{In}(2,\ \pi)\otimes)),\ \otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes)))$$

$$[\underline{C(y,\ x),\ C(x,\ S(y))}$$

$$\underline{C(0,\ y),\ C(u,\ v)}]$$

– by *Ri-d.*2 (*Ri-a* is not applicable):

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{S}(4,\ \mathbf{Intset}(\mathbf{Pe}(1\,3\,4\,2,\ \otimes\,\mathbf{In}(2,\ \pi),\ \mathbf{In}(2,\ \pi)\otimes),\ \mathbf{R}(\mathbf{S}\downarrow(4,\ \otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes))))))$$

i. e.

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{S}(4,\ \mathbf{Intset}(\mathbf{Pe}(1\,3\,4\,2,\ \otimes\,\mathbf{In}(2,\ \pi),\ \mathbf{In}(2,\ \pi)\otimes),\ \otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes))))$$

$$[\underline{C(y,\ x),\ C(x,\ S(y))}$$

$$\underline{C(0,\ y),\ C(u,\ S(v))}]$$

– by *Ri-e.*2 (*Ri-a* is not applicable):

$$E\,1\,[n\,1!(1)] = \mathbf{In}(2,\ \pi), \qquad E\,1\,[n\,1!(1)] = 0$$

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{S}(4,\ \mathbf{Intset}(\mathbf{R}(\mathbf{Pe}(1\,3\,4\,2,\ \otimes\,\mathbf{In}(2,\ \underline{0}),\ \mathbf{In}(2,\ \pi)\otimes)),\ R(\otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes)))))$$

i. e.

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{S}(4,\ \mathbf{Intset}(\otimes\,\underline{0},\ \mathbf{In}(2,\ \pi),\ \underline{0}\otimes,\ \otimes\,\underline{0},\ \pi,\ \pi,\ \pi\otimes))))$$

$$[\underline{C(0,\ x),\ C(x,\ S(0))}$$

$$\underline{C(0,\ y),\ C(u,\ S(v))}]$$

– by *Ri-e.*1 (*Ri-a* is not applicable):

$$E\,1\,[n\,1!(2)] = \mathbf{In}(2,\ \pi), \qquad E\,1\,[n\,1!(2)] = \pi$$

$$\mathbf{C}(1,\ \mathbf{C}(3,\ \mathbf{S}(4,\ \mathbf{Intset}(R(\otimes \underline{0},\ \mathbf{In}(2,\ \pi),\ \underline{0} \otimes),\ R(\otimes \underline{0},\ \mathbf{In}(2,\ \pi),\ \pi \otimes)))))$$

$$\underline{[C(0,\ x),\ C(x,\ S(0))}$$

$$\underline{C(0,\ y),\ C(y,\ S(v))]}$$

— by *Ri-e*. 3 (*Ri-a* is not applicable):

$$E1\,[n\,1!(3)] = 0, \qquad E1\,[n\,1!(3)] = \pi$$

$$\mathbf{C}(1,\ \mathbf{C}(3, \mathbf{S}(4,\ \mathbf{Intset}(R(\otimes \underline{0},\ \mathbf{In}(2,\ \pi), \underline{0} \otimes), R(\otimes \underline{0},\ \mathbf{In}(2,\ \pi),\ \underline{0} \otimes)))))$$

$$\underline{[C(0,\ x),\ C(x,\ S(0))}$$

$$\underline{C(0,\ y),\ C(y,\ S(0))]}$$

— by *Ri-a* the computation terminates with:

$$\otimes \underline{0},\ \mathbf{In}(2,\ \pi),\ \underline{0} \otimes.$$

Note that

$$\rho_e(\otimes \underline{0},\ \mathbf{In}(2,\ \pi),\ \underline{0} \otimes) = \underline{C(0,\ x),\ C(x,\ S(0)).}$$

PROPOSITION 5.7: *Let* $E1$, $E2$ *be constant normal form expressions* (*without occurrences of* •). *If there exists* $Ri(\mathbf{Intset}(E1,\ E2))$ *then*

$$Ri(\mathbf{Intset}(E1,\ E2)) \subseteqq E1$$

and

$$Ri(\mathbf{Intset}(E1,\ E2)) \subseteqq E2.$$

Proof: Follows immediately from Proposition 5.6 and the construction of $Ri(\mathbf{Intset}(E1,\ E2))$. (Note that if $E = \mathbf{c}_k(h,\ E1')$, then $Ri(\mathbf{Intset}(E1,\ E2)) = Ri(\mathbf{Intset}(\mathbf{c}_k(h,\ E1'),\ E2))$)

PROPOSITION 5.8: *Let* $E1$, $E2$ *be in constant normal form* (*without occurrences of* •). *If there exists a constant expression* E *such that* $R(E) \neq \varnothing$ *and* $E \subseteqq E1$ *and* $E \subseteqq E2$, *then* $E \subseteqq Ri(\mathbf{Intset}(E1,\ E2))$.

Proof: Since $Ri(\mathbf{Intset}(E1,\ E2))$ exists we can use induction on the number of the reduction steps that are required by Proposition 5.4 to decide set inclusion of E in $Ri(\mathbf{Intset}(E1,\ E2))$.

