The n-nested simulation equivalences are not congruences for the priority operators*

Pedro Ruben D'Argenio[1] Juan Vicente Echagüe[1,2] Cecilia Pertino[3]

[1] LIFIA
Depto. de Informática
Fac. de Ciencias Exactas
U. Nacional de La Plata
La Plata, Argentina

[2] InCo
Fac. de Ingeniería
U. de la República
Montevideo, Uruguay

[3] CeSPI
U. Nacional de La Plata
La Plata, Argentina

Abstract

In this paper we are interested in the *semantics of reactive systems*. We deal with the family of *priority operators* induced by partial orders over the set of actions. We study the compatibility of these operators with regards to semantic equivalences on labelled transtion systems.

Our main result is that n-nested simulation equivalence is not a congruence for this family of operators [GV89]. We show that the coarsest congruence contained in this equivalence is strictly coarser than $n+1$-nested simulation equivalence.

Finally ready-simulation equivalence [BIM88, Gla90] is shown to be a congruence for priority operators.

1 Introduction.

In this paper we are interested in the *semantics of reactive systems* [Pnu85]. These systems, whose classic examples are industrial control systems, drivers of computer networks and components of operating systems, are mainly characterized by the fact that they can be described as maintaining an interaction with their environment.

It is not suitable for these programs to be represented as a function or a relation that can be calculated, as is the case in many other systems which thus get the name of *functional* or *relational systems*. An example of these are the traditional programs of numeric calculus, compilers and all of which the interaction with the environment can be seen as a unique point of input and a unique point of output.

The most common models for reactive systems are based in *labelled transition systems (LTS)* [Kel76]. These are directed graphs where nodes represent states and edges represent transitions between states, labelled by actions over which the interaction of the system with the environment is based.

The popularity of the LTS is due to their simplicity, grafic representation, easiness to define operations which represent the usual constructors of reactive systems (alternative composition, parallel composition, comunication and priority among others) and the existence of a simple technique, *Structured Operational Semantics* [Plo81] to give semantics over the LTSs to a programming language (not necessarily reactive) [Hen90].

*E-mail: pedro@unlp.edu.ar, echague@fing.edu.uy, ceciliap@cespivm2.unlp.edu.ar

Many simple languages, called *process description languages*, have been proposed for reactive systems. For example CCS [Mil80, Mil89], CSP [BHR84, Hoa85] and ACP [BK85, BW90], all of which posses semantics based on LTSs.

In general LTS cannot be used directly to give semantics to a language because they are too concrete. For example they make a distinction between systems a and $a + a$, represented in Example 1.1, which have the same behaviour: both start executing the action a and finish inmediately after.

Example 1.1

The usual technique to deal with this problem of low level of abstraction is to give the semantics in two steps. First associate the system to an LTS (its concrete semantics) and then consider the set of LTS module a relation of equivalence (called semantic equivalence). In this way we associate an equivalence class of the LTS to the system. This class is the abstract semantics. Numerous semantic equivalences have been proposed. Within the most popular are traces [OH83], failure [BHR84] and bisimulation equivalence [Par81]. An extensive study comparing these equivalences can be found in [Gla90] and [Gla93].

The choice of the semantics in two steps demands a little care when defining the operations and equivalences to be used. This is due to the fact that, in general, the semantics of the constructors of languages are given as operations over the LTS. For each semantic equivalence it is necessary to ask whether these constructors continue to have sense in the equivalence classes. In other words, if we have an operator α over the LTS which adecuately represents a certain construction of a language and we are interested in the semantic equivalence $\sim$: could we lift the operation α from the LTS to the equivalence class given by $\sim$?

It is enough to prove that the equivalence $\sim$ is a *congruence* for the operator α, that is: if $\mathcal{G}_1 \sim \mathcal{G}_2$ then $\alpha(\mathcal{G}_1) \sim \alpha(\mathcal{G}_2)$. This property is also known as *compatibility* of the operator with the equivalence, and intuitively tells us that if two LTSs belong to the same equivalence class their images through the operator remain together. In other words, two equivalent systems should remain equivalent after the application of the operator.

As we can see, being a congruence for the considered operations is a natural condition and in fact, necessary to use the equivalence within the LTS as semantic equivalence.

A well known family of operators within the LTS is the family of priority operators. The aplication of a priority operation consists of, cutting the options of "low priority" in alternative compositions. Up to our knowledge this family of operators appeared for the first time in the literature of reactive systems in [BBK86]. Formally a priority operator is associated to a partial order, the priority order, within the set of actions. When applied to an LTS, the labels are considered of all outgoing transitions, and for each state only the transitions with $\geq$-maximal labels are preserved.

It is well known that bisimulation is a congruence for the family of priority operators [BBK86, BBK85]. It is also known that neither trace equivalence, nor failure equivalence are congruences for them [BBK85], and that the coarsest congruence for priority contained in these is ready-trace equivalence [BBK85]. A more extensive study of the relation within the usual semantics and the family of priority operators can be found in [Per94].

