REWRITING WITH A NONDETERMINISTIC CHOICE OPERATOR : FROM ALGEBRA TO PROOFS

Stéphane Kaplan
Department of Applied Mathematics
The Weizmann Institute of Science
76100 Rehovot (Israel)

LRI. Bat. 490
Université des Sciences
91405 Orsay (France)

Abstract :

The privileged field of classical algebra and term rewriting systems is that of strictly deterministic systems : the *confluence* property is generaly assumed to hold, which ensures determinism about the result of the computations, even if there exist several different computation paths. In this paper, we develop a new formalism introducing a bounded nondeterministic choice operator '↑' into algebraic specifications and related term rewriting systems ; nondeterminism about the result becomes allowed in this framework. We define the algebraic and the operational aspects of such systems, and investigate their relationship. Methods *à la Knuth-Bendix* are developed for automatic theorem proving in such theories. Several examples are considered, including a toy concurrent language, for which non-trivial properties may be automatically proved.

INTRODUCTION

The field of term rewriting systems has greatly developed during the past several years. It provides an appropriate operational description for algebraic specifications, defining abstract implementation via symbolic evaluation. Also, powerful theorem proving tools exist for such systems. However, an essential constraint on those systems is that they must be *confluent*. Roughly speaking, the confluence property means that, even if different computation paths are possible in order to evaluate a given term, the result must be unique. Therefore, nondeterministic and concurrent specifications have seemed banished from that framework.

In this paper, we propose a new formalism that allows nondeterminism to be introduced in term rewriting systems. This formalism extends the classical (strictly deterministic) one, while maintaining its most important properties : coherence between the algebraic and the operational aspects, possibilities of automatic theorem proving. Our main idea is to introduce a special purpose operator $\uparrow$, realizing bounded nondeterministic choice. All the non-confluence that is authorized in such a system is compelled to derive from the sole operator $\uparrow$. This operator undergoes a particular treatment in the procedures that we shall define, whereas the other operators will behave classically.

We had to define a new algebraic and operational formalism, the technicalities of which are sometimes intricate. We advise readers who do not wish to concern themselves with these details to skip directly to example 4.1, that consists of the proof of a property in a nondeterministic specification. The main idea behind this paper should be understandable via that example for a reader who knows about classical term rewriting systems. For similar reasons, most of the proofs of our results are not given in the text itself, but postponed to appendices - except particularly relevant ones.

Our formalism can cope with a large class of concurrent specifications. In that respect, we are very much indebted in the work of the ACP group in Amsterdam. Actually, the results of this paper apply to a simplified version (strictly functional, no explicit sequencing) of ACP. An important difference between our approaches is that a model, for us, must satisfy *either* $a \uparrow b = a$ *or* $a \uparrow b = b$; the semantics of our formalization consists of the class of all such models. For the ACP group, semantics is defined via certain particular (initial) models, in which the previous property need not be verified : the operator $\uparrow$ is not systematically eliminated. The benefit of such simplifications applied to the ACP framework is that *automatic* theorem proving is now possible via the techniques developed in this paper.

A crucial hypothesis of our approach is that all the *computations eventually terminate*, as it is the case for classical term rewriting systems. Thus, we need not consider the properties of infinite nondeterministic calculi, as found for instance in [Nivat 80], [BW 81], [Poigné 81], [Hennessy 1982].

Part 1 defines the algebraic basis of our approach. Part 2 introduces our notion of rewriting, and establishes its connection with the algebraic aspects of the formalism via a Birkhoff theorem. Termination and normal form computation issues are discussed. Part 3 defines our extended notion of $\uparrow$-confluence. A Knuth-Bendix theorem and completion procedure are given, allowing the $\uparrow$-confluence of terminating nondeterministic term rewriting systems to be checked. Lastly, part 4 considers the

application of these methods to theorem proving in structured nondeterministic specifications. Several examples, including a toy concurrent language, are given. We assume that the reader has a basic knowledge of algebras, and term rewriting systems (cf. [ADJ 78], [HO 80] for basic references), nondeterminism (cf. [Nivat 80], [Apt 84]) and concurrency (cf. [Hoare 78], [Milner 80], [Brookes 83], [Boudol 84]). However, notations and concepts are systematically redefined, and the paper should be self-contained.

1. THE ALGEBRAIC FRAMEWORK

1.1. The Free Models

An $\uparrow$-*signature* consists of :

- a set of domain names S, called *sorts*,
- a set of operator names Σ, with an arity function on S. For some sorts $s \in S$, there may exist a distinguished operator : $\uparrow_s : s \times s \to s$, that will represent the *nondeterministic choice* between two elements of sort s. For such a sort s, there may exist a distinguished constant : $\delta_s : \to s$, standing for the *deadlock constant*. Such operators appear in several algebraic frameworks (cf. e.g. [Hoare 78], [Milner 80], [Boudol 84]). We shall write $\uparrow$ and δ instead of $\uparrow_s$ and δ_s when no confusion is possible.

Moreover, we consider only well-formed signatures (cf. [HH 80], [GM 81]), imposing for instance that there is at least one constant operator per sort s (different from δ_s).

Let $X = (X_s)_{s \in S}$ be an infinite set of typed variables. We define, as usual, the set $T_{S,\Sigma}(X)$ of the *terms* that are well-formed on the signature $(S,\Sigma \cup X)$. We define $T_{S,\Sigma}^{\Theta}(X)$ as the subset of $T_{S,\Sigma}(X)$ containing no $\uparrow$ symbol. $T_{S,\Sigma}$ and $T_{S,\Sigma}^{\Theta}$ stand for the two similar sets of terms containing no variables. Such terms are often called *ground* terms.

We consider *occurrences* in terms as finite strings of integers in the usual manner. For a term t and an occurrence ω in t, $t_{|\omega}$ stands for the subterm of t the root of which is at occurrence ω. $t[\omega \leftarrow t']$ is the term t, where $t_{|\omega}$ is replaced by the term t'. *Substitutions* are defined in the usual way. The application of a substitution σ to a term t is written $t\sigma$. We say that a term t matches the term G, called a pattern, at occurrence ω via the substitution σ if $t_{|\omega} = G\sigma$. Two terms t and t' are *unifiable* if there exists a substitution σ such that $t\sigma = t'\sigma$. In that case, they admit a most general unifier-- i.e. a substitution μ that is a unifier of t and t', and such that for every unifier σ of t and t', there exists a substitution σ' such that $\sigma = \mu\sigma'$. A *context* is a term $K \in T_{S,\Sigma}(X)$ with a distinguished variable occurrence. K[t] denotes K where this variable occurrence is replaced by the term t.

For a given relation $\to$, $\to^*$ will denote its reflexive and transitive closure, and $\to^{-1}$ its inverse. Thus, $(\to \cup \to^{-1})^*$ denotes the reflexive, symmetric and transitive closure of $\to$.

An $\uparrow$-*model* M of a given signature (S,Σ) consists of :

- a family of sets $(M_s)_{s \in S}$ indexed by S;
- a family $\text{eval}^M = (\text{eval}_s^M)_{s \in S}$ of applications $\text{eval}_s^M : (T_{S,\Sigma})_s \to M_s$ such that, for any context K[X], for any $t,t',t'' \in T_{S,\Sigma}$:

(0)	either :	$eval^M(K[\, t \uparrow t' \,])$	=	$eval^M(K[\, t \,])$,
	or :	$eval^M(K[\, t \uparrow t' \,])$	=	$eval^M(K[\, t' \,])$
(1)		$eval^M(K[\, (t \uparrow t') \uparrow t'' \,])$	=	$eval^M(K[\, t \uparrow (t' \uparrow t'') \,])$
(2)		$eval^M(K[\, t \uparrow t' \,])$	=	$eval^M(K[\, t' \uparrow t \,])$
(3)		$eval^M(K[\, t \uparrow t \,])$	=	$eval^M(K[\, t \,])$
(4)		$eval^M(K[\, t \uparrow \delta \,])$	=	$eval^M(K[\, t \,])$
		$eval^M(K[\, \delta \uparrow t \,])$	=	$eval^M(K[\, t \,])$

Note :

• Property (0) means that in an $\uparrow$-model, a choice has to be done between the two arguments of the nondeterministic choice operator $\uparrow$.

• Properties (1) to (4) respectively state the associativity, commutativity, idempotency (actually, (0) implies (3)) and neutrality of δ for the nondeterministic choice in the $\uparrow$-models. This will be henceforth referred to as the ACIN property. This assigns the meaning of *bounded nondeterministic choice* to the $\uparrow$ operator.

• In the classical, strictly deterministic case (cf. e.g. [ADJ 78]), models are defined providing a family $(M_s)_{s \in S}$ of sets, and for each operator $f : s_1 \times \cdots \times s_n \to s'$ in Σ an *interpretation* $f^M : M_{s_1} \times \cdots M_{s_n} \to M_{s'}$. However, this is not possible in our formalism, since we need to be able to describe a model M in which $a^M \uparrow^M b^M = a^M$, while $f^M(a^M \uparrow^M b^M) = f^M(b^M)$. This is clearly not compatible with the classical definition of an interpretation f^M. However, there is no contradiction in our formalization, since we simply have :

$$eval^M[a \uparrow b] = eval^M[a] \text{ , and } eval^M[f(a \uparrow b)] = eval^M[f(b)] .$$

This is why the notion of model had to be re-defined in our framework.