Let $E = \mathbf{Pe}(n'!,\ E')$, $E1 = \mathbf{Pe}(n\,1!,\ E1')$, $E2 = \mathbf{Pe}(n\,1!,\ E1')$. If we assume $E \subseteqq E1$ and $E \subseteqq E2$, we have 4×4 different cases to examine in order to show that for each $i \in [1,\ n]$, $E[i] \subseteqq Ri(\mathbf{Intset}(E1,\ E2))[i]$. This is tediuos, but

easy to give. Thus, according to Proposition 5.4, we conclude that if E, $E1$ and $E2$ do not contain constructor functions then $E \subseteq Ri(\mathbf{Intset}(E1, E2))$.

Let $E = \mathbf{c}_k(h, E')$, $E1 = \mathbf{Pe}(n1!, E1')$, $E2 = \mathbf{Pe}(n2!, E2')$. From the assumption $E \subseteq E1$, and $E \subseteq E2$ we have $E' \subseteq R(\mathbf{c}_k \downarrow (h, E1'))$, $E' \subseteq R(\mathbf{c}_k \downarrow (h, E2'))$. and assumed E' not containing constructor functions both $R(\mathbf{c}_k \downarrow (h, E1'))$ and $R(\mathbf{c}_k \downarrow (h, E2'))$ are not containing constructor functions because of Proposition 5.4, then

$$E' \subseteq Ri(\mathbf{Intset}(R(\mathbf{c}_k \downarrow (h, E1)), R(\mathbf{c}_k \downarrow (h, E2)))).$$

Due to properties of **Intset**,

$$Ri(\mathbf{Intset}(R(\mathbf{c}_k \downarrow (h, E1)), R(\mathbf{c}_k \downarrow (h, E2)))) = \mathbf{c}_k \downarrow (h, Ri(\mathbf{Intset}(E1, E2)))$$

and, since constructor inverses are weak inversions

$$\mathbf{c}_k(h, \mathbf{c}_k \downarrow (h, Ri(\mathbf{Intset}(E1, E2)))) \subseteq Ri(\mathbf{Intset}(E1, E2)).$$

Then we have

$$\mathbf{c}_k(h, E') \subseteq \mathbf{c}_k(h, \mathbf{c}_k \downarrow (h, Ri(\mathbf{Intset}(E1, E2)))) \subseteq Ri(\mathbf{Intset}(E1, E2)).$$

If E' contains constructor functions both $R(\mathbf{c}_k \downarrow (h, E1'))$ and $R(\mathbf{c}_k \downarrow (h, E2'))$ can contain constructor functions and the following considerations hold.

Let $E = \mathbf{c}_k(h, E')$, $E1 = \mathbf{c}_{k1}(h1, E1')$, $E2 = \mathbf{c}_{k2}(h2, E2')$:

if $E \subseteq E1$ we have two cases:

$$a1 - \quad \mathbf{c}_k = \mathbf{c}_{k1}, \quad h = h1 \quad \text{and} \quad E' \subseteq E1',$$

or

$$a2 - \quad h < h1 \quad \text{and} \quad E' \subseteq R(\mathbf{c}_k \downarrow (h, \mathbf{c}_{k1}(h1, E1')))$$

if $E \subseteq E2$ we have two cases:

$$b1 - \quad \mathbf{c}_k = \mathbf{c}_{k2}, \quad h = h2 \quad \text{and} \quad E' \subseteq E2',$$

or

$$b2 - \quad h < h2 \quad \text{and} \quad E' \subseteq R(\mathbf{c}_k \downarrow (h, \mathbf{c}_{k2}(h2, E2')))$$

Let us examine the possible cases:

*a*1-*b*1:

$$Ri(\mathbf{Intset}(E1, E2)) = \mathbf{c}_k(h, Ri(\mathbf{Intset}(E1, E2)))$$

then

$$E \subseteq Ri(\mathbf{Intset}(E1, E2)), \quad \text{because } E' \subseteq Ri(\mathbf{Intset}(E1', E2'));$$

a 1-*b* 2:

$$Ri(\mathbf{Intset}(E1, E2)) = \mathbf{c}_k(h, Ri(\mathbf{Intset}(E1', R(\mathbf{c}_k \downarrow (h, \mathbf{c}_{k2}(h2, E2'))))))$$

then

$$E \subseteq Ri(\mathbf{Intset}(E1, E2)), \text{ because } E' \subseteq Ri(\mathbf{Intset}(E1', R(\mathbf{c}_k \downarrow (h, \mathbf{c}_{k2}(h2, E2'))))));$$

a 2-*b* 1: symmetrical of *a* 1-*b* 2;

a 2-*b* 2: analogous to the case $E = \mathbf{c}_k(h, E')$,

$$E1 = \mathbf{Pe}(n1!, E1'), \qquad E2 = \mathbf{Pe}(n1!, E1').$$

Note that Proposition 5.8 gives a sufficient condition only for the termination of *Ri*. However, *Ri* could terminate even if for no E,

$$R(E) \neq \varnothing \qquad \text{and} \qquad E \subseteq E1 \qquad \text{and} \qquad E \subseteq E2.$$

In this case it terminates with $\varnothing$.

Example 5.5: Let us consider the following expression: **Intset**(**S**(2, $\otimes \pi$, $\underline{0} \otimes$), $\otimes \pi$, $\underline{0} \otimes$), according to *Ri* results: by *Ri-c*. 1 (*Ri-a* is not applicable),

$$\mathbf{S}(2, \mathbf{Intset}(\otimes \pi, \underline{0} \otimes, R(\mathbf{S} \downarrow (2, \otimes \pi, \underline{0} \otimes))))$$

and, by *R* 6 *b*,

$$\mathbf{S}(2, \mathbf{Intset}(\otimes \pi, \underline{0} \otimes, \varnothing))$$

finally, by *Ri-a*, $\varnothing$

When *Ri* terminates, it correctly computes **Intset**. However, when **Intset** computes $\varnothing$, *Ri* could not terminate. For a nonterminating *Ri*, the following proposition holds.

