Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Strong Bisimilarity is decidable for LOTOS context-free processes

A. Fantechi, S. Gnesi, R. Sacchelli

Nota Interna B4-31
Ottobre 1993

Strong Bisimilarity is decidable for LOTOS context free processes

A. Fantechi♣♠, S. Gnesi♣, R Sacchelli♦

♣I.E.I. - C.N.R, via S Maria 46, I-56126 Pisa, Italy

♠Dip. di Ingegneria dell'Informazione, Univ. di Pisa, via Diotisalvi 2, I-56126 Italy

♦Dip. di Informatica, Univ. di Pisa, corso Italia 40, I-56125 Italy

Abstract

In this paper we show how an existing technique to decide strong bisimulation on context free processes can be adapted to work on a subset of LOTOS describing this class of non-finite state processes. This decision procedure, originally defined for the Basic Process Algebra is based on the construction of semantic tableaux following the structure of a LOTOS term.

1 Introduction

If we consider a LOTOS behaviour expression [BoB87] as the set of its computations, different classes of processes can be identified. The maximal expressive power that a LOTOS subset can exhibit is that of Turing Machines. The presence of values, conditional expressions and recursion in LOTOS is enough to reach the expressive power of Turing Machines. However, even without resorting to the use of values (that is, even when using the so called "Basic LOTOS"), it is possible to express Turing Machines; in fact, different expressivity results can be given for different LOTOS operators [FGM90].

The expressive power of LOTOS becomes an important issue when considering the problem of automatic verification of specification properties. In fact several methodologies and most automatic tool support for LOTOS can only be used if the specifications considered describe finite-state machines [BoC89,FGMR92,MaV89]. For example, an equivalence verifier can only manipulate specifications in this class; an attempt to verify the equivalence of two non finite-state processes may not terminate with an answer. More powerful verification techniques have recently been exploited to extend the verification problem to non-finite state systems (see [BaB90,BBK88,BS92,DeI91,HüS91]).

Among these verification techniques we wish to recall the bisimulation decision algorithms on context free processes of Basic Process Algebra (BPA) [BeK84] due to Baeten, Bergstra and

Klop and to Hüttel and Stirling [BBK88,HüS91]. In this paper we adapt the decision procedure by Hüttel and Stirling to a subset of LOTOS, that describes context free processes. This decision procedure is based on the construction of semantic tableaux following the structure of a LOTOS term and it allow us to check the strong bisimilarity between processes of this subset.

The organization of the paper is as follows: in Section 2 we present the subset of LOTOS, which coincides with context free languages; the algorithm to decide the strong bisimulation on this subset is described and discussed in Section 3; an example of its application is given in Section 4; the Appendix contains the correctness proof of the given algorithm.

2 Context free LOTOS processes

In this section we present the subset of LOTOS which can generate all context free processes [FGM90]. This subset, called from now on $LOTOS_{CF}$, employs a finite alphabet of observable actions G. Actions occur at gates; since no value communication is present, we can identify the observable actions with the gate at which they occur. The alphabet Act of actions also includes an unobservable (or internal) action i and a "successful termination" action δ used to give semantics to the enabling operator. Let μbe an element of Act and a an element of Act$\cup\{\delta\}$, the syntax and the operational semantics of $LOTOS_{CF}$ operators are given in Table 1, whereP is a process identifier, BE_i a behaviour expression and g_i/h_i the syntactical substitution of parameters. In the operational semantics of the exit operator the null process stop appears, which is not included in the syntax of $LOTOS_{CF}$.

OPERATOR	SYNTAX	SEMANTICS
successful termination	exit	exit -δ->stop
action prefix	μ;BE	μ;BE-μ->BE
choice	$BE_1[]BE_2$	BE_1-a->BE_1' implies $BE_1[]BE_2$ -a->BE_1' BE_2-a->BE_2' implies $BE_1[]BE_2$ -a->BE_2'
process instantiation	$P[g_1,...,g_n]$	$BE_P[g1/h1,...gn/hn]$ -μ->BE' implies P[g1,...gn] -μ->BE'
enabling	$BE_1>>BE_2$	BE_1 -δ->BE_1' implies $BE_1>>BE_2$ -i->BE_2 BE_1 -μ->BE_1' implies $BE_1>>BE_2$ -μ->BE_1'>>BE_2

Table 1

When enabling is used within recursion, this subset is as expressive as context free languages. In fact in [FGM90] it was proved that each process in $LOTOS_{CF}$ corresponds to a context free grammar and viceversa.

