Nordic Journal of Computing 2(1995), 338–357.

SYNTHESIZING DISTINGUISHING FORMULAE FOR REAL TIME SYSTEMS

JENS CHR. GODSKESEN
Tele Danmark Research
Lyngsø Allé 2
DK–2970 Hørsholm, Denmark
jcg@tdr.dk

KIM G. LARSEN
Aalborg University
Fredrik Bajers Vej 7
9220 Aalborg, Denmark
kgl@iesd.auc.dk

Abstract. This paper describes a technique for generating diagnostic information for the *timed* bisimulation equivalence and the *timed* simulation preorder. More precisely, given two (parallel) networks of regular real–time processes, the technique will provide a logical formula that differentiates them in case they are not timed (bi)similar. Our method may be seen as an extension of the algorithm by Čerāns for deciding timed bisimilarity in that information of time–quantities has been added sufficient for generating distinguishing formulae.

CR Classification: D.2.2, D.2.4, F.1.2, F.3.1, F.4.3

Key words: tools and techniques, verification, process algebra, real time

1. Introduction

Research in the area of process algebras has created interest in *behavioural relations* as a tool for verifying correctness of processes [24, 19, 16, 3]. In this approach, specifications as well as implementations are formalized as process algebraic expressions, and verification consists of establishing a suitable behavioural relationship between an implementation and its specification. A number of equivalences has been proposed in the literature (for an overview see [31, 32]), and several automated tools support verification for finite–state systems based on such equivalences (e.g. [30, 9, 14, 29]).

However, for a tool to be of real assistance during a design process it is crucial that *diagnostic* information is offered in case of erroneous design as undoubtedly this troublesome situation will occur more frequently than not. Moreover, the diagnostic information offered must be useful in the subsequent debugging. For a variety of *(bi)simulation*[1] *equivalences* [27, 24, 33] the theoretical basis for generation of such diagnostic information is given in terms of a *logical* characterization of the equivalence: two systems are equivalent exactly when they satisfy the same formulas in a particular modal logic [18, 11]. Thus when two systems are found not to be equivalent,

[1] We use the abbreviation (bi)simulation instead of repeatedly writing simulation and bisimulation.

Received September 20, 1994; accepted June 30, 1995.

one may explain why by giving a formula satisfied by one but not the other. Algorithms for generating distinguishing formulae for finite–state systems has been described in [17, 8, 21, 4, 28] and implemented in at least two tools [14, 9].

During the last few years a number of real–time process algebras has been introduced in order to handle quantitative aspects of processes [10, 34, 25, 2, 7]. In addition a number of time–sensitive and time–abstracting (bi)simulation equivalences and preorders has been introduced and studied [34, 23, 12, 10]. Due to the use of the non–negative reals as time domain, even the simplest processes describe infinite state systems. Thus, decidability of (bi)similarity cannot be achieved using the standard algorithmic techniques for finite–state systems. However, decidability of (bi)simulation equivalence between networks of timed regular processes has recently been established by Čerāns [5] in the time–sensitive case and by Larsen and Wang [23] in the time–abstracted case. The underlying algorithmic techniques has later been implemented in the automatic verification tool EPSILON [6] and applied in the analysis of a simple protocol [13] and a steam generator [20].

Our goal in this paper is to describe the theoretical basis for the method used in EPSILON for generating diagnostic information for the timed (bi)simulation equivalence. The diagnostic information relies on a logical characterization of timed (bi)simulation equivalence using a real–timed version of the well known Hennessy–Milner Logic [18]. Thus in case of two non–(bi)similar processes, the diagnostic information will consist of a distinguishing formula. Our method may be seen as an extension of Čerāns' algorithm for deciding timed bisimilarity to which information of time–quantities has been added sufficient (but no more) for generating distinguishing formulae. The presentation we give of Čerāns's algorithm differs from the one given in [5] and hence may be considered a contribution in itself.

The remainder of this paper is organized as follows. The next section introduces the syntax and operational semantics of a simplified version of Wang's real–time process algebra TCCS [34]. Also, the section defines timed (bi)similarity and examines the connection between it and a real–time version of Hennessy–Milner Logic. Section 3 contains a much simplified account of Čerāns' algorithm for the (simplified) class of TCCS processes. Thus, the section introduces the notion of *symbolic* process providing a finite–state representation of real–time processes, and the notion of symbolic (bi)simulation providing the basis for an algorithm deciding timed (bi)similarity. Section 4 introduces the notions of *pointed* symbolic process and *pointed* symbolic (bi)similarity providing a representation of real–time processes with explicit information of certain time–quantities yet sufficiently finitary that it provides the basis for an algorithm generating distinguishing formulae. Section 5 explains how the technique can be extended to all of TCCS. The final section contains our concluding remarks.

2. Timed Processes

Our language for timed processes is based on the real–time calculus TCCS [34] by Wang. In particular, we apply the *two–phase functioning principle* outlined in [26], which in short means that the behaviour of a system is regarded as being split in two alternating phases: one where all the components of the system agree to let time progress and one where (some of) the components of the system compute. In order to explain our algorithmic ideas most clearly we have made certain simplifications in comparison to the calculus by Wang. In particular, we assume that all actions are observable, hence avoiding the issue of maximal progress. For the same reason, the parallel operator is simplified to that of pure interleaving without any synchronization on actions. Also, we assume recursive definitions of regular processes to be in simple sum normal form — a form into which all regular process expressions of TCCS can easily be transformed. Finally, we assume all processes to initially have integral delays. [2] Despite all the simplifications we impose in order to obtain simplicity and clarity the algorithmic ideas presented extend to the full calculus of TCCS. This will be explained in Section 5.

2.1 Syntax and Semantics

(Initial Integral) Regular Processes

Let $\mathcal{A}$ be a fixed set of actions ranged over by $a, b, c, \ldots$. We denote by $\mathcal{R}_{>0}$ the set of positive reals ranged over by $d, d_1, d_2, \ldots, d', d'', \ldots$. Similarly $\mathcal{R}_{\geq 0}$ denotes the set of non–negative reals ranged over by $e, e_1, e_2, \ldots, e', e'', \ldots$. NAT denotes the set of natural numbers (including 0), and finally, $\mathcal{D}$ denotes the set of delays $\{\epsilon(d) \mid d \in \mathcal{R}_{>0}\}$. We use $\sigma, \sigma', \ldots$ to range over elements of the set $\mathcal{A} \cup \mathcal{D}$.

Assume a *finite* set of process variables $\mathcal{V}$ and for each process variable X a defining equation of the following (normal) form:

$$X \stackrel{def}{=} \sum_{i=1}^{n} \epsilon(e_i).a_i.X_i \tag{1}$$

where $e_i \in$ NAT and $X_i \in \mathcal{V}$. We denote by Δ the set of recursive definitions and the process variables are called *initial integral regular processes* (IIR) [3].

Intuitively, the definition of X in Eq. (1) describes a (real–time) behaviour which after a delay of d will "offer" to its environment all actions a_i for which $e_i \leq d$ (thus e_i is the enabling time of the action a_i); if the environment

[2] However, as in the algorithm by Čerāns, our extended algorithm would apply also to processes having positive rational delays as we may simply scale time. This is further explained in [15].

[3] In the syntax for IIR processes we restrict ourselves to the use of integer delays. However, the semantic time domain is of course that of the positive reals, thus derivatives of IIR processes will not in general be IIR — hence the terminology *initial* integral.

"accepts" the offered action a_i, X may evolve to the behaviour determined by X_i by performing a_i.

Formally, the behaviour of an (initial integral) regular process described by an equation system Δ is given in terms of a transition system with transitions labelled by actions ($\mathcal{A}$) or delays ($\mathcal{D}$). First, let for $d \in \mathcal{R}_{>0}$, X^d denote the term:

$$\sum_{i=1}^{n} \epsilon(e_i \dot{-} d).a_i.X_i$$

assuming that X is defined as in Eq. (1). [4] Also, we let $X^0 = X$.

Given a set Δ of (recursive) definitions of the form Eq. (1), the following labelled transition system is induced:

$$\langle \mathbf{X}, \mathcal{A} \cup \mathcal{D}, \longrightarrow \rangle$$

where

$$\mathbf{X} = \{X^e \mid X \in \mathcal{V}, e \in \mathcal{R}_{\geq 0}\}$$

and $\longrightarrow \subseteq \mathbf{X} \times (\mathcal{A} \cup \mathcal{D}) \times \mathbf{X}$ is defined by the following two axioms assuming Eq. (1) is the defining equation for X:

$$X^e \xrightarrow{a_i} X_i \text{ when } e_i \leq e$$

$$X^e \xrightarrow{\epsilon(d)} X^{e+d}$$

We shall use $P, Q, \ldots$ to range over $\mathbf{X}$.