PROPOSITION 5.9: *Let* $E1$ *and* $E2$ *be in constant normal form (without occurrences of* •*). There exists an integer K, depending on* $E1$ *and* $E2$, *such that Ri*(**Intset** ($E1$, $E2$)) *either terminates in at most K reductions with rules* (*c*) *and* (*d*), *or does never terminate.*

Proof: As pointed out before, due to the presence of rules (*c*) and (*d*), *Ri* could result in a non-terminating sequence of reductions. Note that, rules (*c*) and (*d*) increase the number of constructor functions which occur in the redex. However, because of the existence of the function ρ_e, which maps $E1$ and $E2$ onto two (tuples of) Herbrand terms, and since the Mgi of two

(tuples of) Herbrand terms has an upper bound K (*see* Proposition 4.1) on the number of constructors which can occur in the Mgi, then after K reductions with rules (*c*) and (*d*), **Intset** ($E1$, $E2$) is reduced to some formula which contains exactly K constructor functions.

Example 5.6: Let us consider **Intset**(**S**(1, **In**(2, π)), **S**(2, **In**(2, π))). When we apply *Ri*, it indefinitely reduces the expression obtaining

$$\mathbf{S}(1, \mathbf{Intset}(\mathbf{In}(2, \pi), \mathbf{S}(2, \mathbf{S}(2, \mathbf{In}(2, \pi))))),$$

then

$$\mathbf{S}(1, \mathbf{S}(2, \mathbf{Intset}(\mathbf{S}(1, \mathbf{In}(2, \pi)), \mathbf{S}(2, \mathbf{In}(2, \pi))))))$$

and so on. Note that after $K = 2 * 2^0 * 1 = 2$ reduction steps with *Ri-c*, the expression is reduced to

$$\mathbf{S}(1, \mathbf{S}(2, \mathbf{Intset}(\mathbf{S}(1, \mathbf{In}(2, \pi)), \mathbf{S}(2, \mathbf{In}(2, \pi))))))$$

which contains 2 constructor functions and rule *Ri-c* is still applicable.

Propositions 5.8 and 5.9 state that *Ri* correctly reduces **Intset** ($E1$, $E2$) to the constant expression which denotes the **Sup** of all the subsets of both $E1$ and $E2$. However we have three remarks to make. First, the limitation to expressions without occurrences of $\bullet$ given in Propositions 5.6-5.9 is merely for convenience in the presentation. The limitation could easily be removed by reformulating all the Propositions according to the fact that the operator $\bullet$ is endomorphic on the structure of constant expressions, and that the endomorphism is preserved by **Intset**, as rule (*b*) in Definition 5.4 shows.

A second remark concerns the use of normal form in the system *Ri*. This use is not fundamental and could be removed with some advantages in efficiency. However, it is convenient in the proofs of the propositions 5.6-5.9.

A final remark about the complexity of the system *Ri*. We know very efficient algorithms [Martelli82, Paterson78] to compute the Mgu on Herbrand terms. Moreover, as pointed out in Section 4, the computation of the Mgi on Herbrand terms could be derived as a byproduct of the unifcation algorithm. Thus it is reasonable to expect to be able to define a system of reduction rules of complexity comparable to that of the best unification algorithms. *Ri* does not really seem to be the best from the efficiency viewpoint. This is mainly due to its termination which is guaranteed only by Proposition 5.9. Refinements of *Ri* could be given, mainly to provide for an explicitely treatment of the *occur check*. However, the complexity of the entire reduction process, *R* plus *Ri*, is outside of the scope of the present paper, and such refinements would unnecessarily complicate the reduction rules.

5.3. An example of evaluation based on R and Ri

We can now reformulate the semantics of our primitive operators directly in terms of reductions of constant expressions. We put together system R and system Ri. In order to evaluate the expression $E = \mathrm{op}(E1, \ldots, En)$, where op is the primitive operator:

•: We apply R to $E1$. If $E1$ results different from $\varnothing$, then we return it and the computation of $\mathrm{op}(E2, \ldots, En)$ is suspended until a further evaluation of E is required. Otherwise, the evaluation proceeds with $\mathrm{op}(E2, \ldots, En)$ or returns $\varnothing$, according to $n > 1$.

$\langle - \rangle$, $\otimes - \otimes$, $\mathbf{c}_k$, $\mathbf{c}_k\downarrow$, **Pe**, **Pr**, **In**: We compute each Ei in order to obtain a constant expression.

Intset: If $n > 2$, we arbitrary select a pair of arguments, Ei and Ej, and computes **Intset** (Ei, Ej).

Then replace the pair with the result of **Intset** (Ei, Ej). If $n = 2$, we apply Ri.

We return the result of Ri.