• The class of the $\uparrow$-models is called MOD. The class of the $\uparrow$-models M such that $eval^M$ is surjective is called the class of the *finitely generated* $\uparrow$-models fgMOD.

In general, $T_{S,\Sigma}$ is not an $\uparrow$-model. Intuitively, to view it as an $\uparrow$-model, we need to specify how choice is done between the arguments of the $\uparrow$ symbol, for every occurrence of $\uparrow$ in every term of $T_{S,\Sigma}$. To do so, we define an *$\uparrow$-choice* as a partial application :

$$C : T_{S,\Sigma} \times N^* \to \{ 1 , 2 \} .$$

Intuitively, for a given term t and an occurrence ω of $\uparrow$ in t, we will have :

(c_1) $eval^M[t] = eval^M[t\{\omega \leftarrow \text{left-son } (t_{|\omega})\}]$ iff $C(t,\omega) = 1$

(c_2) $eval^M[t] = eval^M[t\{\omega \leftarrow \text{right-son}(t_{|\omega})\}]$ iff $C(t,\omega) = 2$

in the $\uparrow$-model determined by C. Of course, an $\uparrow$-choice must satisfy certain properties reflecting the ACIN constraints concerning choice.[1]

An $\uparrow$-choice C being given, we define the class MOD^C of the $\uparrow$-models as those models which satisfy the two previous conditions (c_1) and (c_2). We define analogously the class $fgMOD^C$ of the finitely generated models of MOD^C.

We define $\sim_C$ as being the smallest equivalence relation on $T_{S,\Sigma}$ that satisfies :

(c'_1) $t \sim_C t\{\omega \leftarrow \text{left-son } (t_{|\omega})\}$ if $C(t,\omega) = 1$

(c'_2) $t \sim_C t\{\omega \leftarrow \text{right-son } (t_{|\omega})\}$ if $C(t,\omega) = 2$

Then, the set-theoretical quotient of $T_{S,\Sigma}$ by $\sim_C$ is likely to be in MOD^C, and to be the *most general* object of MOD^C. Formally, we define an *$\uparrow$-morphism* from an $\uparrow$-model M into an $\uparrow$-model M' as a family φ of applications : $\varphi_s : M_s \to M'_s$ such that : $eval^{M'}(t) = \varphi_s(eval^M(t))$ (for any $t \in (T_{S,\Sigma})_s$).

We also define an *$\uparrow$-equivalence* as an equivalence relation on $T_{S,\Sigma}$ that respects the ACIN properties of $\uparrow$ and δ.

---

[1] For the sake of readability, this is precisely done in Appendix 1.

Theorem 1.1

- MOD and fgMOD, with ↑-morphisms, are non-empty categories. They admit no initial object.[2]
- For a given ↑-choice C, MOD^C and $fgMOD^C$, with ↑-morphisms, are categories that both admit $T_{S,\Sigma}/\sim_C$ as initial object.
- MOD $= \cup_{C:\uparrow\text{-choice}} MOD^C$
  fgMOD $= \cup_{C:\uparrow\text{-choice}} fgMOD^C$
- Every ↑-model of $fgMOD^C$ is isomorphic to the set-theoretical quotient of $T_{S,\Sigma}/\sim_C$ by an ↑-equivalence.

1.2. The Equations

Definition

Let M and N be two terms of $T_{S,\Sigma}(X)$ of the same sort. We shall say that an ↑-model A *satisfies the equation* M = N , which we write : $A \models M = N$, if and only if : for any context K, and for any ground substitution σ, $eval^A(K[M\sigma]) = eval^A(K[N\sigma])$.

Note : if $A \models M = N$, then in particular $A \models M\uparrow P = N\uparrow P$, for any $P \in T_{S,\Sigma}(X)$.

For a set E of equations, MOD_E (resp. $fgMOD_E$) is the class of the models (resp. the finitely generated models) that satisfy every equation of E. For a given ↑-choice C, MOD_E^C and $fgMOD_E^C$ are defined analogously. The couple formed by a ↑-signature and a set of equations will often be called a nondeterministic specification.
Let $\sim_E$ be the equivalence relation on $T_{S,\Sigma}$ generated by the pairs $K[M\sigma] \sim_E K[N\sigma]$. We define $\sim_{E\cup C}$ as the smallest equivalence relation on $T_{S,\Sigma}$ containing $\sim_E$ and $\sim_C$, and respecting the ACIN properties of ↑ and δ. Then :

Theorem 1.2

- MOD_E and $fgMOD_E$ are non-empty categories.
- For a given ↑-choice C, MOD_E^C and $fgMOD_E^C$ are categories that admit $T_{S,\Sigma}/\sim_{E\cup C}$ as initial object.
- MOD $= \cup_{C:\uparrow\text{-choice}} MOD_E^C$
  fgMOD $= \cup_{C:\uparrow\text{-choice}} fgMOD_E^C$
- Every ↑-model of $fgMOD_C^E$ is ↑-isomorphic to the set-theoretical quotient of $T_{S,\Sigma}/\sim_{E\cup C}$ by a ↑-equivalence.
- Let M and N be in $T_{S,\Sigma}$. Then :
  $fgMOD_E^C \models M = N$ iff $T_{S,\Sigma}/\sim_{E\cup C} \models M = N$
  $MOD_E^C \models M = N$ iff $M \sim_{E\cup C} N$ [Birkhoff Theorem. Weak form]

The last point is similar to the classical Birkhoff theorem for "deterministic" algebras. However, it considered to be a *weak form* because it assumes a given ↑-choice, and it applies on *ground* terms. We subsequently provide a stronger form, that is choice independent and applies to terms with variables. This last point is proven in Appendix 2, the other points being quite straightforward.

Until now, we defined the *algebraic framework* that is needed for the expression of the nondeterminism. In the next section, we deal with the *operational aspect* of the question, via the introduction of a generalized notion of rewrite rules.

---

[2]Except in the pathological cases, where there is at most one term per sort with an ↑ symbol.

2. REWRITING WITH NONDETERMINISM

2.1. In order to take into account the properties of $\uparrow$, we say that two terms M and M' in $T_{S,\Sigma}(X)$ are ACIN-equal, and we write $M \overset{ACIN}{=} M'$ if they are equal modulo the *A*ssociativity, the *C*ommutativity, the *I*dempotence of $\uparrow$, and the *N*eutrality of δ for $\uparrow$. Note that ACIN unification and pattern-matching are decidable (cf. [Kirchner 84]). Classically, we say that a term is *flattened* when adjacent occurrences of the symbol $\uparrow$ are merged, identical terms under an $\uparrow$ symbol are assimilated, and all the occurrences of δ are removed (cf. [BP 85]). For instance, a flattened form of the term $a\uparrow((b\uparrow c)\uparrow(\delta\uparrow((c\uparrow b)\uparrow H(e\uparrow f,g))))$ is $\uparrow\{a,b,c,H(\uparrow\{e,f\},g)\}$ (where braces "{}" recall that the arguments of $\uparrow$ are to be considered as a set).

Definition

An $\uparrow$ *term rewriting system* (or, shortly, an $\uparrow$–TRS) R is a finite set of couples (λ,ρ) of terms in $T_{S,\Sigma}(X)$.

- We define the associated *rule-reduction relation* as being the smallest binary predicate $\twoheadrightarrow_R^r$ on $T_{S,\Sigma}(X)$ such that :

$$\frac{M \overset{ACIN}{=} K[\lambda\sigma]\ ,\ N \overset{ACIN}{=} K[\rho\sigma]\ ,\ (\lambda,\rho)\in R}{M \twoheadrightarrow_R^r N}$$

- We define the associated *choice-reduction relation* as being the smallest binary predicate $\twoheadrightarrow_R^c$ on $T_{S,\Sigma}(X)$ such that :

$$K[M\uparrow N] \twoheadrightarrow_R^c K[M], \quad K[M\uparrow N] \twoheadrightarrow_R^c K[N],$$
$$K[M\uparrow\delta] \twoheadrightarrow_R^c K[M], \quad K[\delta\uparrow M] \twoheadrightarrow_R^c K[M].$$

- We define the *reduction relation* associated to R as being $\twoheadrightarrow_R = \twoheadrightarrow_R^r \cup \twoheadrightarrow_R^c$.
- Its $\uparrow$-*closure* is the relation $\rightarrowtail\!\!\!\!\twoheadrightarrow_R = [\ (\twoheadrightarrow_R^r \cup \twoheadrightarrow_R^{r-1}) \cup \twoheadrightarrow_R^c\ ]^*$

M and N stand for any terms in $T_{S,\Sigma}(X)$. K is any context and σ any substitution.

Note :

- For the $\twoheadrightarrow_R$ relation, one step of reduction is often called a *transition*. Transitions associated with $\twoheadrightarrow_R^r$ and $\twoheadrightarrow_R^c$ are respectively called *rule*-transitions and *choice*-transitions. The G's are the left-hand sides and the D's are the right-hand sides of the rules of R.
- *From an operational point of view,* $\twoheadrightarrow_R$ is strictly equivalent to the classical term rewriting system generated by

$$R \cup \{\ x\uparrow y\rightarrow x\ ,\ x\uparrow y\rightarrow y\ ,\ x\uparrow\delta\rightarrow x\ ,\ \delta\uparrow x\rightarrow x\ \}.$$

Nevertheless, these two interpretations of R are viewed, algebraically, in a completely different manner. This is particularly visible in how the closure is computed, as in the following.

- Intuitively, $\rightarrowtail\!\!\!\!\twoheadrightarrow_R$ plays the role of :
  - reflexive, transitive closure w.r.t. the $\uparrow$ operator,
  - reflexive, transitive and symmetric closure w.r.t. the rules of R viewed as equations.

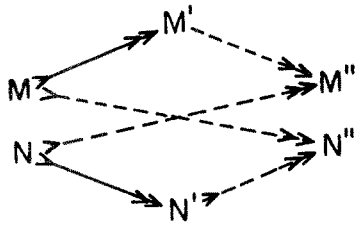
This is how $\rightarrowtail\!\!\!\!\twoheadrightarrow_R$ will capture both the equational aspect of the equations of R, and the asymmetric (irreversible) aspect of a choice-transition. This leads to the following result :

Theorem 2.1 [Birkhoff Theorem. Strong form :]

$$MOD_R \models M = N$$
iff

for all M' such that $M \rightarrowtail\!\!\!\!\twoheadrightarrow_R M'$, there exists M'' such that $N \rightarrowtail\!\!\!\!\twoheadrightarrow M''$ and $M' \rightarrowtail\!\!\!\!\twoheadrightarrow M$

for all N' such that $N \rightarrow\!\!\gg_R N'$, there exists N'' such that $M \rightarrow\!\!\gg N''$ and $N' \rightarrow\!\!\gg N'$
Schematically :



[fig. 1]

(Throughout the paper, in this kind of diagrams, full arrows stand for arrows existing by hypothesis, while dashed arrows stand for conclusions of the property being pictured).

Proof :
The proof of that theorem is long and tedious, and of no particular interest for the purposes of this paper. It can be found in [Kaplan 85].
Note that Theorem 2.1 extends the classical Birkhoff theorem for strictly deterministic specifications. As opposed to Theorem 1.2 (weak form), it does not rely on a specific $\uparrow$-choice.

Example 2.1
In this specification, $!\langle int \rangle \mapsto \langle proc \rangle$ stands for : emit the integer $\langle int \rangle$ and then behave like the process $\langle proc \rangle$. We algebraically specify an operator P such that :

$$P(0) = \delta \text{ and } P(n+1) = !0 \mapsto \delta \uparrow !2 \mapsto \delta \uparrow \ldots \uparrow !2n \mapsto \delta$$

So, P(n+1) can emit, nondeterministically, any even integer between 0 and 2n. The operator 'π' is an auxiliary function, such that $\pi(n) = 2n$.

$$\begin{array}{lcl} P(0) & \rightarrow\!\!\gg & \delta \\ P(s(n)) & \rightarrow\!\!\gg & P(n) \uparrow ![\pi(n)] \mapsto \delta \\ \pi(0) & \rightarrow\!\!\gg & 0 \\ \pi(s(n)) & \rightarrow\!\!\gg & s(s(\pi(n))) \end{array}$$

We will prove properties about that specification in section 4.

2.2. Termination

For a given nondeterministic term rewriting system R, we say that $\rightarrow\!\!\gg_R$ is *finitely terminating* if there exists no infinite chain :

$$t_1 \rightarrow\!\!\gg_R t_2 \rightarrow\!\!\gg_R \cdots \rightarrow\!\!\gg_R t_n \rightarrow\!\!\gg_R \cdots$$

Now, as in section 1.2, $\rightarrow\!\!\gg_R$ is operationally equivalent to the *classical* term rewriting system R^{class} generated by :

$$R \cup \{ x \uparrow y \rightarrow x \ , \ x \uparrow y \rightarrow y \ , \ x \uparrow \delta \rightarrow x \ , \ \delta \uparrow x \rightarrow x \ \}.$$

For classical systems, powerful criteria have been developed in order to check the finite termination property (cf. e.g. [Dersh 79],[DF 85]). Some works considered termination of systems with ACIN-like properties ([BK 84a], [BP 85]). These criteria, based on partial orders in $T_{S,\Sigma}(X)$ are applicable to our case. Moreover, it is easy to check that the $\{ x \uparrow y \rightarrow x \ , \ x \uparrow y \rightarrow y \ , \ x \uparrow \delta \rightarrow x \ , \ \delta \uparrow x \rightarrow x \ \}$ classical rewrite rules do not introduce specific non-termination. Thus :

Theorem 2.2
$\rightarrow\!\!\gg_R$ is finitely terminating if and only if the *classical* term rewriting system R^{class} is finitely terminating.

The ↑–TRS appearing in this paper have been shown to be terminating, using simplification orderings on the corresponding classical term rewriting systems.

2.3. The Normal Form Function

We recall that a term t is a *normal form*, or *irreducible* when there exists no t' such that $t \twoheadrightarrow t'$. Otherwise, a term u is a normal form of a term t if $t \twoheadrightarrow^* u$ and u is a normal form. Let $\twoheadrightarrow$ be the reduction relation associated to a *finitely terminating* ↑–TRS R. As in the classical case, the set of the normal forms of a term $t \in T_{S,\Sigma}(X)$, that we denote {NF(t)}, is finite. One thus has the following version of the Birkhoff Theorem 2.1, that provides an effective criterion to check equality in MOD_R :

Theorem 2.3

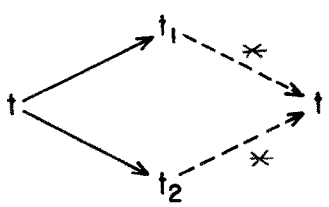
$$MOD_R \models M = N \quad \text{iff} \quad \{NF(M)\} = \{NF(N)\}$$

The proof easily follows from Theorem 2.1, under the previous stronger hypothesis of finite termination.

3. ↑ CONFLUENCE

3.1. Confluence is known to be a crucial property for *classical* term rewriting systems. Given a binary predicate →, this predicate is *confluent* iff for any term $t \in T_{S,\Sigma}(X)$,[3]

if there exist t_1 and t_2 such that $t \rightarrow^* t_1, t_2$,
then there exists t' such that $t_1, t_2 \rightarrow^* t'$.



[fig. 2]

Of course, in our framework, a relation $\twoheadrightarrow_R$ will never be confluent because of reductions such as $t_1 \uparrow t_2 \twoheadrightarrow_R t_1$ and $t_1 \uparrow t_2 \twoheadrightarrow_R t_2$. We are thus led to define a specific notion of confluence.

Before doing this, we shall add a restriction to the ↑–TRS that we consider. We will suppose that they are *left-↑-free*, which means that the left-hand sides of the rules are in $T_{S,\Sigma}^{\Theta}(X)$ (i.e. contain no ↑ symbol). The reasons for that are threefold.

- Firstly, the statement of our definitions, and of our subsequent results, will be much more readable : we will not have to consider ACIN pattern matching or ACIN unification. Also, the corresponding algorithms will be far less time consuming.
- Secondly, all the examples that we were led to consider are naturally left-↑-free. This is mainly why all the forthcoming results, that we obtained first in the general case, will be presented under the assumption of left-↑-freedom.

---

[3] $\rightarrow^*$ is the reflexive and transitive closure of →. a→b,c stands for a→b and a→c. Similarly for a,b→c.

• Finally, a non-left-↑-free rule such as f(a↑b)→≫c expresses some context-sensitive constraint about the choice operator. This would be contrary to our conception of what the ↑ operator is. In particular, it should be able to choose between its arguments in a context-free manner.

Theorem and Definition 3.1

Given a finitely terminating ↑–TRS R, the following properties are equivalent :

(i) $\forall t \in T_{S,\Sigma}^{\Theta}(X)$

for any t_1 and t_2 in $T_{S,\Sigma}(X)$ such that $t \to\!\!\gg_R t_1, t_2$,

for any t'_1 such that $t_1 \to\!\!\gg_R^* t'_1$,

there exists t_3 such that $t'_1, t_2 \to\!\!\gg_R^* t_3$

for any t'_2 such that $t_2 \to\!\!\gg_R^* t'_2$,

there exists t_4 such that $t'_2, t_1 \to\!\!\gg_R^* t_4$

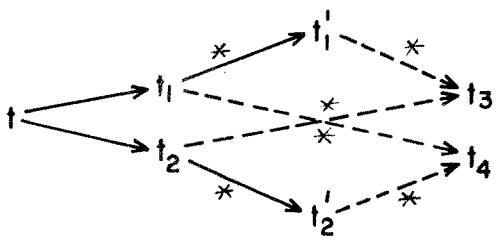
(cf. [fig. 3])

(ii) $\forall t \in T_{S,\Sigma}^{\Theta}(X)$, if there exist t_1 and t_2 in $T_{S,\Sigma}(X)$ such that $t \to\!\!\gg_R t_1, t_2$, then t_1 and t_2 have *the same set of normal forms* for $\to\!\!\gg$.

(iii) $\forall t \in T_{S,\Sigma}^{\Theta}(X)$, if there exist t_1 and t_2 in $T_{S,\Sigma}(X)$ such that $t \to\!\!\gg_R t_1, t_2$, then $R \models t_1 = t_2$.

In that case, $\to\!\!\gg_R$ is said to be ↑-*confluent*

Schematically :



[fig. 3]

Proof : Implications (i) ⟹ (ii) and (iii) ⟺ (i) are just a matter of definition. Equivalence (ii) ⟺ (iii) is a consequence of theorem 2.3. ▪

Note :

• It follows from Definition 3.1 that for a ↑-confluent system, the following property is satisfied :

$\forall t \in T_{S,\Sigma}^{\Theta}(X)$, $\forall t_1, t_2 \in T_{S,\Sigma}(X)$,

$t \to\!\!\gg t_1$ and $t \to\!\!\gg^* t_3$ ⟹ $\exists t_3 \in T_{S,\Sigma}(X)$ s.t. $t_1 \to\!\!\gg^* t_3$ and $t_2 \to\!\!\gg^* t_3$ (†).

This *does not* imply the confluence, in the classical sense, of $\to\!\!\gg$[4] as it would have been the case if (†) had been quantified over $t \in T_{S,\Sigma}(X)$ (cf. [Huet 77]). Here, the hypothesis that t does not contain any '↑' symbol is crucial.

• It should be noticed that, compared with the classical deterministic case, ↑-confluence is a kind of *local* confluence : consideration of global confluence would be irrelevant in our framework.

• Intuitively, the ↑-confluence condition means that all the non-confluence in a specification comes from choice-transitions (at the occurrence of ↑ symbols), and

[4] For instance, the simple system a → b↑c satisfies (†) but is not confluent. Nevertheless, it is ↑-confluent.

not from rule-transitions. We illustrate this notion on several examples.

Example 3.1
A trivial example of a specification that is not ↑-confluent is the following :

$$f(a,b) \twoheadrightarrow a \ , \ f(a,b) \twoheadrightarrow b \ .$$

However, the following system is equivalent to the previous one on $T_{S,\Sigma}^{\ominus}$, that is ↑-confluent :

$$f(a,b) \twoheadrightarrow a \uparrow b$$

This shows again the interest of a specific operator ↑, that conveys all the nondeterminism allowed in a (↑-)confluent system.

Example 3.2
In this example, we specify a toy concurrent language, along the lines of the ACP framework (cf. [BK 84a,b]). The operators are the following :

δ	-- The deadlock
!<integer> ⊢ <process>	-- Emit <integer> and behave like <process>
<process>‖<process>	-- Interleaved execution of processes
<process>⫡<process>	-- (Technical)
<bool>:<process>‖<bool>:<process>	-- Guarded choice

The equations are :

p ‖ p'	↠	p ⫡ p' ↑ p' ⫡ p
(!i ⊢ p) ⫡ p'	↠	!i ⊢ (p ‖ p')
δ ⫡ p'	↠	δ
True:p ‖ True:p'	↠	p ↑ p'
True:p ‖ False:p'	↠	p
False:p ‖ True:p'	↠	p'
False:p ‖ False:p'	↠	δ

Criteria developed in the next section allow us to show that this specification is ↑-confluent. We shall also prove various properties about it.

3.2. Knuth-Bendix Theorem for ↑-Convergence

In this section, we consider under which conditions a system is ↑-convergent. As in the classical case, we need to consider the notion of *critical pair*.

Definition 3.2
Consider two rules $G_1 \twoheadrightarrow D_1$ and $G_2 \twoheadrightarrow D_2$ such that, for a *non-variable* occurrence ω of G_1, $G_{1|\omega}$ and G_2 are unifiable. Let σ be their most general unifier. The pair :

$$< G_1[\omega \leftarrow D_2]\sigma \ , \ D_1\sigma >$$

is called a critical pair.

Notes :
• We use the fact that the system under consideration is left-↑-free. Thus, neither G_1 nor G_2 contain the ↑ symbol. For non left-↑-free systems, we would have considered rules such that $G_{1|\omega}$ and G_2 are ACIN-unifiable.
• We also need to suppose that the rules under consideration are *left-linear*. This means that a variable may appear at most once in the left-hand side of a rule. We now have the following result :

Theorem 3.3 [Knuth-Bendix theorem for ↑-TRS]

Given a finitely terminating, left-↑-free and left-linear ↑–TRS R, the following properties are equivalent :
(i) $\twoheadrightarrow_R$ is ↑-confluent
(ii) For every critical pair <t,t'> of R, $\{NF_R(t)\} = \{NF_R(t')\}$

This is the central result of this paper. In particular, it enables us to decide whether a finitely terminating ↑–TRS is confluent or not, via the following method :
- compute all the critical pairs of the system (which are in finite number);
- compute and compare their normal forms.

Note : The implication (ii) $\Longrightarrow$ (i) of theorem 3.3 is not true if the left-linearity hypothesis is not satisfied. A counter-example is given in Appendix 3.
Proof : (cf. Appendix 4).

3.3. Completion procedure

As in the classical case, it is interesting to consider a *completion procedure*, that transforms a set of equations E into a left-↑-free, left-linear and ↑-confluent ↑–TRS R. We now concentrate on the left-↑-freedom hypothesis. In order to maintain the left-↑-freedom assumption, we shall suppose that the equations of E are already oriented, so that the left-hand sides are ↑-free. Then left-↑-freedom has to be incrementally ensured.

Example 3.3
We give here a naive example, in order to illustrate the previous point. Realistic examples are provided in the next chapter. Let E be the set :

P(x) = a(x) ↑ b(x) ↑ c(x)
Q(P(x)) = d(x) ↑ e(x) ↑ f(x) ↑ g(x)
Q(a(x)) = d(x)

The system (when left-to-right oriented) has the unique following critical pair :
< Q(a(x)↑b(x)↑c(x)) , d(x)↑e(x)↑f(x)↑g(x) >. The completion procedure should generate new equations, in order to ensure that these two terms have the same set of normal forms. These normal forms currently are :
for Q(a(x)↑b(x)↑c(x)) : { d(x), Q(b(x)), Q(c(x)) }
for d(x)↑e(x)↑f(x)↑g(x)} : { d(x), e(x), f(x), g(x) }
Notice that the term d(x) appears in the previous two sets. In order to ensure the ↑-confluence, it is thus sufficient to generate the equation Q(b(x))↑Q(c(x)) = e(x)↑f(x)↑g(x) (which is sound). However, this is not acceptable here since we are restricted to left-↑-free rules. Our experiment also shows that such rules tend to cause infinite loops in the completion procedure. For instance, it is often the case that for a critical pair <a↑b,c↑d>, just adding the rules (a$\twoheadrightarrow$c,b$\twoheadrightarrow$d) or (a$\twoheadrightarrow$d,b$\twoheadrightarrow$d) (when they are sound) allows the completion procedure to stop, whereas adding the rule a↑b$\twoheadrightarrow$c↑d would lead to non-termination. However, it is not possible to systematically decide which group of rules is sound and may be added.
Notice now that there is one situation in which it is possible to add a left-↑-rule, namely when one of the sets of normal forms is reduced, after elimination of the common elements, to a singleton. We shall give the completion procedure in this case. In spite of this restriction, it appeared -- surprisingly enough -- that all the examples drawn from "natural" theories that we considered correspond to that situation.

Completion procedure
Given
- a set E of equations M=N ,
- a simplification ordering > such that for each equation M=N, M>N and M is ↑-free and left-linear.

0. $E_0 \leftarrow E$; $R_0 \leftarrow \phi$; $i \leftarrow 0$.

1. If $E_i = \phi$, then STOP-WITH-SUCCESS.

2. $E_{aux} \leftarrow \phi$
For each equation M=N in E, do
$NF_{M-N} \leftarrow \{NF(M)\} - \{NF(N)\}$;
$NF_{N-M} \leftarrow \{NF(N)\} - \{NF(M)\}$;
if $NF_{M-N} = NF_{N-M}$ then SKIP ;
if $|NF_{M-N}|>1$ and $|NF_{N-M}|>1$ then STOP-WITH-FAILURE
else -- Suppose that NF_{M-N} is the singleton {m}
$E_{aux} \leftarrow E_{aux} + (m = \uparrow\{\mu, \mu \in NF_{N-M}\})$
$E_i \leftarrow E_{aux}$

3. Choose an equation M=N in E_i.

4. If M and N are uncomparable via >, then
if $E_i \neq \phi$, then choose another rule in E_i
else STOP-WITH-FAILURE.

5. -- We suppose that for instance M > N, and that M is ↑-free and
-- linear.
$R_{modif} \leftarrow \{(\lambda \twoheadrightarrow \rho) \in R_i \mid \lambda$ or ρ contains an instance of M$\}$
$R_{i+1} \leftarrow R_i - R_{modif} + \{M \twoheadrightarrow N\}$;
Y $\leftarrow$ the set of the critical pairs of R_{i+1};
$E_{i+1} \leftarrow E_i + R_{modif} - \{M=N\} + Y$;
$i \leftarrow i+1$; GOTO 1.

With respect to the classical Knuth-Bendix completion procedure, only step **2.** is new. In the classical case, the corresponding "do" loop is reduced to the following :
$\overline{M} \leftarrow NF(M)$; $\overline{N} \leftarrow NF(N)$ -- Both $\overline{M}$ and $\overline{N}$ are singletons.
$E_{aux} \leftarrow E_{aux} + (\overline{M} = \overline{N})$
Examples of the application of the completion procedure are given in the next section.

Theorem 3.4
The completion procedure takes a system E of equations as input, and produces, when it stops, a system RR of left-↑-free, left-linear rules that is finitely terminating, ↑-confluent, and such that :
$$\forall t,t' \in T_{S,\Sigma}(X), \quad E \models t=t' \text{ iff } \{NF_{RR}(t)\} = \{NF_{RR}(t)\}$$

Proof : The correctness of the procedure derives from the following facts :

(1) At the end of every step **5.**, one has $\equiv_E = \equiv_{E_i \cup R_i}$. Thus, for the final step, $\equiv_E = \equiv_{RR}$. This leads to the conclusion of the theorem, using Theorem 2.3.

(2) Finite termination, left-↑-freedom and left-linearity are incrementally ensured via the ordering >.

(3) At STOP-WITH-SUCCESS, R_i = RR is ↑-confluent, because it admits no more critical pairs. ∎

Of course, this completion procedure may be given a much more efficient form, as in [Huet80]. The optimizations proposed for the classical completion procedure, based on adequate marking and fair rule consideration, carry over to our framework.

4. THEOREM PROVING IN NONDETERMINISTIC THEORIES

4.1. In this section, we address the problem of proving inductive properties in nondeterministic theories presented by an $\uparrow$–TRS R. A property M = M' is an *inductive* theorem if and only if it is valid in the class fgMOD_R of the finitely generated models of the theory. Inductive properties are usually proved in two different ways :

- explicitly using induction techniques, as in [BM 79], [Bidoit 81],
- via so-called *inductionless induction* techniques, that involve Knuth-Bendix algorithms, and in theories that are specified in appropriate ways (cf. references below).

In this section, we rely on the second approach. It has been carefully investigated in the case of classical equational rewrite rules ([Goguen 80], [Musser 80], [HH 80], [Fribourg 84], ...), rewriting modulo equations ([JK 84]), conditional equations ([Kaplan 84], [RZ 85]), etc. Those investigations led to the design of large systems for theorem proving (cf. [HH 80], [Lescanne 83]). We are going to extend such methods to our formalism of nondeterministic systems.

From now on, we suppose that the theory is defined by a left-$\uparrow$-free finitely terminating $\uparrow$–TRS R on the signature (S,Σ) , such that :

(i) Σ may be partioned in Constr $\cup$ Der $\cup$ $\{(\uparrow)_s,(\delta)_s\}$, where operators in Constr are called *constructors* and operators in Der are called *derived operators*. We add the technical condition that every $(T_{S,Constr})^s$ [the set of terms of sort s formed with constructors only] must be non-empty, for any s$\in$S.

(ii) For any t,t' $\in T_{S,Constr}$, then $t \equiv_R t'$ iff t = t'.

(iii) For any f$\in$Der, for any vector $\vec{t}$ of terms of $T_{S,Constr}$ of suitable arity, for any $\tau\in\{NF_R[f(\vec{t})]\}$, then τ is in $T_{S,Constr}$.

These conditions are commonly imposed in the classical framework. We have to see how they may be realized in the nondeterministic case.

- Condition (i) is just a methodological choice.
- Condition (ii) is sometimes formulated by saying that R generates no equations between constructors. This condition is generaly ensured by forbidding, in R, rules of the form M$\rightarrow\gg$M', where symbols occurring at the root of M and M' are both in Constr.
- Condition (iii) is far more difficult to ensure. It states that functions corresponding to the derived operators are well-defined with respect to constructors. One can also say that their definition is *sufficiently complete* (w.r.t. Constr). The methodologies that have been developed in the classical case (cf. [Bidoit 81], [Fribourg 84],...) are still applicable in our framework. They were sufficient to develop the examples in this paper, and their principles inspire the examples developed by the ACP group.

We return now to the question of the proofs in theories specified as previously. As in [HH 80], the completion procedure of section 3 is modified into an *inductive completion procedure*, in the following way : the following step **4'.** is introduced between step **4.** and step **5.** of the completion procedure. Note that step **4'.** realizes an exhaustive case analysis on the equation M = M' chosen (and oriented in step **4.**).

4'. -- M > M' and M is ↑-free.

if M = c($M_1, \cdots, M_n$) and M' = c($M'_1, \cdots, M'_n$) with c∈Constr,
then $E_i \leftarrow E_i + (M_i = M'_i)_{i\in[1..n]}$; GOTO 2.
else

if M = c($M_1, \cdots, M_n$) and M' = c'($M'_1, \cdots, M'_n$) with c,c'∈Constr and c≠c',
then STOP-WITH-DISPROOF
else

if M = c($M_1, \cdots, M_n$) with c∈Constr and M' is a variable
(or symmetrically in M and M'),
then STOP-WITH-DISPROOF
else

if M = c($M_1, \cdots, M_n$) with c∈Constr and M' = ↑{$t_1, \ldots, t_p$}, -- M' is flattened
then
- if the root symbol of at least one of the t_i is a constructor c' ≠ c,
  then STOP-WITH-DISPROOF else
- if every t_j, for j∈[1..n], is under the form $t_j = c(t_{j,1}, \cdots, t_{j,n})$,
  then $E_i \leftarrow E_i \cup (M_j = \uparrow\{t_{1,j} \cdots t_{p,j}\})_{j\in[1..n]}$; GOTO 2
  else
- if the root symbol of at least one of the t_i (but not all of them) is the constructor c,
  then STOP-WITH-FAILURE else GOTO 5

else GOTO 5

Now, one has the following result, stating the correctness of that procedure.

Theorem 4.1

Let M = M' be a property to be verified. We suppose that the inductive completion procedure is applied to R ∪ { M = M' }, and eventually stops.

- **If** the procedure stops with "STOP-WITH-DISPROOF", **then** M = M' does not hold in $fgMOD_R$.
- **If** the procedure stops with "STOP-WITH-SUCCESS", **then** M = M' is a theorem of $fgMOD_R$.

When the procedure stops with STOP-WITH-FAILURE, no conclusion can be drawn about the veracity of M=N in the inductive theory.

The proof of theorem 4.1 is similar to the proof for the classical case (cf. e.g. [HH 80]). Actually, it should be emphasized that this principle is rather independent of the formalism (classical rewriting, rewriting modulo equations, conditional rewriting) that is understated, if a correct Knuth-Bendix completion procedure may be provided for the framework under consideration. Note that there is a new failure case, when an equation such as c(t) = c(t') ↑ d is chosen (c being a constructor and the root symbol of d being a derived operator). In that case, nothing can be said about the theorem to be proved. We are now going to show how this method applies to different examples.

4.2. Proof Examples

Example 4.1

We consider again the specification of example 2.1, for which we wish to prove that, for any n, the signal that P(n) nondeterministically emits is an *even* number. To that effect, we first enrich the specification with the predicates 'Q' and 'even', with the

following rules :

$Q(\delta)$	$\twoheadrightarrow$	True
$Q(!x \mapsto p)$	$\twoheadrightarrow$	even(x)
even(0)	$\twoheadrightarrow$	True
even(s(0))	$\twoheadrightarrow$	False
even(s(s(x)))	$\twoheadrightarrow$	even(x)

The predicate 'even' checks whether an integer is even, and Q checks if the signal emitted by a process is even. It should be noted that, for a process p = a↑b (in flattened form), the set of the normal forms of Q(p) is exactly the union of the normal forms of Q(a) and Q(b). This allows the fact that Q has a "sufficiently complete" definition (in the sense of this chapter) to be verified.
Now, one wants to prove the inductive theorem : Q(P(n)) = True , $(\forall n)$. It is thus added to the whole system, that currently consists of :

P(0)	$\twoheadrightarrow$	δ	(1)
P(s(n))	$\twoheadrightarrow$	$P(n) \uparrow ![\pi(n)]\mapsto\delta$	(2)
$\pi(0)$	$\twoheadrightarrow$	0	(3)
$\pi(s(n))$	$\twoheadrightarrow$	$s(s(\pi(n)))$	(4)
$Q(\delta)$	$\twoheadrightarrow$	True	(5)
$Q(!x \mapsto p)$	$\twoheadrightarrow$	even(x)	(6)
even(0)	$\twoheadrightarrow$	T	(7)
even(s(0))	$\twoheadrightarrow$	F	(8)
even(s(s(0))	$\twoheadrightarrow$	even(x)	(9)
Q(P(n))	$\twoheadrightarrow$	True	(10)

Note that rules (1) to (9) define a specification that verifies the hypotheses of 4.1 : the constructors are T and F for the booleans, and <int>↦<proc> for the processes. Other operators may easily be checked to be derived operators. Note also that rules (1) to (9) admit no critical pair. We now follow the application of the completion procedure to the whole system (1) to (10).

- There is a critical pair between (1) and (10) :
  <Q(δ) , True >, that normalizes into <True,True>.
- There is a critical pair between (2) and (10) :
  <$Q(P(n) \uparrow ![\pi(n)]\mapsto\delta)$, True >. The normal forms of $Q(P(n) \uparrow ![\pi(n)]\mapsto\delta)$ are {True , $even(\pi(n))$}. Thus, the completion procedure generates the new rule :

$$even(\pi(n)) \twoheadrightarrow True \quad (11)$$

- There is a critical pair between (3) and (11) :
  <even(0) , True>, that is eliminated, as before, by normalization.
- There is a critical pair between (4) and (11) :
  $<even(s(s(\pi(n)))$, True > . Now, $even(s(s(\pi(n))) \twoheadrightarrow_{9-11} True$. Thus, that critical pair is eliminated, and the inductive completion procedure stops with success, having generated no new equation between constructors. This concludes the proof. ∎

Example 4.3 (Example 2.4 continued)
For the specification given in example 2.4, the constructors are True and False for the booleans, 0 and succ for the integers, and !<int>↦<proc> for the processes (plus $\uparrow_{processes}$ and $\delta_{processes}$). The previous proof method allowed us to prove the following properties :

$(p \lfloor\!\lfloor p') \lfloor\!\lfloor p''$	=	$p \lfloor\!\lfloor (p' \parallel p'')$
$p \parallel (p' \parallel p'')$	=	$(p \parallel p') \parallel p''$
$p \| (p' \| p'')$	=	$(p \| p') \| p''$

The proof scripts, being much longer than for example 4.2, are not given here. This is also true for the next examples. They are provided, though, to illustrate the interest of our methods.

Example 4.3
We slightly modify the previous example in order to include synchronous communication, still in the spirit of the ACP works. The construction 'c !! i ↦ p' stands for sending message i (an integer) along channel c (conventionally represented by an integer), and then behaves like the process p. 'c ?? i ↦ p' is defined analogously. Synchronous communication (defined by "|") may occur only via two channels the sum of the labels of which is less than or equal to three (recall that this is intended to be a toy concurrent language !). The construct !<int>↦<proc> exists as before (communication to the outside world). The specification itself is :

x + 0	$\twoheadrightarrow$	x
x + s(y)	$\twoheadrightarrow$	s(x + y)
$p \parallel p'$	$\twoheadrightarrow$	$(p \lfloor\!\lfloor p') \uparrow (p' \lfloor\!\lfloor p) \uparrow (p \mid p')$
$(!i \mapsto p) \lfloor\!\lfloor p'$	$\twoheadrightarrow$	$!i \mapsto (p \parallel p')$
$\delta \lfloor\!\lfloor p'$	$\twoheadrightarrow$	δ
$(c!!i \mapsto p) \lfloor\!\lfloor p'$	$\twoheadrightarrow$	δ
$(c??i \mapsto p) \lfloor\!\lfloor p'$	$\twoheadrightarrow$	δ
$(c!!i \mapsto p) \mid (c'!!i' \mapsto p)$	$\twoheadrightarrow$	δ
$(c??i \mapsto p) \mid (c'??i' \mapsto p)$	$\twoheadrightarrow$	δ
$(c??i \mapsto p) \mid (c'!!i' \mapsto p)$	$\twoheadrightarrow$	Aux(c + c', p, p')
$(c??i \mapsto p) \mid (c'??i' \mapsto p)$	$\twoheadrightarrow$	Aux(c + c', p, p')
Aux(0,p,p')	$\twoheadrightarrow$	$!0 \mapsto p \parallel p'$
Aux(s(0),p,p')	$\twoheadrightarrow$	$!s(0) \mapsto p \parallel p'$
Aux(s(s(0)),p,p')	$\twoheadrightarrow$	$!s(s(0)) \mapsto p \parallel p'$
Aux(s(s(s(0))),p,p')	$\twoheadrightarrow$	$!s(s(s(0))) \mapsto p \parallel p'$
Aux(s(s(s(s(x)))),p,p')	$\twoheadrightarrow$	δ
True:p ‖ True:p'	$\twoheadrightarrow$	$p \uparrow p'$
True:p ‖ False:p'	$\twoheadrightarrow$	p
False:p ‖ True:p'	$\twoheadrightarrow$	p'
False:p ‖ False:p'	$\twoheadrightarrow$	δ

Note that the equations defining synchronous communication might have been given more simply by :

$$x \leq s(s(s(0))) = True \quad \Longrightarrow \quad Aux(x,p,p') \quad \twoheadrightarrow \quad !x\vdash\!\!\rightarrow p \| p'$$
$$x \leq s(s(s(0))) = False \quad \Longrightarrow \quad Aux(x,p,p') \quad \twoheadrightarrow \quad \delta$$

These are conditional rewrite rules, which we do not know, yet, how to include in our framework.
The constructors of the processes in the specification are !<int>⊢→<proc>, <int>!!<int>⊢→<proc>, <int>??<int>⊢→<proc>. We are able to prove the following properties :

$$\begin{array}{lcl} p \mid (p' \mid p'') & = & (p \mid p') \mid p'' \\ p \| (p' \| p'') & = & (p \| p') \| p'' \\ b{:}p \parallel (b'{:}p' \parallel b''{:}p'') & = & (b{:}p \parallel b'{:}p') \parallel b''{:}p'' \\ b{:}(p \uparrow q) \parallel b'{:}p' & = & (b{:}p \parallel b'{:}p') \uparrow (b{:}q \parallel b'{:}p') \\ b{:}p \parallel b'{:}(p' \uparrow q') & = & (b{:}p \parallel b'{:}p') \uparrow (b{:}p \parallel b'{:}q') \end{array}$$

An other interesting property to establish would be :

$$c!!x\vdash\!\!\rightarrow p \parallel c??y\vdash\!\!\rightarrow p' \;=\; p \parallel p'[y\backslash x]$$

However, this is not formalizable via a finite number of equations, because of the substitution operation p'[y\x]. This may be *simulated* in the following way : one explicitly specifies the sort of the arithmetic expressions, and one redefines adequately the sort of processes. Then, the substitution operation becomes an operation of the specification :

$$\_[\_\backslash\_] \;:\; \text{process} \times \text{arith-expr} \times \text{integer} \rightarrow \text{process},$$

that is finitely axiomatizable. This is done in details in [Kaplan 86]. We were then able to prove the previous theorem via our methods.

5. CONCLUSIONS

In this paper, we have considered the feasibility of term rewriting systems taking into account nondeterminism. In particular, we have managed to maintain the coherence of the following "waterfall" scheme :

algebraic behaviour : terms, equations, models
$\Longrightarrow$ simulation of equations via rewriting. Birkhoff theorem
$\Longrightarrow$ termination, confluence. Knuth-Bendix Theorem
$\Longrightarrow$ completion procedure. Inductive completion
$\Longrightarrow$ **proof in well structured theories**

The main characteristic of this work is that it leads to *automatic* theorem proving methods, extending principles now widely used in the deterministic framework, and implemented in large systems, to nondeterministic specifications. We have developed several examples that illustrate the applicability of the method to non-trivial specifications.

The questions that need further attention are essentially the following :
- how can the procedures we have designed be efficiently implemented ? We have already shown the soundness of several optimizations : avoiding ACIN pattern-matching and unification through the assumption of left-↑-freedom, etc. However, computations are still time and space consuming ;
- we also have to determine the precise range of application of our methods. We feel that we have shown their applicability to nondeterministic computations. Our long-term project is the consideration of concurrent computations, in the spirit of our last examples. The proof methods that we have designed apply to properties slightly different from those usually considered in this field. In particular, we seem to be able to deal with almost arbitrarily complex *data* structures. For instance, we could specify and prove facts about a complicated kind of communication channel in example 4.3. However, we cannot take into account non-terminating computations. A possible approach would be to consider only their finite approximations. For

example, in order to specify a process such that $P \equiv !x \mapsto P$, we can synthesize the process $\overline{P}$ such that $\overline{P}(0) = \delta$ and $\overline{P}(s(n)) = !x \mapsto \overline{P}(n)$, and prove properties about $\overline{P}(n)$, for every n. But this will not allow to deal with properties such as fairness or partial correctness. The work of [PF 85] might be relevant in this respect.

Acknowledgments : We wish to thank C.Choppy whose work [CJ 85] stimulated our personal research in the field. It is also a pleasure to thank a reviewer of a previous version of this work for very careful reading, and suggestions.
It has been partially supported by the Esprit METEOR project. We also thank the Laboratoires de Marcoussis for their hardware support.

BIBLIOGRAPHY

[ADJ 78] J.A.Goguen, J.W. Thatcher, E.G.Wagner,
An initial algebra approach to the specification, correctness and implementation of abstract data types, in Current Trends in Programming Methodology, Prentice-Hall N-J. (1978)

[Apt 84] K.Apt,
Ten years of Hoare's logic : a survey. Part II : nondeterminism, *in* Theoretical Computer Science 28 (1984).

[BK 84a] J.Bergstra, J.Klop,
Algebra of communicating processes with abstraction, CWI Report CS-R8403, Amsterdam (1984).

[BK 84b] J.Bergstra, J.Klop,
Algebra of communicating processes. Part II, CWI Report, Amsterdam (1984)

[Bidoit 81] M. Bidoit,
Une methode de presentation de types abstraits : applications, These de 3e cycle, Orsay (1981)

[BM 79] R. Boyer, J.S. Moore,
A computational logic, Academic Press, (1979).

[Boudol 84] G.Boudol,
An "asynchronous" calculus MEIJE, *in* NATO Summer School, La-Colle-sur-Loup, France (1984).

[BP 85] L.Bachmai, D.Plaisted,
Associative path orderings, Proc. 1^{st} RTA Conf., Dijon (1985).

[Brookes 83] S.D.Brookes,
On the relationship between CCS and CSP, Proc. 10^{th} ICALP Conf., L.N.C.S., Springer Verlag (1983).

[Broy 84] M.Broy,
On the Herbrand Kleene universe for nondeterministic computations, Proc. MFCS 84 Conf., L.N.C.S., Springer Verlag (1984).

[BW 81] M.Broy, M.Wirsing,
On the algebraic specification of nondeterministic programming languages, Proc. CAAP-81 Conf., L.N.C.S. N.112 (1981).

[CJ 85] C.Choppy, C.Johnen,
Petrireve : proving Petri net properties with rewriting systems, Proc. 1^{st} RTA Conf., Dijon (1985).

[Dersh 79] N. Dershowitz,
Orderings for term rewriting systems, Proc. 20^{th} Symposium on Foundation of Computer Science, pp.123-131 (1979).

[DF 85] D.Detlefs, R.Forgaard,
A procedure for automatically proving the termination of a set of rewrite rules, Proc. 1^{st} RTA Conf., Dijon (1985).

[Fribourg 84] L.Fribourg,
A narrowing procedure for theories with constructors, Proc. CADE Conf., Napa (1984).

[GM 81] J.Goguen, J.Meseguer,
Completeness of many-sorted equational logic, SIGPLAN Notices (1981).

[Goguen 80] J.Goguen,
How to prove algebraic inductive hypotheses without induction, 5^{th} CAD, Les Arcs- France (1980).

[Hennessy 1982] M.Hennessy,
Powerdomains and nondeterministic recursive definitions, Symposium on Programming, L.N.C.S N.137 (1982)

[Huet 77] G. Huet,
Confluent reductions : abstract properties and applications to term rewriting systems, Proc. 18^{th} FOCS Conf., Providence (1978).

[HH 80] G. Huet, J-M. Hullot,
Proofs by induction in equational theories with constructors 21^{st} FOCS (1980).

[HO 80] G.Huet, D.C. Oppen,
Equations and rewrite rules : a survey , Formal languages : Perspective and open problems, R. Book

Ed., Academic Press (1980).
[Hoare 78] C.A.R.Hoare,
Communicating sequential processes, CACM 21 666-677 (1978).
[JK 84] J.P.Jouannaud, C.Kirchner,
Completion of a set of rules modulo a set of equations, Proc. of the 11^{th} POPL Conference (1984).
[Kaplan 84] S. Kaplan,
Unification, narrowing with fair conditional term rewriting systems, Internal L.R.I. Report (to appear).
[Kaplan 86] S. Kaplan,
A Birkhoff theorem for nondeterministic specifications, Weizmann Institute Internal Note, Rehovot (to appear- 1986)
[KB 70] D.E. Knuth, P.B. Bendix,
Simple word problems in universal algebra, Computational problems in abstract algebra, J.Leech Ed., Pergammon Press (1970).
[Kirchner 84] C.Kirchner,
A new equational unification method : a generalization of Martelli-Montanari's algorithm, Proc. CADE Conf. (1984).
[Lescanne 83] P.Lescanne,
Computer experiment with the REVE term rewriting systems generator, Proc. of the 10^{th} POPL Conference (1983).
[Milner 80] R.Milner,
A calculus of communicating systems, L.N.C.S. N.92, Springer Verlag (1980).
[Musser 80] D.L.Musser,
On proving inductive properties of abstract data types, Proc. 7^{th} Conf., Las Vegas (1980)
[Nivat 80] M.Nivat,
Nondeterministic programs : an algebraic overview, Proc. of the IFIP 80 Conf., North-Holland Publishing Company (1980).
[PF 85] S.Porat, N.Francez,
Fairness in term rewriting systems, Proc. 1^{st} RTA Conf., Dijon (1985).
[Poigné 81] A.Poigné,
On effective computations of nondeterministic schemes, Proc. of the CAAP-81 Conference, L.N.C.S. N.112 (1981)
[RZ 85] J.L.Remy, H.Zhang,
Contextual rewriting, Proc. 1^{st} RTA Conf., Dijon (1985)

Appendix 1 : Definition of a ↑-choice

Let S,Σ be a ↑-signature. A ↑-*choice* C is an application :

$$C : T_{S,\Sigma} \times N^* \to \{ 1 , 2 \}$$

such that :

• let ω be in N^*, and $t \in T_{S,\Sigma}$. If ω is not an occurrence of ↑ in $T_{S,\Sigma}$, then $C(t,\omega)$ is undefined.

• Let t,t',t'' be in $T_{S,\Sigma}$ and let K be a context. A ↑-choice must perform the same choices in τ=K[t↑(t'↑t'')] and in τ'=K[(t↑t')↑t'']. Let ω stand for the occurrence of the generic variable in the context K. Then, for instance, the occurrence of the t' is $\omega.2.1$ in τ, and is $\omega.1.2$ in τ'. Thus, the following constraints must be fulfilled :

- If $C(\tau,\omega) = 1$ -- t is chosen in τ
  then $C(\tau',\omega) = 1$ and $C(\tau',\omega.1) = 1$
- If $C(\tau,\omega) = 2$ and $C(\tau,\omega.2) = 1$ -- t' is chosen in τ
  then $C(\tau',\omega) = 1$ and $C(\tau',\omega.1) = 2$
- If $C(\tau,\omega) = 2$ and $C(\tau,\omega.2) = 2$ -- t'' is chosen in τ
  then $C(\tau',\omega) = 2$

• One must have also, with the same conventions :

- $C(K[x↑y],\omega) = \neg C(K[y↑x],\omega)$,$\forall x,y \in T_{S,\Sigma}$[5]
- $C(K[x↑\delta],\omega) = 1$,$\forall x \in T_{S,\Sigma}$
- $C(K[\delta↑x],\omega) = 2$,$\forall x \in T_{S,\Sigma}$

It is then easy to check that $T_{S,\Sigma}/\sim_C$ is a ↑-model, just taking as $eval^{T_{S,\Sigma}}$ the application associating to each term of $T_{S,\Sigma}$ its class for $\sim_C$.

Appendix 2 : Proof of Theorem 1.2

We just need to prove the last point, namely that :

$$MOD_E^C \models M = N \text{ iff } M \sim_{E \cup C} N, \quad \text{for } M,N \in T_{S,\Sigma}$$

Here, M and N are ground terms. We can also suppose that E contains only equations between ground terms, possibly in infinite number. This is because a system of equations with variables is equivalent to the set of its ground instances.

To establish the property, we shall use the classical Birkhoff Theorem in classical equational theories. We introduce the new sorts 'int' of the integers, and 'int-string' of the integer strings with the signature :

$0 : \to$ int, s : int → int,

$\varepsilon : \to$ int-string, u : int → int-string, . : int-string int-string → int-string

and the equations :

$\varepsilon.s = s$, $s.\varepsilon = s$, $s.(s'.s'') = (s.s').s''$ $(\forall s,s',s'' \in$ int-string$)$.

It is easy to check that the initial model, in the classical sense, of the previous specification has domains isomorphic to the integers and the integer strings.

We introduce a new operator < , > : s int-string → new-s , 'new-s' being a new sort. We shall also use the new constant 'Error' of 'new-s'. Now, a ↑-choice being given, we define :

(a) the following (infinite) set of equations $\bar{C}$ (with the conventions of Appen-

[5] with ¬1 = 2 and ¬2 = 1

dix 1) :

- If $C(t,\omega)$ is undefined, then $\langle t,\overline{\omega}\rangle$ = error
  (where $\overline{\omega}$ stands for the element of int-string representing ω).
- If $C(K[t\uparrow t'],\omega) = 1$, then $\langle K[t\uparrow t'],\overline{\omega}\rangle = \langle K[t],\overline{\omega}.s(0)\rangle$
- If $C(K[t\uparrow t'],\omega) = 2$, then $\langle K[t\uparrow t'],\overline{\omega}\rangle = \langle K[t],\overline{\omega}.s(s(0))\rangle$

and

(b) the (infinite) set of equations $\overline{E}$:
for every $t,t'\in T_{S,\Sigma}$ such that $MOD_{\mathcal{C}}^{E}\models t=t'$, for every $\omega\in$int-string, then $\langle t,\omega\rangle = \langle t',\omega\rangle$

We now build, for every $\uparrow$-model A of $MOD_{E}^{\mathcal{C}}$, a model (in the classical sense) $\varphi(A)$ of $\overline{C}\cup\overline{E}$ in the following way : for any $m\in A$,

- if $m=eval^{M}(\mu)$, for a $\mu\in T_{S,\Sigma}$, then m is mapped into $\varphi(m) = class_{\overline{C}\cup\overline{E}}(\langle\mu,0\rangle)$,
- else m is mapped into $class_{\overline{C}\cup\overline{E}}(error)$.

One can check, in the previous definition, that the choice of μ does not matter. Then,

- Let $A\in MOD_{E}^{\mathcal{C}}$. It is clear that $A\models M = N$ if and only if the associated model $\varphi(A)$ validates, in the classical sense, the equation $\langle\varphi(M),0\rangle = \langle\varphi(N),0\rangle$.
- By the classical Birkhoff theorem, the previous condition is true for every A if and only if $\langle\varphi(M),0\rangle \equiv_{\overline{C}\cup\overline{E}} \langle\varphi(N),0\rangle$ (here, $\equiv_{\overline{C}\cup\overline{E}}$ is the classical congruence generated by $\overline{C}\cup\overline{E}$).
- The last property is again easily showed to be equivalent to $M \sim_{C\cup R} N$.

This concludes the proof of the theorem. ∎

Appendix 3 : The Hypothesis of Left-Linearity in Theorem 3.3

Consider the following system R : -- a, b, c and k are constants. x is a variable.

$$\begin{array}{lll} g(x,x) & \twoheadrightarrow & x \\ k & \twoheadrightarrow & a\uparrow b \\ g(a,b) & \twoheadrightarrow & c \\ g(b,a) & \twoheadrightarrow & c \end{array}$$

R is a finitely terminating, left-$\uparrow$-free, but *not* left-linear system. Condition (ii) of Theorem 3.3 is trivially verified, since R has no critical pairs. However, R is not $\uparrow$-confluent, since $g(k,k)\in T_{S,\Sigma}^{\Theta}(X)$ is such that $g(k,k)\twoheadrightarrow k$, $g(a\uparrow b,k)$, and $\{NF(k)\}=\{a,b\}$ while $\{NF(g(a\uparrow b,k))\}=\{a,b,c\}$.

∎

Appendix 4 : Proof of Theorem 3.3

(i) $\Longrightarrow$ (ii)
This comes directly from Theorem 3.1.
(ii) $\Longrightarrow$ (i)
Let t,t' be in $T_{S,\Sigma}(X)$. We say that $t\twoheadrightarrow^{r}_{strict} t'$ if and only if there exists a rule $\lambda\twoheadrightarrow\rho$, a substitution $\sigma : X\rightarrow T_{S,\Sigma}^{\Theta}(X)$ and an occurrence ω in t such that $t_{|\omega} = \lambda\sigma$ and $t' = t[\omega\leftarrow\rho\sigma]$. We have the following lemma :

Lemma :
Under the assumption of theorem 3.3 and hypothesis (ii),

$$\text{if } t\twoheadrightarrow^{r}_{strict} t', \text{ then } \{NF(t)\} = \{NF(t')\}$$

Intuitively, the lemma states that along a rule transition in which no '$\uparrow$' symbol

disappears, the set of the normal forms remains unchanged. Note that t is allowed to contain '↑' symbols, but not in the subtree on which the strict reduction operates. Now, implication (ii) $\Longrightarrow$ (i) comes directly from the lemma : if $t \in T_{S,\Sigma}^{\Theta}(X)$ and $t \twoheadrightarrow t_1, t_2$, then $\{NF(t_1)\} = \{NF(t_2)\} = \{NF(t)\}$. ▪

Proof of the lemma :

Proof is by *reductio ad absurdum*. Let P stand for the following predicate on $T_{S,\Sigma}(X)$:

$$P(z) \quad \text{iff} \quad \forall t' \in T_{S,\Sigma}(X),\ z \twoheadrightarrow^{r}_{strict} t' \implies \{NF(z)\} = \{NF(t')\}.$$

If the lemma is false, then there exists at least one $t \in T_{S,\Sigma}(X)$ such that :

$$\neg P(t) \quad \text{and} \quad (\forall t' \in T_{S,\Sigma}(X),\ t \twoheadrightarrow^{+} t' \implies P(t')\ ).$$

($\twoheadrightarrow^{+}$ stands for the irreflexive and transitive closure of $\twoheadrightarrow$).

Else, one could construct an infinite sequence of terms $(t_i)_{i \in N}$ such that $\neg P(t_i)$ and $t_i \twoheadrightarrow t_{i+1}$. This contradicts the finite termination hypothesis on $\twoheadrightarrow$.

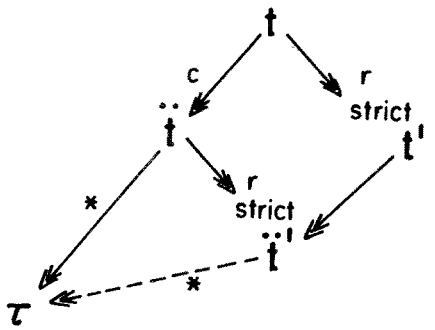
Now, for such a t, we shall prove that, actually, P(t) is true, which is contradictory. We suppose now that $t \twoheadrightarrow^{r}_{strict} t'$ and we prove that $\{NF(t)\} = \{NF(t')\}$.

α) It is clear that $\{NF(t)\} \supseteq \{NF(t')\}$, since $t \twoheadrightarrow t'$.

β) In order to prove the converse, let τ be in $\{NF(t)\}$. If $t = \tau$, then we obtain a contradiction since t is a normal form and $t \twoheadrightarrow t'$. Thus, there exists a term $\ddot{t} \in T_{S,\Sigma}(X)$ such that $t \twoheadrightarrow \ddot{t} \twoheadrightarrow^{*} \tau$. The proof proceeds by case analysis on the transition $t \twoheadrightarrow \ddot{t}$.

β.1) If $t \twoheadrightarrow^{c} \ddot{t}$, (i.e. the transition is a choice-transition), there exist an occurrence $\ddot{\omega}$ in t, and two terms M,N in $T_{S,\Sigma}(X)$ such that $t_{|\ddot{\omega}} = M \uparrow N$, and $\ddot{t} = t[\ddot{\omega} \leftarrow M]$ or $\ddot{t} = t[\ddot{\omega} \leftarrow N]$. Now, for two occurrences v and v' of a term ϑ, we say that $v \propto v'$ if v' is on the path from v unto the root of ϑ, and that $v \perp v'$ if neither $v \propto v'$ nor $v' \propto v$. We consider three subcases.

- **If** $\omega \perp \ddot{\omega}$, then suppose that, for instance, $\ddot{t} = t[\ddot{\omega} \leftarrow M]$. Let $\ddot{t}'$ stand for $t[\omega \leftarrow \rho\sigma, \ddot{\omega} \leftarrow M]$ (which makes sense since $\omega \perp \ddot{\omega}$). Now, $t \twoheadrightarrow \ddot{t}$, thus $P(\ddot{t})$ holds. Since $\ddot{t} \twoheadrightarrow^{r}_{strict} \ddot{t}'$, $\ddot{t}$ and $\ddot{t}'$ have the same set of normal forms, and thus $\tau \in \{NF(\ddot{t}')\}$. But, because $t' \twoheadrightarrow \ddot{t}'$, we also have : $\tau \in \{NF(t')\}$.
  Pictorially :



- **If** $\ddot{\omega} \propto \omega$, then $t_{|\omega} = \lambda\sigma$ contains a symbol '↑', which contradicts the strictness of the transition $t \twoheadrightarrow^{r}_{strict} t'$.
- **If** $\omega \propto \ddot{\omega}$, then we suppose, for instance, that $\ddot{t} = t[\ddot{\omega} \leftarrow M]$ ("M is chosen"). Then, there are two possibilities :
  - ω is an occurrence of M (more rigourously, the prefix $\omega_{|\ddot{\omega}}$ of ω is an occurrence of M). Then, let $\ddot{t}' = \ddot{t}[\omega \leftarrow \rho\sigma]$. Then $\ddot{t} \twoheadrightarrow^{r}_{strict} \ddot{t}'$, and we conclude as before.
  - ω is an occurrence of N. Then $t' \twoheadrightarrow^{c} \ddot{t}$ and, thus, $\tau \in \{NF(t')\}$.

This concludes the case *β*.1.

β.2) If $t \gg\!\!\to^{r} t'$, then there exist an occurrence ω in t, a rule $\lambda \gg\!\!\to \rho$ and a substitution $\sigma: X \to T_{S,\Sigma}(X)$, such that $t_{|\omega} = \lambda\sigma$ and $t = t[\omega \leftarrow \rho\sigma]$. As before, we proceed with the analysis of the respective locations of ω and ω in t.

- **If** $\omega \perp \omega$, then let $t' = t[\omega \leftarrow \rho\sigma, \omega \leftarrow \rho\sigma]$ (which makes sense since $\omega \perp \omega$). As before, $t \gg\!\!\to^{r}_{strict} t'$, thus $\{NF(t)\} = \{NF(t')\}$. So $\tau \in \{NF(t')\}$, which implies $\tau \in \{NF(t')\}$.
- **If** $\omega \propto \omega$, there are two subcases to consider.
  - ω corresponds to a *non-variable* occurrence of $\lambda\sigma$. Let μ be the smallest unifier of $\lambda_{|\omega}$ and λ. There exists a substitution ν such that $\mu\nu = \sigma \cup \sigma$ (we can assume that the variables appearing in σ and σ are distinct). Then $\langle \rho\mu , \lambda\mu[\omega \leftarrow \rho\mu] \rangle$ is a critical pair. By (ii), the two elements have the same set of normal forms, and it is also the case for $\rho\sigma = t'_{|\omega}$ and $\lambda\sigma[\omega \leftarrow \rho\sigma] = t_{|\omega}$. This leads easily to : $\tau \in \{NF(t')\}$.
  - ω corresponds to a *variable* occurrence x of $\lambda\sigma$. Let $Var(\lambda) = \{x, y_1, \cdots, y_n\}$. t still matches the rule $\lambda \gg\!\!\to \rho$ at occurrence ω via the substitution : $\sigma' = \{x \to \rho\sigma, y_1 \to y_1\sigma, \cdots, y_n \to y_n\sigma\}$. Let $t' = t[\omega \leftarrow \rho\sigma']$. Then : $t \gg\!\!\to^{r}_{strict} t'$. Here, the fact that R si left-linear is important in order to apply P to t ; otherwise, we would just have : $t (\gg\!\!\to^{r}_{strict})^{*} t'$. Thus, as before, $\tau \in \{NF(t')\}$, and $\tau \in \{NF(t')\}$.
  - **If** $\omega \propto \omega$, then we conclude as in the previous subcase.

This concludes the proof of theorem 3.3.

■