In this paper we are interested in the relation between the priority operators in the LTS and certain

equivalences found between ready-trace and bisimulation equivalence: ready-simulation [BIM88, Gla90] and the family of n-nested simulation equivalences [GV89].

The n-nested simulations (for $n = 0, 1, ...$) form a secuence of equivalences which become strictly finner when n grows. They extend from simulation, coinciding with 1-nested simulation, to bisimulation: Two finitely branching LTSs are bisimilar only if they are n-nested similar for an n large enough. Intuitively two systems are bisimilar when they can succesfully participate of a refined game of imitation (in a very precise sense) which allows the participants to interchange, at any time of the game and any number of times, the roles of imitator and imited. In the same context, two systems are n-nested similar when they can participate succesfully of the same game in the case that the number of interchanging of roles is limited to a maximun of $n - 1$.

As bisimualtion, n-nested simulations posses a very simple characterization in logic HML [HM85]: it is easy to prove that two finitely branching LTSs are n-nested similar if and only if they satisfy exaclty the same HML formulas with at most $n - 1$ nested negations [GV89].

Finally, 2-nested simulation plays an important role in the theory of Structured Operatioanl Semantics [Plo81]. In [GV89] this equivalence is shown to be the coarsest congruence with respect to a large family of operators contained in completed trace equivalence. This operators can be defined in *tyft/tyxt format*.

In this paper three original results are presented.

The main result is that none of the n-nested simulations are congruences for the priority operators. This was unexpected to the authors because, from $n = 2$, these equivalences are finner than ready-simulation and thus "posses all the necessary information" to behave correctly with respect to priority.

The second is the study of the coarsest congruences for the priority operators contained in the n-nested simulations. We show that these n-nested congruences are placed between the n-nested and the $n + 1$-nested simulations.

The last result is that ready-simulation equivalence is a congruence for the family of priority operators.

Some immediate observations can be made from this results. The first is that n-nested simulations cannot be considered semantic equivalences, at least in contexts where priority operations have sense. The second is related to Structured Operational Semantics: as priority operators are not compatible with 2-nested simulation, they cannot be represented in *tyft/tyxt format*. Finally, the place of n-nested congruences, in the semantic lattice over the LTS, tells us that a new structure exists in this area that is worth studying.

The structure of this paper is as follows. The section 2 is dedicated to basic notions and presentation of the notation. It deals with the notion of LTS, the definitions of the semantic equivalences, the characterization of these equivalences in a logical way and the presentation of the family of priority operators. Also some well known results are mentioned. Section 3 contains the contribution to this paper. It begins with the result of congruence for ready-simulation. Afterwards shows that n-nested is not a congruence for the priority operators and finishes stating the relation of the n-nested congruences with known equivalences. We conclude in section 4 presenting some consecuences of these results and perspectives of further work.

2 Notions of LTS, semantic equivalences, formulas and priority operators.

This section is divided into four parts: the first part 2.1, dealing with the definition and notation of the LTS. Part 2.2 recalls the equivalences used. Part 2.3 explains the basics of HML logic. And the last part 2.4 introduces the priority operators.

2.1 LTS.

We give the semantic model, the Labelled Transition Systems [Kel76]. Over this model we define the different semantic equivalences.

Definition 2.1.1 *A* labelled transition system *(LTS) is a structure* $\mathcal{G} = (S, i, Act, \longrightarrow)$ *where* S *is a set of* states *with inicial state* i, Act *is the* alphabet of actions, *and* $\longrightarrow \subseteq S \times Act \times S$, *the* transition relation. *Elements* $(p, a, q) \in \longrightarrow$, *notation* $p \xrightarrow{a} q$, *are called* transitions.

The set of *initial actions* of a state is defined by: $I(p) = \{a \in Act : \exists q . p \xrightarrow{a} q\}$. An LTS is *image finite* if for all $p \in S$ and $a \in Act$ the set $\{q : p \xrightarrow{a} q\}$ is finite. All the LTSs considered through this paper are image finite. Let us use $\mathcal{C}_{LTS}$ to denote the set of LTSs.

In order to provide some examples along the paper it is usefull to have a sintax for LTS. We use a small part of CCS [Mil80, Mil89], just with combinators nil (0), prefixing ($a._-$) and non-deterministic choice ($\_ + \_$). 0 represents the LTS with one state and no arrows; $a.p$ adds to p an initial state and an arrow labelled a to connect to it; $p + q$ identifies the initial states of the given of p and q into a new one.

2.2 Equivalences.

Intuitively an LTS *simulates* another when a strategy can be found that allows the first LTS to imitate the evolution of the second. This strategy, called *simulation*, is a set of pairs (p, q), implying state p can imitate the evolutions of state q in a very precise sense.

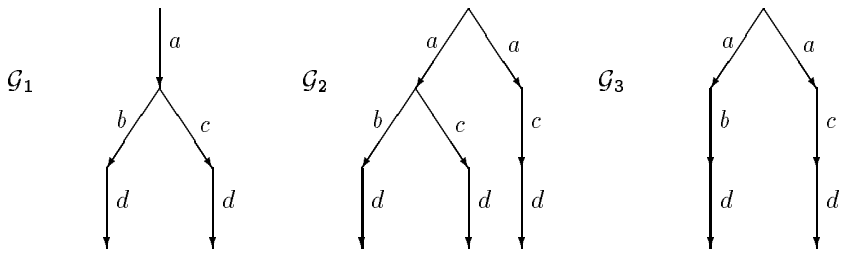
Definition 2.2.1 (Simulation) *Given* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$, *where* $\mathcal{G}_1 = (S_1, i_1, Act, \longrightarrow_1)$ *and* $\mathcal{G}_2 = (S_2, i_2, Act, \longrightarrow_2)$. *A* simulation $R \subseteq S_1 \times S_2$ *from* $\mathcal{G}_1$ *to* $\mathcal{G}_2$, *notation* $R : \mathcal{G}_1 \underset{\sim}{\subseteq} \mathcal{G}_2$, *is a binary relation satisfying* $i_1 R i_2$ *and*

$$\textit{if } pRq \textit{ and } p \xrightarrow{a}_1 p', \textit{ then } \exists q' : q \xrightarrow{a}_2 q' \textit{ and } p' R q'$$

An LTS $\mathcal{G}_1$ *is simulated by* $\mathcal{G}_2$, *notation* $\mathcal{G}_1 \underset{\sim}{\subseteq} \mathcal{G}_2$ *if there exists* $R : \mathcal{G}_1 \underset{\sim}{\subseteq} \mathcal{G}_2$. *We say that* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are* similar, *notation* $\mathcal{G}_1 \rightleftharpoons \mathcal{G}_2$, *if* $\mathcal{G}_1 \underset{\sim}{\subseteq} \mathcal{G}_2$ *and* $\mathcal{G}_2 \underset{\sim}{\subseteq} \mathcal{G}_1$.

That is, two LTSs are similar if a strategy can be found for each LTS to imitate the other. These two strategies need not be simetric. So the pair (p, q) means that p can imitate q but not necessarily q can imitate p. In general, the roles of the imitator state (p) and the imited state (q) are not interchangable. This is the case only in the initial state of the LTS, where the roles of imitator and imited are assigned and thus, the strategy to be used is chosen.

Example 2.2.2 *In this example we have* $\mathcal{G}_1 \rightleftharpoons \mathcal{G}_2, \mathcal{G}_1 \not\rightleftharpoons \mathcal{G}_3$ *and* $\mathcal{G}_2 \not\rightleftharpoons \mathcal{G}_3$.



$a(bd + cd)$ $\qquad$ $a(bd + cd) + acd$ $\qquad$ $abd + acd$

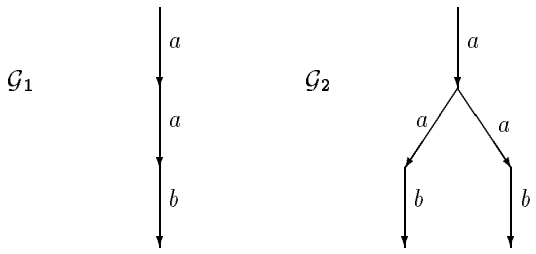
Allowing the interchangeability of roles at any moment a finner equivalence, the *bisimulation* [Par81], is obtained. This interchange can be done as many times as desired along the imitation game. It is enough to ask for both simulations to coincide, in the sense that every pair (p, q) means p simulates q and q simulates p.

Definition 2.2.3 (Bisimulation) *Given $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. A* bisimulation $R : S_1 \times S_2$ *between $\mathcal{G}_1$ and $\mathcal{G}_2$, notation $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$ is a binary relation satisfying $R : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2$ and $R^{-1} : \mathcal{G}_2 \subseteq_{\rightarrow} \mathcal{G}_1$.*

$\mathcal{G}_1$ and $\mathcal{G}_2$ are bisimilar, *notation $\mathcal{G}_1 \leftrightarrows \mathcal{G}_2$, if there exists $R : \mathcal{G}_1 \leftrightarrows \mathcal{G}_2$.*

It is well known that bisimulation equivalence is strictly finner than simulation. Note that the similar LTSs of Example 2.2.2 are not bisimilar. Two bisimilar LTSs are shown in Example 2.2.4.

Example 2.2.4



aab $\qquad\qquad$ $a(ab + ab)$

An interesting family of equivalences that can be found between simulation and bisimulation are the n-nested equivalences. These equivalences are defined by allowing only a limited number of changes of roles. For each natural n, arises the notion of *n-nested simulation* [GV89] where roles can be exchanged $n - 1$ times along the imitation game.

Definition 2.2.5 (n-nested simulation) *Given $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$, such that $\mathcal{G}_1 = (S_1, i_1, Act, \longrightarrow_1)$ and $\mathcal{G}_2 = (S_2, i_2, Act, \longrightarrow_2)$. An* n-nested simulation $R \subseteq S_1 \times S_2$ *from $\mathcal{G}_1$ to $\mathcal{G}_2$, notation $R : \mathcal{G}_1 \subseteq_{n\rightarrow} \mathcal{G}_2$, is a relation satisfying:*

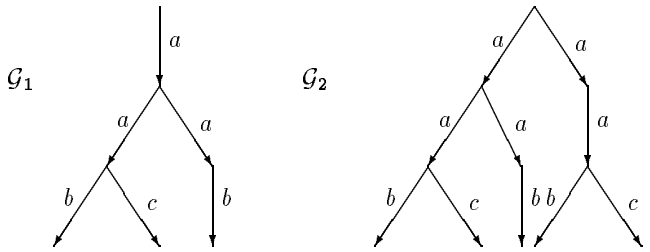
1. $R : \mathcal{G}_1 \subseteq_{0\rightarrow} \mathcal{G}_2$ *iff* $i_1 R i_2$
2. $R : \mathcal{G}_1 \subseteq_{n+1\rightarrow} \mathcal{G}_2$ *iff* $R : \mathcal{G}_1 \subseteq_{\rightarrow} \mathcal{G}_2 \wedge \exists Q : \mathcal{G}_2 \subseteq_{n\rightarrow} \mathcal{G}_1$ *with* $R^{-1} \subseteq Q$

$\mathcal{G}_1$ is n-nested simulated by *$\mathcal{G}_2$, notation $\mathcal{G}_1 \subseteq_{n\rightarrow} \mathcal{G}_2$, if there exists $R : \mathcal{G}_1 \subseteq_{n\rightarrow} \mathcal{G}_2$.*

$\mathcal{G}_1$ and $\mathcal{G}_2$ are n-nested simulation equivalent, *notation $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$, if $\mathcal{G}_1 \subseteq_{n\rightarrow} \mathcal{G}_2$ and $\mathcal{G}_2 \subseteq_{n\rightarrow} \mathcal{G}_1$.*

Notice that if we reinforce, in this definition, the condition of "being contained" with "coincide" we obtain bisimulation. Also note that $\rightleftharpoons_1$ coincides with $\rightleftharpoons$.

Example 2.2.6 ([GV89]) *The following LTSs are 2-nested equivalent.*



$a(a(b+c)+ab)$ $\qquad$ $a(a(b+c)+ab)+aa(b+c)$

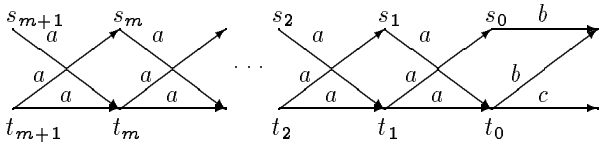
Let I be the identity relation. Take $R_1 = I \cup \{(a(a(b+c)+ab), a(a(b+c)+ab)+aa(b+c))\}$, $R_2 = R_1^{-1} \cup \{(a(b+c)+ab, a(b+c))\}$. *And* $S_1 = R_2$, $S_2 = S_1^{-1} \cup \{(b, b+c)\}$.

For every n we get different equivalences, finner when n grows and all of them strictly coarser than bisimulation (the LTSs in Example 2.2.6 are not bisimilar). The next example generalizes Example 2.2.6 to obtain, for each n, a pair of n-nested equivalent LTSs (in Example 2.3.4 we show that they are not $n+1$-nested equivalent).

Example 2.2.7 *In* [GV89] *the LTSs of Example 2.2.6 are generalized to $\mathcal{G}_1^n$ and $\mathcal{G}_2^n$, in order to obtain $\mathcal{G}_1 \rightleftharpoons_n \mathcal{G}_2$ and $\mathcal{G}_1 \not\rightleftharpoons_{n+1} \mathcal{G}_2$ for $n \geq 2$.*

LTSs $\mathcal{G}_1^n$ and $\mathcal{G}_2^n$ are obtained by terms s_n and t_n defined for $m \geq 0$ by:

$$\begin{array}{llll} s_0 & = b & t_0 & = b + c \\ s_{m+1} & = a.t_m & t_{m+1} & = a.s_m + a.t_m \end{array}$$



Another equivalence mentioned in the literature is *ready-simulation* [BIM88]. It is a refinement of simulation, where we ask for the preservation of initial actions.

Definition 2.2.8 (Ready-simulation.) *Let $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. A* ready-simulation $R : S_1 \times S_2$ *from $\mathcal{G}_1$ to $\mathcal{G}_2$, notation $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$, is a simulation $R : \mathcal{G}_1 \subseteq \mathcal{G}_2$ satisfying: if pRq then $I(p) = I(q)$.*

An LTS $\mathcal{G}_1$ is ready-simulated by $\mathcal{G}_2$, notation $\mathcal{G}_1 \subseteq_R \mathcal{G}_2$, if there exists $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$. $\mathcal{G}_1$ and $\mathcal{G}_2$ are ready-simulation equivalent, *notation $\mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2$ if $\mathcal{G}_1 \subseteq_R \mathcal{G}_2$ and $\mathcal{G}_2 \subseteq_R \mathcal{G}_1$.*

Ready-simulation requires the set of inicial actions to be the same on the two simulated processes but, as simulation, does not allow the interchange of roles.
Ready-simulation equivalence is between simulation (the LTSs $\mathcal{G}_1$ and $\mathcal{G}_2$ in Example 2.2.2 are similar but not ready-similar) and bisimulation.

Note that for $n \geq 2$, $\leftrightarrows \subset \rightleftharpoons_{n+1} \subset \rightleftharpoons_n \subset \rightleftharpoons_R \subset \rightleftharpoons$.

2.3 HML logic.

All the equivalences we already saw can be characterized very simply in the Hennessy and Milner logic. In this framework is that we base most of our proofs.

Definition 2.3.1 (HML logic [HM85]) *The set $\mathcal{L}_{HML}$ of* Hennessy-Milner logic (HML) formulas *(over a given alphabet set $\{a, b, \ldots\}$) is given by the grammar: $\phi ::= T \mid \phi \wedge \phi \mid \neg\phi \mid \langle a \rangle \phi$*

The satisfaction relation $\models \subseteq S \times \mathcal{L}_{HML}$ *is defined in the standard way:*

- $p \models T$, *for every* p.
- $p \models \varphi \wedge \psi$ *iff both* $p \models \varphi$ *and* $p \models \psi$.
- $p \models \neg\varphi$ *iff* $p \not\models \varphi$.
- $p \models \langle a \rangle \varphi$ *iff there exists a* p' *such that* $p \xrightarrow{a} p'$ *and* $p' \models \varphi$.

By $\mathcal{L}_i \subset \mathcal{L}_{HML}$, $i \geq 0$ we understand the set of HML formulas which contain at most $i-1$ (nested) occurrences of $\neg$.

Definition 2.3.2 (Equivalence relation induced by a logic) *Let* χ *be a set of HML formulas. With* $\sim_\chi$ *we denote the equivalence relation in* $\mathcal{C}_{LTS}$ *induced by* χ*:*

$$\mathcal{G}_1 \sim_\chi \mathcal{G}_2 \stackrel{\text{def}}{\Leftrightarrow} (\forall \phi \in \chi : i_1 \models \phi \Leftrightarrow i_2 \models \phi)$$

Where i_1 *and* i_2 *correspond to the initial states of* $\mathcal{G}_1$ *and* $\mathcal{G}_2$.

We call this relation χ-formula equivalence.

Using subsets of $\mathcal{L}_{HML}$ we can characterize the equivalences defined in this section.

The first result, owed to Hennessy and Milner, was on bisimulation and afterwards extended to the rest of the equivalences (see [HM80, GV89]).

Theorem 2.3.3 (Equivalences in HML) *Given* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *image finite LTSs. Then:* $\mathcal{G}_1 \leftrightarrows \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_{HML}} \mathcal{G}_2$, $\mathcal{G}_1 =_n \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_n} \mathcal{G}_2$ *and* $\mathcal{G}_1 = \mathcal{G}_2 \Leftrightarrow \mathcal{G}_1 \sim_{\mathcal{L}_1} \mathcal{G}_2$.

A similar result can be found for ready-simulation using a simple extension of HML [BIM88, Gla90].

Example 2.3.4 *Back to the* $\mathcal{G}_1^n$ *and* $\mathcal{G}_2^n$ *of Example 2.2.7 we can see that the LTSs are not bisimilar, not even* $n+1$*-nested equivalent.*

Using $f_1 = \langle c \rangle T$ *and* $f_{n+1} = \langle a \rangle \neg f_n$ *we have a formula* $f_n \in \mathcal{L}_n : \mathcal{G}_2^n \models f_{n+1}$ *and* $\mathcal{G}_1^n \not\models f_{n+1}$. *Thus,* $\mathcal{G}_1^n \not=_{n+1} \mathcal{G}_2^n$. *As* $f_{n+1} \in \mathcal{L}_{HML}$ *we also have* $\mathcal{G}_1^n \not\leftrightarrows \mathcal{G}_2^n$.

2.4 Priority operators.

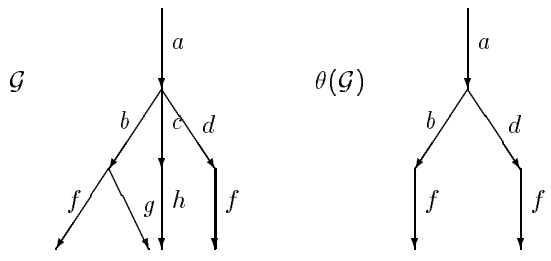
The family of priority operators $\theta_\geq$ are well known in the literature of LTS [BBK85, BBK86, BW90]. Each operator $\theta_\geq$ is associated to a partial order $\geq$ on Act ($a > b$ meaning a has priority over b). The application of a priority operation consists of cutting transitions of "low priority". That is, for each state of the LTS the outgoing transitions are considered and only the $\geq$-maximal are maintained.

Definition 2.4.1 (Priority operator) *Given an LTS* $\mathcal{G} = (S, i, Act, \longrightarrow)$, *for each partial order* $\geq$ *on* Act, *we define a* priority operator $\theta_\geq : \mathcal{C}_{LTS} \to \mathcal{C}_{LTS}$ *by:*

$$\theta_\geq(S, i, Act, \longrightarrow) = (S, i, Act, \{p \xrightarrow{a} q : p \xrightarrow{a} q \wedge \forall b.p \xrightarrow{b} \text{ implies } b \not> a\})$$

We use θ instead of $\theta_\geq$ when there is no ambiguity.

Example 2.4.2 *We show* $\mathcal{G}$ *and* $\theta_\geq(\mathcal{G})$, *with* $\geq$ *defined by* $\{a > d, b > c, f > g\}$.



$a(b(f+g)+ch+df)$ $\qquad$ $a(bf+df)$

3 n-nested simulation equivalence is not a congruence for the priority operator.

This section contains the contributions of this paper. In part 3.1 we prove that ready-simulation is a congruence for the priority operators and that this is not the case for n-nested simulation. Part 3.2 studies the coarsest congruence for priority operators contained in n-nested simulation.

3.1 Ready-simulation and n-nested simulation.

When introducing operators in a model it is natural to ask for congruence results for the different semantic equivalences.

Definition 3.1.1 (Congruence) *Given $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$, we say that an equivalence $\approx$ is a* congruence *for the operator $\alpha : \mathcal{C}_{LTS} \to \mathcal{C}_{LTS}$ iff $\mathcal{G}_1 \approx \mathcal{G}_2 \Rightarrow \alpha(\mathcal{G}_1) \approx \alpha(\mathcal{G}_2)$.*

It is well known that simulation is not a congruence for the priority operators. On the contrary, bisimulation is known to be one. The following results show that ready-simulation is also a congruence for this operators whereas n-nested simulation is not.

Theorem 3.1.2 *Let $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$ and a priority operator θ. If $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ then $R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$.*

Corollary 3.1.3 (Ready-simulation is a congruence for the priority operators) *Given $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$: $\mathcal{G}_1 \rightleftharpoons_R \mathcal{G}_2 \Rightarrow \theta(\mathcal{G}_1) \rightleftharpoons_R \theta(\mathcal{G}_2)$*

Theorem 3.1.4 *n-nested is not a congruence for the priority operators*

Proof: We take from Example 2.2.7 the LTSs $\mathcal{G}_1^n, \mathcal{G}_2^n$ which are n-nested similar. And θ with the partial order $\{c > b\}$. It is easy to see that $\theta(\mathcal{G}_1^n) \not\rightleftharpoons_n \theta(\mathcal{G}_2^n)$. $\theta(\mathcal{G}_1^n)$ and $\theta(\mathcal{G}_2^n)$ correspond to terms s'_n and t'_n respectively.

$$\begin{array}{ll} s'_0 = b & t'_0 = c \\ s'_{n+1} = a.t'_n & t'_{n+1} = a.s'_n + a.t'_n \end{array}$$

Then we give formulas $f_n \in \mathcal{L}_n$ such that $\theta(\mathcal{G}_2^n) \models f_n$ and $\theta(\mathcal{G}_1^n) \not\models f_n$: $f_1 = \langle a \rangle \langle b \rangle T$ and $f_{n+1} = \langle a \rangle \neg f_n$. ■

3.2 n-nested congruence.

We study in this section the coarsest congruence for the priority operators included in n-nested simulation.

Definition 3.2.1 (n-nested congruence) *We define the* n-nested congruence, *notation $\sim_n^\theta$, as the coarsest congruence for the priority operators included in n-nested simulation. That is, $\mathcal{G}_1 \sim_n^\theta \mathcal{G}_2 \Rightarrow \theta(\mathcal{G}_1) \sim_n^\theta \theta(\mathcal{G}_2)$.*

From this definition and Theorem 3.1.4 we have that $\sim_n^\theta \subset \rightleftharpoons_n$. Also $\sim_{n+1}^\theta \subset \sim_n^\theta$. Theorem 3.2.7 sets with more precision n-nested congruence in the semantic lattice.

Lemma 3.2.2 (An n-nested simulation R is a ready-simulation) *Given $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. If $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2$, $n \geq 2$, then $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$.*

From Theorem 3.1.2 and Lemma 3.2.2 we have:

Lemma 3.2.3 *Given* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2, n \geq 2 \Rightarrow R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2) \Rightarrow R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$.

Theorem 3.2.4 *Take* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$, *such that* $\mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2$. *Then for any* θ: $\theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$, *for* $n \geq 2$.

Proof: We prove even more because we prove: $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2 \Rightarrow R : \theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$.

By induction n: When $n = 2$ we have $R : \mathcal{G}_1 \subseteq_2 \mathcal{G}_2$ implies $R : \mathcal{G}_1 \subseteq_R \mathcal{G}_2$ (by Lemma 3.2.2). By Theorem 3.1.2 $R : \theta(\mathcal{G}_1) \subseteq_R \theta(\mathcal{G}_2)$ then $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$ (Lemma 3.2.3).

Now suppose $n \geq 2$. By hypothesis: $R : \mathcal{G}_1 \subseteq_n \mathcal{G}_2 \Rightarrow R : \theta(\mathcal{G}_1) \subseteq_{n-1} \theta(\mathcal{G}_2)$. $R : \mathcal{G}_1 \subseteq_{n+1} \mathcal{G}_2 \stackrel{\text{def}}{\Rightarrow} \exists R' \subseteq S_2 \times S_1$ such that $R' : \mathcal{G}_2 \subseteq_n \mathcal{G}_1 \wedge R^{-1} \subseteq R'$. By Lemma 3.2.3 $R : \theta(\mathcal{G}_1) \subseteq \theta(\mathcal{G}_2)$ and by hypothesis $R' : \theta(\mathcal{G}_2) \subseteq_{n-1} \theta(\mathcal{G}_1)$. Thus $R : \theta(\mathcal{G}_1) \subseteq_n \theta(\mathcal{G}_2)$. ■

Corollary 3.2.5 (Behaviour of θ in n-nested) *Given* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. *For* $n \geq 2 : \mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2$ *implies* $\theta(\mathcal{G}_1) \rightleftharpoons_n \theta(\mathcal{G}_2)$.

Remark 3.2.6 (The reverse of Corollary 3.2.5 does not hold) *Let* $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{C}_{LTS}$. $\theta(\mathcal{G}_1) \rightleftharpoons_n \theta(\mathcal{G}_2)$, *for any* θ *does not imply* $\mathcal{G}_1 \rightleftharpoons_{n+1} \mathcal{G}_2$.

Given $\mathcal{G}_1 = a(aa + a(aa + a))$ and $\mathcal{G}_2 = aa(aa + a) + a(aa + a(aa + a))$, obtained from terms s_2 and t_2 of Example 2.2.7 where we only changed $s_0 = b$ by $s_0 = a$ and $t_0 = c$ by $t_0 = aa$. $\mathcal{G}_1$ and $\mathcal{G}_2$ are 2-nested similar but not 3-nested similar. It is easy to see that for any θ, $\theta(\mathcal{G}_1) = \mathcal{G}_1 \rightleftharpoons_2 \mathcal{G}_2 = \theta(\mathcal{G}_2)$, but $\theta(\mathcal{G}_1) = \mathcal{G}_1 \not\rightleftharpoons_3 \mathcal{G}_2 = \theta(\mathcal{G}_2)$.

This example can be generalized for n-nested simulation equivalences as done in 2.2.7.

Theorem 3.2.7 (n-nested congruence in the semantic lattice) $\rightleftharpoons_{n+1} \subset \sim_n^{\theta} \subset \rightleftharpoons_n$.

We prove this theorem for a more general set of operators.

Definition 3.2.8 ($\mathcal{F}_f$: a family of unary operators) *Let* H *be the set of functions* $f : \mathcal{P}(Act) \to \mathcal{P}(Act)$ *satisfying* $f(X) \subseteq X$ *for all* $X \subseteq Act$. *We define the* operator $\mathcal{F}_f : \mathcal{C}_{LTS} \to \mathcal{C}_{LTS}$, *for all* $f \in H$, *for cutting branches of an LTS, as:*

$$\mathcal{F}_f((S, i, Act, \longrightarrow)) \stackrel{\text{def}}{=} (S, i, Act, \longrightarrow')$$

where $p \stackrel{a}{\longrightarrow}' q \stackrel{\text{def}}{\Leftrightarrow} p \stackrel{a}{\longrightarrow} q \wedge a \in f(I(p))$.

Definition 3.2.9 ($\mathcal{F}_f$ congruence) *We define the* $\mathcal{F}_f$ congruence, *notation* $\sim_n^{\mathcal{F}_f}$, *as the coarsest congruence for the* $\mathcal{F}_f$ *operators included in* n*-nested simulation.*

Now it is easy to prove:

Lemma 3.2.10 ($\mathcal{F}_f$ congruence in the semantic lattice) $\rightleftharpoons_{n+1} \subset \sim_n^{\mathcal{F}_f} \subset \rightleftharpoons_n$.

As each θ (related to a partial order $\geq$) is a particular case of $\mathcal{F}_f$ obtained with $f(X) = \{a \in X : \forall b \in X. b \not> a\}$, we have Theorem 3.2.7 as a corollary of this lemma.

4 Concluding remarks.

In this paper we studied the relationship between the priority operators and some equivalence relations in the LTS. We were particularly interested in knowing which equivalences are congruences for these family of operators.

The main results presented are three: in the first place, ready-simulation is a congruence for the priority operators. This was an expected result after the results on traces and ready-traces (see

[BBK85]) saying that demanding equal initial action sets, to an equivalence which is not a congruence, is enough to obtain a congruence. We include it here because were not able to find it in the literature.

In a second place, our most important contribution, the n-nested simulations are not congruences for the priority operators. Thus, leaving these equivalences out of consideration as semantic equivalences, at least in contexts where priority operations have sense. This result was rather unexpected because, for $n > 1$, n-nested equivalences are strictly finner than ready-simulation and the "information" about the inicial action set is "available".

The first conclusion we arrive to after this result is that the priority constructors in languages cannot be supplied of a structured operational semantics within the *tyft/tyxt* format. This is because all the constructors to which a semantics can be given in that format are compatible with 2-nested simulation, as shown in [GV89].

The intuitive presentation of the n-nested simulation in terms of a game of simulation with a limited number of changes of roles, and its characterization through HML makes us believe that the family of n-nested is a secuence approaching to bisimulation. We think it is worth to dedicate more effort to understand the behaviour of these equivalences and look for an equivalence, convergent to bisimulation, with better behaviour.

This idea is reinforced by the third contribution of this paper: the coarsest congruence for priority operators contained in n-nested simulation is placed between n-nested and $n+1$-nested simulation. Apparently in that area of the semantic lattice exists an interesting structure to study.

Various lines of research are proposed by these results. The first is to look for a characterization of the n-nested congruence in the style of the ones given in 2.2. This kind of definition will give us a better understanding of n-nested simulations.

Another is the search of other operators over the LTS compatible with bisimulation and not with n-nested simulations. Take as example the broadcast combiner $\bowtie$ defined in [Pnu85]. As for today we do not know whether the coarsest congruence for this operator contained in n-nested simulation is n-nested congruence.

An approach to this matter, in which we are working at the moment, deals with the function $\mathcal{F}$ (See definition 3.2.8). We know that both θ and $\bowtie$ can be constructed from $\mathcal{F}$ for suitable elections of f. At the moment we are trying to characterize the family of f such that $\mathcal{F}_f$ is compatible with n-nested simulation. And when this is not the case characterize the coarsest congruences This work will be subject of publication in the near future [Per94].

Acknowledgement.

We would like to thank Javier Blanco who encouraged us to work on priority operators and gave advice throughout our work. We also thank Daniel Yankelevich because of his useful comments.

This work was partially supported by the "Programa de Desarrollo de Ciencias Básicas", Uruguay.

References

[BBK85] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Ready trace semantics for concrete process algebra with the priority operator. Report CS-R8517, Centrum voor Wiskunde en Informatica, Amsterdam, August 1985.

[BBK86] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. In *Fundamentae Informaticae IX*, pages 127–168, 1986.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BIM88] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: preliminary report. In *Conference Record of the 15th ACM Symposium on Principles of Programming Languages*, pages 229–239, California, San Diego, 1988.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, (37):77–121, 1985.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra*. Cambridge University Press, 1990.

[Gla90] R.J. van Glabbeek. The linear time - branching time specrtrum. Report CS-R9029, Centrum voor Wiskunde en Informatica, Amsterdam, July 1990.

[Gla93] R.J. van Glabbeek. The linear time - branching time specrtrum II. The semantics of sequential systems with silent moves (preliminary version), 1993.

[GV89] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Proceedings ICALP 89*, pages 423–438, Stressa, 1989. LNCS 372, Springer-Verlag.

[Hen90] M. Hennessy. *The semantics of programming languages: an elementary introduction using structural operational semantics*. Jhon Wiley & Sons, Sussex, 1990.

[HM80] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In J. de Baker and J. van Leeuwen, editors, *Proceedings ICALP 80*, pages 299–309. LNCS 85, Springer-Verlag, 1980.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Hoa85] C.A.R. Hoare. *Communicating sequential process*. Prentice Hall, 1985.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 8(19):371–384, 1976.

[Mil80] R. Milner. *A calculus of communicating systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and concurrency*. Prentice Hall, 1989.

[OH83] E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. In J. Diaz, editor, *Proceedings 10th ICALP*, pages 561–572, Barcelona, 1983. LNCS 154, Springer-Verlag.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequence. In P. Deussen, editor, *Proceedings 5th. GI Conference*, pages 167–183. LNCS 104, Springer-Verlag, 1981.

[Per94] C. Pertino. Degree thesis, Departamento de Informática, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, 1994. In preparation.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI-FN-19, Computer Science Department, University of Århus, 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantic and logic of the reactive systems. In W. Brauer, editor, *Proceedings 12th ICALP*, pages 15–32, Nafplion, 1985. LNCS 194, Springer-Verlag.