Consider the definition of F_{ADD} in Section 4.1

$$F_{\mathrm{ADD}}(w) = u \bullet v \qquad \text{where} \quad \mathbf{Intset}(w, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes)) = u,$$

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w'))) = v$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w', \qquad \mathbf{Intset}(w, \mathbf{S}(2, \mathbf{S}(3, \otimes \pi, \pi, \pi \otimes))) = z.$$

The evaluation of $F_{\mathrm{ADD}}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes)$ proceeds as follows:

1. (by Y and β reductions) as step 1 of Section 4.1.
2. We apply Ri to reduce the subexpression

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi \otimes, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes)),$$

which contains only constant expressions. The subexpression is first reduced to

$$\mathbf{Intset}(\mathbf{S}(1, \mathbf{S}(2, \otimes \pi, \underline{0}, \pi \otimes)), \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0} \otimes))$$

then, (by two steps with rule $Ri.d$) to $\mathbf{S}(1, \mathbf{S}(2, \mathbf{Intset}(\otimes \pi, \underline{0}, \pi \otimes, \varnothing))$ which is finally, reduced to $\varnothing$.

The (main) expression is reduced to

$$\mathbf{S}(2, \mathbf{S}(3, F_{\mathrm{ADD}}(w')))$$

where

$$\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, z)) = w',$$

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi\otimes, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \pi, \pi\otimes))) = z$$

3. (by Y and β reductions) as step 3 of Section 4.1
4. We apply Ri to reduce the subexpression

$$\mathbf{Intset}(\otimes \mathbf{S}(\pi), \mathbf{S}(\underline{0}), \pi\otimes, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \pi, \pi\otimes)))$$

(by two steps with rule $Ri.c$ and by one step with $Ri.d$ and $Ri.e$) to

$$\mathbf{S}(1, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \underline{0}, \pi\otimes)))$$

5. We apply Ri to reduce the subexpression

$$\mathbf{Intset}(\mathbf{S}\downarrow(2, \mathbf{S}\downarrow(3, \mathbf{S}(1, \mathbf{S}(2, \mathbf{S}(3, \otimes\pi, \underline{0}, \pi\otimes))))),$$

$$\mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0}\otimes))$$

(by two steps with rule $Ri.d$) to

$$\mathbf{S}(1, \mathbf{S}(3, \mathbf{Pe}(1\,3\,2, \otimes \mathbf{In}(2, \pi), \underline{0}\otimes))),$$

and we obtain the value and the suspension of step 5 in Section 4.1.

6. CONCLUSIONS

In [Bellia88] we introduced retractions to unify into a functional paradigm, logic and functional programming. The approach was discussed in a pure functional language which provides set operators as the language primitives for programming with retractions. The language semantics was defined in a abstract way which expresses the operators semantics through an equivalence relation. In the present paper we have considered the definition of the language operational semantics. The operational semantics is defined in two distinct ways which are proved equivalent and only differ in the involved techniques. The first definition inherits from predicative languages, concepts and techniques and it introduces:

— structures of Herbrand terms as normal forms for constant expressions, and;

– the Mgi algorithm, byproduct of unification, as an algorithm to compute with the set intersection operator, **Intset**.

The second definition is a more conventional semantics for functional languages. It explicitly defines:

– the concept of constant normal form, and, correspondingly provides for a system of reduction rules, *R*, which reduces constant expressions to the corresponding normal forms, and;

– a system of rules, *Ri*, which reduces expressions which contain occurrences of the set intersection operator to constant expressions.

The comparison of the two distinct semantics shows that, the *Mgi* algorithm, which is something more than pattern-matching (in fact, is bi-directional matching), corresponds to the system of rules *Ri*. Thus, *Mgi* on Herbrand terms has a clear functional counterpart.

A language implementation along the lines of the first definition results easily to design in conventional machine architecture. An experimental implementation of the language has been realized in PASCAL. Constant expressions are internally represented by structures of Herbrand terms. The function ρ_e is embodied into the *read* routine, and is called by the Mgi algorithm each time β reductions produce constant expressions which have to be reduced into normal form. The function η is embodied into the *write* routine.

The second definition is oriented to reduction machines and can be easily extended to parallel architectures.

APPENDIX

THE SYSTEM *R*

R 1 *a:*

$$\langle t_1, \ldots, t_j, \langle t'_1, \ldots, t'_n \rangle, t_{j+1}, \ldots, t_m \rangle = \langle t_1, \ldots, t_j, t'_1, \ldots, t'_n, t_{j+1}, \ldots, t_m \rangle$$

R 1 *b:*

$$\langle t_1, \ldots, t_j, t' \bullet t'', t_{j+1}, \ldots, t_m \rangle = \langle t_1, \ldots, t_j, t', t_{j+1}, \ldots, t_m \rangle \bullet \langle t_1, \ldots, t_j, t'', t_{j+1}, \ldots, t_m \rangle$$

R 1 *c:*

$$\langle \underline{c_0} \rangle = \underline{c_0}$$

R 1 *d:*

$\langle t_1, \ldots, t_j, \varnothing, t_{j+1}, \ldots, t_m \rangle = \varnothing$

R 1 *e:*

$\langle t_1, \ldots, t_j, \mathbf{c}_k(h, t), t_{j+1}, \ldots, t_m \rangle = \mathbf{c}_k(h', \langle t_1, \ldots, t_j, t, t_{j+1}, \ldots, t_m \rangle)$

where

$$h' = h + \sum_{i=1, j} \# t_i$$

R 2 *a:*

$t \bullet \varnothing = \iota$

R 2 *b:*

$t \bullet \pi = \pi$

R 3 *a:*

$\mathbf{In}(h, t' \bullet t'') = \mathbf{In}(h, t') \bullet \mathbf{In}(h, t'')$

R 3 *b:*

$\mathbf{In}(h, \underline{c_0}) = \langle \underline{c_0}, \ldots, \underline{c_0} \rangle$ (h-tuple)