Example 2.1

The following context free $LOTOS_{CF}$ process definition models the behaviour of a system managing nested windows; the process Opened_window can be seen as the manager of an open window, which accepts both its closure and the creation of a new nested window. At the closure of a window the control is returned to its "parent" window.

```
process Nested_windows [openclick, draw_win, closeclick, cancel_win]: noexit :=
    openclick; draw_win; Opened_window [openclick, draw_win, closeclick, cancel_win]
            >> Nested_windows [openclick, draw_win, closeclick, cancel_win]
where
    process Opened_window [openclick, draw_win, closeclick, cancel_win, ]: exit :=
     (closeclick; cancel_win; exit
     []
     (openclick; draw_win; Opened_window [openclick, draw_win, closeclick, cancel_win]
            >> Opened_window [openclick, draw_win, closeclick, cancel_win]
     ))
    endproc
endproc
```

On LOTOS terms various equivalence relations can be defined. Among them we recall the notions of maximal trace equivalence and of strong bisimulation equivalence [Mil80].

Definition 1

$\sigma \in Act^*$ is a *maximal trace* of a process P, if there exist actions $a_1, a_2, ..., a_n \in Act$ and processes $P_1, P_2, ..., P_n$ such that $\sigma = a_1.a_2.....a_n$, and $P \text{—}a_1\rightarrow P_1 \text{—}a_2\rightarrow P_2 \text{ ....—}a_n\text{-}\rightarrow P_n$ and there exist no transition leading from P_n .

Two processes P and Q are called maximal trace equivalence if the related sets of maximal traces are equal, and we write $P \sim Q$.

Definition 2

A relation $\Re$ between processes is a *strong bisimulation* if whenever $P \Re Q$ then for each $a \in Act \cup \{\delta\}$ we have:

1. $P \text{—}a\rightarrow P' \Rightarrow \exists\ Q'$: $Q \text{—}a\rightarrow Q'$ and $P' \Re Q'$
2. $Q \text{—}a\rightarrow Q' \Rightarrow \exists\ P'$: $P \text{—}a\rightarrow P'$ and $P' \Re Q'$.

Two processes P and Q are said to be strongly bisimilar, written P<=>Q, if there is a strong bisimulation $\Re$ such that P$\Re$Q.

On $LOTOS_{CF}$ the problem to decide the maximal trace equivalence between two processes is unsolvable, since it corresponds to the decision of language equivalence that is unsolvable for context free languages [HoU69]. In any case, the undecidability result for maximal trace equivalence does not extend in $LOTOS_{CF}$ to bisimulation equivalence between processes; in fact bisimulation equivalence is decidable on processes expressing context free languages [BBK88,HüS91,CHS92]. The proof of this statement is based on the regularity properties of the computation trees, starting from the system of recursive equations defining such processes. $LOTOS_{CF}$ processes can be expressed as a system of recursive equations [FGM90], and therefore for these classes we can apply the bisimulation decision algorithm of [BBK88] or the more efficient algorithm, based on the construction of semantic tableaux, presented by Hüttel and Stirling [HüS91].

In the following section we will present this algorithm. For its definition some extensions to the original algorithm have been necessary; also some conditions on $LOTOS_{CF}$ processes have been introduced.

3 The Tableau Decision Method

We consider the class of *guarded normed recursive* $LOTOS_{CF}$ processes, which are expressed as a set of recursive equations, each having at the left hand side the process identifier and at the right hand side the behaviour expression. A process is therefore defined as $(X_i|B)$, where X_i is the recursion variable and $B=\{X_i=T(X_1,\ldots,X_n)\colon i:1\ldots n\}$ is a set of recursive equations. $T(X_1,\ldots,X_n)$ are terms defined in (Act,Var,>>,[],;), and Var is a set of variables. In this formal system the recursion is *guarded* , i.e. all variables appearing within $T(X_1,\ldots,X_n)$ in B must be preceded by an atomic action.

A normal form, similar to the Greibach Normal Form of context free grammar, can be given for recursive $LOTOS_{CF}$ processes:

Definition 3
A $(X_i|B)$ system is in Greibach Normal Form if the recursive equations in B are defined as: $X_i=[]_k\ a_k;\alpha_k$, where $a_k \in$ Act and α_k is a sequence of process variables.
If *lth* $(\alpha_k)\leq 2$ then we say that $(X_i|B)$ is in 3-GNF, where the function *lth* (α) gives the number of symbols of the string α.

Proposition 1

Every recursive system in Greibach Normal Form can be written as a 3-GNF system, by introducing new process identifiers.

