

Unfold/fold program transformations

Laurent Kott

▶ **To cite this version:**

Laurent Kott. Unfold/fold program transformations. [Research Report] RR-0155, INRIA. 1982. inria-00076404

HAL Id: inria-00076404
https://inria.hal.science/inria-00076404

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. 954 90 20

Rapports de Recherche

N° 155

UNFOLD/FOLD PROGRAM TRANSFORMATIONS

Laurent KOTT

Août 1982

UNFOLD/FOLD PROGRAM TRANSFORMATIONS

Laurent KOTT

Publication Interne n° 173 - Juillet 1982

RÉSUMÉ : Nous étudions ici la correction des transformations par pliage/dépliage des programmes récursifs. En gardant un point de vue pratique, nous donnons des conditions assurant cette correction. Puis nous montrons comment utiliser les transformations de programme comme un outil de preuve.

ABSTRACT:

Here we study the correctness of unfold/fold transformations of recursive programs. With a practical mind, we give conditions ensuring this correctness. Then we show how to use program transformations as a proof tool.

* *IRISA*

Secrétariat de Rédaction : F. MOINET - Laboratoire d'Informatique
Campus de Beaulieu
35042 RENNES

This text was presented at Seminar on the Application of Algebra to Language Definition and Compilation Sponsored by CNRS and NSF, Ecole Nationale Supérieure des Mines de Paris, Fontainebleau, France, June 1982.

PAPIER RECUPÉRÉ ET RECYCLÉ

1. INTRODUCTION

It is well-known that programming is difficult because the ability of conceiving algorithms is limited by the amount of superfluous complexity introduced in order to express algorithms as programs. For two decades the advantages of fonctionnal style of programming have been appreciated to fight against this situation. Here functionnal programming means program schemes as well as functionnal programming languages (LISP, APL, FP,...). However the lack of clarity, the consumption of time and space limit the acceptance of functionnal style.

To increase efficiency, several authors have suggested high level program transformations. But this approach raises a big problem : how ensure that any transformation applied to a program preserves its meaning. A transformation system which cannot certify its transformations is dangerous and must be rejected. Then, there are two basic approaches to solve the correctness problem. One way is to provide a catalogue of program schemes and transformations. This approach was followed by J. Arsac , S. Gerhard, G. Huet and B. Lang, D. Loveman.. [1,10,13,19].

The catalogue must specify conditions under which particular transformation may be performed and certified correct.

The other way is to design a system with a small set of transformations rules wich discovers the transformations on the fly. Such a system was described by R. Burstall and J. Darlington [4] and based on the so-called "unfold/fold" technique (also mentionned by Z. Manna and R. Waldinger [20]).

In this fight Catalogue versus Discovery a lot of arguments have been developped ; let us mention three of them :

- the catalogue may become so large than it cannot be used without a large consumption of time and space
- the discovery system perform many times the same transformation ;
- the discovery system implies the correctness proof of each transformation or a general proof that it perform only correct transformations.

This last point is the origin of our work. Of course, we do not give a proof of the general problem -it is a second order problem- yet we provide conditions, found by the system itself, under which the transformation is correct.

This work has two parts : in the first one (sections 3 and 4) we study the unfold/fold transformations rules and give conditions ensuring the corrections. In the second part (section 5), we show how a transformation system may be used as an equivalence proof system. Finally, the section 6 is devoted to some further researchs and concluding remarks. Before we recall briefly definitions and results of the algebraic theory of recursive program schemes.

2. ALGEBRAIC FRAMEWORK

In this section, we briefly recall formalism and main results of the algebraic semantics of recursive programs. For more details the reader should refer to [6,7,11,12,22].

2.1. Notations

We shall use the following notations :
$\mathbb{N}$ is the set of positive integers ;
for any k of N greater than 0 $[k]$ denotes the set $\{i \in N \mid 1 \leq i \leq k\}$;
$[0]$ may denote the empty set ;
for any no empty set the free monoid generated by X is X^*, the empty word Λ, and the integer $|u|$ is the length of the word u
for any subset L of X^* and any word W in X^* $L/w = \{u \mid u \in X^* \text{ and } W.u \in L\}$

2.2. C.p.o. set

Definition 1 $(D, \sqsubseteq, \perp)$ is a complete partially order (abbreviated in c.p.o.) iff $\perp$ is the least element of the partially ordered set $(D, \sqsubseteq)$ and each directed subset Δ of D has a least upper bound denoted $\sqcup\Delta$

Let us recall that Δ is a directed subset of D iff for any pair (d,d') of elements of Δ there exists d'' in Δ such that $d \sqsubseteq d''$ and $d' \sqsubseteq d''$.

Remark Let be $(D, \sqsubseteq)$ a partially ordered set, the set D^k (k in $\mathbb{N}$, k>0) will be always ordered component wise and we shall use the same notation $\sqsubseteq$ for the order relation over D^k.

Definition 2 Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be partially ordered sets. A mapping f from D^k to D' is increasing iff $(d_1,\ldots,d_k),(e_1,\ldots,e_k) \in D^k$ $(d_1,\ldots,d_k) \sqsubseteq (e_1,\ldots e_k) \Rightarrow f(d_1,\ldots,d_k) \sqsubseteq' f(e_1,\ldots,e_k)$. Further more f is continuous iff for any directed subsets of D, say $\Delta_1,\ldots,\Delta_k$, having least upper bound, then the set $f(\Delta_1,\ldots,\Delta_k)$ is directed (with respect to $\sqsubseteq'$) and admits $f(\sqcup\Delta_1,\ldots,\sqcup\Delta_k)$ as least upper bound.

2.3. F-magma

In order to built recursive program schemes which are systems of equations on terms, we need a set of functions symbols, say F, whose elements are symbols with arity (a non negative integer). We call F_k the set of elements of F with the same arity k.

Well formed terms (with respect to arities) are obtained by composition of these symbols, applied to variables, and may be viewed as particular cases of finite trees and the set of treees -finite or infinite- is a special case of F-magma (or F-algebra).

Definition 1 An ordered F-magma is a structure $M=\langle D_M, \sqsubseteq_M, \perp_M, \{f_M \mid f \in F\} \rangle$ where $(D_M, \sqsubseteq_M)$ is a partial ordered set with $\perp_M$ as least element and, for any f of F_k, f_M is an increasing mapping from D_M^k to D_M. M is said complete if $\langle D_M, \sqsubseteq_M, \perp_M \rangle$ is a c.p.o. set and each f_M is continuous.

A morphism h between two complete ordered F-magma, M and M', is a continuous mapping from D_M to $D_{M'}$ wich preserves their structure.

Definition 2 A complete ordered F-magma (abbreviated in coF-M) M is free over V if and only if V is included in D_M (up to a canonical injection) and, for any other coF-M M' and any mapping h from V to $D_{M'}$, there exists a unique morphism h^{∞}_M, from M to M' whose the restriction to V is identical to h.

2.4. Trees

We note V a set of variable disjoint from F and described by $\{x_n, n \in \mathbb{N}\}$

Definition 1 A wellformed F-tree on V (abbreviated in a F-tree on V) is a mapping t from a subset of $\mathbb{N}^*$, called dom(t), to F U V. such that $\forall w \in N^*$, $\forall f \in F_k$, $\forall i \in \mathbb{N}$ w. i $\in$ dom(t) implies :

(i) w $\in$ dom(t)

(ii) t(w)=f=>dom(t)/w $\cap$ N =[k]

A F-tree on V is said finite iff its domain is finite, otherwise it is said infinite. We note $M^{\infty}(F,V)$ the set of all F-trees -finite and infinite- on V. We note M(F,V) the set of finite F-trees on V.

By adding a new symbol Ω, with a null arity, we obtain the set $M^{\infty}(F \cup \{\Omega\}, V)$ of $F \cup \{\Omega\}$-trees on V. We shall write $M^{\infty}_{\Omega}(F,V)$ and define on this set the syntactic order by :

$t,t' \in M^{\infty}_{\Omega}(F,V)$ $t \leq t'$ iff (i) $dom(t) \subseteq dom(t')$

(ii) $w \in dom(t)$ $t(w) \neq \Omega \Rightarrow t(w)=t'(w)$

Obviously Ω is the least element of $M^{\infty}_{\Omega}(F,V)$

Proposition 1 The structure $H = < M^{\infty}_{\Omega}(F,V), \leq, \Omega, \{f_H \mid f \in F\}>$ is the free coF-M over V.