R 3 *c:*

$\mathbf{In}(h, \varnothing) = \varnothing$

R 3 *d:*

$\mathbf{In}(1, t) = t$

R 3 *e:*

$\mathbf{In}(h, \mathbf{c}_k(1, t)) = \mathbf{c}_k(k_1, \mathbf{c}_k(k_2, \ldots, \mathbf{c}_k(k_h, t_1) \ldots))$

where $k_j = (j-1)_* k + 1$ and, t_1 is such that

if t (after reduction to normal form) is:

(*i*) $\mathbf{c}_p(q, t_2)$

then $t_1 = \mathbf{c}_p(q_1, \ldots, \mathbf{c}_p(q_h, t_3)$ where $q_j = (j-1) * \# t_3 + q$ and

t_3 is the reduction of t_2 according to (*i*)-(*ii*)

(*ii*) $\mathbf{Pe}(r!, \otimes t_1, \ldots, t_n \otimes)$

then $t_1 = \mathbf{Pe}(m!, \otimes \mathbf{In}(h_1, t_1'), \ldots, \mathbf{In}(h_n, t_n') \otimes))$ where $m = r_* h$ and is such that

$$m! = m1_1, m2_1, \ldots, mr_1, m1_2, \ldots, mr_2, \ldots, m1_h, \ldots, mr_h \quad \text{with}$$

$mi_j = (r\,[i]-1) * h + j$

and,

$$h_j = \begin{cases} h *_{} h'_j & \text{if} \quad t_j = \mathbf{In}(h'_j, t'_j) \\ h, & \text{if } t_j \text{ is a constant} \quad \text{(in this case, } t'_j = t_j) \end{cases}$$

[in the other cases, t can be considered as a special case of $\mathbf{Pe}(r!, \otimes t_1, \ldots, t_n \otimes)$ where r is the identity permutation and/or $n=1$]. The computation of t_1 corresponds to the computation of a more general **Injection** operation, i. e. injection of tuples of any length.

Example I1:

$$\mathbf{In}(2, \mathbf{c}_3(1, \mathbf{c}_2(2, \mathbf{Pe}(m!, (\otimes \mathbf{In}(2, \pi), \underline{c1_0}, \underline{c2_0} \otimes)))) \qquad \text{with} \quad m! = 3\,1\,4\,2$$

by $R\,3\,e$, is reduced to

$$\mathbf{c}_3(1, \mathbf{c}_3(4, \mathbf{c}_2(2, \mathbf{c}_2(6, \mathbf{Pe}(m!, \otimes \mathbf{In}(4, \pi), \mathbf{In}(2, \underline{c1_0}), \mathbf{In}(2, \underline{c2_0}) \otimes)))$$

with

$$m! = 5\,1\,7\,3\,6\,2\,8\,4$$

by $R\,3\,b$ and $R\,4\,a$, is further reduced to

$$\mathbf{c}_3(1, \mathbf{c}_3(4, \mathbf{c}_2(2, \mathbf{c}_2(6, \mathbf{Pe}(m, \otimes \mathbf{In}(4, \pi), \underline{c1_0}, \underline{c1_0}, \underline{c2}, \underline{c2_0}) \otimes))),$$

which corresponds (under ρ_e) to the pair of Herbrand terms

$$\underline{c_3(c1_0, c_2(x, c2_0), x)}, \qquad \underline{c_3(c1_0, c_2(x, c2_0), x)}$$

R 4 a:

$$\otimes t_1, \ldots, t_j, \langle t'_1, \ldots, t'_n \rangle, t_{j+1}, \ldots, t_m \otimes = \otimes t_1, \ldots, t_j, t'_1, \ldots, t'_n, t_{j+1}, \ldots, t_m \otimes$$

R 4 b:

$$\otimes t_1, \ldots, t_j, t' \bullet t'', t_{j+1}, \ldots, t_m \otimes = \otimes t_1, \ldots, t_j, t', t_{j+1}, \ldots, t_m \otimes \bullet \otimes t_1, \ldots, t_j, t'', t_{j+1}, \ldots, t_m \otimes$$

R 4 c:

$$\otimes \underline{c1_0}, \ldots, \underline{cn_0} \otimes = \langle \underline{c1_0}, \ldots, \underline{cn_0} \rangle$$

R 4 d:

$$\otimes t_1, \ldots, t_j, \varnothing, t_{j+1}, \ldots, t_m \otimes = \varnothing$$

R 4 e:

$$\otimes t_1, \ldots, t_j, \otimes t'_1, \ldots, t'_n \otimes, t_{j+1}, \ldots, t_m \otimes$$
$$= \otimes t_1, \ldots, t_j, t'_1, \ldots, t'_n, t_{j+1}, \ldots, t_m \otimes$$

R 4 f:

$$\otimes t_1, \ldots, t_j, \mathbf{c}_k(h, t), t_{j+1}, \ldots, t_m \otimes$$
$$= \mathbf{c}_k(h, \otimes t_1, \ldots, t_j, t, t_{j+1}, \ldots, t_m \otimes),$$

where

$$h' = h + \sum_{i=1, j} \# t_i$$

R 4 g:

$$\otimes t_1, \ldots, t_h, \mathbf{Pe}(m!, t), t_{h+1}, \ldots, t_m \otimes$$
$$= \mathbf{Pe}(n!, \otimes t_1, \ldots, t_h, t, t_{h+1}, \ldots, t_m \otimes),$$

where if

$$\sum_{i=1, h} \# t_i = k1 \quad \text{and} \quad \sum_{i=h+1, n} \# t_i = k2 \quad \text{then,} \quad n = k1 + m + k2$$

and is such that

$$n!(i) = \begin{cases} i & \text{if } i \in [1, k1] \cup [k1+m+1, k2] \\ m!(i-k1+1)+k1, & \text{otherwise.} \end{cases}$$