Definition 4

For each variable $X_i \in Var$, we define the *norm* of X_i, $|X_i|$, as the length of the minimum w, i.e.

$$|X_i| = \min\{ lth(w) \mid X_i \text{---}w\rightarrow stop \}.$$

We assume that for all X_i there exists a sequence of actions 'w' such that $X_i \text{---}w\rightarrow stop$

We can easily compute the norm of a process, starting from the expression which defines it, using inductively these rules:

$|a;p| = 1+|p| \quad \forall p \in P \ \forall a \in Act$

$|p[]q| = \min(|p|,|q|) \quad \forall p,q \in P$

$|p>>q| = |p|+|q| \quad \forall p,q \in P$

$|exit| = 1$

3.1 The algorithm

The bisimulation decision algorithm on $LOTOS_{CF}$ terms applies for each transition the definition of strong bisimulation equivalence itself: for each action performed by a process it checks if the other process is able to perform the same action, and then it checks the bisimilarity of the processes resulting from the execution of this action. The algorithm goes on in this way until an unsuccessful or successful termination is reached.

We give now a more formal description of the algorithm. Let $(X|B_X)$ and $(Y|B_Y)$ be two $LOTOS_{CF}$ systems of recursive equations, normed and 3-GNF. To test whether $X <=> Y$, we build a tableau using a goal-directed technique as defined in [HüS91].

The rules are built around equations of the form $E>>\alpha = F>>\beta$, where α and β can be also empty sequences of variables. In the case α or β are empty we consider equations having only the left side of the corresponding enabling operator. Each rule has this form:

$$\frac{E>>\alpha = F>>\beta}{E_1>>\alpha_1 = F_1>>\beta_1 \cdots E_n>>\alpha_n = F_n>>\beta_n}$$

The premise of the rule represents the goal (i.e. $E>>\alpha <=> F>>\beta$) while the consequents are the subgoals. Each of these rules is backwards sound, insofaras the consequents are true (i.e. their equations relate strongly bisimilar processes) then so is the antecedent.

To determine whether $X>>\alpha <=> Y>>\beta$ a tableau for the equation $X>>\alpha = Y>>\beta$ is created using the rules in Table 2. The rules relative to the recursion, choice, and action prefix were

directly obtained by those given in [HüS91] for the corresponding operators of BPA, as well as the rules for the generation of new subtableaux (i.e. SUBL and SUBR). As far as concerns the enabling and exit operators, we have had to define new rules since the operational semantics of the sequential composition of $LOTOS_{CF}$ is different from BPA sequential composition.

Rules within subtableaux:

REC $\quad \dfrac{X>>\alpha = Y>>\beta}{E>>\alpha = F>>\beta}$ where $X =_{def} E$, $Y =_{def} F$

CHOICE $\quad \dfrac{([]^{n}_{i=1}\ a_i;\alpha_i)>>\alpha = ([]^{m}_{j=1} b_j;\beta_j)>>\beta}{\forall i\in I\ \exists\, j\in J : a_i;\alpha_i>>\alpha = b_j;\beta_j>>\beta}$ with $I=\{1...n\}$, $J=\{1...m\}$

PREFIX $\quad \dfrac{a;\alpha = a;\beta}{\alpha = \beta}$ where $a\in Act$

PREFIX $\quad \dfrac{a;exit = a;exit}{- = -}$ where $a\in Act$

EXIT $\quad \dfrac{exit>>\alpha = exit>>\beta}{\alpha = \beta}$

SUBLEX $\quad \dfrac{\alpha = exit>>\beta}{\gamma = \beta}$ where $\alpha = i;\gamma$ or $\alpha = X$, and $X=_{def} i;\gamma$

SUBREX $\quad \dfrac{exit>>\alpha = \beta}{\alpha = \gamma}$ where $\beta = i;\gamma$ or $\beta = X$, and $X=_{def} i;\gamma$

Rules for new subtableaux:

SUBL $\quad \dfrac{\alpha_i>>\alpha = \beta_i>>\beta}{\alpha_i>>\gamma = \beta_i}$ where $\alpha = \gamma>>\beta$ is the residual.

SUBR $\quad \dfrac{\alpha_i>>\alpha = \beta_i>>\beta}{\alpha_i = \beta_i>>\gamma}$ where $\beta = \gamma>>\alpha$ is the residual.

Table 2

3.1 How to Construct Subtableaux

A tableau for the equation $X>>\alpha=Y>>\beta$ can be seen as a finite proof tree whose root is labelled $X>>\alpha=Y>>\beta$, and where the equations which label the immediate successors of a node are determined by an application of one of the rules in Table 2.