For any f of F_k, f_H is an application from $M^{\infty}_{\Omega}(F,V)^k$ to $M^{\infty}_{\Omega}(F,V)$ s.t. for any $(t_1,\ldots,t_k)$ of $M^{\infty}_{\Omega}(F,V)^k$ $f_H(t_1,..,t_k)$ is the tree t defined by

$dom(t)=\{\Lambda\} \cup \bigcup_{i \in [k]} i.dom(t_i)$

$t(\Lambda)=f$; $t(i.w)=t_i(w)$ for any i in $[k]$ and w in $dom(t_i)$.

To achieve this result it suffices to check (and checking is rather tedious) the conditions of the definition 2 of § 2.3.

For any mapping ν from V to $M^{\infty}_{\Omega}(F,V)$, we note ν^* (instead of ν^{∞}_H as written in def. 2 of § 2.3.) the unique endomorphism of H whom the restriction to V is ν. We call it a substitution. Sometimes we write $t[t_1/xi_1,\ldots,t_p/xi_p]$ instead of $\nu^*(t)$, where $\{x_{i_j} \mid j \in [p]\}$ is the set of elements of V occuring at least once in t and, for any j in $[p]$, $\nu(x_{i_j})=t_j$.

2.5. Congruences on H

We shall use the concept of congruences F-magmas and, more precisely, congruences on H stable by substitution (see [14]). To define this restricted notion we need the concept of subtree and subtree replacement.

Definition 1 For any w in dom(t) the subtree of t at node w, denoted by t/w, is defined as follows :

(i) $dom(t/w)=dom(t)/w$

(ii) $\forall u \in dom(t/w)$ $t/w(u)=t(w.u)$

For any t' in $M^{\infty}_{\Omega}(F,V)$, the tree $t[t'/w]$ obtained by the replacement of t/w by t' in t is defined as follows :

(i) $dom(t|t'/w|)=(dom(t)-w.\ N^*)\ U\ w.dom(t')$

(ii) $\forall u \in dom(t)-w.\ N^*\ \ t[t'/w](u)=t(u)$

(iii) $\forall u \in dom(t')\qquad t[t'/w](wu)=t'(u)$

Definition 2 A precongruence on $M_\Omega^\infty(F,V)$ is a preorder (a reflexive and transitive relation) R or $M_\Omega^\infty(F,V)$ such that, for any t in $M_\Omega^\infty(F,V)$, any w in dom(t), any mapping ν from V to $M_\Omega^\infty(F,V)$ and any pair (s,s') of trees, (s,s') belongs to R implies that the pair $(t[\nu^*(s)/w],t[\nu^*(s')/w])$ belongs to R too.

A congruence on $M_\Omega^\infty(F,V)$ is a symetric precongruence.

Proposition 1 For any subset R of the cartesian product $M_\Omega^\infty(F,V)xM_\Omega^\infty(F,V)$ the precongruence generated by R (i.e. the least precongruence including R) is the reflexive and transitive closure of the rewriting relation $\xrightarrow[R]{}$ defined by :

$t \xrightarrow[R]{} t'$ iff $\exists (s,s')\in R, \exists w\in dom(t)\ \exists \nu: V \longrightarrow M_\Omega^\infty(F,V)$

$t/w = \nu^*(s)$ and $t'=t[\nu^*(s')/w]$

The congruence generated by R is the reflexive and transitive closure of the relation $\xrightarrow[R]{} \cup \xrightarrow[R^{-1}]{}$

As usual we shall note $\xrightarrow[R]{*}$ (resp. $\xleftrightarrow[R]{*}$) the precongruence (resp. congruence) generated by R.

2.6. Recursive program schemes

A Recursive Program Schemes (RPS) is a triple $\Sigma=(A,\Phi,R)$ where

- A is the base functions symbols'alphabet and is some finite subset of $\bigcup_{n\in N} A_n$; $A_n=\{a_p^n \mid p \in \mathbb{N}$ and the arity of a_p^n is p$\}$;
- Φ is the procedure symbols'alphabet and is equal to $\{\phi_1,\ldots,\phi_N\}$; each ϕ_i has an arity equal to n_i ;
- R is a functionnal binary relation over $M(A\cup\Phi,V)$ such that if (s,t) belongs to R there exists an integer i (in $[N]$) such that s is equal to $\phi_i x_1 \ldots x_{n_i}$ and t belongs to $M(A\cup\Phi,\{x_1,\ldots,x_{n_i}\})$.

For sake of clarity, we shall write $\xrightarrow[\Sigma]{}$ instead of $\xrightarrow[R]{}$ and use the alternative presentation for RPSs

$$\sum_{1\leq i\leq N} \phi_i x_1 \ldots x_{n_i} = t_i,\ t_i \in M(A \cup \Phi, \{x_1, \ldots, x_{n_i}\})$$

Computations of a term s in a RPS $\sum,(A,\Phi,R)$, are needed to define the semantics (see below) and are sequences of terms rewriten from s in $\sum$. The direct or immediate information contained in a term s is all what we can know about this term without making any computation, that is to say by ignoring the value of procedure symbols occuring in s. Whence the definition of immediate information $\pi(s)$ of a term s :

$$\pi(x) = x \text{ for any } x \text{ in } V$$
$$\pi(ft_1 \ldots t_p) = f\pi(t_1) \ldots \pi(t_p) \text{ for } f \text{ in } A_p$$
$$\pi(\phi_i t_1 \ldots t_{n_i}) = \Omega$$

Thus π is a mapping from $M(A \cup \Phi, V)$ to $M_\Omega(A,V)$.

2.7. Semantics of RPSs

Let $\sum$, (A,Φ,R) be a RPS, an interpretation of $\sum$ is a complete ordered A-magma $M = \langle D_M, \sqsubseteq_M, \perp_M, \{f_M \mid f \in A\}\rangle$. We call D_M^V thet set of "data mappings" naturally order by the following relation, also denoted $\sqsubseteq_M$:

$$\forall \nu,\nu' \in D_M^V \quad \nu \sqsubseteq_M \nu' \iff \forall x \in V \quad \nu(x) \sqsubseteq_M \nu'(x).$$

Since $M_\Omega^\infty(A,V)$ is the free coA-M over V, for each t belonging to $M_\Omega^\infty(A,V)$ and each interpretation M, we are able to define a continuous function t_M from D_M^V to D_M by $t_M(\nu)$ equals to $\nu_M^\infty(t)$ for any ν in D_M^V

Theorem 1 (M. NIVAT [21])

For any RPS $\sum$, (A,Φ,R), any s of $M(A \cup \Phi, V)$ and any interpretation of $\sum$ the set $\{\pi(t)_M \mid s \xrightarrow[\sum]{*} t\}$ is directed with respect to $\sqsubseteq_M$.

From now on, we note $L(\sum,s)$ the set $\{\pi(t) \mid s \xrightarrow[\sum]{*} t\}$. We define the function computed by any term s of $M(A \cup \Phi, V)$ for a given RPS $\sum$ and an interpretation M.

Definition 1 Let $\sum$, (A,Φ,R), be a RPS, M be an interpretation of $\sum$ and s be a term of $M(A \cup \Phi,V)$, the function computed by s is noted $s_{<\sum,M>}$ and is the least upperbound of the set $\{t_M | t \in L(\sum,s)\}$

When M is the free interpretation H, $<M_\Omega^\infty(A,V),\subseteq,\Omega,\{a_H | a \in A\}>$ (cf.§2.4.), we write $s_\sum$ or $T(\sum,s)$ instead of $s_{<\sum,H>}$.

Proposition 1 Let $\sum,(A,\Phi,R)$ be a RPS, M be an interpretation of $\sum$ and s be a term of $M(A\cup\Phi,V)$, $s_\sum$ is an element of $M_\Omega^\infty(A,V)$ so we can define the function $(s_\sum)_M$ which satisfies the following equality $(s_\sum)_M = s_{<\sum,M>}$

Now we rely computed function and rule of computation.