R 5 a:

$$\mathbf{c}_k(h, t' \bullet t'') = \mathbf{c}_k(h, t') \bullet \mathbf{c}_k(h, t'').$$

R 5 b:

$$\mathbf{c}_k(h, \varnothing) = \varnothing.$$

R 5 c:

$$\mathbf{c}_{k1}(h1, \mathbf{c}_{k2}(h2, t)) = \mathbf{c}_{k2}(h2, \mathbf{c}_{k1}(h1 - k2 + 1, t)) \quad \text{if} \quad h2 < h1.$$

R 6 a:

$$\mathbf{c}_k\downarrow(h, t' \bullet t'') = \mathbf{c}_k\downarrow(h, t') \bullet \mathbf{c}_k\downarrow(h, t'').$$

R 6 b:

$$\mathbf{c}_k\downarrow(1, v) = \mathbf{c}_k\downarrow(v) = \varnothing \quad \text{if } v \text{ is a constant different from } \pi.$$

R 6 c:

$$\mathbf{c}_k\downarrow(1, \pi) = \mathbf{c}_k\downarrow(\pi) = \pi.$$

R 6 d:

$$\mathbf{c}_k\downarrow(p, \mathbf{In}(q, v)) = \varnothing \quad \text{if } v \text{ is a constant different from } \pi.$$

$$\mathbf{c}_k\downarrow(p, \mathbf{In}(q, \pi)) = \mathbf{c}_k(p_1, \ldots, \mathbf{c}_k(p_{p-1},$$

$$\mathbf{c}_k(p_{p+1}, \ldots, \mathbf{c}_k(p_q, \mathbf{Pe}(m, \otimes I_1, \ldots, I_k \otimes))\ldots)$$

where

$$I_1 = \ldots = I_k = \mathbf{In}(q, \pi), \qquad p_j = (j-1)*k+1, \qquad m = k*q$$

and

$$m! = m1_1, m2_1, \ldots, mk_1, m1_2, \ldots, mk_2, \ldots, m1_q, \ldots, mk_q$$

$$\text{with} \quad mi_j = (i-1)*q+j.$$

Example I2: $\mathbf{c}_2\downarrow(3, \mathbf{In}(5, \pi))$, by *R 6 d*, is reduced to

$$\mathbf{c}_2(1, \mathbf{c}_2(3, \mathbf{c}_2(7, \mathbf{c}_2(9, \mathbf{Pe}(m!, \otimes \mathbf{In}(5, \pi), \mathbf{In}(5, \pi) \otimes)))))$$

with

$$m! = 1\ 6\ 2\ 7\ 3\ 8\ 4\ 9\ 5\ 10,$$

which corresponds (under ρ_e) to the 6-tuple of Herbrand terms $\underline{c_2(x, y)}$, $\underline{c_2(x, y)}$, $\underline{x}$, $\underline{y}$, $\underline{c_2(x, y)}$, $\underline{c_2(x, y)}$

R 6 e:

$\mathbf{c}_k\downarrow(h, \otimes t_1, \ldots, t_m \otimes) = \otimes t_1, \ldots, t_{j-1}, \mathbf{c}_k\downarrow(h', t_j), t_{j+1}, \ldots, t_m \otimes,$

where

$$h = h' + \sum_{i=1,\, j} \# t_i$$

R 6 f:

$\mathbf{c}_k\downarrow(h, \mathbf{c}_k(h, t)) = t$

$\mathbf{c}_k\downarrow(h, \mathbf{c}_{k'}(h', t)) = \varnothing$ if $h = h'$ and $\mathbf{c}_k \neq \mathbf{c}_{k'}$

$\mathbf{c}_k\downarrow(h, \mathbf{c}_{k'}(h', t)) = \mathbf{c}_{k'}(h'', \mathbf{c}_k\downarrow(h, t))$

where if $h' < h$ then $h'' = h'$, otherwise $h'' = h' + k - 1$.

R 6 g:

$\mathbf{c}_k\downarrow(h, \mathbf{Pe}(m!, \otimes t_1, \ldots, t_n \otimes) = \mathbf{Pe}(m!, \otimes t_1, \ldots, \mathbf{c}_k\downarrow(h', t_j), \ldots, t_n \otimes)$

where,

$$t_j = \otimes t_1, \ldots, t_n \otimes [m!(h)]$$

and, h' is such that $m!(h) = h' + \sum_{i=1,\, j-1} \# t_i$

R 7 a:

$\mathbf{Pr}(p, q, \langle t_1, \ldots, t_m \rangle) = \langle t_p, \ldots, t_q \rangle.$

R 7 b:

$\mathbf{Pr}(p, q, t' \bullet t'') = \mathbf{Pr}(p, q, t') \bullet \mathbf{Pr}(p, q, t'').$

R 7 c:

$\mathbf{Pr}(1, 1, v) = v.$

R 7 d:

$$\mathbf{Pr}(p, q, \mathbf{In}(k, t)) = \mathbf{In}(q, t).$$

R 7 e:

$$\mathbf{Pr}(p, q, \otimes t_1, \ldots, t_m \otimes) = \otimes \mathbf{Pr}(p', q', t_h), t_{h+1}, \ldots, t_{h+r-1}, \mathbf{Pr}(1, q'', t_{h+r}) \otimes$$

where

$$t_h = \otimes t_1, \ldots, t_m \otimes [h], \quad p' = p - \sum_{i=1, h-1} \# t_i, \quad q' = \min(q, \# t_h - p' + 1),$$

if $q' = q$ then $r = 0$, otherwise is such that

$$\sum_{i=1, r-1} \# t_{h+i} < q - q' \leqq \sum_{i=1, r} \# t_{h+i}$$

and

$$q'' = q - (q' + \sum_{i=1, r-1} \# t_{h+i}).$$

Example I3:

$$\mathbf{Pr}(2, 2, \otimes \mathbf{In}(2, \underline{c_0}), \mathbf{In}(3, \pi) \otimes),$$

by *R 7 e*, is reduced to

$$\otimes \mathbf{Pr}(2, 1, \mathbf{In}(2, \underline{c_0})), \mathbf{Pr}(1, 1, \mathbf{In}(3, \pi)) \otimes$$

which, by *R 7 d*, is further reduced to

$$\otimes \mathbf{In}(1, \underline{c_0}), \pi \otimes$$

which finally, by *R 3 d*, results $\otimes \underline{c_0}, \pi \otimes$.

R 7 f:

$$\mathbf{Pr}(p, q, \mathbf{c}_k(h, t)) = \begin{cases} \mathbf{Pr}(p+k-1, q, t) & \text{if } p > h \\ \mathbf{Pr}(p, q, t) & \text{if } p+q-1 < h \\ \mathbf{c}_k(h-p+1, \mathbf{Pr}(p, q+k-1, t)) & \text{if } p \leqq h \leqq p+q. \end{cases}$$

R 7 g:

$$\mathbf{Pr}(p, q, \mathbf{Pr}(p', q', t)) = \mathbf{Pr}(p+p'-1, q, t).$$

R 7 h:

$$\mathbf{Pr}(p, q, \mathbf{Pe}(n!, \otimes t_1, \ldots, t_k \otimes)) = \mathbf{Pe}(m!, \otimes \mathbf{Pr}(1, q_1, u_1), \ldots, \mathbf{Pr}(1, q_r, u_r) \otimes)$$

where,

$$\{u_i = \mathbf{Pe}(n, \otimes t_1, \ldots, t_k \otimes))[p+i+1] \mid 1 < i < q\}$$

(with the u_i's indexed, according to the order in which they occur in $\otimes t_1, \ldots, t_k \otimes$),

q_i is the number of the different components which fall in u_i,

$m = q$ and, if $N(i) = \{n!(i) \mid p < i < p+q-1\}$

then

$m!(i) = n!(p+i-1) +$ cardinality-of $\{j \notin N(i) \mid j < n!(p+i-1)\}$.

In contrast to the other rules, $R\,7\,h$ has to be restricted to expressions $\otimes t_1, \ldots, t_p \otimes$ which are in normal form, that is each t_i is either a constant or an **Injection** of constants. However, a more general rule could be given and the restriction affects only the efficiency of the reduction process.

Example I4.

$\mathbf{Pr}(3, 3, \mathbf{Pe}(n!, \otimes \mathbf{In}(2, \pi), \underline{c_0}, \mathbf{In}(3, \pi) \otimes))$ with $n! = 4\ 5\ 1\ 6\ 2\ 3$,

by $R\,7\,h$, is reduced to

$\mathbf{Pe}(m!, \otimes \mathbf{Pr}(1, 2, \mathbf{In}(2, \pi)), \mathbf{Pr}(1, 1, \mathbf{In}(3, \pi)) \otimes)$ with $m! = 1\ 3\ 2$,

which, by $R\,7\,d$, is reduced to

$$\mathbf{Pe}(m!, \otimes \mathbf{In}(2, \pi), \mathbf{In}(1, \pi) \otimes),$$

which, by $R\,3\,d$, is finally, reduced to

$$\mathbf{Pe}(m!, \otimes \mathbf{In}(2, \pi), \pi \otimes).$$

$R\,8\,a$:

$$\mathbf{Pe}(n!, \langle t_1, \ldots, t_n \rangle) = \langle t_{n!(1)}, \ldots, t_{n!(n)} \rangle.$$

$R\,8\,b$:

$$\mathbf{Pe}(n!, t' \bullet t'') = \mathbf{Pe}(n!, t') \bullet \mathbf{Pe}(n!, t'').$$

$R\,8\,c$:

$\mathbf{Pe}(n!, t) = t$ if $n!(i) = i$ for each $i \in [1, n]$

$\mathbf{Pe}(n!, t) = \mathbf{Pe}(n'!, t')$ if $t = \otimes t_1, \ldots, t_p \otimes$, and there exists $h \in [1, n]$ such that $n!(h) \in \mathfrak{I}(t, j)$ (i.e. index set of t_j in t) and for each k such that $n!(K) \in \mathfrak{I}(t, i)$ results $k > h$, $t' = \otimes t_1, \ldots, t_p \otimes$, $n = n'$ and,

$$n'!(i) = \begin{cases} n!(i) & \text{if } n!(i) < \sum_{h=1,\, i-1} \#t_h \text{ or } n!(i) > \sum_{h=1,\, j} \#t_h \\ n!(i) + \sum_{h=i+1,\, j} \#t_h & \text{if } n!(i) \in \mathfrak{I}(t, i) \\ n!(i) - \sum_{h=i,\, j-1} \#t_h & \text{if } n!(i) \in \mathfrak{I}(t, j) \\ n!(i) + (\#t_j - \#t_i) & \text{if } \sum_{h=1,\, i} \#t_h < n!(i) < \sum_{h=1,\, j} \#t_h \end{cases}$$

R 8 d:

$\mathbf{Pe}(n!, \mathbf{c}_k(h, t)) = \mathbf{c}_k(h', \mathbf{Pe}(n'!, t))$

where

$$h' \text{ is such that } n!(h') = h, \qquad n' = n + k - 1$$

and,

$$n'! \text{ is such that:}$$

for $i < h'$,

$$n'!(i) = n!(i) \quad \text{if} \quad n!(i) < k$$

$$n'!(i) = n!(i) + k - 1 \quad \text{if} \quad n!(i) > k$$

for $i > h'$,

$$n'!(i + k - 1) = n!(i) \quad \text{if} \quad n!(i) < k$$

$$n'!(i + k - 1) = n!(i) + k - 1 \quad \text{if} \quad n!(i) > k$$

for $i \in [h', h' + k - 1]$, $n'!(i) = i$.

R 8 e:

$\mathbf{Pe}(n!, \mathbf{Pe}(n'!, t)) = \mathbf{Pe}(n''!, t)$

where $n''!$ is such that $n''!(i) = n'!(n!(i))$.

Example I5: Let

$$c = \{\mathrm{nil}_0, s_1, \mathrm{cons}_2\},$$

$$\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{s}_1 \downarrow (4, \mathbf{Pe}(1, 3, 4, 2, 5, \otimes \mathbf{In}(2, \pi), \pi, \pi, \pi \otimes))))$$

is reduced by *R 6 g*

to:

$$\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{Pe}(1, 3, 4, 2, 5, \otimes \mathbf{In}(2, \pi), \pi, \mathbf{s}_1 \downarrow (1, \pi), \pi \otimes)))$$

which is reduced by *R 6 c*,

to:

$$\mathbf{cons}_2(1, \mathbf{cons}_2(4, \mathbf{Pe}(1, 3, 4, 2, 5, \otimes \mathbf{In}(2, \pi), \pi, \pi, \pi \otimes)))$$

which is irreducible.

REFERENCES

[Apt82] K. R. Apt and M. H. van Emdem, *Contribution to the Theory of Logic Programming*, J. ACM, Vol. 29, 1982, pp. 841-862.

[Bellia88] M. BELLIA, E. DAMERL, P. DEGANO, G. LEVI and M. MARTELLI, *A Formal Model for Lazy Implementation of a PROLOG Compatible Functional Language.* In *Implementations of PROLOG*, J. A. CAMPBELL Ed., Ellis Horwood, 1984, pp. 309-326.

[Bellia88] M. BELLIA, *Logic and Functional Programming by Retractions*, RAIRO Inf. Théorique et Applications, Vol. 22, 1988, pp.

[Berkling75] K. BERKLING, *Reduction Languages for Reduction Machines*, Proc. 2nd Int. Symp. on Computer Architectures, IEEE Comp. Society Press, 1975, pp. 133-140.

[Berkling82] K. BERKLING, J. A. ROBINSON and E. E. SIBERT, *A Proposal for a Fifth Generation Logic and Functional Programming System, based on Highly Parallel Reduction Machine Architecture*, Syracuse University, November 1982.

[Berkling85] K. BERKLING, *Epsilon-reduction: Another view of Unification*, CASE Center, Syracuse University, 1985.

[Dershowitz82] N. DERSHOWITZ, *Ordering for Term-rewriting Systems*, Theoretical Computer Science, Vol. 17, 1982, pp. 279-301.

[Friedman76] D. FRIEDMAN and D. WISE, *CONS Should not Evaluate its Arguments.* In *Automata, Languages and Programming P*, S. MICHELSON Ed., Edinburgh Univ. Press, 1977, pp. 256-284.

[Henderson76] P. HENDERSON and J. H. MORRIS, *A Lazy Evaluator*, Proc. Third ACM Symp. on Principles of Programming Languages, 1976, pp. 95-103.

[Henderson80] P. HENDERSON, *Functional Programming*, Application and Implementation, Prentice-Hall, Englewood Cliffs, N.J., 1980.

[Huet80] G. HUET and D. C. OPPEN, *Equations and Rewrite Rules.* A survey. INRIA Tech, Report 15, also as SRI Rep. STAN-CS-80-785, january 1980.

[Kowalski74] R. A. KOWALSKI, *Predicate Logic as a Programming Language*, Proc. IFIP Congress, 1974, pp. 569-574.

[Martelli82] A. MARTELLI and U. MONTANARI, *An Efficient Unification Algorithm*, ACM TOPLAS, Vol. 4, 1982, pp. 258-282.

[O'Donnell77] M. J. O'DONNELL, *Computing in System Described by Equations*, LNCS 50, Springer-Verlag, Berlin, 1977.

[Paterson78] M. S. PATERSON and M. N. WEGMAN, *Linear Unification*, J. Comp. System Science, Vol. 16, 1978, pp. 158-167.

[Stoy77] J. E. STOY, *Denotational Semantics*, The Scott-Strachey Approach to Programming Languages, MIT Press, Cambridge, 1977.

[Treleaven82] P. C. TRELEAVEN, *Computer Architecture for Functional Programming*, In *Functional Programming and its Applications*, J. Darlington, P. HENDERSON and D. A. TURNER Eds., Cambridge Univ. Press, 1982, pp. 281-306.