A proof tree consists of a number of subtableaux; we construct a subtableau using the rules described in Table 2 as shown in Figure 1, where it is assumed that $E = []^n_{k=1}\ a_k;\alpha_k$, $F = []^n_{k=1}\ a_k;\beta_k$, $\alpha_i = exit>>\alpha_j$ and $\beta_i = exit>>\beta_j$.

$$\frac{X>>\alpha=Y>>\beta}{E>>\alpha=F>>\beta}\ REC$$

$$\frac{a_1;\alpha_1 = a_1;\beta_1 \quad \ldots\ldots \quad a_i;\alpha_i = a_i;\beta_i \quad \ldots\ldots \quad a_n;\alpha_n = a_n;\beta_n}{\alpha_1 = \beta_1 \quad \ldots\ldots \quad \alpha_j = \beta_j \quad \ldots\ldots \quad \alpha_n = \beta_n}\ CHOICE,\ PREFIX,\ EXIT$$

Figure 1

Figure 1 represents a *basic step* for $X>>\alpha=Y>>\beta$, it involves an application of the REC rule followed by an application of the CHOICE, PREFIX, EXIT, SUBLEX, SUBREX rules in sequence. The basic step aims to capture the set of matching single transition steps in the operational semantics of both sides of the equation,such as $X>>\alpha\text{-}a_i\text{->}\alpha_i$ and $Y>>\beta\text{-}a_i\text{->}\beta_i$.

These rules are only applied to nodes that are not terminal. Terminal nodes are either successful or unsuccessful. A tableau node is an unsuccessful terminal if it has one of these forms:

1. $\alpha=\beta$ and $|\alpha|\neq|\beta|$;
2. $a;\alpha=b;\alpha$ and $a\neq b$.

Clearly, such nodes cannot relate bisimilar processes.

A tableau node is a successful terminal if it has one of these forms:

1. $\alpha=\beta$ and there is another node above it in either the same or another subtableau also labelled $\alpha=\beta$;
2. $\alpha=\alpha$.

With reference to Figure 1, an important observation is that $lth\ (\alpha_i)\leq lth\ (X>>\alpha)$, since X can only introduce a sequence of at most two variables via the REC rule; similarly $lth\ (\beta_i)\leq lth\ (Y>>\beta)$.

Assume that $k=\min(|X|,|Y|)$, a subtableau for $X>>\alpha=Y>>\beta$ iterates the construction of basic steps until reaching a depth of k steps.

If $|X|\leq|Y|$, each leaf of a subtableau for $X>>\alpha=Y>>\beta$ is either labelled $\alpha=\gamma>>\beta$, which is called *residual* of the subtableau, since X is eliminated, or $\alpha_i>>\alpha=\beta_i>>\beta$ where α_i and β_i need

not be empty. On the other hand if |Y|<|X|, similar remarks would apply, except that the residual is then $\gamma >> \alpha = \beta$.

After constructing a subtableau, we have to identify a residual node and we apply one of the SUB rules defined in Table 2 to every leaf of the subtableau which is neither the residual nor a terminal.

If each consequent of the SUB rule and the residual aren't terminals we build a new subtableau, as described above, starting from the expressions which characterize them.

The proof goes on as a succesion of subtableaux; if at any point in the construction of the subtableau we reach an unsuccessful terminal, we terminate and consider the resulting tableau as being unsuccessful. On the other hand, a successful tableau is a finite depth proof tree, all of whose leaves are successful terminals.

The proof of the correctness of this algorithm is reported in the Appendix.

4 An Example

Let X_0 and Y_0 be two $LOTOS_{CF}$ processes so defined:

```
process X0[a,b,c,d]:exit:= a;exit[]b;exit>>X0>>X1 endproc
where
        process X1[a,b,c,d]:exit:= c;exit[]d;exit>>X0>>X1 endproc
endproc

process Y0[a,b,c,d]:exit:= a;exit[]b;exit>>Y1
where
        process Y1[a,b,c,d]:exit:= c;exit[]d;exit>>Y3>>Y0
        where
                process Y2[a,b,c,d]:exit:= c;exit[]d;exit>>Y3
                where
                        process Y3[a,b,c,d]:exit:= a;exit>>Y2[]b;exit>>Y1>>Y2 endproc
                endproc
        endproc
endproc
```