Definition 2 A computation rule ρ associates to each RPS $\sum$, (A,Φ,R), an application from $M(A\cup\Phi,V)$ to itself such that, for any s of $M(A\cup\Phi,V)$, $s \xrightarrow[\sum]{*} \rho_\sum(s)$

We note $L^\rho(\sum,s)$ the subset $\{\pi\rho_\sum^n(s)/n \in \mathbb{N}\}$ of $M_\Omega(A,V)$.

Definition 3 We define two rules of computation : σ the full substitution rule and α the parallel outermost rule. For any RPS $\sum$, (A,Φ,R), we define $\alpha_\sum$ and $\sigma_\sum$ as follows :

(i) $\alpha_\sum(x)=\sigma_\sum(x)$, $\forall x \in V$

(ii) $\alpha_\sum(ft_1\ldots t_n)=f\alpha_\sum(t_1)\ldots\alpha_\sum(t_n)$

(iii) $\sigma_\sum(ft_1\ldots t_n)=f\sigma_\sum(t_1)\ldots\sigma_\sum(t_n)$, $\forall f \in A_n$

(iv) $\alpha_\sum(\phi_i t_1\ldots t_{n_i})=\tau_i[t_1/x_1\ldots t_{n_i}/x_{n_i}]$

$\sigma_\sum(\phi_i t_1\ldots t_{n_i})=\tau_i\,[\sigma_\sum(t_1)/x_1,\ldots,\sigma_\sum(t_{n_i})/x_{n_i}]$,

$\forall \phi_i \in \Phi$, $(\phi_i x_1\ldots x_{n_i},\tau_i) \in R$.

Proposition 2 Let $\sum,(A,\Phi,R)$, be a RPS, M be an interpretation of $\sum$ and s be a term of $M(A\cup\Phi,V)$, $s_{<\sum,M>}$ is the least upperbound of $\{t_M | t \in L^\alpha(\sum,s)\}$ and $\{t_M | t \in L^\sigma(\sum,s)\}$

In others words α et σ are (universally) correct computation rules [8,24].

2.8. Equivalence and inequality modulo a class of interpretations

One of the main advantages of the algebraic semantics is related to the equivalence of RPSs ; so we introduce a preorder and an equivalence over $M(A \cup \Phi, V)$.

Definition 1 Let $\sum, (A, \Phi, R)$, be a RPS and C be a class of interpretations, we define the preorder relation $\leq_{<\sum, C>}$ over $M(A \cup \Phi, V)$ by
$\forall s, s' \in M(A \cup \Phi, V)$ $s \leq_{<\sum, C>} s'$ iff $\forall M \in C$ $s_{<\sum, M>} \leq_M s'_{<\sum, M>}$
We note $\equiv_{<\sum, C>}$ the associated equivalence relation

A class C of interpretations is said equationnal if there exists a binary relation S over $M_\Omega(A,V)$ such that, for any interpretation M, M belongs to C iff $s_M = t_M$ for each pair (s,t) of S.

Given a relation S, we note C_S the equationnal class associated to S and write $\leq_{<\sum, S>}$ (resp. $\equiv_{<\sum, S>}$) instead of $\leq_{<\sum, C_S>}$ (resp. $\equiv_{<\sum, C_S>}$).

Now we state a fruitful theorem used very often in the sequel.

Theorem 2 (I. GUESSARIAN [11])

Let $\sum, (A, \Phi, R)$, be a RPS, (s,s') be a pair of terms of $M(A \cup \Phi, V)$ and S be a subset of $M_\Omega(A,V) \times M_\Omega(A,V)$ then :

$$s \leq_{<\sum, S>} s' \iff \forall t \in L(\sum, s)\ \exists t' \in L(\sum, s')\ t \leq_S t'$$

where $\leq_S$, is the precongruence $(\leq \cup <\!\!\!\!\!\!\rightarrow_S)^*$ over $M_\Omega(A,V)$

Corollary Let $\sum, (A, \Phi, R)$, be a RPS, (s,s') be a pair of terms of $M(A \cup \Phi, V)$ and S be a subset of $M_\Omega(A,V) \times M_\Omega(A,V)$ then
$s \leq_{<\sum, S>} s' \iff \forall t \in L^\alpha(\sum, s)\ \exists t' \in L^\alpha(\sum, s')$ s.t. $t \leq_S t'$ in other words
$s \leq_{<\sum, S>} s' \iff \forall n \in N\ \exists n' \in N\ \pi\alpha^n_\sum(s) \leq_S \pi\alpha^{n'}_\sum(s')$

Definition 2 We define over $M^\infty_\Omega(A,V)$ the preorder relation $\leq_C$, given a class C of interpretations, as follows : for any s, s' of $M^\infty_\Omega(A,V)$
$s \leq_C s' \iff \forall M \in C \quad s_M \leq_M s'_M$
We note $\equiv_C$ the associated equivalence relation

Proposition 1 Let $\sum$, (A,Φ,R) be a RPS, C be a class of interpretations, (s,s') a pair of terms of $M(A,\Phi,V)$, then

$$s \leq_{<\sum,C>} s' \iff s \leq_C s'$$

Proposition 2 Let s,s' be terms of $M_\Omega^\infty(A,V)$ and S be a subset of $M_\Omega(A,V)\times M_\Omega(A,V)$ then

$$s \leq_S s' \iff \forall t \in M_\Omega(A,V): t \sqsubseteq s \exists t' \in M_\Omega(A,V): t' \sqsubseteq s' \text{ s.t. } t \sqsubseteq_S t'$$

This proposition is analogous to the theorem of GUESSARIAN.

Let us recall if $\sum,(A,\Phi,R)$, is a RPS $\{\phi_i x_1 \ldots x_{n_i} = t_i / i \in |N|\}$, and C is a class of interpretations $\sum$ may be viewed as a system of equations in $M_\Omega^\infty(A,V)/\equiv_C$ whom the least solution is $<[T(\sum,\phi_1 x_1 \ldots x_{n_1})]_{\equiv_C}, \ldots, [T(\sum,\phi_N x_1 \ldots x_{n_N})]_{\equiv_C}>$
Further more we have the equivalence : $\phi_i x_1 \ldots x_{n_i} =_{<\sum,C>} \phi_j x_1 \ldots x_{n_j}$ iff $T(\sum_1,\phi_i x_1 \ldots x_{n_i}) \equiv_C T(\sum,\phi_j x_1 \ldots x_{n_j})$

We give now a technical result

Proposition 3 Let $\sum,(A,\Phi,R)$, be a RPS, s,s' be terms of $M(A\ \Phi,V)$, and S be a subset of $M_\Omega(A,V)\times M_\Omega(A,V)$ then

$$s \leq_S s' \Rightarrow \forall n \in \mathbb{N}\ \exists n' \in \mathbb{N}\quad \pi\alpha_{\sum}^{n}(s) \cdot \sqsubseteq_S \quad \pi\alpha_{\sum}^{n'}(s') \text{ and } n \leq n'$$

3. UF-TRANSFORMATIONS. FIRST RESULTS

Let us recall the transformation method of BURSTALL and DURLINGTON [4] using the so-called "unfold/fold" technique (also mentionned by MANNA and WALDINGER [20]).

In this section, we use freely the following notations :

V is a set of variables

A is a set of base functions

S is a subset of $M_\Omega(A,V) \times M_\Omega(A,V)$ which expresses properties of base functions

Φ is a set of procedure symbols.

Definition 1 Let $\sum$ and $\sum'$ be two RPSs, $\sum$ is transformed into $\sum'$ by UF-transformation if and only if

(i) $\sum$ is $\underset{1 \leq i \leq N}{F_i x_1 \ldots x_{n_i}} = t_i$

(ii) $\sum'$ is $\underset{1 \leq i \leq N}{F_i x_1 \ldots x_{n_i}} = t'_i$

(iii) $\forall i \in [N] \quad t_i \overset{*}{\underset{RUS}{\longrightarrow}} t'_i$

We denote this transformation by $\sum \underset{UF(S)}{\longrightarrow} \sum'$

To rely this definition to transformation rules of [4,9] ; let us detail the point (iii).