EXAMPLE 1. Consider the following equation system for the four variables Z, Z_b, Z_a and X: [5]

$$Z \stackrel{def}{=} \epsilon(1).a.Z_b + \epsilon(1).b.Z_a + b.X \qquad Z_b \stackrel{def}{=} b$$

$$Z_a \stackrel{def}{=} a \qquad X \stackrel{def}{=} \epsilon(1).a$$

Applying the operational semantics of regular processes yields the following transition sequence of Z:

$$Z \xrightarrow{\epsilon(\frac{1}{2})} \epsilon(\tfrac{1}{2}).a.Z_b + \epsilon(\tfrac{1}{2}).b.Z_a + b.X \xrightarrow{b} X \xrightarrow{\epsilon(\frac{1}{2})} \epsilon(\tfrac{1}{2}).a$$

Note that the final state ($\epsilon(\frac{1}{2}).a$) cannot perform an a–transition. □

The above defined transition system enjoys a number of useful properties stated below — for proofs and more information we refer the reader to [34, 35]. Note also the (uncountably) infinite state and dense nature of the transition system determined by an equation system Δ.

[4] $\dot{-}$ denotes *monus* on $\mathcal{R}_{\geq 0}$. That is for $e, e' \in \mathcal{R}_{\geq 0}$, $e \dot{-} e' = \max\{e - e', 0\}$.

[5] *nil* denotes the variable defined to be the empty sum — thus for all d, nil^d is the empty sum and the only transitions are delay transitions. We use the abbreviations $a.P$ for $\epsilon(0).a.P$, and a for $a.nil$.

PROPOSITION 1.

(1) *(Time Determinism) Whenever* $P \xrightarrow{\epsilon(d)} P'$ *and* $P \xrightarrow{\epsilon(d)} P''$ *then* $P' = P''$.

(2) *(Strong Persistency) If* $P \xrightarrow{a} Q$ *and* $P \xrightarrow{\epsilon(d)} P'$ *then* $P' \xrightarrow{a} Q$.

(3) *(Time Continuity) For all* d_1, d_2 *and* P'', $P \xrightarrow{\epsilon(d_1+d_2)} P''$ *iff* $P \xrightarrow{\epsilon(d_1)} P' \xrightarrow{\epsilon(d_2)} P''$ *for some* P'.

Networks

Assume Δ is a system of equations of the form Eq. (1) over the finite set of variables $\mathcal{V}$. Then a (parallel) *network* (over Δ and $\mathcal{V}$) is a term of the form:

$$\overline{X} = (X_1 \,|\, \ldots \,|\, X_n)$$

where $X_i \in \mathcal{V}$.

The set of n–ary networks induces a labelled transition system:

$$\mathcal{N}_n = \langle \mathbf{Net}_n, \mathcal{A} \cup \mathcal{D}, \longrightarrow \rangle$$

where

$$\mathbf{Net}_n = \left\{ (X_1^{e_1} \,|\, \ldots \,|\, X_n^{e_n}) \;\; | \, X_i \in \mathcal{V}, e_i \in \mathcal{R}_{\geq 0} \right\}$$

and $\longrightarrow \subseteq \mathbf{Net}_n \times (\mathcal{A} \cup \mathcal{D}) \times \mathbf{Net}_n$ is defined by the following axiom and inference rule:

$$(X_1^{e_1} \,|\, \ldots \,|\, X_n^{e_n}) \xrightarrow{\epsilon(d)} (X_1^{e_1+d} \,|\, \ldots \,|\, X_n^{e_n+d})$$

$$\frac{X_i^{e_i} \xrightarrow{a} X_i'}{(X_1^{e_1} \,|\, \ldots \,|\, X_i^{e_i} \,|\, \ldots \,|\, X_n^{e_n}) \xrightarrow{a} (X_1^{e_1} \,|\, \ldots \,|\, X_i' \,|\, \ldots \,|\, X_n^{e_n})}$$

Thus, in a network, the regular components synchronize on delay transitions and interleave on action transitions. We shall use $\overline{P}, \overline{Q}, \ldots$ to range over $\mathbf{Net}_n$ for arbitrary n.

EXAMPLE 2. Consider the following equation system for the two variables X and Y:

$$X \stackrel{def}{=} \epsilon(1).a \qquad Y \stackrel{def}{=} b$$

Applying the operational semantics for networks yields the following transition sequence of $X \,|\, Y$:

$$X \,|\, Y \xrightarrow{\epsilon(\frac{1}{2})} (\epsilon(\tfrac{1}{2}).a \,|\, b) \xrightarrow{b} (\epsilon(\tfrac{1}{2}).a \,|\, nil) \xrightarrow{\epsilon(\frac{1}{2})} (a \,|\, nil) \xrightarrow{a} (nil \,|\, nil)$$

□

The properties of time determinism and time continuity (see proposition 1) also hold for networks. However, the persistency property only holds in the following weaker form:

PROPOSITION 2.

(1) *(Persistency) If* $\overline{P} \xrightarrow{a} \overline{Q}$ *and* $\overline{P} \xrightarrow{\epsilon(d)} \overline{P}'$ *then* $\overline{P}' \xrightarrow{a} \overline{Q}'$ *for some* $\overline{Q}'$.

2.2 (Bi)simulation and Distinguishing Formulae

As networks semantically constitute labelled transition systems we may compare them with respect to a number of behavioural relations such as bisimilarity and similarity [24, 27]. Below we recall the definition of these classical notions:

DEFINITION 1. *Let $\mathcal{T} = \langle S, A, \longrightarrow \rangle$ be a labelled transition system. Then a simulation $\mathcal{S}$ is a binary relation on S such that whenever $p\mathcal{S}q$ and $a \in A$ then the following holds:*

$$\text{Whenever } p \xrightarrow{a} p' \text{ then } q \xrightarrow{a} q' \text{ for some } q' \text{ with } p'\mathcal{S}q'$$

We say that q is simulated by p (or p simulates q) whenever $p\mathcal{S}q$ for some simulation $\mathcal{S}$. We write $p \leq q$ in this case.
A binary relation $\mathcal{B}$ is a bisimulation if both $\mathcal{B}$ and $\mathcal{B}^{-1}$ are simulations. [6]
We say that p and q are bisimilar if $p\mathcal{B}q$ for some bisimulation $\mathcal{B}$ in which case we write $p \sim q$.

EXAMPLE 3. Consider the union of the equation systems from Examples 1 and 2 (note the agreement with respect to the definition of X). Then the (unique) execution sequences demonstrated in the two examples clearly proves that $X \,|\, Y \not\sim Z$ and $X \,|\, Y \not\leq Z$. [7] On the other hand it may be argued that $Z \leq X \,|\, Y$ by checking that the following collection of pairs constitutes a simulation:

$$\begin{aligned}
&\{(Z^e, X^e|Y^e) \mid e \in \mathcal{R}_{\geq 0}\} \\
&\quad \cup \quad \{({Z_b}^e, nil|Y^e) \mid e \in \mathcal{R}_{\geq 0}\} \\
&\quad \cup \quad \{({Z_a}^e, X^{e+1}|nil) \mid e \in \mathcal{R}_{\geq 0}\} \\
&\quad \cup \quad \{(X^e, X^{e'}|nil) \mid e' \leq e \in \mathcal{R}_{\geq 0}\} \\
&\quad \cup \quad \{(nil, nil|nil)\}
\end{aligned}$$

□

The above example illustrates two cases of non–(bi)similar networks. Ideally, an automatic verification tool should not only report this fact but also provide *explanations* as to why there is a lack of (bi)similarity. The well known Hennessy–Milner Logic [18], which is suitable for describing properties of process behaviour, provides the key to such explanations:

DEFINITION 2. *The formulas of Hennessy–Milner Logic, $\mathcal{M}$, are given by the following abstract syntax:*

$$F ::= \mathrm{tt} \mid F \wedge F \mid \neg F \mid \langle \sigma \rangle F$$

[6] $\mathcal{B}^{-1} = \{(q, p) \mid (p, q) \in \mathcal{B}\}$.

[7] Strictly speaking we are comparing networks from two different transition systems, namely: $\mathbf{Net}_1$ and $\mathbf{Net}_2$. However, the notion of (bi)similarity may be applied to states of different transition systems by first forming their (disjoint) union.

where σ ranges over actions and delays, i.e. the set $\mathcal{A} \cup \mathcal{D}$. Also, let $\mathcal{L}$ denotes the set of negation free formulae.

We interpret Hennessy–Milner Logic relative to the labelled transition system for networks, $\mathcal{N}_n$. I.e. we define a *satisfaction* relation $\models$ between networks ($\mathbf{Net}_n$) and formulae ($\mathcal{M}$). For propositional constructs the definition is straightforward, and for the modality, we define:

$$\overline{P} \models \langle\sigma\rangle F \;\;\Leftrightarrow\;\; \exists \overline{P}'.\overline{P} \xrightarrow{\sigma} \overline{P}' \wedge \overline{P}' \models F$$

Now let $\mathcal{M}(\overline{P})$ be the set of properties satisfied by $\overline{P}$. Also, let $\mathcal{L}(\overline{P})$ be the set of negation free properties satisfied by $\overline{P}$. Then the following characterization result [18] shows that Hennessy–Milner Logic can be applied for explanations:

THEOREM 1. *Let $\overline{P}$ and $\overline{Q}$ be networks. Then $\overline{P} \sim \overline{Q}$ if and only if $\mathcal{M}(\overline{P}) = \mathcal{M}(\overline{Q})$. Also, $\overline{P} \leq \overline{Q}$ if and only if $\mathcal{L}(\overline{P}) \subseteq \mathcal{L}(\overline{Q})$.* [8] $\square$

EXAMPLE 4. Consider the networks $X \,|\, Y$ and Z from Example 3. The facts that $X \,|\, Y \not\sim Z$ and $X \,|\, Y \not\leq Z$ are both "explained" by the formula $\langle\epsilon(\frac{1}{2})\rangle\langle b\rangle\langle\epsilon(\frac{1}{2})\rangle\langle a\rangle\mathrm{tt}$, which is satisfied by $X \,|\, Y$ but not by Z. $\square$

3. Symbolic Processes

It is obvious that the standard semantics of networks is infinitary (even under quotient with bisimilarity and similarity); thus decidability of bisimilarity and similarity between networks is beyond the standard algorithmic techniques for finite state systems. However, in [5] Čerāns presented a very clever algorithm for deciding timed bisimilarity for a class of parallel timer processes using the region–graph technique devised by Alur and Dill [1]. In the following we shall give a much simplified presentation of Čerāns' algorithm for the very simple type of networks we are dealing with in this paper. Then in the next section we shall show how to extend the algorithm in order that distinguishing formulae may be generated.

3.1 Symbolic States

Given two networks:

$$\overline{X} = (X_1 \,|\, \ldots \,|\, X_m) \quad \text{and} \quad \overline{Y} = (Y_1 \,|\, \ldots \,|\, Y_n)$$

over some equation system Δ and variables $\mathcal{V}$ their bisimilarity (or similarity) will be reduced to deciding a suitable property of an induced joint

[8] The theorem assumes that the labelled transition system satisfies a technical condition called *image–finiteness*; i.e. for all $\overline{P}$ and σ the set $\{\overline{Q} \mid \overline{P} \xrightarrow{\sigma} \overline{Q}\}$ is finite. However, this condition is met by networks due to the property of Time Determinism and because every component of a network has only finitely many transitions.

finite–state symbolic transition system, in which states represent *sets* of pairs of networks.

A *symbolic state* (of arity (m, n)) over Δ and $\mathcal{V}$ is a finite list:

$$\phi = \Big[(M_1, N_1), \ldots, (M_k, N_k)\Big]$$

where M_i and N_i are multisets of $\{X^l \mid X \in \mathcal{V},\, l \in \mathrm{N{\scriptsize AT}}\}$ with $|\uplus_i M_i| = m$ and $|\uplus_i N_i| = n$, and with $M_i \uplus N_i \neq \emptyset$ for $i > 1$. [9] ϕ is said to be *interior* in case $M_1 \uplus N_1 = \emptyset$ and *boundary* otherwise.

A symbolic state $\phi = [(M_1, N_1), \ldots, (M_k, N_k)]$ *represents* a whole family of pairs of networks. More precisely, all pairs of the form:

$$\Big((M_1^{v_1} \,|\, \ldots \,|\, M_k^{v_k})\ ,\ (N_1^{v_1} \,|\, \ldots \,|\, N_k^{v_k})\Big) \tag{2}$$

where $0 = v_1 < v_2 < \cdots < v_k < 1$, and where for a multiset $M = \{X_1, \ldots X_j\}$ and $e \in \mathcal{R}_{\geq 0}$, $M^e = (X_1^e \,|\, \ldots \,|\, X_j^e)$.

Now, call $\overline{v} = (v_1, \ldots, v_k) \in \mathcal{R}^k_{\geq 0}$ *well–ordered and fractional* (W) if $0 = v_1 < v_2 < \cdots < v_k < 1$. Then for $\phi = [(M_1, N_1), \ldots, (M_k, N_k)]$ a symbolic state and $\overline{v} = (v_1, \ldots, v_k)$ a well–ordered and fractional tuple, $\phi(\overline{v}) = (\phi_1(\overline{v}), \phi_2(\overline{v}))$ denotes the pair in Eq. (2). [10] Thus, the set of pairs represented by ϕ is

$$\|\phi\| = \{\phi(\overline{v}) \mid \overline{v} \in W\}$$

As all delay–constants in the definitions of variables are integers it follows that $\phi_1(\overline{v})$ and $\phi_1(\overline{v}')$ ($\phi_2(\overline{v})$ and $\phi_2(\overline{v}')$) are able to perform precisely the same actions for $\overline{v}$ and $\overline{v}'$ being arbitrary well-ordered and fractional tuples.

EXAMPLE 5. Consider once more the union of the equation systems from Examples 1 and 2. Then the symbolic state $[(\{X, Y\}, \{Z\})]$ represents the single pair of network $(X|Y,\, Z)$. The symbolic state $[(\emptyset, \emptyset),\ (\{X, Y\}, \{Z\})]$ represents all pairs of the form $(\epsilon(1-d).a|b,\ \ \epsilon(1-d).a.Z_b + \epsilon(1-d).b.Z_a + b.X)$, where $0 < d < 1$. □

We let $\mathrm{SS}_{m,n}$ denote the set of symbolic states (of arity (m, n)) over Δ and $\mathcal{V}$. Assuming that X is defined using Eq. (1) it is clear that the set $\{X^l \,|\, l \in \mathrm{N{\scriptsize AT}}\}$ is finite as $X^l = X^{l'}$ for $l, l' \geq \max\{e_i \,|\, i = 1 \ldots n\}$ (intuitively X^l will at some point become invariant under delay–transitions). Hence, it may be concluded that the set of symbolic states $\mathrm{SS}_{m,n}$ is indeed finite.

Moreover, it will follow from the following sections, that the pairs represented by a symbolic state ϕ are either all bisimilar or all non–bisimilar; i.e.:

$$\forall \overline{v} \in W.\, \phi_1(\overline{v}) \sim \phi_2(\overline{v}) \quad \text{or} \quad \forall \overline{v} \in W.\, \phi_1(\overline{v}) \not\sim \phi_2(\overline{v})$$

[9] $\uplus$ denotes disjoint union.

[10] We take advantage of the fact that $|$ is commutative and associative wrt. bisimilarity and similarity.

What remains is to determine which symbolic states represent bisimilar pairs. In particular, in order to determine bisimilarity between two networks

$$\overline{X} = (X_1 \,|\, \ldots \,|\, X_m) \text{ and } \overline{Y} = (Y_1 \,|\, \ldots \,|\, Y_n)$$

we will apply the method to be presented to the following (initial) symbolic state:

$$\phi_{\overline{X},\overline{Y}} = \big[(\{X_1, \ldots, X_m\}\ ,\ \{Y_1, \ldots, Y_n\})\big]$$

3.2 Symbolic Semantics

In order to determine which symbolic states represent bisimilar pairs we provide in the following a symbolic semantics for $\mathrm{SS}_{m,n}$. Let $\phi = [(M_1, N_1), \ldots, (M_k, N_k)]$ be a symbolic state (of arity (m, n)), then $\longmapsto_1$, $\longmapsto_2$ and $\overset{w}{\longmapsto}$ are defined by the rules below:

$$\frac{X \xrightarrow{a} X'}{\phi \overset{a}{\longmapsto}_1 \langle\!\langle (\{X'\} \uplus M_1{}', N_1), \ldots, (M_k{}', N_k) \rangle\!\rangle} \quad \begin{array}{l} M_i = M_i{}' \uplus \{X\} \\ M_j = M_j{}'\ (i \neq j) \end{array}$$

$$\frac{Y \xrightarrow{a} Y'}{\phi \overset{a}{\longmapsto}_2 \langle\!\langle (M_1, \{Y'\} \uplus N_1{}'), \ldots, (M_k, N_k{}') \rangle\!\rangle} \quad \begin{array}{l} N_i = N_i{}' \uplus \{Y\} \\ N_j = N_j{}'\ (i \neq j) \end{array}$$

$$\phi \overset{w}{\longmapsto} [(\emptyset, \emptyset), (M_1, N_1), \ldots, (M_k, N_k)] \quad \phi \text{ is boundary}$$

$$\phi \overset{w}{\longmapsto} \langle\!\langle (M_k{}^{-1}, N_k{}^{-1}), (M_1, N_1), \ldots, (M_{k-1}, N_{k-1}) \rangle\!\rangle \quad \phi \text{ is interior}$$

where $\langle\!\langle \ldots \rangle\!\rangle$ denotes the list resulting from removing all pairs $(\emptyset, \emptyset)$ from the original list (in order to ensure that the result of the transition is a symbolic state); we denote by $\overset{a}{\hookrightarrow}_1$ and $\overset{a}{\hookrightarrow}_2$ the two transition relations which are defined using the rules for $\overset{a}{\longmapsto}_1$ and $\overset{a}{\longmapsto}_2$ except that pairs of empty sets are *not* removed in the resulting "symbolic" state; finally, for M a multiset and $d \in \mathcal{R}_{>0}$, $M^{-d} = \{X^{n+d} \mid X^n \in M\}$.

EXAMPLE 6. Recall the definitions of Examples 1 and 2. Fig. 1 illustrates (part of) the symbolic transition system for the initial symbolic state induced by the network pair $X \,|\, Y$ and Z. Symbolic states are indicated by boxes with symbolic state tuples occurring just below their associated box. For additional information we have indicated inside the boxes the families of network pairs represented by the symbolic state. □

The close relationship between the (symbolic) semantics of a symbolic state and the (standard) semantics of the (pairs of) networks it represents are as follows:

LEMMA 1. (CORRESPONDENCE)

(1) Whenever $\phi \overset{a}{\longmapsto}_1 \phi'$ and $(\overline{P}, \overline{Q}) \in \|\phi\|$, then for some $\overline{P}'$, $\overline{P} \xrightarrow{a} \overline{P}'$ such that $(\overline{P}', \overline{Q}) \in \|\phi'\|$;

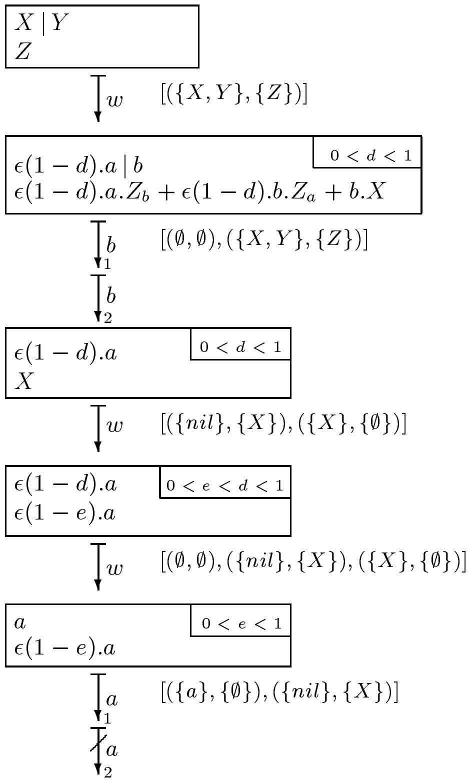

Fig. 1: Part of the symbolic transition system induced by $X \mid Y$ and Z.

(2) *Whenever* $(\overline{P},\overline{Q}) \in \|\phi\|$ *and* $\overline{P} \stackrel{a}{\longrightarrow} \overline{P}'$ *then for some* ϕ', $\phi \stackrel{a}{\longmapsto}_1 \phi'$ *with* $(\overline{P}',\overline{Q}) \in \|\phi'\|$;

(3) *Whenever* $\phi \stackrel{w}{\longmapsto} \phi'$ *and* $(\overline{P},\overline{Q}) \in \|\phi\|$ *then for some* $d \in \mathcal{R}_{>0}$, $(\overline{P}^d,\overline{Q}^d) \in \|\phi'\|$ [11];

(4) *Whenever* $(\overline{P},\overline{Q}) \in \|\phi\|$ *then for all* $d \in \mathcal{R}_{>0}$ *there exists* ϕ' *such that* $\phi(\stackrel{w}{\longmapsto})^*\phi'$ *with* $(\overline{P}^d,\overline{Q}^d) \in \|\phi'\|$;

Obviously, Lemma 1 *1* and *2* may be dualised with $\longmapsto_2$ replacing $\longmapsto_1$.

3.3 Symbolic Bisimulation

We may now define the notion of symbolic simulation and bisimulation:

DEFINITION 3. $\mathcal{B} \subseteq SS_{m,n}$ *is a symbolic simulation if whenever* $\phi \in \mathcal{B}$ *and* $a \in \mathcal{A}$ *the following holds:*

[11] Here $\overline{P}^d$ denotes the unique network such that $\overline{P} \stackrel{\epsilon(d)}{\longrightarrow} \overline{P}^d$.

(1) Whenever $\phi \overset{a}{\longmapsto}_1 \phi'$ *then* $\phi' \overset{a}{\longmapsto}_2 \phi''$ *for some* ϕ'' *with* $\phi'' \in \mathcal{B}$,

(2) Whenever $\phi \overset{w}{\longmapsto} \phi'$ *then* $\phi' \in \mathcal{B}$

If in addition $\mathcal{B}$ *satisfies the following condition:*

(3) Whenever $\phi \overset{a}{\longmapsto}_2 \phi'$ *then* $\phi' \overset{a}{\longmapsto}_1 \phi''$ *for some* ϕ'' *with* $\phi'' \in \mathcal{B}$

$\mathcal{B}$ *is a symbolic bisimulation. We write* $\phi \in \leq_s$ *whenever* ϕ *is contained in some symbolic simulation. Similarly we write* $\phi \in \sim_s$ *whenever* ϕ *is contained in some symbolic bisimulation.*

As $\mathrm{SS}_{m,n}$ is finite–state and both $\leq_s$ and $\sim_s$ are maximal fixedpoints of simple monotonic functions on sets of symbolic states (which constitutes a complete finite lattice with respect to set inclusion) it follows from standard techniques (iterative, local, on–the–fly, etc.) that questions of the form $\phi \in \leq_s$ and $\phi \in \sim_s$ are decidable. Moreover, using the previous Correspondence Theorem 1 the following close relationships between (ordinary) (bi)simulation and the symbolic counterparts may be easily established:

THEOREM 2. *Let* $\ll$ *be either* $\leq$ *or* $\sim$. *Then whenever* $\overline{P} \ll \overline{Q}$ *and* $(\overline{P}, \overline{Q}) \in \|\phi\|$ *then* $\phi \in \ll_s$. *Also, whenever* $\phi \in \ll_s$ *and* $(\overline{P}, \overline{Q}) \in \|\phi\|$ *then* $\overline{P} \ll \overline{Q}$.

It follows that in order to decide whether two networks $\overline{X}$ and $\overline{Y}$ are (bi)similar we may alternatively decide whether the initial symbolic state $\phi_{\overline{X},\overline{Y}}$ is a member of the symbolic (bi)simulation.

EXAMPLE 7. Using the definition of symbolic (bi)similarity it follows clearly from Fig. 1 and Theorem 2 that $X \,|\, Y \not\sim Z$ and $X \,|\, Y \not\leq Z$. □

4. Pointed Symbolic Processes

The previous section provides a finite symbolic semantics based on which (bi)similarity between networks can be decided. However, the semantics has completely abstracted away from all time–quantities, and it is not possible to synthesize distinguishing formulae (and in particular not the time–quantities of modalities) based on this semantics. Therefore, we provide in this section a more informative yet still (sufficiently) finitary *pointed* symbolic semantics with explicit information of time–quantities. We show that this semantics provides the basis for an algorithm constructing distinguishing formulae (an algorithm which in fact has been implemented in the EPSILON tool [6]).

4.1 Pointed Symbolic States and Semantics

A *pointed symbolic state* (of arity (m, n) and over Δ and $\mathcal{V}$) is a pair $\langle \phi, \overline{v} \rangle$, where ϕ is a symbolic state and $\overline{v}$ is a (corresponding) well–ordered fractional. We denote by $\mathrm{PSS}_{m,n}$ the set of all pointed symbolic states (of arity (m, n) and over Δ and $\mathcal{V}$). Observe that $\mathrm{PSS}_{m,n}$ in contrast to $\mathrm{SS}_{m,n}$ is infinite (in fact uncountable).

A pointed symbolic state $\langle \phi, \overline{v} \rangle$ *represents* both the family of network pairs represented by ϕ, i.e. $\|\phi\|$, as well as the particular network pair pointed out by $\overline{v}$, i.e. $\phi(\overline{v})$.

The symbolic semantics of $\mathrm{PSS}_{m,n}$ refines that of $\mathrm{SS}_{m,n}$ in that symbolic w–transitions are parameterized explicitly with time–quantities in order to capture precisely the corresponding delay transition between the represented networks.

Before giving the symbolic semantics of $\mathrm{PSS}_{m,n}$ we need some notation for well–ordered and fractional tuples: for $\overline{v} = (v_1, \ldots, v_k) \in W$ we define $\overline{v}^* = (1 - v_k)$ and

$$\mathrm{next}_B(\overline{v}) = (0, v_1 + (\tfrac{\overline{v}^*}{2}), \ldots, v_k + (\tfrac{\overline{v}^*}{2}))$$

$$\mathrm{next}_I(\overline{v}) = (0, v_1 + \overline{v}^*, \ldots, v_{k-1} + \overline{v}^*)$$

Note that the tuples $\mathrm{next}_B(\overline{v})$ and $\mathrm{next}_I(\overline{v})$ are well–ordered and fractional.

Now let $\langle \phi, \overline{v} \rangle \in \mathrm{PSS}_{m,n}$. Then the $\Longrightarrow_1$, $\Longrightarrow_2$ and $\overset{w(d)}{\Longrightarrow}$ transition relations are defined by the rules below:

$$\frac{\phi \overset{a}{\hookrightarrow}_1 \phi'}{\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_1 \langle\!\langle \phi', \overline{v} \rangle\!\rangle}$$

$$\frac{\phi \overset{a}{\hookrightarrow}_2 \phi'}{\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_2 \langle\!\langle \phi', \overline{v} \rangle\!\rangle}$$

$$\frac{\phi \overset{w}{\longmapsto} \phi'}{\langle \phi, \overline{v} \rangle \overset{w(\overline{v}^*/2)}{\Longrightarrow} \langle \phi', \mathrm{next}_B(\overline{v}) \rangle} \quad \phi \text{ is boundary}$$

$$\frac{\phi \overset{w}{\longmapsto} \phi'}{\langle \phi, \overline{v} \rangle \overset{w(\overline{v}^*)}{\Longrightarrow} \langle \phi', \mathrm{next}_I(\overline{v}) \rangle} \quad \phi \text{ is interior}$$

where $\langle\!\langle \phi, \overline{v} \rangle\!\rangle$ denotes the pointed symbolic state resulting from removing all pairs $(\emptyset, \emptyset)$ from ϕ and at the same time removing the corresponding component v_i from $\overline{v}$.

EXAMPLE 8. Fig. 2 illustrates (part) of the pointed symbolic transition system for the initial pointed symbolic state induced by the network pair $X \mid Y$ and Z. Pointed symbolic states are indicated by boxes with pointed symbolic state pairs occurring just below their associated box. For additional information we indicate inside a box the network pair represented by the pointed symbolic state. □

Though $\mathrm{PSS}_{m,n}$ is an infinite set the above symbolic semantics is finitary in the following important sense: the total set of immediate $\Longrightarrow$–derivatives of any pointed symbolic state $\langle \phi, \overline{v} \rangle$ is finite! In particular, for each pair $\langle \phi, \overline{v} \rangle$ there is a unique d, ϕ' and $\overline{v}'$ for which $\langle \phi, \overline{v} \rangle \overset{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle$ (in contrast to the standard semantics) [12]. As we shall see this will enable the pointed symbolic semantics to be used for algorithmically generating distinguishing formulae.

The following Correspondence Lemma shows that the pointed symbolic semantics indeed is an extension of the symbolic semantics from the previous section:

LEMMA 2. (CORRESPONDENCE)

(1) Whenever $(\overline{P}, \overline{Q}) = \phi(\overline{v})$ *and* $\overline{P} \xrightarrow{a} \overline{P}'$ *then for some* ϕ' *and* $\overline{v}'$, $\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_1 \langle \phi', \overline{v}' \rangle$ *with* $(\overline{P}', \overline{Q}) = \phi'(\overline{v}')$.

(2) Whenever $\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_1 \langle \phi', \overline{v}' \rangle$ *then* $\phi_1(\overline{v}) \xrightarrow{a} \phi_1'(\overline{v}')$ *and* $\phi_2(\overline{v}) = \phi_2'(\overline{v}')$.

(3) Whenever $\phi \overset{w}{\longmapsto} \phi'$ *then for all* $\overline{v}$, $\langle \phi, \overline{v} \rangle \overset{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle$ *for some* $\overline{v}'$ *and* d.

(4) Whenever $\langle \phi, \overline{v} \rangle \overset{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle$ *then* $\phi_1(\overline{v}) \xrightarrow{\epsilon(d)} \phi_1'(\overline{v}')$, $\phi_2(\overline{v}) \xrightarrow{\epsilon(d)} \phi_2'(\overline{v}')$ *and* $\phi \overset{w}{\longmapsto} \phi'$.

Obviously, clause *1* and *2* have analogous counterparts for $\Longrightarrow_2$.

4.2 Pointed Symbolic Bisimulation

DEFINITION 4. $\mathcal{B} \subseteq PSS_{m,n}$ *is a* pointed symbolic simulation *if whenever* $\langle \phi, \overline{v} \rangle \in \mathcal{B}$, $a \in \mathcal{A}$ *and* $d \in \mathcal{R}_{>0}$ *the following holds:*

(1) Whenever $\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_1 \langle \phi', \overline{v}' \rangle$ *then* $\langle \phi', \overline{v}' \rangle \overset{a}{\Longrightarrow}_2 \langle \phi'', \overline{v}'' \rangle$ *for some* ϕ'' *and* $\overline{v}''$ *with* $\langle \phi'', \overline{v}'' \rangle \in \mathcal{B}$,

(2) Whenever $\langle \phi, \overline{v} \rangle \overset{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle$ *then* $\langle \phi', \overline{v}' \rangle \in \mathcal{B}$

If in addition $\mathcal{B}$ *satisfies the following condition:*

(3) Whenever $\langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_2 \langle \phi', \overline{v}' \rangle$ *then* $\langle \phi', \overline{v}' \rangle \overset{a}{\Longrightarrow}_1 \langle \phi'', \overline{v}'' \rangle$ *for some* ϕ'' *and* $\overline{v}''$ *with* $\langle \phi'', \overline{v}'' \rangle \in \mathcal{B}$

$\mathcal{B}$ *is a* pointed symbolic bisimulation. *We write* $\langle \phi, \overline{v} \rangle \in \leq_{ps}$ *whenever* $\langle \phi, \overline{v} \rangle$ *is contained in some pointed symbolic simulation. Similarly we write* $\langle \phi, \overline{v} \rangle \in \sim_{ps}$ *whenever* $\langle \phi, \overline{v} \rangle$ *is contained in some pointed symbolic bisimulation.*

For $\mathcal{B} \subseteq \mathrm{SS}_{m,n}$, let $\mathcal{B}^{\uparrow} = \{\langle \phi, \overline{v} \rangle \mid \phi \in \mathcal{B}\}$. Dually, for $\mathcal{B} \subseteq \mathrm{PSS}_{m,n}$ let $\mathcal{B}^{\downarrow} = \{\phi \mid \langle \phi, \overline{v} \rangle \in \mathcal{B}\}$.

LEMMA 3. *Whenever* $\mathcal{B}$ *is a symbolic (bi)simulation then* $\mathcal{B}^{\uparrow}$ *is a pointed symbolic (bi)simulation. Dually, whenever* $\mathcal{B}$ *is a pointed symbolic (bi)simulation then* $\mathcal{B}^{\downarrow}$ *is a symbolic (bi)simulation.* □

[12] The set of derivatives of $\langle \phi, \overline{v} \rangle$ is the union of the sets $\{\langle \phi', \overline{v}' \rangle \mid \exists a. \exists i. \langle \phi, \overline{v} \rangle \overset{a}{\Longrightarrow}_i \langle \phi', \overline{v}' \rangle\}$ and $\{\langle \phi', \overline{v}' \rangle \mid \exists d \in \mathcal{R}_{>0}. \langle \phi, \overline{v} \rangle \overset{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle\}$.

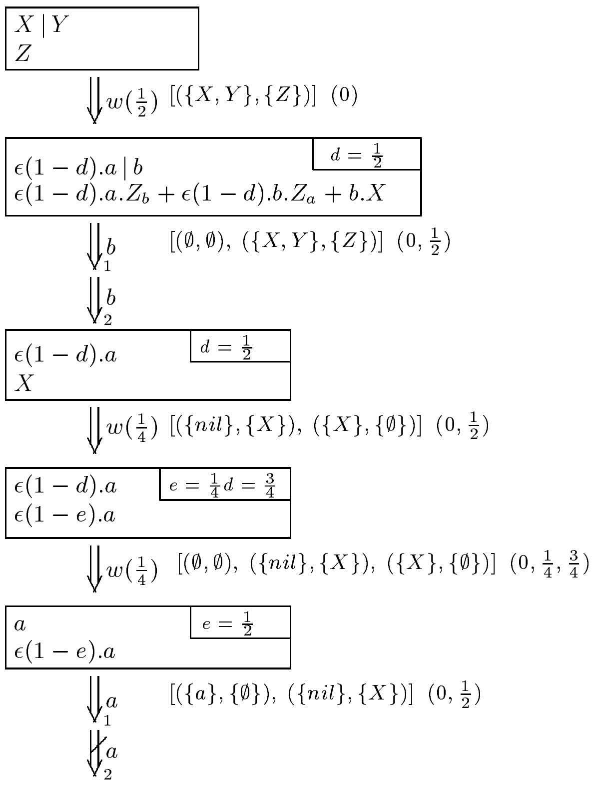

Fig. 2: Part of the *pointed* symbolic transition system induced by $X \mid Y$ and Z.

This leads directly to the following using the existing result of Theorem 2:

THEOREM 3. *$\langle \phi, \overline{v} \rangle \in \leq_{ps}$ if and only if $\phi_1(\overline{v}) \leq \phi_2(\overline{v})$. Also, $\langle \phi, \overline{v} \rangle \in \sim_{ps}$ if and only if $\phi_1(\overline{v}) \sim \phi_2(\overline{v})$.* $\square$

Thus, two networks $\overline{X}$ and $\overline{Y}$ are (bi)similar just in case the initial pointed symbolic state $\langle \phi_{\overline{X},\overline{Y}}, (0) \rangle$ is a member of the pointed symbolic (bi)simulation. However, due to the infinite nature of $\mathrm{PSS}_{m,n}$ this does not directly lead to an alternative algorithm for (bi)similarity, a problem we will deal with in the following subsection.

4.3 Computing Distinguishing Formulae

In standard fashion we may define for n a natural number the n'th approximations of the various (pointed) symbolic relations ($\leq_s$, $\sim_s$, $\leq_{ps}$ and $\sim_{ps}$). The 0'th approximate is simply the entire set of (pointed) symbolic states, and the $(n+1)$'th approximate is defined using the definition schemas of Definitions 3 and 4 with the n'th approximate substituting B in the (relevant) defining clauses. For $\equiv$ one of the above four relations the corresponding n'th approximate will be denoted $\equiv^n$. It is easy to see that in all four cases $\equiv^n$ will be decreasing in n. Also, in all four cases $\equiv^n$ approximates $\equiv$ in the sense that $\equiv$ equals the intersection of all its approximations (i.e. $\equiv = \bigcap_{n\in\omega} \equiv^n$).

As the set of symbolic states is finite, there exists a K (being simply the number of symbolic states of the given arity) such that $\leq_s^n$ and $\sim_s^n$ will be constant when n exceeds K. This fact leads directly to an iterative algorithm for deciding $\leq_s$ and $\sim_s$ and hence — due to Theorem 2 — for deciding $\leq$ and $\sim$. In contrast, the set of pointed symbolic states is infinite and there is therefore no a priori guarantee that the decreasing sequences $\langle\leq_{ps}^n\rangle_n$ and $\langle\sim_{ps}^n\rangle_n$ will converge at any finite stage. However, finite convergence is ensured by the following Lemma relating symbolic and pointed symbolic approximates:

LEMMA 4. *For all* n, $(\leq_{ps}^n)^\downarrow = \leq_s^n$ *and* $(\leq_s^n)^\uparrow = \leq_{ps}^n$. *Also, for all* n, $(\sim_{ps}^n)^\downarrow = \sim_s^n$ *and* $(\sim_s^n)^\uparrow = \sim_{ps}^n$.

Thus the pointed symbolic approximates converges at the same iteration as the symbolic counterparts. Furthermore, as any pointed symbolic state has only a finite (and computable) set of immediate derivatives, it follows that $\leq_{ps}^n$ and $\sim_{ps}^n$ are decidable for any n. Hence (bi)similarity between networks may alternatively be computed by deciding $\leq_{ps}^K$– ($\sim_{ps}^K$–) membership problems for a sufficiently large K.

In addition, the use of *pointed* symbolic (bi)simulation enables the generation of distinguishing formulae in cases of non–(bi)similarity as stated by the following theorem. The proof given is constructive and constitutes the generation algorithm.

THEOREM 4.

(1) Whenever $\langle\phi,\overline{v}\rangle \notin \leq_{ps}$ *then* $\phi_1(\overline{v}) \models F$ *and* $\phi_2(\overline{v}) \not\models F$ *for some* $F \in \mathcal{L}$.

(2) Whenever $\langle\phi,\overline{v}\rangle \notin \sim_{ps}$ *then* $\phi_1(\overline{v}) \models F$ *and* $\phi_2(\overline{v}) \not\models F$ *for some* $F \in \mathcal{M}$.

PROOF. We only give the proof of *2* (being slightly more involved). We proceed by induction in n, where $\langle\phi,\overline{v}\rangle \notin \sim_{ps}^n$.

BASIS: As $\langle\phi,\overline{v}\rangle \in \sim_{ps}^0$ no distinguishing formula is required.

INDUCTION STEP: Assume $\langle\phi,\overline{v}\rangle \notin \sim_{ps}^{n+1}$. There are three cases to consider depending on which of the conditions for membership of $\sim_{ps}^{n+1}$ that fails:

Case 1: Assume $\langle\phi,\overline{v}\rangle \overset{a}{\Longrightarrow}_1 \langle\phi',\overline{v}'\rangle$ but whenever $\langle\phi',\overline{v}'\rangle \overset{a}{\Longrightarrow}_2 \langle\phi^j,\overline{v}^j\rangle$ ($j = 1\ldots m$) then $\langle\phi^j,\overline{v}^j\rangle \notin \sim_{ps}^n$. Now, using the Induction Hypothesis we may conclude that there exists some formula F_j such that ${\phi^j}_1(\overline{v}^j) \models F_j$ but ${\phi^j}_2(\overline{v}^j) \not\models F_j$. Now, let $F = \langle a\rangle(F_1 \wedge \ldots \wedge F_m)$ then it follows easily using the Correspondence Lemma 2 that $\phi_1(\overline{v}) \models F$ whereas $\phi_2(\overline{v}) \not\models F$.

Case 2: Assume $\langle \phi, \overline{v} \rangle \stackrel{w(d)}{\Longrightarrow} \langle \phi', \overline{v}' \rangle$ but $\langle \phi', \overline{v}' \rangle \not\in \sim_{ps}^{n}$. Now, using the Induction Hypothesis we may conclude that there exists a formula F' such that $\phi_1'(\overline{v}') \models F'$ but $\phi_2'(\overline{v}') \not\models F'$. Now let $F = \langle \epsilon(d) \rangle F'$, then it follows from the Correspondence Lemma 2 that $\phi_1(\overline{v}) \models F$ whereas $\phi_2(\overline{v}) \not\models F$.

Case 3: Assume $\langle \phi, \overline{v} \rangle \stackrel{a}{\Longrightarrow}_2 \langle \phi', \overline{v}' \rangle$ but whenever $\langle \phi', \overline{v}' \rangle \stackrel{a}{\Longrightarrow}_1 \langle \phi^i, \overline{v}^i \rangle$ $(i = 1 \ldots l)$ then $\langle \phi^i, \overline{v}^i \rangle \not\in \sim_{ps}^{n}$. Now, using the Induction Hypothesis we may conclude that there exists some formula F_i such that ${\phi^i}_1(\overline{v}^i) \models F_i$ but ${\phi^i}_2(\overline{v}^i) \not\models F_i$. Now, let $F = \neg \langle a \rangle (\neg F_1 \wedge \ldots \wedge \neg F_l)$ then it follows easily using the Correspondence Lemma 2 that $\phi_1(\overline{v}) \models F$ whereas $\phi_2(\overline{v}) \not\models F$. □

EXAMPLE 9. Applying the construction of the above Theorem 4 to the pointed symbolic transition system induced by $X \mid Y$ and Z (partially shown in Fig. 2) we may generate the formula $\langle \epsilon(\frac{1}{2}) \rangle \langle b \rangle \langle \epsilon(\frac{1}{4}) \rangle \langle \epsilon(\frac{1}{4}) \rangle \langle a \rangle \mathrm{tt}$ distinguishing $X \mid Y$ and Z. □

The construction of Theorem 4 has been implemented in the verification tool EPSILON which provides automatic computation of distinguishing formulae for a variety of equivalences and refinements (in particular time–abstracting and observational ones). The implementation uses the (efficient) local correctness checking described in [22] yielding a polynomial complexity in the size of the (finite) number of pointed symbolic states to be considered and the (finite) set of transitions among those states. To give the reader some idea of the resulting implementation we offer below a very informal outline.

In checking the pointed symbolic (ps–) (bi)similarity of a pointed symbolic state $\langle \phi, \overline{v} \rangle$ the implementation makes use of two *data structures*: (1) a set B of symbolic states (assumed to be symbolic (bi)similar); and (2) a set N containing pointed symbolic states $\langle \phi', \overline{v}' \rangle$ paired with a formula F distinguishing the networks $\phi_1'(\overline{v}')$ and $\phi_2'(\overline{v}')$. Both B and N are initialized to the empty set $\emptyset$. Now, if either ϕ belongs to B or $\langle \phi, \overline{v} \rangle$ occurs in N the algorithm terminates immediately (with success in the first case and failure in the latter). Otherwise, the assumption set B is augmented with ϕ and the (three) two conditions for ps–(bi)similarity (Definition 4) are checked one by one. For each condition all the relevant transitions for $\langle \phi, \overline{v} \rangle$ are treated in turn. If all conditions turn out to be successful $\langle \phi, \overline{v} \rangle$ *is* indeed ps–(bi)similar relative to B and N; otherwise $\langle \phi, \overline{v} \rangle$ represents a non–(bi)similar pair of networks and is hence added to N together with a distinguishing formula obtained from the construction of the above proof — also the assumption set B is reset to its initial value (i.e. ϕ and all subsequently added symbolic states are (logically) removed from B).

5. Extension to Full TCCS

In this section we indicate how the techniques devised in the previous sections may be generalized in a straightforward manner to all of TCCS [35].

For a full explanation we refer the reader to [15].

In TCCS, as in CCS, it is assumed that $\mathcal{A} = \Delta \cup \overline{\Delta} \cup \{\tau\}$, where $\Delta \cup \overline{\Delta}$ is the set of *observable* actions and τ is a distinguished *unobservable* action denoting internal computation. The sets Δ and $\overline{\Delta}$ are isomorphic; for $a \in \Delta$ ($a \in \overline{\Delta}$) we denote by $\overline{a}$ the isomorphic action in $\overline{\Delta}$ (Δ). $\overline{a}$ is called the *complement* of a. Whenever $L \subseteq \Delta \cup \overline{\Delta}$, $\overline{L} = \{\overline{a} \mid a \in L\}$. As in CCS, complementary actions are used for synchronization between components in a network.

Now, a crucial design principle of TCCS is that of *maximal progress*; i.e. whenever a process can perform an internal action it should do so immediately without any further delay. Thus in addition to the properties of Proposition 1 we want the following to hold:

(4) (Maximal Progress) Whenever $P \xrightarrow{\tau}$ *then* $P \overset{\epsilon(d)}{\not\longrightarrow}$ *for all* $d \in \mathcal{R}_{>0}$.

To ensure maximal progress the semantics for delay transitions of *regular processes* must be changed slightly from that of Subsection 2.1:

$$X^e \xrightarrow{\epsilon(d)} X^{d+e} \quad \text{if } \ d + e \leq M(X)$$

where $M(X) = \min\{e_j | a_j = \tau\}$ is the maximal delay of X (i.e. the maximum X can be delayed before an internal action is enabled).

In TCCS *networks* are terms of the form $(X_1 | \ldots | X_n)\backslash L$, where L is a set of restricting actions (forcing synchronization). The semantic rules for action transitions are similar to those for CCS:

$$\frac{X_i^{e_i} \xrightarrow{a} X_i'}{(X_1^{e_1} \mid \ldots \mid X_i^{e_i} \mid \ldots \mid X_n^{e_n})\backslash L \xrightarrow{a} (X_1^{e_1} \mid \ldots \mid X_i' \mid \ldots \mid X_n^{e_n})\backslash L} \; a \notin L \cup \overline{L}$$

$$\frac{X_i^{e_i} \xrightarrow{a} X_i' \qquad X_j^{e_j} \xrightarrow{\overline{a}} X_j'}{(\ldots \mid X_i^{e_i} \mid \ldots \mid X_j^{e_j} \mid \ldots)\backslash L \xrightarrow{\tau} (\ldots \mid X_i' \mid \ldots \mid X_j' \mid \ldots)\backslash L}$$

In order that maximal progress also holds for networks the rule for delay transitions must ensure that no pair of components can synchronize during a delay. This is achieved by the following rule:

$$\frac{\{X_i^{e_i} \xrightarrow{\epsilon(d)} X_i{}'\}_{i=1,\ldots,n}}{(X_1^{e_1} \mid \ldots \mid X_n^{e_n})\backslash L \xrightarrow{\epsilon(d)} (X_1{}' \mid \ldots \mid X_n{}')\backslash L} \; \mathrm{Sort}_{e_i+d}(X_i) \cap \overline{\mathrm{Sort}_{e_j+d}(X_j)} = \emptyset \quad (i \neq j)$$

where, for $e \in \mathcal{R}_{>0}$, $\mathrm{Sort}_e(X)$ is the set of actions which X enables strictly before a delay of e; i.e. $\mathrm{Sort}_e(X) = \{a_i \mid e_i < e\}$.

Now, similarity and bisimilarity between the above type of TCCS networks may be decided using slight variations of the notions of symbolic state and symbolic semantics from Section 3. A *symbolic state* is now of the form $\phi_{L,L'}$, where L and L' are sets of observable actions (to be restricted). The (extended) symbolic state $\phi_{L,L'}$ represents all pairs of networks of the form

$(\overline{P}\backslash L, \overline{Q}\backslash L')$, where $(\overline{P}, \overline{Q}) \in \|\phi\|$. Compared with Subsection 3.2 the symbolic semantics must be altered so that the restriction sets are respected and synchronization allowed. This is achieved by rules of the form:

$$\frac{X \xrightarrow{a} X'}{\phi_{L,L'} \overset{a}{\longmapsto}_1 \langle\!\langle (\{X'\} \uplus M_1{}', N_1), \ldots, (M_k{}', N_k) \rangle\!\rangle_{L,L'}} \quad \begin{array}{l} M_i = M_i' \uplus \{X\} \\ M_j = M_j' \ (i \neq j) \\ a \notin L \cup \overline{L} \end{array}$$

$$\frac{X \xrightarrow{a} X' \quad Y \xrightarrow{\overline{a}} Y'}{\phi_{L,L'} \overset{\tau}{\longmapsto}_1 \langle\!\langle (\{X', Y'\} \uplus M_1{}', N_1), \ldots, (M_k{}', N_k) \rangle\!\rangle_{L,L'}} \quad \begin{array}{l} M_i = M_i' \uplus \{X\} \\ M_j = M_j' \uplus \{Y\} \\ M_l = M_l' \ (l \neq i, j) \end{array}$$

To be consistent with maximal progress the rules for $\overset{w}{\longmapsto}$ is modified as follows:

$$\phi_{L,L'} \overset{w}{\longmapsto} [(\emptyset, \emptyset), (M_1, N_1), \ldots, (M_k, N_k)] \quad \text{if} \quad \begin{cases} \phi \text{ boundary} \\ \phi_{L,L'} \overset{\tau}{\not\longmapsto}_1 \\ \phi_{L,L'} \overset{\tau}{\not\longmapsto}_2 \end{cases}$$

It may now be proved that the above (simple) modifications maintain the close relationship between the symbolic and standard semantics similar to the one expressed in the Correspondence Lemma 1. Thus the relationships between (ordinary) (bi)simulation and the symbolic counterparts stated in Theorem 2 remains valid.

Using as *pointed symbolic states* pairs of the form $\langle \phi_{L,L'}, \overline{v} \rangle$ the results from Section 4 may be seen to extend directly. In this extension it is crucial that the Correspondence Lemma 1 is valid for our extension.

6. Conclusion

In this paper we have described a constructive technique for generating diagnostic information for the timed bisimulation equivalence and the timed simulation preorder for a subclass of TCCS defined in [34]. More precisely, given two networks of initially integral regular processes the technique will provide a logical formula in Hennessy–Milner Logic [18] that distinguishes them in case they are not timed (bi)similar. Our method may be seen as an extension of the algorithm by Čerāns for deciding timed bisimilarity in that information of time–quantities has been added sufficient for generating distinguishing formulae.

The technique devised in this paper generalizes straightforward to all of TCCS and moreover it generalizes easily to the *modal* extension of TCCS TMS defined in [6] in which a variety of equivalences and preorders is considered. A detailed presentation of the technique and its coverage of TMS can be found in [15]. The technique has been added to the automatic verification tool for TMS, Epsilon, and it has been applied in the analysis and verification of a simple protocol using timing considerations in its retransmission strategy [13] as well as in the analysis and verification of a steam generator [20].

Acknowledgements

This work has been supported by the Danish Basic Research Foundation project BRICS and the ESPRIT Basic Research Action 7166, CONCUR2. The first author has been supported by a scolarship from Aalborg University.

References

[1] R. Alur and D. Dill. Automata for modelling real–time systems. In *Automata, Languages and Programming: Proceedings of the 17th ICALP*, LNCS 443, pages 322–335. Springer-Verlag, 1990.

[2] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. Technical Report P8916, University of Amsterdam, 1989.

[3] J. A. Bergstra, J. Heering, and P. Klint. Algebra of communicating processes. In *CWI symposium on Mathematics and Computer Science*, pages 89–138. North–Holland, 1986.

[4] U. Celikkan and R. Cleaveland. Diagnostic information for behavioral preorders. In *Proceedings of CAV'92*, volume 663 of *Lecture Notes In Computer Science*, pages 370–383. Springer-Verlag, 1992.

[5] K. Čerāns. Decidability of bisimulation equivalences for processes with parallel timers. In *Proceedings of CAV'92*, volume 663 of *Lecture Notes In Computer Science*, pages 302–315. Springer-Verlag, 1992.

[6] K. Čerāns, J.C. Godskesen, and K.G. Larsen. Timed modal specifications — theory and tools. In *Proceedings of CAV'93*, volume 697 of *Lecture Notes In Computer Science*, pages 253–267. Springer-Verlag, 1993.

[7] L. Chen. An interleaving model for real–time systems. In *Second International Symposium on Logical Foundation of Computer Science*, volume 620 of *Lecture Notes In Computer Science*, pages 81–92, Tver, 1992. Springer-Verlag.

[8] R. Cleaveland. On automatically distinguishing inequivalent processes. In *Proceedings of CAV'90*, volume 531 of *Lecture Notes In Computer Science*. Springer-Verlag, 1990.

[9] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench. Technical Report ECS-LFCS-89-83, LFCS, University of Edinburgh, Scotland, 1989.

[10] Jim Davis and Steve Schneider. An introduction to timed CSP. Technical Report PRG–75, Oxford University Computing Laboratory, 1989.

[11] R. DeNicola and F.W. Vaandrager. Three logics for branching bisimulation (extended abstract). In *Proceedings of Fifth Annual Symposium on Logic in Computer Science*, pages 118–129. IEEE Computer Society Press, 1991.

[12] J.C. Godskesen and K.G. Larsen. Real–time calculi and expansion theorems. In *Twelfth Conference on the FST and TCS*, volume 652, pages 302–315. Springer-Verlag, December 1992.

[13] J.C. Godskesen, K.G. Larsen, and A. Skou. Automatic verification of real–timed systems using epsilon (extended abstract). In *Proceedings of the Fourtienth International IFIP Symposium on Protocol Specification, Testing and Verification*, pages 323–330, June 1994.

[14] J.C. Godskesen, K.G. Larsen, and M. Zeeberg. TAV (tools for automatic verification) — users manual. Technical Report R 89-19, University of Aalborg, Denmark, 1989.

[15] J.C. Godskesen. *Timed Modal Specifications –A Theory for Verification of Real-Time Concurrent Systems.* Phd. thesis, Aalborg University, Denmark, 1994.

[16] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[17] M. Hillerström. *Verification of CCS–processes.* PhD thesis, Aalborg University Center, Denmark, 1987. R 87–27.

[18] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, pages 137–161, 1985.

[19] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[20] C.H. Kristensen and J.T. Lang. Applying epsilon to a steam generator case study. Technical Report R 94–2015, Aalborg Universitetscenter, March 1994.

[21] H. Korver. Computing distinguishing formulas for branching bisimulation. In *Proceedings of CAV'91*, volume 575 of *Lecture Notes In Computer Science*, pages 13–23. Springer-Verlag, 1991.

[22] K.G. Larsen. Efficient local correctness checking. In *Proceedings of CAV'92*, volume 663, pages 30–43. Springer-Verlag, 1992.

[23] K.G. Larsen and Y. Wang. Time abstracted bisimulation: Implicit specifications and decidability. In *Proceedings of MFPS'93*, volume 802 of *Lecture Notes In Computer Science*, pages 160–176. Springer-Verlag, 1993.

[24] R. Milner. *Calculus of Communicating Systems*, volume 92 of *Lecture Notes In Computer Science*. Springer Verlag, 1980.

[25] X. Nicollin and J. Sifakis. The algebra of timed processes atp: Theory and application. Technical Report RT–C26, IMAG, Grenoble, December 1990.

[26] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Proceedings of CAV'91*, volume 575 of *Lecture Notes In Computer Science*, pages 376–398, Aalborg, Denmark, 1991. Springer-Verlag.

[27] D. Park. Concurrency and automata on infinite sequences. In *5th GI Conference*, volume 104 of *Lecture Notes In Computer Science*, pages 167–183. Springer-Verlag, 1981.

[28] E.E. Polak. An efficient implementation of branching bisimulation and distinguishing formulae. Technical Report P9216, University of Amsterdam, 1992.

[29] J.L. Richier, C. Rodriguez, J. Sifakis, and J. Voiron. Xesar: a tool for protocol validation. users' guide. Technical report, LGI–IMAG, 1987.

[30] R. De Simone and D. Vergamini. Aboard AUTO. Technical Report 111, INRIA, Sofia–Antipolis, 1989.

[31] R.J. van Glabbeek. The linear time — branching time spectrum. In *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes In Computer Science*, pages 278–297. Springer-Verlag, 1990.

[32] R.J. van Glabbeek. The linear time — branching time spectrum II (the semantics of sequential systems with silent moves). In *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes In Computer Science*, pages 66–81. Springer-Verlag, 1992.

[33] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). *Information Processing 89*, pages 613–618, 1989.

[34] Y. Wang. Real–time behaviour of asynchronous agents. In *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes In Computer Science*, pages 502–520. Springer-Verlag, 1990.

[35] Y. Wang. *A Calculus of Real Time Systems*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 1991.