We want to apply the algorithm, described in Section 3, to prove the bisimilarity of X_0 and Y_0. To do this we need to define the systems of recursive equations in Greibach Normal Form, corresponding to the definitions of X_0 and Y_0:

$(X_0|B_X)=(X_0|\ X_0=a;exit[]b;X'_0>>X_0,\quad X'_0=exit>>X_1,$

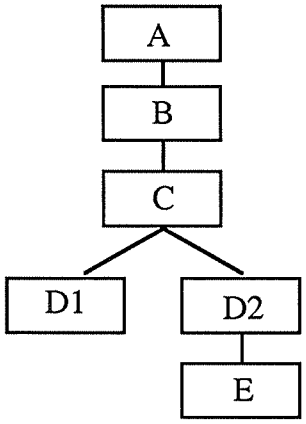
$X_1=c;exit[]d;X'_1>>X_1,\quad X'_1=exit>>X_0)$

$(Y_0|B_Y)=(Y_0|\ Y_0=a;exit[]b;exit>>Y_1,\quad Y_1=c;exit>>Y_0[]d;Y_4>>Y_0,$

$Y_2=c;exit[]d;exit>>Y_3,\quad Y_3=a;exit>>Y_2[]b;Y_5>>Y_2,$

$Y_4=exit>>Y_3,\ Y_4=exit>>Y_1).$

We construct then a tableau whose root is an equation between the recursion variables X_0 and Y_0. This tableau consists of six subtableaux, called A, B, C, D_1, D_2, E, and its structure is:



(A)

$$\frac{X_0 = Y_0}{\text{REC}}$$

$$\frac{a;exit\ []\ b;\ X'_0>>\ X_0\ =\ a;exit\ []\ b;exit>>\ Y_1}{\text{CHOICE}}$$

$$\frac{a;exit=a;exit}{\text{PREFIX}} \qquad \frac{b;X'_0>>X_0 = b;exit>>Y_1}{\text{PREFIX}}$$

$$\boxed{-\ =\ -}$$

$$\frac{X'0>>\ X0\ =exit>>\ Y1}{\text{REC}}$$

$$\frac{exit>>X_1>>X_0 = exit>>Y_1}{\text{EXIT}}$$

$$X_1>>X_0 = Y_1$$

(B)

$X_1 >> X_0 = Y_1$

REC

$(c;exit \; [] \; d;X'_0 >> X_1) >> X_0 = c;exit >> Y_0 \; [] \; d;Y_4 >> Y_0$

CHOICE

$c;exit >> X_0 = c;exit >> Y_0$ | $d;X'_0 >> X_1 >> X_0 = d;Y_4 >> Y_0$

PREFIX | PREFIX

$exit >> X_0 = exit >> Y_0$ | $X'_0 >> X_1 >> X_0 = Y_4 >> Y_0$

EXIT | REC

$X_0 = Y_0$ | $exit >> X0 >> X1 >> X0 = exit >> Y3 >> Y0$

EXIT

$X_0 >> X_1 >> X_0 = Y_3 >> Y_0$

SUBL

$X_0 >> X_1 >> Y_0 = Y_3 >> Y_0$

(C)

$X_0 >> X_1 >> Y_0 = Y_3 >> Y_0$

REC

$(a;exit[]b;X'_0 >> X_0) >> X_1 >> Y_0 = (a;exit >> Y_2 [] b;Y_5 >> Y_2) >> Y_0$

CHOICE

$a;exit >> X_1 >> Y_0 = a;exit >> Y_2 >> Y_0$ | $b;X'_0 >> X_0 >> X_1 >> Y_0 = b;Y_5 >> Y_2 >> Y_0$

PREFIX | PREFIX

$exit >> X_1 >> Y_0 = exit >> Y_2 >> Y_0$ | $X'_0 >> X_0 >> X_1 >> Y_0 = Y_5 >> Y_2 >> Y_0$

EXIT | REC

$X_1 >> Y_0 = Y_2 >> Y_0$ | $exit >> X_1 >> X_0 >> X_1 >> Y_0 = exit >> Y_1 >> Y_2 >> Y_0$

EXIT

$X_1 >> X_0 >> X_1 >> Y_0 = Y_1 >> Y_2 >> Y_0$

SUBL

$X_1 >> X_0 >> Y_2 = Y_1 >> Y_2$

(D1)

$X_1 >> Y_0 = Y_2 >> Y_0$

REC

$(c;exit \; [] \; d;X'_1 >> X_1) >> Y_0 = (c;exit \; [] \; d;exit >> Y_3) >> Y_0$