For some i of [N], $t_i \overset{*}{\underset{RUS}{\longrightarrow}} t'_i$ means there exists a sequence of terms of $M(A \cup \Phi, V)$ $\{s_j \mid 0 \leq j \leq p\}$ such that

- $s_o = t_i$ and $s_p = t'_i$
- $\forall j \in [p]$ $\quad s_{j-1} \underset{R}{\longrightarrow} s_j$ "unfolding"
  $\quad s_j \underset{R}{\longrightarrow} s_{j-1}$ "folding"
  $\quad s_{j-1} \underset{S}{\longrightarrow} s_j$ "laws about primitives"

Definition 2 Let $\sum$ and $\sum'$ be two RPSs such that $\sum \underset{UF(S)}{\longrightarrow} \sum'$, the performed transformation is correct if, and only if, for any s of $M(A \cup \Phi, V)$ and any M of C_S, $s_{<\sum,M>}$ is equal to $s_{<\sum',M>}$.

That means that the transformation preserves the function computed by any program under any interpretation of C_S. A straightforward application of the results of the section 2 leads us to the following result.

Proposition 1 Let $\sum$ and $\sum'$ be two RPS$_S$, $\{F_i x_1 \ldots x_{n_i} = t_i \mid i \in [N]\}$ and $\{F_i x_1 \ldots x_{n_i} = t'_i \mid i \in [N]\}$, such that $\sum \xrightarrow[UF(S)]{} \sum'$, then the performed transformation is correct if, and only if ;

$\forall i \in [N]\ \forall t \in L(\sum, F_i x_1 \ldots x_{n_i})\ \exists t' \in L(\sum', F_i x_1 \ldots x_{n_i})$ s.t. $t \sqsubseteq_S t'$

$\forall t' \in L(\sum', F_i x_1 \ldots x_{n_i})\ \exists t \in L(\sum, F_i x_1 \ldots x_{n_i})$ s.t. $t' \sqsubseteq_S t$

We say that $\sum$ and $\sum'$ are S-equivalent ($\sum \equiv_S \sum'$)

The main problem of this article is to study the correctness of UF-transformations. We give now a first result.

Proposition 2 Let $\sum$ and $\sum'$ be two RPS$_S$ such that $\sum \xrightarrow[UF(S)]{} \sum'$; in general the performed transformation is not correct.

Proof It is sufficient to give an example.

Let $\sum$ be a RPS, $\{F_i x_1 \ldots x_{n_i} = t_i \mid i \in [N], t_i(\Lambda) \in A\}$ (in Greibach form) and let $\sum'$ be the RPS $\{F_i x_1 \ldots x_{n_i} = F_i x_1 \ldots x_{n_i} \mid i \in [N]\}$ it is clear that $\sum \xrightarrow[UF]{} \sum'$ and, for any interpretation M, $F_i x_1 \ldots x_{n_i} < \sum', M>$ is the constant function $\perp_M$. Obviously it is not the case for $F_i x_1 \ldots x_{n_i} < \sum, M>$; thus the transformation is not correct.

Proposition 3 Let $\sum$ and $\sum'$ be two RPS$_S$ such that $\sum \xrightarrow[UF(S)]{} \sum'$ then

$\forall i \in [N]\ \forall t' \in L(\sum', F_i x_1 \ldots x_{n_i})\ \exists t \in L(\sum, F_i x_1 \ldots x_{n_i})\ t' \sqsubseteq_S t$

We note this property $\sum' \leq_S \sum$

We omit the proof in this draft and recall this result is mentionned in [4,5] also.

Intuitively the result means that UF-transformations preserve partial correctness of programs but not termination.

Our goal is to give conditions ensuring the property $\sum \leq_S \sum'$ which implies the equivalence $\sum \equiv_S \sum'$ with the converse inequality of the proposition 3 above.

Here we would mention a result of Courcelle [5].

Definition 3 Let $\sum$ be a RPS and S be a subset of $M_\Omega(A,V) \times M_\Omega(A,V)$; $\sum$ is S-univocal if, and only if, all its solutions (as a system of equations over $M_\Omega^\infty(A,V)$) are S-equivalent .

Proposition 4 [5] Let $\sum$ and $\sum'$ be two RPSs such that $\sum \xrightarrow[UF(S)]{} \sum'$; if $\sum'$ is S-univocal then the transformation is correct.

Proof Proposition 3 implies that $\langle T(\sum, F_1 x_1 \ldots x_{n_1}), \ldots, T(\sum, F_N x_1 \ldots x_{n_N}) \rangle$ is a solution of $\sum'$ and by definition of S-univocality, we have the equivalences $T(\sum, F_i x_1 \ldots x_{n_i}) \equiv_C T(\sum', F_1 x_1 \ldots x_{n_i}), i \in [N]$ and, $\sum \equiv_C \sum'$.

In his article, B. Courcelle gives sufficient conditions over S such that any RPS $\sum$ is S-univocal. This approach is very interesting since it gives sufficient conditions ensuring the correctness of any UF(S)-transformation but it deals with complex properties of terms rewriting systems (confluence, finite termination,...) which are difficult to check.

Our aim is less global and more pragmatic : we want give conditions for each UF-transformation and, if possible, find out these conditions automatically.

To reach this goal, we are going to make some assumptions on the performed transformations and define the WUF-transformations (W stands for weak).

Definition 4 Let $\sum$ and $\sum'$ be two RPSs, $\{F_i x_1 \ldots x_{n_i} = t_i \mid i \in [N]\}$ and $\{F_i x_1 \ldots x_{n_i} = t_i' \mid i \in [N]\}$, such that $\sum \xrightarrow[UF(S)]{i} \sum'$: this transformation is a WUF-transformation if, and only if, the following conditions are fulfilled

(i) $\forall i \in [N]\ i \geq 2 \Rightarrow t_i = t_i'$

(ii) $\exists s_1, s_2 \in M(A \cup \Phi, V)$ s.t. $t_1 \xrightarrow[\sum]{*} s_1, s_1 \xrightarrow{*} s_2$ and $t_1' \xrightarrow[\sum]{*} s_2$

Notations : We precise our notations (used in the sequel). If $\sum$ and $\sum'$ are two RPSs such that $\sum \xrightarrow[WUF(S)]{} \sum'$ then this transformation is fully defined by the 4-tuple $\langle s_1, s_2, s_3, F_j \rangle$ where s_1, s_2, s_3 belong to $M(A \cup \Phi, V)$ and F_j to Φ and

(1) $t_1 \xrightarrow[\sum]{*} s_1$ (unfolding)

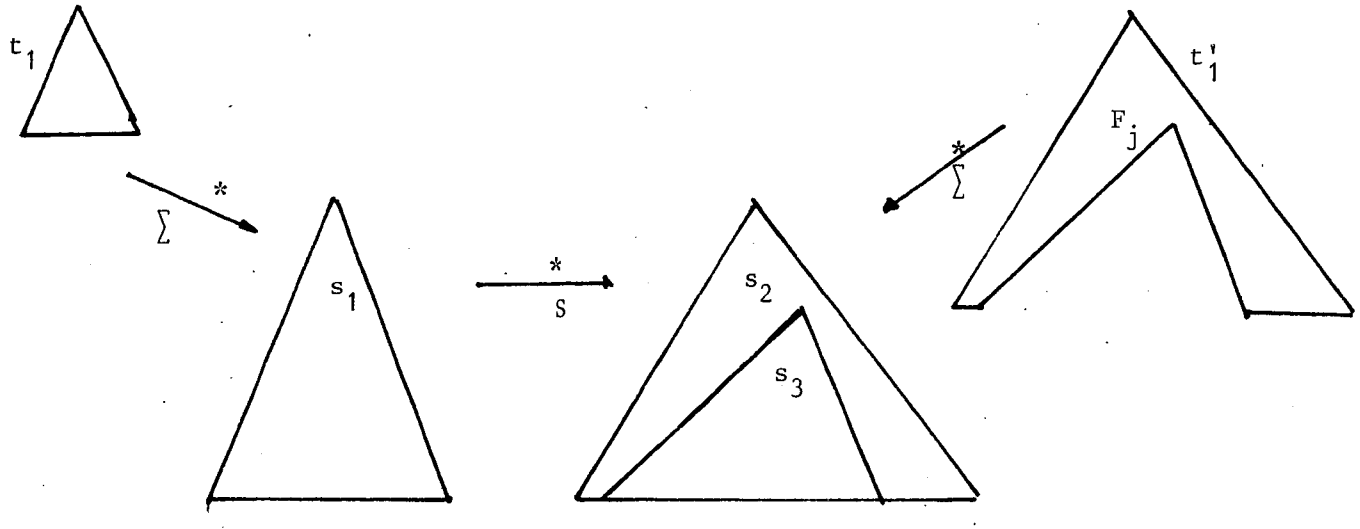
(2) $s_1 \xrightarrow[S]{*} s_2$ (laws)

(3) $\exists\, m \in dom(s_2)$ $s_3 = s_2/m$

(4) $\exists t^1, \ldots t^{n_j} \in M(A \cup \Phi, V)$ $t_1' = s_2[F_j t^1 \ldots t^{n_j}/m]$

(5) $F_j t^1 \ldots t^{n_j} \xrightarrow[\sum]{*} s_3$, thus $t_1' \xrightarrow[\sum]{*} s_2$ (folding)

(6) If $j=1$ then $F_1 t^1 \ldots t^{n_1} \xrightarrow[\sum]{+} s_3$ and there is exactly one rewriting related to F_1. Sometimes we shall use a graphical representation



In the next section, we are going to study the correctness of WUF-transformations.

4. WUF-TRANSFORMATIONS

First of all, we must mention that the propositions2 and 3 hold for the WUF-transformation. So, our goal is to give some conditions ensuring this correction .

We distinguish two cases : Let $\sum$ and $\sum'$ be two RPSs and $\langle s_1, s_2, s_3, F_j \rangle$ a WUF-transformation from $\sum$ to $\sum'$ then

(1) either j is equal to 1

(2) or j is not.

4.1. Study of a WUF-transformation $\langle s_1, s_2, s_3, F_j \rangle, j \neq 1$

That is the easy case and we have the following property.

Proposition 1 Let $\sum$ and $\sum'$ be two RPSs such that $\sum \xrightarrow[WUF(S)]{} \sum'$; if the transformation is given by $\langle s_1, s_2, s_3, F_j \rangle$ with $j \neq 1$ then the inequality $\sum \leq_S \sum'$ holds.

Proof : It is omitted in this version. Let us precise it is sufficient to prove the following inequalities

$$T(\sum, s_2) \leq_S T(\sum', s_2)$$

$$T(\sum, t_i) \leq_S T(\sum', t_i), \quad i \in [N], \ i \neq 1$$

As a corrolary of this proposition and the proposition 3 of section 3, we get

Theorem 1 Let $\sum$ and $\sum'$ be two RPSs such that $\sum \xrightarrow[WUF(S)]{} \sum'$; if the transformation is defined by $\langle s_1, s_2, s_3, F_j \rangle$ with $j \neq 1$, then the equivalence $\sum \equiv_S \sum'$ holds and the transformation is correct.

Example 1 Let $\sum$ be the RPS

$$\begin{cases} G(x) = R(X_1 f(g(x), G(k(x)))) \\ H(x,y) = f(y, G(x)) \end{cases}$$

We perform the folding

$$h(x, f(g(x), G(k(x)))) \xleftarrow[\sum]{} h(x, H(k(x), g(x)))$$

and the RPS $\sum'$ obtained is

$$\begin{cases} G(x)=h(x,H(k(x),g(x))) \\ H(x,y)=f(y,G(x)) \end{cases}$$

and $\sum$ and $\sum'$ are equivalent (S is the empty set)

4.2. Study of a WUF-transformation $\langle s_1,s_2,s_3,F_1\rangle$

It is the interesting case where the proposition 2 of section 3 holds. Let us recall the assumption made in the end of section 3 : in the folding operations from s_2 to t'_1 there is exactly one folding related to F_1 and this is the last one.

In this case we know that a WUF-transformation does not preserve termination of programs. The basic idea is to find out a new rule added to the set S and to show the equivalence of programs modulo a subclass of C_S.

Algorithm Let $\langle s_1,s_2,s_3,F_1\rangle$ be a WUF-transformation and let V' be the set of variables occurring in $\sum$: we know that s_1 and s_3 belong to $L(\sum,F_1x_1\ldots x_{n_1})$ (up to a substitution) and we define a term s_4 of $M_\Omega(A,V)$ in the following way

- $dom(s_4)=dom(s_1)\cap dom(s_3)$
- $\forall w\in dom(s_4)$ either $s_3(w)\in V'$ then $s_4(w)=s_3(w)$

  or $s_3(w)\in A$ if $s_1(w)=s_3(w)$ then $s_4(w)=s_3(w)$
  if $s_1(w)\in V'$ then $s_4(w)=s_1(w)$
  if $s_1(w)\in\Phi$ then $s_4(w)=\Omega$

  or $s_3(w)\in\Phi$ if $s_1(w)\neq s_3(w)$ then $s_4(w)=\Omega$
  else $s_4(w)=y$
  where y is a new variable of V/V'

Example 2 Let $\sum$ be the RPS

$F(x)=f(G(x),H(x))$
$G(x)=g(x\ ,G(x))$
$H(x)=h(H\ (x))$

(1) if $s_1=f(G(x),H(x))$ and $s_3=f(G(x),H(x))$ then $s_4=f(y_1,y_2)$
(2) if $s_1=f(g(x,G(x)),\ H(x))$ and $s_3=f(G(x),H(x))$ then $s_4=f(\Omega,y)$
(3) if $s_1=f(g(x,G(x)),h(H(x)))$ and $s_3=f(G(x),H(x))$ then $s_4=f(\Omega,\Omega)$

We are able to define the rule associated to a WUF-transformation.

Definition 1 Let be $<s_1,s_2,s_3,F_1>$ a WUF(S)-transformation from $\sum$ to $\sum'$; let us note s_4 the term defined by the above algorithm, we shall call "extra-rule" the pair (s_4,Ω) and S_s the new set of rules $SU\{(s_4,\Omega)\}$

We need a technical lemma

Lemma 1 Let be $<s_1,s_2,s_3,F_1>$ a WUF(S)-transformation from $\sum$ to $\sum'$ and s_4 be the term constructed by the algorithms above ; for any integer n there exists p, $p \leq n-1$, such that

$$\pi\alpha^{n}_{\sum}(s_3) \xi_{\{<s_4,\Omega>\}} \pi\alpha^{p}_{\sum}(s_1)$$

Proof omitted in this version

With this lemma we reach the crucial proposition.

Proposition 1 Let be $<s_1,s_2,s_3,F_1>$ a WUF(S)-transformation from $\sum$ to $\sum'$ and s_4 be the term constructed by the algorithm above ; then the following inequality holds

$$\sum \leq_{S_s} \sum'$$

Proof For sake of brievity we do not give this very technical proof. But, in the example 2 below, we detail a sketch of this proof.

That means by adding an extra-rule we obtain the desired inequality.

As a corrolary, with the proposition 3 of section 3, we have

Theorem 1 Let be $<s_1,s_2,s_3,F_1>$ a WUF(S)-transformation from $\sum$ to $\sum'$ and s_4 be the term of $M_\Omega(A,V)$ constructed by the algorithm ; then the following equivalence holds $\sum \equiv_{S_s} \sum'$.

Example 2 Let $\sum$ be the RPS

$$F_1(X)=f(F_2(x),F_3(x))$$
$$F_2(X)=g(x,F_2(x))$$
$$F_3(x)=h(F_3(x))$$

and S be the set of rules $\{<f(g(x,y),y'),g(x,f(y,y'))>\}$; we perform the following WUF(S)-transformation

$$f(F_2(x),F_3(x)) \xrightarrow[\sum]{} f(g(x,F_2(x)),F_3(x)) \qquad \text{unfolding}$$

$\xrightarrow[S]{}$ $g(x,f(F_2(x),F_3(x)))$ law

$\xleftarrow[\sum]{}$ $g(x,F_1(x))$ folding

and obtain the RPS $\sum'$

$F_1(x)=g(x,F_1(x))$
$F_2(x)=g(x,F_2(x))$
$F_3(x)=h(H_3(x))$

With our notations $s_1=f(g(x,F_2(x)),F_3(x))$
$s_2=g(x,f(F_2(x),F_3(x)))$
$s_3=f(F_2(x),F_3(x))$

Using the algorithm we find out $s_4=f(\Omega,x)$ and $S_s=\{<f(g(x,y),y'),$ $g(x,f(y,y'))>,<f(\Omega,x),\Omega>\}$. We are going to prove the inequality $\sum \leq_{S_s} \sum'$. The proof of this particular inequality may be viewed as a sketch of the proof of proposition 1.