CHOICE

$c;exit >> Y_0 = c;exit >> Y_0$ | $d;X'_1 >> X_1 >> Y_0 = d;exit >> Y_3 >> Y_0$

PREFIX | PREFIX

$Y_0 = Y_0$ | $X'1 >> X1 >> Y0 = exit >> Y3 >> Y0$

REC

$exit >> X_0 >> X_1 >> Y_0 = exit >> Y_3 >> Y_0$

EXIT

$X_0 >> X_1 >> Y_0 = Y_3 >> Y_0$

(D2)

$X_1>>X_0>>Y_2 = Y_1>>Y_2$

REC

$(c;exit [] d;X'_1>>X_1) >>X_0>>Y_2 = (c;exit>>Y_0 [] d;Y_4>>Y_0)>>Y_2$

CHOICE

$c;exit>>X_0>>Y_2 = c;exit>>Y_0>>Y_2$ PREFIX

$d;X'_1>>X_1>>X_0>>Y_2 = d;Y_4>>Y_0>>Y_2$ PREFIX

$exit>>X_0>>Y_2=exit>>Y_0>>Y_2$

EXIT

$X_0>>Y_2=Y_0>>Y_2$

SUBL

$X_0>>X_1>>Y_0 = Y_3>>Y_0$

$X'_1>>X_1>>X_0>>Y_2=Y_4>>Y_0>>Y_2$

REC

$exit>>X0>>X1>>X0>>Y2=exit>>Y3>>Y0>>Y2$

EXIT

$X_0>>X_1>>X_0>>Y_2=Y_3>>Y_0>>Y_2$

(E)

$X_0>>Y_2 =Y_0>>Y_2$

REC

$(a;exit [] b;X'_0>>X_0) >>Y_2 = (a;exit [] b;exit>>Y_1)>>Y_2$

CHOICE

$a;exit>>Y_2 = a;exit>>Y_2$ PREFIX

$b;X'_0>>X_0>>Y_2=b;exit>>Y_1>>Y_2$ PREFIX

$Y_2 =Y_2$

$X'0>>X0>>Y2=exit>>Y1>>Y2$

REC

$exit>>X_1>>X_0>>Y_2=exit>>Y_1>>Y_2$

EXIT

$X_1>> X0>>Y_2 = Y_1>>Y_2$

In the subtableau E two leaves are successful terminal nodes; they are enclosed in boxes to make them more evident. Also the other terminal leaves in the whole tableau are successful. We can hence conclude that X_0 and Y_0 are strongly bisimilar.

5 Conclusions

The algorithm, presented in this paper, allows us to verify the strong bisimulation equivalence between $LOTOS_{CF}$ processes. The worst case time complexity of the algorithm, considered as the maximal depth of the proof tree in terms of the number of basic steps, is $O(m^4n^{3(m+1)})$, where n is the number of the variables and m the maximal norm of a variable [HüS91].

An implementation of this algorithm is planned inside the verification environment LITE, developed inside the project LOTOSPHERE [Eij91], to extend the verification functionality of this environment to cover a class of non finite state systems, in order to experiment on "real" specifications whether the worst case exponential complexity is really problematic in practice.
We wish also to extend, if possible, the class of application of the algorithm presented in this paper to context free processes defined outside $LOTOS_{CF}$. In fact other subset of LOTOS can describe context free processes, for example when the disabling operator is used together with recursion, action prefix and choice operators [FGM90].
Another topic under study is the applicability to $LOTOS_{CF}$ of a similar algorithm to decide branching bisimilarity on context free BPA processes. It is an open question instead as to whether weak bisimulation equivalence (or observational equivalence) is decidable on this class of processes [Hüt91].

References

[BaB90] Barbeau, M., Bochmann, G. V.: Extension of the Karp and Miller Procedure to LOTOS Specifications, Proc. 2nd Workshop on Computer-Aided Verification, Vol. 1, 1990.

[BBK88] Baeten, J. C. M., Bergstra, J. A., Klop, J. W.: Decidability of Bisimulation Equivalence for Processes generating Context-Free Languages, REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Model for Concurrency, Noordwijkerhout, 1988.

[BeK84] Bergstra, J. A.,. Klop, J. W: The Algebra of Recursively Defined Processes and the Algebra of Regular Processes, ICALP '84, LNCS vol. 172, pp. 82-94, 1984.

[BoB87] Bolognesi, T., Brinskma, E.: Introduction to the ISO Specification Language LOTOS, Computer Networks & ISDN Systems, 14, 1, 25-29 (1987).

[BoC89] Bolognesi, T., Caneve, M.: Equivalence Verification: Theory, Algorithms and a Tool, in van Eijk, P.H.J., Vissers C.A., Diaz M., eds., The Formal Description Technique LOTOS, pp. 303-326, North-Holland, 1989.

[BS92] Burkart, O., Steffen, B.: Model Checking for Context-Free Processes, Proc. CONCUR'92, LNCS vol. 630, pp. 123-137, 1992.

[CHS92] Christensen, S., Hüttel, H., Stirling, C.: Bisimulation Equivalence is Decidable for All Context-Free Processes, Proc. CONCUR'92, LNCS vol. 630, pp. 138-147, 1992.

[DeI91] De Francesco, N., Inverardi, P.: A Semantics Driven Method to Check the Finiteness of CCS Processes, Proc. 3rd Workshop on Computer-Aided Verification, pp.342-354, 1991.

[Eij91] van Eijk, P.: The Lotosphere Integrated Tool Environment LITE, in Proceedings 4th International Conference on Formal Description Techniques (FORTE '91), Sidney, November 1991, North-Holland, pp. 473-476.

[FGM90] Fantechi, A., Gnesi, S., Mazzarini, G., "How Much Expressive are LOTOS Behaviour Expressions?" Formal Description Techniques - III, pp. 17-32, North-Holland, 1990.

[FGMR92] Fernandez, J.C., Garavel, H., Mounier, L., Rasse, A., Rodriguez, C., Sifakis, J.: A Toolbox for the Verification of LOTOS Programs. 14th ICSE, Melbourne, 1992, pp. 246-261.

[HoU69] Hopcroft, J. E., Ullman, J. D.: Formal Languages and their Relation to Automata, Addison-Wensley, 1969.

[HüS91] Hüttel, H., Stirling, C.: Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes, Proc. LICS 91, Computer Society Press, pp. 376-386, 1991.

[Hüt91] Hüttel, H., Silence is Golden: Branching Bisimilarity is Decidable for Context-Free Processes, Proc. 3rd Workshop on Computer-Aided Verification, LNCS vol. 575, pp. 2-12, 1991.

[MaV89] Madeleine, E., Vergamini, D.: AUTO: A Verification Tool for Distributed Systems Using Reduction of Finite Automata Networks, Formal Description Techniques - II, pp. 61-66, North-Holland, 1989.

[Mil80] Milner, R.: A Calculus of Communicating Systems, LNCS vol. 92, 1980.

Appendix

In the following we prove the soundness and completeness of the presented algorithm. The proof closely resembles the one given for context free processes expressed in the BPA [HüS91].

Proposition 1

Let $(X|B_1)$ and $(Y|B_2)$ be two DeLOTOS+>> systems of recursive equations, normed and 3-GNF. Let Var_1 and Var_2 be the sets of variables respectively in B_1 and B_2; if $m=\max\{|Z|: Z\in Var_1\cup Var_2\}$ then:

1) $|\alpha|<|X>>\alpha|$ and $|\gamma>>\beta|<|Y>>\beta|$ for each SUBL rule application;

2) $|\beta|<|Y>>\beta|$ and $|\gamma>>\alpha|<|X>>\alpha|$ for each SUBR rule application;

3) $lth\ (\alpha_i)+lth\ (\gamma)+lth\ (\beta_i)\leq 3(m+1)$ for each SUB rules application.

Proof

1) Obviously $|\alpha|<|X>>\alpha|=|X|+|\alpha|$.

Since $X>>\alpha=Y>>\beta$ and $\alpha=\gamma>>\beta$ aren't unsuccessful terminals then $|X>>\alpha|=|Y>>\beta|$ and $|\alpha|=|\gamma>>\beta|$, and consequently $|\gamma>>\beta|=|\alpha|<|X>>\alpha|=|Y>>\beta|$.

2) This is analogous to the previous proof except that now we have $|\gamma>>\alpha|<|\beta|$.

3) Let $X>>\alpha=Y>>\beta$ be the root of a subtable, and $k=\min(|X|,|Y|)$; we build such subtableau iterating a basic step k times. Every time we apply the REC rule every variable is substituted with a sequence of at most two variables. Then after k steps we have $lth\ (\alpha_i), lth\ (\gamma), lth\ (\beta_i)\leq k+1$. Since $k\leq m$ then $lth\ (\alpha_i)+lth\ (\gamma)+lth\ (\beta_i)\leq 3(k+1)\leq 3(m+1)$. ♦

Theorem 1

(i) Every tableau for $X>>\alpha=Y>>\beta$ is finite.

(ii) There is a finite succession of subtableaux for $X>>\alpha=Y>>\beta$.

Proof

(i) If a tableau for $X>>\alpha=Y>>\beta$ were infinite then there would be an infinite path in the tree. This path would contain neither successful terminals nor unsuccessful terminal nodes.

From point 3) of Proposition 1 this path can not pass through infinitely many nodes which are consequents of a SUB rule, otherwise in this path there would have to be a successful terminal node which is reached infinitely many times.

Then this path must pass through a residual, but from points 1) and 2) of Proposition 1 this path cannot be infinite; we can therefore conclude that this path must be finite.

(ii) follows from (i) and from the fact that there is an upper bound on the number of basic steps along any path, namely $O\ (m^4 n^{3(m+1)})$, where n is the number of the variables and m the maximal norm of a variable. ♦

To prove the next theorem we need this definition:

Definition 4

The sequence of bisimulation approximations $\{<=>_n\}^{\omega}_{n=1}$ is defined inductively as follows:

- $p<=>_0 q \quad \forall\ p,q \in P$
- $p<=>_{n+1} q$ iff for each $a \in Act$:
  - $p\text{-}a\text{->}p' \Rightarrow \exists\ q'$: $q\text{-}a\text{->}q'$ and $p'<=>_n q'$
  - $q\text{-}a\text{->}q' \Rightarrow \exists\ p'$: $p\text{-}a\text{->}p'$ and $p'<=>_n q'$.

It is a standard result (see [M80] for instance) that any image-finite labelled transition (i.e. where for each $p \in P$ and $a \in Act$ the set $\{q|\ p\text{-}a\text{->}q\}$ is finite):

$$<=>\ = \bigcap_{n=0}^{\omega} <=>_n$$

Theorem 2

$X>>\alpha<=>Y>>\beta$ iff there exists a successful tableau for $X>>\alpha=Y>>\beta$.

Proof

($\Rightarrow$)

Suppose $X>>\alpha<=>Y>>\beta$; we build a tableau for $X>>\alpha=Y>>\beta$ which preserves the property that for each node $\alpha'=\beta'$ we have $\alpha'<=>\beta'$.

Clearly, in the case of the PREFIX and REC rule if the antecedent is true then so is the consequent. In SUB rules, if the residual is true and the antecedent is true, the consequent is also true .

This is not immediate when we consider the CHOICE rule. However, from the definition of strong bisimulation it follows that if $([]^n_{i=1} a_i;\alpha_i)>>\alpha<=>([]^m_{j=1} b_j;\beta_j)>>\beta$ then for each i there is a j such that $a_i;\alpha_i>>\alpha<=>b_j;\beta_j>>\beta$, and for each j there is an i with the same feature.

From these observations and from Theorem 1 this tableau construction must terminate with success.

($\Leftarrow$):

Suppose we had a successful tableau for $X>>\alpha=Y>>\beta$ but $X>>\alpha<\neq>Y>>\beta$.

Since the transition systems for $X>>\alpha$ and $Y>>\beta$ are image-finite, there exists an m such that $X>>\alpha<\neq>_m Y>>\beta$ and $\forall$ n<m $X>>\alpha<=>_n Y>>\beta$.

The rules for the basic steps are backwards sound with respect to each $<=>_m$, then within a subtableau for $X>>\alpha=Y>>\beta$ a leaf must exist $\alpha'=\beta'$ such that $\alpha'<\neq>_n\beta'$ with $n\leq m$, because the PREFIX rule must have been applied at least once.

After the application of the SUB rule there is at least one new root $\alpha"=\beta"$ of the subsequent subtableau with $\alpha"<\neq>_n\beta"$.

From these equations we choose $X_1>>\alpha=Y_1>>\beta$, and we choose the minimum n such that $X_1>>\alpha<\neq>_n Y_1>>\beta$.

Proceeding in this way we find a contradiction because we must find a successful terminal node. It cannot be $\alpha=\alpha$ since $\alpha<=>\alpha$ then $\forall m\geq 0$ $\alpha<=>_m\alpha$, however it cannot be $\alpha_i=\beta_i$ with $\alpha<\neq>_{n_i}\beta$ (where n_i is the minimum number such that $\alpha<=>_{n_i}\beta$ $\forall i$), since the node above it labelled $\alpha_i=\beta_i$ must have the property that $\alpha_i<=>_{n_i}\beta_i$. We therefore conclude that $X>>\alpha<=>Y>>\beta$.. ♦