The inequality $\sum \leq_{S_s} \sum'$ means that

$\forall i \in \{1,2,3\}$ $T(\sum,F_i(x)) \leq_{S_s} T(\sum',F_i(x))$

In this example, it is obvious that both equivalences $T(\sum,F_2(x))\equiv_S T(\sum',F_2(x))$ and $T(\sum,F_3(x)) \equiv_S T(\sum',F_3(x))$ hold. Morover, we know $T(\sum,F_1(x))\equiv_S T(\sum,s_2)$ holds too ; whence it is sufficient to prove the following inequality $T(\sum,s_2) \leq_{S_s} T(\sum',E(x))$; in other words

$\forall n\in \mathbb{N}\ \exists n'\in \mathbb{N}\ \pi\alpha^n_{\sum}(s_2) \Xi_{S_s} \pi\alpha^{n'}_{\sum'}(F_1(x))$

This proof is made by induction over n

. <u>n=0</u> $s_2 = g(x_1,f(\Omega,\Omega)) \xrightarrow[<s_4,\Omega>]{} g(x,\Omega)$

Thus $s_2 \Xi_{S_s} \pi\alpha_{\sum'}(F_1(x))$

Assume that the result holds for any integer less than n+1.

. <u>n+1</u> $\pi\alpha^{n+1}_{\sum}(s_2)=g(x,\pi\alpha^{n+1}_{\sum}(s_3))$

By lemma 1, we know there exists an integer p such that

$$\pi\alpha_{\sum}^{n+1}(s_3) \sqsubseteq_{\{<s_4,\Omega>\}} \pi\alpha_{\sum}^{p}(s_1) \text{ and } p \leq n.$$

But the proposition 3 (cf. § 2.8.) ensures us there exists an integer q such that $\pi\alpha_{\sum}^{p}(s_1) \sqsubseteq_S \pi\alpha_{\sum}^{q}(s_2)$ and $q \leq p \leq n$.

By induction hypothesis, there exists r of $\mathbb{N}$ such that $\pi\alpha_{\sum}^{q}(s_2) \sqsubseteq_{S_s} \pi\alpha_{\sum}^{r},(F_1(x))$

If we sum up all these inequalities, we obtain

$$\pi\alpha_{\sum}^{n+1}(s_3) \sqsubseteq_{\{<s_4,\Omega>\}} \pi\alpha_{\sum}^{p}(s_1) \sqsubseteq_S \pi\alpha_{\sum}^{q}(s_2) \sqsubseteq_{S_s} \pi\alpha_{\sum}^{r},(F_1(x))$$

and $q \leq p \leq n$.

So we reach the two following inequalities

$$\pi\alpha_{\sum}^{n+1}(s_3) \sqsubseteq_{S_s} \pi\alpha_{\sum}^{r},(F_1(x))$$

and $g(x, \pi\alpha_{\sum}^{n+1}(s_3)) \sqsubseteq_{S_s} g(x, \pi\alpha_{\sum}^{r},(F_1(x)))$

We have just proved there exists some integer n' such that

$$\pi\alpha_{\sum}^{n+1}(s_2) \sqsubseteq_{S_s} \pi\alpha_{\sum}^{n'},(F_1(x))$$

Then the inequality $\sum \leq_{S_s} \sum'$ holds and the equivalence $\sum \equiv_{S_s} \sum'$ holds too.

Let us remark the equivalence $\sum \equiv_S \sum'$ does <u>not</u> hold.

<u>Example 3</u> Let $\sum$ be the RPS $\{G(x)=f(G(x))\}$ and S be the set $\{<f(x),g(f(x))>\}$; we perform the following WUF(S)-transformation

$s_1=f(G(x)), s_2=g(f(G(x))), s_3=f(G(x))$ and reach the RPS $\sum'$ $\{G(x)=g(G(x))\}$. Here s_4 is equal to $f(x)$ and S_s to $\{<f(x),g(f(x))>, <f(x),\Omega>\}$. Again the equivalences $\sum \equiv_{S_s} \Omega \equiv_{S_s} \sum'$ hold too.

Obviously, the extra rule ensures the correctness of WUF(S)-transformation (modulo S_s) but, sometimes, this involves the "trivialisation" of RPSs.

<u>Definition 2</u> Let $<s_1,s_2,s_3,F_1>$ be a WUF(S)-transformation from $\sum$ to $\sum'$; we say the extra-rule $<s_4,\Omega>$ gets trivial the RPS $\sum$ iff there is no occurrence of Ω in s_4.

With this definition we state.

Proposition 2 Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a WUF(S)-transformation from $\sum$ to $\sum'$; if the rule $\langle s_4, \Omega \rangle$ gets trivial the RPS $\sum$ then the equivalence $T(\sum, F_1 x_1 .. x_{n_1}) \equiv_{\{\langle s_4, \Omega \rangle\}} \Omega$ holds.

Now we give a very simple condition which involves a "bad" extra-rule.

Proposition 3 With the usual notations the rule $\langle s_4, \Omega \rangle$ gets trivial the RPS $\sum$ if, and only if, the number of folding is greater than the number of unfolding.

This proposition is very important if we consider strict interpretations.

Definition 3 Let S be a set of rules, we note ST(S) the set of interpretations which satisfy S and are strict.

Theorem 2 [10] Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a WUF(S)-transformation from $\sum$ to $\sum'$; if the number of unfolding is greater or equal to the number of folding then the equivalence $\sum \equiv_{ST(S)} \sum'$ holds.

This result is of practical interest if we think about programs written in Lisp for instance ([18]).

Now, we propose another extra-rule which is necessary under some assumptions.

Definition 4 Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a WUF(S)-transformation from $\sum$ to $\sum'$ and s_4 be the term defined by our algorithm ; we note $\pi(s_4)$ the term $\pi s_2 [s_4/s_3]$ and for some t' of $L^{\alpha}(\sum; F_1 x_1 \ldots x_{n_1})$ we consider the rule $\langle \pi(s_4), t' \rangle$. The set $S \cup \{\langle \pi(s_4), t' \rangle\}$ is denoted by $S_{t'}$.

This definition leads to a proposition which generalizes the proposition 1 above.

Proposition 4 Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a WUF(S)-transformation from $\sum$ to $\sum'$ the inequality $\sum \leq_{S_{t'}} \sum'$ ($S_{t'}$ is defined as above) holds

Definition 5 Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a WUF(S)-transformation from $\sum$ to $\sum'$; this transformation is said <u>safe</u> iff in the construction of s_4 by our algorithm there is no need of auxiliary variables i.e.variables occuring in s_4 already occured in s_1 and s_3.

With the above definitions and notations we can state.

Theorem 3 Let $\langle s_1, s_2, s_3, F_1 \rangle$ be a safe WUF(S)-transformation from $\sum$ to $\sum'$: the equivalence $\sum \equiv_S \sum'$ holds if, and only if, there exists t' of $L^{\alpha}(\sum', F_1 x_1 \ldots x_{n_1})$ such that $h(s_4) \sqsubseteq_S t'$.

In the case of a safe WUF(S)-transformation add an extra-rule is a necessary and sufficient condition which ensures us the correctness of this safe WUF(S)-transformation.

Here we stop the study of WUF-transformations and UF-transformations. It is possible to give some results about UF-transformation with some assumptions weaker than the assumptions made for WUF-transformations. However it is clear there is no general result in the general case.

5. UF-TRANSFORMATIONS AND EQUIVALENCE PROOFS

In section 3, we said a UF(S)-transformation from $\sum$ to $\sum'$ is correct if, and only if, $\sum$ and $\sum'$ are S-equivalent. Conversely it sounds good to prove the S-equivalence between two RPSs by exhibiting a correct UF(S)-transformation from one to the other. Then any UF-transformation system become a system for proving equivalence of recursive programs.

In this section, we present two proof methods [17] without any recursion induction principle (see also [2,5,13]).

5.1. The Mc Carthy method

Proposition 1 Let $\sum_1$ and $\sum_2$ be two RPSs, $\{F_i x_1..x_{n_i}=t_i^1 \mid i\in[N]\}$ and $\{F_i x_1..x_{n_i}=t_i^2 \mid i\in[N]\}$, if there exists a RPS $\sum_3$ and two UF(S)-transformation from $\sum_1$ to $\sum_3$ and from $\sum_2$ to $\sum_3$ which are correct then $\sum_1$ and $\sum_2$ are S-equivalent.

Proof : Obvious (definition of correctness and transitivity of S-equivalence).

Example 1 Let $\sum_1$ and $\sum_2$ be the RPSs

$$\sum_1 \begin{cases} F_1(x)=g(F_1(x)) \\ F_2(x)=g(F_1(x)) \end{cases} \qquad \sum_2 \begin{cases} F_1(x)=g(F_1(x)) \\ F_2(x)=f(F_1(x)) \end{cases}$$

and S be the set of rules $\{<f(g(x)),g(g(f(x)))>,<f(\Omega),g(\Omega)>\}$, let us consider the RPS $\sum_3$

$$\sum_3 \begin{cases} F_1(x)=g(F_1(x)) \\ F_2(x)=g(g(F_2(x))) \end{cases}$$

We have $\sum_1 \xrightarrow[UF(S)]{} \sum_2$ because

$$g(F_1(x)) \xleftarrow{\sum_1} g(g(F_1(x))) \xrightarrow{\sum_1} g(g(g(F_1(x)))) \xleftarrow{\sum_1} g(g(F_2(x)))$$

and $\sum_2 \xrightarrow[UF(S)]{} \sum_3$ because

$$f(F_1(x)) \xleftarrow{\sum_2} f(g(F_1(x))) \xrightarrow{S} g(g(f(F_1(x)))) \xleftarrow{\sum_2} g(g(F_2(x)))$$

The results of the section 4 ensure us the correctness of the performed transformations, whence the equivalence $\sum_1 \equiv_S \sum_2$ holds.

Example 2 shows us the usefulness of this method.

Example 2 Let $\sum$ be the RPS $\{F(x,y)=h(x,y,f(F(g(x),y)))\}$ and S be the set of rules $\{<f(h(x,y,z)),h(x,f(y),f(z))>,<f(\Omega),\Omega>\}$

Assume we want to prove the equivalence $f(F(x,y)\equiv_{<\sum,S>} F(x,f(y))$, we may use the Mc Carthy method in the following way :

1. Introduction a new procedure symbol of arity 2, say G
2. Define two RPSs $\sum_1$ and $\sum_2$

$$\sum_1 \begin{cases} G(x,y)=f(F(x,y)) \\ F(x,y)=h(x,y,f(F(g(x),y))) \end{cases}$$

$$\sum_2 \begin{cases} G(x,y)=F(x,f(y)) \\ F(x,y)=h(x,y,f(F(g(x),y))) \end{cases}$$

3. Apply the Mc Carthy method to $\sum_1$ and $\sum_2$

Let us consider the RPS $\sum_3$

$$\sum_3 \begin{cases} G(x,y)=h(x,f(y),f(G(g(x),y))) \\ F(x,y)=h(x,y,f(F(g(x),y))) \end{cases}$$

We have $\sum_2 \overrightarrow{UF(S)} \sum_3$ since

$$F(x,f(y)) \xrightarrow[\sum_2]{} h(x,f(y),f(F(g(x),f(y)))) \xleftarrow[\sum_2]{} h(x,f(y),f(G(g(x),y)))$$

and we have $\sum_1 \overrightarrow{UF(S)} \sum_3$ since

$$f(F(x,y)) \xrightarrow[\sum_1]{} f(h(x,y,f(F(g(x),y))) \xrightarrow[S]{} h(x,f(y),f(f(F(g(x),y))))$$

$$\xleftarrow[\sum_1]{} h(x,f(y),f(G(g(x),y)))$$

Again the results of section 4 guarantee the correctness of UF(S)-transformations, so the equivalence $\sum_1 \equiv_S \sum_2$ holds and, as a corollary, we get the equivalence $f(F(x,y))\equiv_{<\sum,S>} F(x,f(y))$

This quite elegant method (nothing else than an implementation of the Mc Carthy induction principle [18]) is incomplete of course. But, in some sense, it is "too incomplete" [3]. That means that very simple equiva-

lence cannot be proved with this method. For instance if $\sum$ is the RPS of Example 2 and $\sum'$ the RPS $F(x,y)=h(x,y),F(g(x),f(y)))$, it is impossible to prove the S-equivalence of $\sum$ and $\sum'$ by the Mac Carthy method. In order to deal with this problem, we introduce a new type of transformation rule which generalizes the UF-transformation rules.

5.2. Second order replacement method

Definition 1 Let $\sum$ be RPS and C be a class of interpretations, the pair (t,t') of elements of $M(A \cup \Phi,V)$ is a $<\sum,C>$-second order replacement (abbreviated in $<\sum,C>$-SOR) iff t and t' are $<\sum,C>$-equivalent.

Definition 2 Let $\sum$ be a RPS, $\{F_i x_1 \ldots x_{n_i} = t_i \mid i \in [N]\}$ and $\{(s_j, s'_j) \mid j \in [N]\}$ be a set of $<\sum,C>$-SORs ; if, for any i in $|N|$, there exist ν_i a substitution and j_i in $[N]$ such that $\nu_i^*(s_{j_i})$ is a subterm of t_i, we can define $t'_i = t_i[\nu_i^*(s'_{j_i})/\nu_i^*(s_{j_i})]$ and the RPS $\sum'$ $\{F_i x_1 \ldots x_{n_i} = t'_i \mid i \in [N]\}$.

We shall say that $\sum$ is transformed into $\sum'$ by a SOR(C)-transformation and write $\sum \xrightarrow{SOR(C)} \sum'$.

Obvisously, the SOR transformation is a generalization of UF-transformations and raises the same problems about its correctness.

Proposition 1 Let $\sum$ and $\sum'$ be two RPSs and C be a class of interpretations ; if there exists a SOR(C)-transformation from $\sum$ to $\sum'$ which is correct then the equivalence $\sum \equiv_C \sum'$ holds.

Of course, proofs of correctness of SOR-transformations are rather difficult to manage but if we restrain to classes of interpretations which are strict and equationnal it is possible to have some general results (see [18]).

Example 1 Let $\sum$ be again the RPS $F(x,y)=h(x,y,f(F(g(x),y)))$ and S be the set $\{<f(h(x,y,z)),h(x,f(y),f(z))>, <f(\Omega,x),\Omega>\}$; the pair $<f(F(x,y)), F(x,f(y))>$ is a $<\sum,C_S>$-SOR as shown in the Example 2 of § 5.1. Whence, by the SOR method, we reach the RPS $\sum'$ $F(x,y)=h(x,y,F(g(x),f(y)))$.

To conclude this section, we describe a method combining the twopreceeding methods.

5.3. A "system" for proving equivalence of RPSs

Let $\sum$ and $\sum'$ be two RPSs and S be a set of laws ; assume we want prove the equivalence $\sum \equiv_S \sum'$ holds then

(1) Find some pairs of terms which transform $\sum$ into $\sum'$ by SOR(S) transformations ; go to (3)

(2) If they do not exist, then $\sum$ and $\sum'$ are not S-equivalent and the "system" halts

(3) (3.1.) Try to prove these pairs are $<\sum,S>$-SORs by the Mac Carthy method
(3.2.) or goto (1)
(3.3.) If (3.1.) and (3.2.) fail then the "system" fails.

Obviously, the "system" is not automatic (even more or less) but rather human. Nevertheless we believe these guidelines are usefull for proving by hand equivalences and may provide automatic systems in particular cases.

6. CONCLUDING REMARKS

First of all, we mention there are many people who developp programs by using the UF-transformation implemented by M. Burstall and J.Darlington in Edinburgh [4], among them M. Feather [9]. We think there are two axes of futur research.

1. In the same vein, program transformations may be used for developping programs written in an applicative style as in [15,25]. In the former paper, the authors use a catalogue of transformations schemes (directly inspired of [14]) ; in the latter. P. Wadler gives a complete set of transformation rules for a given applicative language (over lists). It would be worthwhile to design a UF-transformation system which transforms FP-programs into FP-programs using parallel operators (as the α or tree operators) as many as possible. The outcome of such transformations will be the design of a highly parallel architecture.

2. Following the idea of [18,25] ; it seems very fruitful to consider a transformation system for developping programs written in a given programming language ; whence an increasing efficiency. Moreover this system could contain a catalogue of the most current transformations ; whence another gain of efficiency and, maybe, the end of the catalogue versus discovery fight.

7. BIBLIOGRAPHY

1 Arsac J., "Syntactic source to source transform", Comm. of Assoc. Comput. Mach., 22, 1, 43-54 (1979)

2 Begay D., Kott L., "Preuve de programme sans induction", 5th Coll. de Lille, Report. Univ. Poitiers (1980)

3 Boudol G., Kott L., "Recursion induction principle revisited", to appear in Theor. Comput. Sc. (1981)

4 Burstall R., Darlington J., "A transformation system for developping recursive programs", J. Assoc. Comput. Mach., 24, 1, 46-67 (1977)

5 Courcelle B., "Infinite trees in normal form and recursive equations having a unique solution", Math. System. Theory, 13, 131-180 (1979)

6 Courcelle B., Guessarian I., "On some classes of interpretations", J. of Comput. System Sci., 17, 388-413 (1978)

7 Courcelle B., Nivat M., "The algebraic semantics of recursive program schemes", MFCS 78, LNCS 64, Springer-Verlag, 16-30 (1978)

8 Downey P., Sethi R., "Correct computation rules for recursive languages", SIAM J. of Computing, 5 (1976)

9 Feather M., "A system for developping programs by transformations", Ph. D. Thesis, Edinburgh (1979)

10 Gerhart S., "Correctness preserving program transformations", 2n POPL Conf., Palo Alto (1975)

11 Guessarian I., "Semantic equivalence of program schemes and its syntactic characterization", 3nd ICALP, Edinburgh, 189-200 (1976)

12 Guessarian I., "Algebraic Semantics", LNCS 99, Springer-Verlag (1981)

13 Huet G., Lang B., "Proving and applying program transformations expressed with 2nd order patterns", Acta -Infor, 11, 31-55 (1978)

14 Huet G., Oppen D., "Equations and rewriterules. A survey "in "Formal Languages perspectives and open problems", R. Book ed., Academic Press (1980)

15 Kieburtz R., Shultis J., "Transformation of FP Program Schemes", Conf. on FP Languages & Computer Architecture, Portsmouth, 41-48 (1981)

16 Kott L., "About transformations System : a theoretical study" in "Program Transformations", B. Robinet ed., Dunod, 232-247 (1978)

17 Kott L., "A system for proving equivalences of recursive programs", 5th Coll. on Automated Deduction, LNCS 87 (1980).

18 Kott L., "The Mac Carthy's induction principle : "oldy" but "goody", to appear in Calcolo.

19 Lovemann D., "Program improvement by source-to-source transformations", J. Assoc. Comput. Mach., 24, 1, 121-145 (1977)

20 Manna Z., Waldinger R., "Knowledge and reasoning in program synthesis", Artif. Intel. 5, 6, 2, 175-208 (1975)

21 Nivat M., "On the interpretation of recursive polyadic program schemes", Instituto Nazionale di Alta Matematica, Symposia Matematica, Vol.XV, 255-281 (1975)

22 Nivat M., "Interpretation universelle d'un schéma de programme récursif", Informatica, 7, 1, 9-16 (1977)

23 Sherlis W., "Program Improvement by internal specialization", 8th POPL, Williamsbourg (1981).

24 Vuillemin J., "Correct and optimal implementation of recursion in a simple programming language", J. of Comput. System Sci., 9, 332-354 (1974)

25 Wadler P., "Application style programming, program transformations, and list operators", Conf. on FP Language & Computer Architectures, Portsmouth, 25-32 (1981).

Liste des Publications Internes IRISA

PI 147 **Deux files d'attente à capacité limitée en tandem**
J. Pellaumail, J. Boyer , 19 pages ; *Juillet 1981*

PI 148 **Programme de classification hiérarchique : 1) Méthode de la vraisemblance des liens, 2) Méthode de la variance expliquée**
I.C. Lerman , 113 pages ; *Juin 1981*

PI 149 **Convergence des méthodes de commande adaptative en présence de perturbations aléatoires**
J.J. Fuchs , 46 pages ; *Juillet 1981*

PI 150 **Construction automatique et évaluation d'un graphe d'«implication» issu de données binaires, dans le cadre de la didactique des mathématiques**
H. Rostam , 112 pages ; *Juin 1981*

PI 151 **Réalisation d'un outil d'évaluation de mécanismes de détection de pannes]-]Projet Pilote SURF**
B. Decouty, G. Michel, C. Wagner, Y. Crouzet , 59 pages ; *Juillet 1981*

PI 152 **Règle maximale**
J. Pellaumail , 18 pages ; *Septembre 1981*

PI 153 **Corrélation partielle dans le cas « qualitatif »**
I.C. Lerman , 125 pages ; *Octobre 1981*

PI 154 **Stability analysis of adaptively controlled not-necessarily minimum phase systems with disturbances**
Cl. Samson , 40 pages ; *Octobre 1981*

PI 155 **Analyses d'opinions d'instituteurs à l'égard de l'appropriation des nombres naturels par les élèves de cycle préparatoire**
R. Gras , 37 pages ; *Octobre 1981*

PI 156 **Récursion induction principle revisited**
G. Boudol, L. Kott , 49 pages ; *Décembre 1981*

PI 157 **Loi d'une variable aléatoire à valeur R^+** réalisant le minimum des moments d'ordre supérieur à deux lorsque les deux premiers sont fixés
M.Kowalowka, R. Marie , 8 pages ; *Décembre 1981*

PI 158 **Réalisations stochastiques de signaux non stationnaires, et identification sur un seul échantillon**
A. Benveniste J.J. Fuchs , 33 pages ; *Mars 1982*

PI 159 **Méthode d'interprétation d'une classification hiérarchique d'attributs-modalites pour l'«explication» d'une variable ; application à la recherche de seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire**
B. Tallur , 34 pages ; *Janvier 1982*

PI 160 **Probabilite stationnaire d'un réseau de files d'attente multiclasse à serveur central et à routages dépendant de l'état**
L.M. Le Ny , 18 pages ; *Janvier 1982*

PI 161 **Détection séqentielle de changements brusques des caractéristiques spectrales d'un signal numérique**
M. Basseville, A. Benveniste , pages ; *Mars 1982*

PI 162 **Actes regroupés des journées de Classification de Toulouse (Mai 1980), et de Nancy (Juin 1981)**
I.C. Lerman , 304 pages ;

PI 163 **Modélisation et Identification des caractéristiques d'une structure vibratoire : un problème de réalisation stochastique d'un grand système non stationnaire**
M. Prévosto, A. Benveniste, B. Barnouin , 46 pages ; *Mars 182*

PI 164 **An enlarged definition and complete axiomatization of observational congruence of finite processes**
Ph. Darondeau , 45 pages ; *Avril 1982*

PI 165 **Accès vidéotex à une banque de données médicales**
A. Chauffaut, M. Dragone, R. Rivoire, J.M. Roger , 25 pages ; *Mai 1982*

PI 166 **Comparaison de groupes de variables définies sur le même ensemble d'individus**
B. Escofier, J. Pages , 115 pages ; *Mai 1982*

PI 167 **Transport en circuits virtuels internes sur réseau local et connexion Transpac**
M. Tournois, R. Trépos , 90 pages ; *Mai 1982*

PI 168 **Impact de l'intégration sur le traitement automatique de la parole**
P. Quinton , 14 pages ; *Mai 1982*

PI 169 **A systolic algorithm for connected word recognition**
J.P. Banâtre, P. Frison, P. Quinton , 13 pages ; *Mai 1982*

PI 170 **A network for the detection of words in continuous speech**
J.P. Banâtre, P. Frison, P. Quinton , 24 pages ; *Mai 1982*

PI 171 **Le langage ADA : Etude bibliographique**
J. André, Y. Jégou, M. Raynal , 12 pages ; *Juin 1982*

PI 172 **Comparaison de plusieurs variables : 2ème partie : un exemple d'application**
B. Escofier, J. Pajès , 37 pages ; *Juillet 1982*

PI 173 **Unfold-fold program transformations**

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique