The Complexity of Deciding Behavioural Equivalences and Preorders

Hans Hüttel
and
Sandeep Shukla

We give an overview of the computational complexity of all the equivalences in van Glabbeek's linear/branching time hierarchy and the preorders in the corresponding hierarchy of preorders. We consider finite state or *regular* processes as well as infinite-state BPA and BPP processes.

A distinction, which turns out to be important in the finite-state processes, is that of *simulation-like* equivalences/preorders vs. *trace-like* equivalences and preorders. Here we survey various known complexity results for these relations. For regular processes, all simulation-like equivalences and preorders are decidable in polynomial time whereas all trace-like equivalences and preorders are PSPACE-Complete. We also consider interesting special classes of regular processes such as *deterministic*, *determinate*, *unary*, *locally unary*, and *tree-like* processes and survey the known complexity results in these special cases.

For infinite-state processes the results are quite different. For the class of *context-free processes* or *BPA processes* any preorder or equivalence beyond bisimulation is undecidable but bisimulation equivalence is polynomial time decidable for *normed* BPA processes and is known to be elementarily decidable in the general case. For the class of *BPP* processes, all preorders and equivalences apart from bisimilarity are undecidable. However, bisimilarity is decidable in this case and is known to be decidable in polynomial time for normed BPP processes.

Categories and Subject Descriptors: F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*mechanical verification*

General Terms: Theory,Verification

Additional Key Words and Phrases: behavioural equivalences, behavioural preorders, complexity bounds, modal logic

1. Introduction

Within concurrency theory, a number of preorders and equivalence relations between processes have been considered in various approaches to the semantics of concurrency and automatic verification.

In this paper, we shall consider the equivalences that have come out of the study of interleaving semantics in the context of process calculi in the tradition of CCS [Mil80] and CSP [Hoa84]. Most of these preorders and equivalences first arose in the literature of *comparative concurrency semantics* [vG90a; BIM90; GV92]. In this particular area of concurrency semantics, the main emphasis is on *full abstraction* and various notions of equivalences have been found to be fully abstract for different language constructs. For example, **bisimulation** equivalence is used in CCS [Mil80; Mil89] to identify processes which are equivalent under a particular semantic notion [Par81; Mil89]. However, in [BIM90], bisimulation has been shown not to be *fully abstract* and the notion of *ready simulation* has been defined. In [GV92], a new notion of 2-nested simulation equivalence has been defined and shown to be fully

Name: Hans Hüttel
Affiliation: BRICS, Department of Computer Science, Aalborg University
Address: Fredrik Bajersvej 9220 Aalborg Ø, Denmark, e-mail – hans@cs.auc.dk
Name: Sandeep Shukla
Affiliation: Department of Computer Science, State University of New York at Albany
Address: Albany NY 12222, USA, e-mail – – sandeep@cs.albany.edu

abstract for languages with a general format called the *tyft/tyxt*.

Within the area of computer-aided verification, there has been a significant amount of work devoted to using these relations to prove the correctness of concurrent systems [BCM92]. The correctness criterion is then that the implementation is equivalent to the specification. Also, establishing that a given simulation relation holds has been used as a partial procedure for proving some safety properties [LV91].

In [vG90a] van Glabbeek proposed the *linear/branching time spectrum* as a unifying framework for classifying all known equivalences in the area of comparative concurrency semantics. We shall follow this classification here. Figure 1 [vG90a] illustrates the classification as a hierarchy with the help of a Hasse diagram. The arrows in the diagram imply strict inclusion. Hence if there is an arrow from a relation R to another relation Q, that means R is less discriminating than Q. In other words, if two processes are related by R then they must be related by Q but the converse is not true in general. The least discriminating equivalences are at the bottom of the diagram.

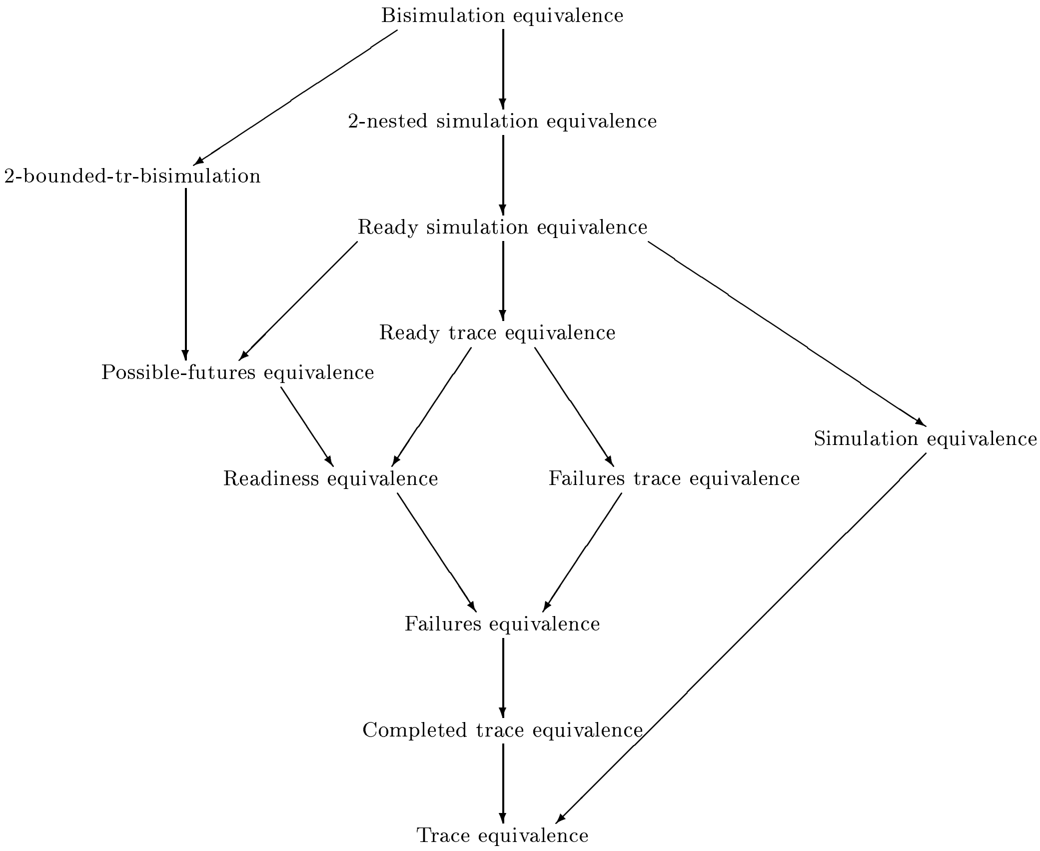

Fig. 1. The linear-time/branching time hierarchy of equivalences.

The coarsest equivalences are trace equivalence and completed trace equivalence (=language equivalence). Directly above them we have the testing/failures equivalences, and at the top of the diagram is bisimulation equivalence.

As all equivalences save bisimilarity are defined as the symmetric closure of a preorder, there is a similar hierarchy for the behavioural preorders, illustrated in Figure 2.

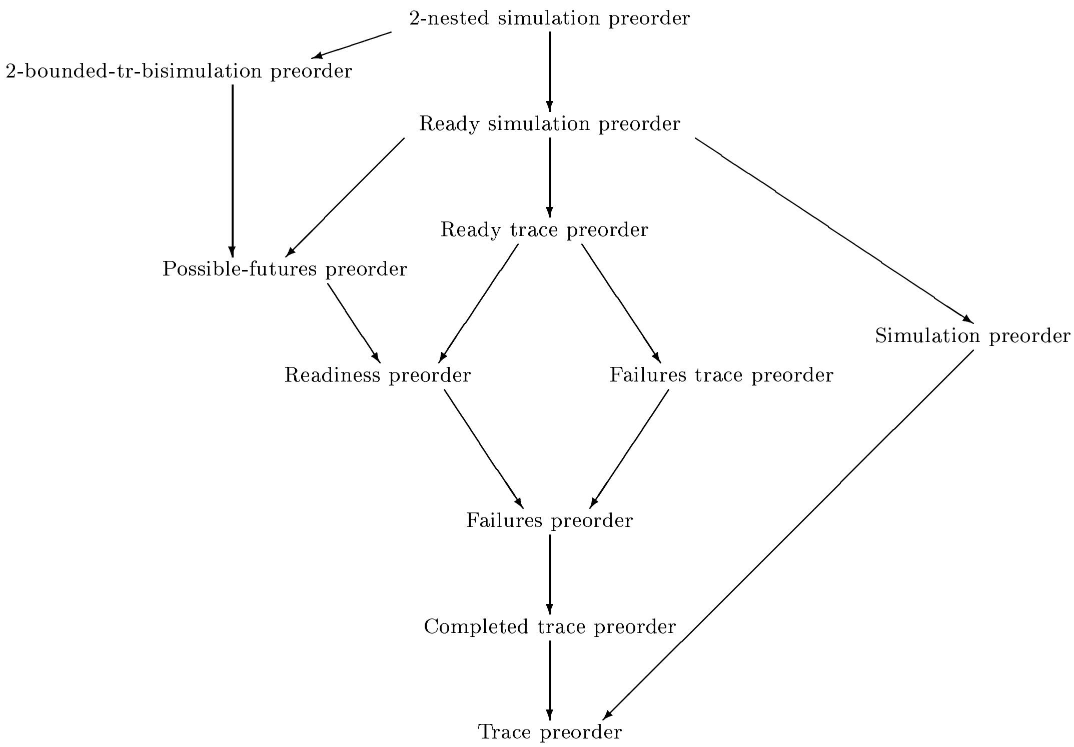

Fig. 2. The linear-time/branching time hierarchy of preorders.

Motivated by the importance of these relations in automated verification, several researchers have studied the decision problems for these relations [KS90; PT87; HT94; SRHS96; BP94; And93; And94; ABGS91; CS91; MS95; Hut91; HT90]. A main distinction has turned out to be that of *finite-state* processes versus *infinite-state processes*. It is well-known that all behavioural relations in the van Glabbeek hierarchy are decidable for finite-state processes, and the main concern is therefore that of the computational complexity of the decision procedures. On the other hand, for sufficiently rich classes of infinite-state processes, no non-trivial behavioural equivalence is decidable. However, in recent years, it has been shown that some behavioural equivalences are indeed decidable for certain interesting classes of infinite-state processes and recently, a number of complexity results have been established for these classes of processes.

Apart from a short survey by Moller and Smolka [MS95] on the complexity of bisimulation equivalence, there has not been any attempt to present all these results in a unifying framework. The fact that these relations are widely used by the computer aided verification community to establish correctness warrants a survey of the complexity results for these relations. The present paper gives an overview of the results and attempts to unify the results within a particular point of view. Here, we shall only consider the so-called *strong* equivalences, i.e. notions of equivalence that do not distinguish between observable and non-observable actions.

In this paper a further distinction between *simulation-like* equivalences (preorders, respectively) and *trace-like* equivalences (preorders) turns out to be important. A common characteristic of simulation-like equivalences is that they are defined using the notion of *simulation* w.r.t. single actions – more precisely, the simulation-like equivalences are bisimilarity, n-nested simulation equivalence, ready-simulation equivalence and simulation equivalence. All other equivalences are trace-like, in that their definitions at some point call upon the notion of a *sequence* of visible actions, i.e. a trace.

In the case of finite-state processes, all simulation-like equivalences are in P, whereas the trace-like equivalences are PSPACE-complete. In the case of infinite-state processes, the picture is different. All equivalences other than bisimulation equivalence are *undecidable* for the classes BPA and BPP. In the case of bisimulation, the equivalence is in P for so-called normed BPA processes and elementarily decidable for arbitrary BPA processes. Bisimulation is also known to be in P for normed BPP processes.

2. Labelled transition systems

The common model of the behaviour of concurrent systems within interleaving semantics is that of a labelled transition system, which describes the state change of a processes and the actions that they can perform at any given instant. The idea of using labelled transition systems for this purpose originates with Milner [Mil80].

Definition 2.1. *A* labelled transition system $(Pr, Act, \to)$*is a triple where* Pr *is the set of* states*,* Act *is the set of* actions *and* $\to$ *is the* transition relation *satisfying*

$$\to \subseteq Pr \times Act \times Pr$$

Instead of writing $(p, a, q) \in\to$ one usually writes $p \xrightarrow{a} q$ and interprets this as 'from state p we can perform an a-action leading to the state q'. Sometimes we consider the reflexive, transitive closure of $\to$, writing $p \xrightarrow{w} p'$ if $w = a_1 \cdots a_n \in Act^*$ it is the case that $p \xrightarrow{a_1} p_1 \cdots \xrightarrow{a_n} p_n = p'$ for some intermediate states $p_1, \ldots, p_n$.

We shall use the predicates '$p \xrightarrow{a}$' denoting '$\exists q : p \xrightarrow{a} q$', '$p \not\xrightarrow{a}$' denoting '$\neg\exists q : p \xrightarrow{a} q$' and $p \not\longrightarrow$ for $\forall a \in Act : p \not\xrightarrow{a}$.

3. Finite-state processes

In this section we examine the complexity of deciding behavioural relation for finite transition systems. When discussing the complexity of deciding an equivalence or preorder w.r.t the finite transition system $(Pr, Act, \rightarrow)$, we shall always assume that the complexity is a function of the *size* of the transition system, $n = |Pr| + |\rightarrow|$, i.e. the sum of the sizes of the state space and the transition relation.

3.1 Regular processes

The class of regular processes was first investigated by Milner [Mil84], who showed that a labelled transition system is finite iff it can be described by means of a regular process.

Regular process expressions [Mil84] are given by the abstract syntax

$$p ::= a \mid X \mid p_1 + p_2 \mid ap \mid \mathbf{0}$$

Here a ranges over a set Act of atomic actions, and X over a set Var of variables. The symbol $+$ is the non-deterministic choice, ap_2 represents prefixing the process p_1 with the action a and $\mathbf{0}$ denotes the empty (inactive) process.

We say that a process expression is *guarded* iff every variable occurrence in p occurs within a prefix, i.e. in a subexpression $a.q$ of p. Regular processes are defined by a finite set Δ of guarded equations

$$\Delta = \{X_i \stackrel{\text{def}}{=} p_i \mid 1 \leq i \leq k\}$$

where the X_i are distinct process variables, and the p_i are guarded expressions with free variables in $Var(\Delta) = \{X_1, \ldots, X_k\}$. One variable (generally X_1) is singled out as the *root*. We shall only consider processes defined by guarded equations.

In what follows we shall feel free to write $\Delta_1 R \Delta_2$ for binary relations R; this should be read as stating that the roots of Δ_1 and Δ_2 are related by R.

The operational semantics of a regular process expression, given a finite system of process equations Δ, is given by the labelled transition system $(Pr, Act, \rightarrow)$where Pr is the set of regular process expressions and $\rightarrow$ is defined as the least relation satisfying the proof rules given below.

$$\frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$$

$$a.p \xrightarrow{a} p \quad a \in Act \qquad \frac{p \xrightarrow{a} p'}{X \xrightarrow{a} p'} \qquad X \stackrel{\text{def}}{=} p \in \Delta$$

We shall usually omit the subscript Δ, when obvious from the context. A process expression together with an associated transition relation is called a process.

We shall mostly consider systems of process equations in *normal form*:

DEFINITION 3.1. *[Mil84] A system of regular process equations is in normal form if each equation is of the form* $X_i \stackrel{\text{def}}{=} \sum a_{ij} X_{ij} + \sum b_{ik}.\mathbf{0}$.

THEOREM 3.1. *[Mil80] Let $\sim$ denote bisimulation equivalence. For any system of guarded regular process equations Δ there is a system of process equations Δ' in normal form such that $\Delta \sim \Delta'$. Moreover, Δ' can be found effectively.*

As bisimulation equivalence is at the top of the van Glabbeek hierarchy, we see that the transformation into normal form preserves all equivalences. Moreover, it is easy to see that the transformation of a system of equations Δ into normal form can be accomplished in time polynomial to the size of Δ. It is therefore enough to consider systems of regular process equations in normal form.

3.2 Simulation-like equivalences and preorders

We shall call the class of equivalences and preorders that have a *simulation-style* definition or whose definition employs a fixed.depth recursive application of such a definitional structure (e.g.,n-nested simulation) **simulation-like** equivalences. The simulation-like equivalences are *bisimulation equivalence*, *simulation*, *ready-simulation*, *n-nested simulation*, *$\frac{m}{2}$-nested simulation*, *complete-simulation* preorders and equivalences.

All simulation-like relations share a common characteristic, namely that they define how a transition by a process must be simulated by another process, in order for the processes to be related. All simulation-like preorders (save bisimulation) are then defined as the largest sets of pairs satisfying their appropriate simulation condition. This indicates that the simulation-like preorders in the preorder hierarchy are definable as fixed-points of functionals over the complete lattice of relations, and the computation of relations then reduces to computing maximal fixed points of certain functionals. We therefore start this section by outline give some basic definitions and results from lattice theory.

3.2.1 *Functions over complete lattices and their fixed points*

Definition 3.2. *A partial order $(D, \sqsubseteq)$ is called a* complete lattice *if for any set of points $Y \sqsubset D$ there exists a least upper bound* $\sup Y$ *and a greatest lower bound* $\inf Y$ *w.r.t. $\sqsubseteq$.*

Proposition 3.1. *Any complete lattice $(D, \sqsubseteq)$ has a least element $\perp$, i.e. a point $\perp$ such that $\perp \sqsubseteq x$ for all $x \in D$, and a largest element, i.e. a point $\top$ such that $x \sqsubseteq \top$ for all $x \in D$.*

It is easy to see that, given any transition system $(Pr, Act, \rightarrow)$, the family of binary relations on processes constitutes a complete lattice $(\mathbf{2}^{Pr \times Pr}, \subseteq)$ with respect to the inclusion ordering. If $Y \subseteq \mathbf{2}^{Pr \times Pr}$ then $\inf Y = \bigcap\{y \mid y \in Y\}$ and $\sup Y = \bigcup\{y \mid y \in Y\}$. The least element of $(\mathbf{2}^{Pr \times Pr}, \subseteq)$ is the empty relation $\emptyset$, and the largest element is the total relation $Pr \times Pr$.

Definition 3.3. *Let $(D, \sqsubseteq)$ be a partial order. An endofunction $f : D \rightarrow D$ is said to be monotonic if whenever $x, y \in D$ and $x \sqsubseteq y$ then $fx \sqsubseteq fy$.*

The following two standard results of basic lattice theory form the basis of the theory of simulation-like equivalences and of their various decision procedures. The results, due to Tarski, are also central to the semantics of the modal mu-calculus (see Section 3.3.2.)

Theorem 3.2. *[Tar55] Let $(D, \sqsubseteq)$ be a complete lattice and let $f : D \rightarrow D$ be a monotonic endofunction. Then f has a least fixed-point* fix f *given by*

$$\text{fix } f = \inf\{x \mid fx \sqsubseteq x\}$$

and a largest fixed-point FIX f *given by*

$$\text{FIX } f = \sup\{x \mid x \sqsubseteq fx\}$$

In the case of *continuous* and *co-continuous* functions over complete lattices, another characterization of these fixed-points is possible. A function is said to be continuous (resp. co-continuous) if it preserves least upper (resp. greatest lower) bounds of totally ordered subsets[1]

DEFINITION 3.4. *Let $(D, \sqsubseteq)$ be a complere lattice. A monotonic endofunction $f : D \to D$ is said to be* continuous *if for any totally ordered subset $Y \subseteq D$ we have that*

$$f(\sup Y) = \sup\{fx \mid x \in Y\}$$

f is said to be co-continuous *if*

$$f(\inf Y) = \inf\{fx \mid x \in Y\}$$

THEOREM 3.3. *[Tar55] Let $(D, \sqsubseteq)$ be a complete lattice and let $f : D \to D$ be a monotonic endofunction. If f is continuous, then f has a least fixed-point* fix f *given by*

$$\text{fix } f = \sup\{f^n\bot \mid n \geq 0\}$$

If f is co-continuous, then f has a largest fixed-point given by

$$FIX f = \inf\{f^n\top \mid n \geq 0\}$$

This latter result allows an iterative characterization of these fixed-points and forms the concrete basis for several equivalence-checking algorithms. In what follows, we shall express any simulation-like preorders as maximal fixed-points of an associated endofunction. In the case of *finite-state* processes, these endofunctions are all co-continuous.

3.2.2 *Simulation equivalence and the simulation preorder.* Intuitively, the simulation preorder relates two processes, if any transition by the former process can be simulated by the latter in such a way that the resulting processes are still related. The notion of simulation equivalence is then simply the equivalence closure of the simulation preorder.

DEFINITION 3.5. *A relation R between processes is a* simulation *iff whenever pRq then for each $a \in Act$ $p \xrightarrow{a} p' \;\Rightarrow\; \exists q : q \xrightarrow{a} q' \wedge p'Rq'$. A process p is simulated by a process q, notation $p \subsetneq q$, iff there is a simulation relation R with pRq. Two processes p and q are* simulation equivalent, *notation $p \leftrightarrows q$, iff $p \subsetneq q$ and $q \subsetneq p$.*

3.2.3 *Bisimulation equivalence.* The notion of bisimulation equivalence was first proposed by Park [Par81] and later used by Milner [Mil89]. Bisimulation equivalence is the only equivalence in the linear-branching time hierarchy *not* defined as the equivalence closure of some corresponding preorder.

[1](aka chains.)

DEFINITION 3.6. *Given a labelled transition system* $(Pr, Act, \rightarrow)$*, a relation* R *is a bisimulation relation if whenever* pRq *then*

—*If* $p \xrightarrow{a} p'$ *then* $\exists q' : q \xrightarrow{a} q'$ *with* $p'Rq'$
—*If* $q \xrightarrow{a} q'$ *then* $\exists p' : p \xrightarrow{a} p'$ *with* $p'Rq'$

.

3.2.4 *$\frac{m}{2}$-nested simulations.* The hierarchy of $\frac{m}{2}$-nested simulations was proposed by Liu in [Liu92]. This hierarchy generalizes that of the hierarchy of n-nested simulations described in the next section. The central notion is that of nesting a simulation within another; the matching condition now requires that one matches transitions within (the inverse of) a simulation.

DEFINITION 3.7. *[Liu92] Let* $(Pr, Act, \rightarrow)$*be a labelled transition system,* $\Re \subseteq Pr \times Pr$ *be a binary relation. Then* $\mathcal{S}$ *is said to be a simulation nested in* $\Re$ *if* $\mathcal{S}$ *is a simulation and* $\mathcal{S} \subseteq \Re^{-1}$ *A process* P *is said to be simulated in* $\Re$ *by another process* Q *just in case* (P, Q) *is contained in some simulation* $\mathcal{S}$ *nested in* $\Re$*. We write* $P\mathcal{N}(\Re)Q$ *in this case.*

The following results from [Liu92] motivate the above definition.

THEOREM 3.4. *[Liu92]* $\mathcal{N}(\Re)$ *is itself a simulation nested in* $\Re$*, in fact the maximal one. If* $\Re$ *s preorder then so is* $\mathcal{N}(\Re)$

LEMMA 3.1. *[Liu92]* $\mathcal{N}$ *is monotonic, that is, for any two relations* $\Re_1$ *and* $\Re_2$*, if* $\Re_1 \subseteq \Re_2$*, then* $\mathcal{N}(\Re_1) \subseteq \mathcal{N}(\Re_2)$.

Hence by Theorem 3.2 $\mathcal{N}$ has a largest fixed point.

THEOREM 3.5. *[Liu92] Any post-fixed point of* $\mathcal{N}$ *is a bisimulation.*

One can apply the nesting operator $\mathcal{N}$ repeatedly to obtain a hierarchy of finer and finer equivalences and preorders.

DEFINITION 3.8. *[Liu92] Let* $(Pr, Act, \rightarrow)$*be a labelled transition system. We define a series of relations* $\subsetneq^{\frac{m}{2}} \subseteq Pr \times Pr$ *and* $\leftrightarrows^{\frac{m}{2}}$ $(m \geq 0)$ *as follows:*

1. $\subsetneq^{0} = Pr \times Pr$.
2. $\subsetneq^{\frac{1}{2}} = \{(P, Q) \mid \forall a \in Act.P \xrightarrow{a}\Rightarrow Q \xrightarrow{a}\}$
3. $\subsetneq^{\frac{m}{2}+1} = \mathcal{N}(\subsetneq^{\frac{m}{2}})$*, for* $m \geq 0$.

And for $m \geq 0$*,* $\leftrightarrows^{\frac{m}{2}} = \subsetneq^{\frac{m}{2}} \cap (\subsetneq^{\frac{m}{2}})^{-1}$.

As mentioned earlier, this hierarchy contains the hierarchies of n-nested equivalences and preorders. Further, $\subsetneq^{\frac{3}{2}}$ coincides with the ready-simulation preorder [BIM90] (a.k.a. the $\frac{2}{3}$-bisimulation of [LS89]!)

3.2.5 *n-nested simulation equivalences and preorders.* The notion of n-nested simulation equivalence was introduced by Groote and Vaandrager [GV89] in their study of the *tyft/tyxt*-format for structured operational semantics because 2-nested simulation equivalence is the completed trace congruence for this format.

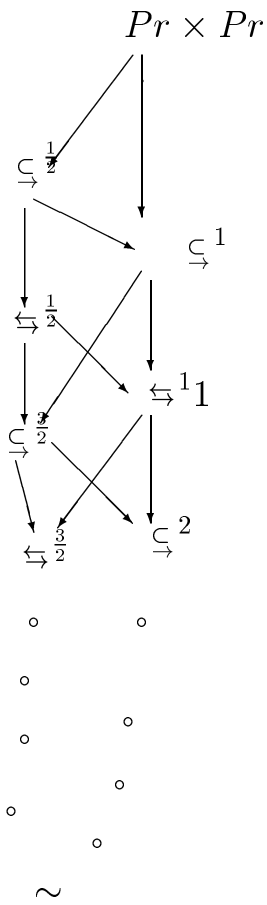

Fig. 3. The hierarchy of $\frac{m}{2}$-nested equivalences and preorders

Definition 3.9. *For all* $n \in \mathbf{N}$, n-nested simulation, *written* $\underset{\rightarrow}{\subset}^n$, *is inductively defined by*

—$p \underset{\rightarrow}{\subset}^0 q$ *for all processes* p *and* q,
—$p \underset{\rightarrow}{\subset}^{n+1} q$ *iff there is a simulation relation* $R \subseteq (\underset{\rightarrow}{\subset}^n)^{-1}$ *with* pRq.

Two processes p *and* q *are* n-nested simulation equivalent, *written* $p \leftrightarrows^n q$, *iff* $p \underset{\rightarrow}{\subset}^n q$ *and* $q \underset{\rightarrow}{\subset}^n p$.

Note that 1-nested simulation is just simulation and that therefore 1-nested simulation equivalence is simulation equivalence.

3.2.6 *Ready-simulation or 2/3-bisimulation.* The notion of ready simulation (or 2/3-bisimulation) originated in work by Bloom, Istrail and Meyer [BIM90] and Larsen and Skou [LS89]. It is the completed trace and the trace congruence induced by the GSOS-format [BIM90]. The matching condition of the preorder definition now states that two processes are related, if the transitions of the former process can be matched by the latter with the resulting processes staying within the relation *and* that the two processes have the same sets of initial actions.

DEFINITION 3.10. *A relation R between processes is a* ready simulation *iff it is a simulation and whenever pRq then for each $a \in Act$ we have $p \xrightarrow{a}$ if $q \xrightarrow{a}$. We say that q ready simulates p, written $p \subsetneq_r q$, iff there is a ready simulation R with pRq. Two processes p and q are* ready simulation equivalent, *written $p \leftrightarrows_r q$, iff $p \subsetneq_r q$ and $q \subsetneq_r p$.*

3.3 Complexity results

We shall now we discuss the complexity results for simulation-like equivalences and focus on how these results can be obtained. The main result is that

THEOREM 3.6. *All simulation-like equivalences of the linear-branching time hierarchy are polynomial time decidable for finite state transition systems.*

There are (at least) four different ways of obtaining this result, namely by means of *approximation techniques*, *characteristic formulae*, *Horn clauses* and *characteristic games*.

A lot of attention has been devoted to establishing good complexity bounds for the bisimulation equivalence problem, as bisimilarity has become widely used for verification purpose. In 1991, Alvarez showed that this particular equivalence problem is **P**-complete.

THEOREM 3.7. *[ABGS91] The bisimulation equivalence problem for finite-state processes is P-complete.*

3.3.1 *Approximation techniques.* Because of Theorem 3.2, for any simulation-like relation pRq the decision problem 'pRq ?' amounts to deciding whether or not (p, q) is a member of the largest fixed-point of a suitable functional, namely the functional inherent in the definition of the relation.

EXAMPLE 3.1. (Ready simulation) The ready simulation functional $\mathcal{F}_r : \mathbf{2}^{Pr \times Pr} \to \mathbf{2}^{Pr \times Pr}$ is defined as follows: $(p, q) \in \mathcal{F}_r(R)$ if for any $a \in Act$, whenever $p \xrightarrow{a} p'$ then there exists a q' such that $(p', q') \in R$. It is easily seen that $\mathcal{F}_r$ is monotonic and that R is a ready simulation iff $R \subseteq \mathcal{F}_r(R)$. Thus, by Theorem 3.2, we have that

$$p \subsetneq_r q \iff (p, q) \in \text{FIX } \mathcal{F}_r$$

and also that

$$\subsetneq_r = \bigcap_{i=0}^{\omega} {\mathcal{F}_r}^i(Pr \times Pr)$$

□

The above example is easily generalizable to all simulation-like preorders; the value of the underlying functional $\mathcal{F}(R)$ is simply the set of pairs (p, q) that have matching transitions (in the sense of the corresponding definition) to processes that fall within R. Membership of a simulation-like relation then amounts to membership of a post-fixed-point of $\mathcal{F}$, i.e. as membership of the largest fixed-point.

A possible decision procedure for any simulation-like relation now consists in computing the largest fixed point of $\mathcal{F}$. Theorem 3.2 immediately gives an iterative refinement strategy that consists in computing the successive approximations of the largest fixed point.

The pragmatic problem is one of finding a computationally efficient strategy for computing the ith approximant $\mathcal{F}^i(Pr \times Pr)$. For example, the polynomial-time algorithm for bisimulation equivalence by Kanellakis and Smolka [KS90] computes the largest bisimulation by means of a bottom-up partition refinement strategy using an algorithm due to Paige and Tarjan. The polynomial-time algorithms for simulation equivalence and ready simulation equivalence by Huynh and Tian [HT94] employs a different strategy for computing the largest fixed-point. The Bloom and Paige algorithm in [BP94] for ready simulation uses yet another variant of the approximation approach.

3.3.2 *Characteristic formulae.* Another approach to equivalence checking appeals to model checking by constructing *characteristic formulae.* Following Hennessy and Milner [HM85], properties of labelled transition systems are usually described by means of modal logic. The modal mu-calculus extends the basic Hennessy-Milner logic with recursive definitions. Here, we shall only consider the *nu-calculus*, which allows maximal recursive definitions and has the syntax

$$F ::= \mathit{tt} \mid \mathit{ff} \mid [a]F \mid \langle a\rangle F \mid F_1 \vee F_2 \mid \neg F_1 \mid X$$

Here X ranges over some set of recursion variables $Rvar$. Nu-calculus-formulae are given by means of *declarations*; a declaration Δ is a family of recursion equations of the form

$$\Delta = \{X_i \stackrel{\text{def}}{=} F_i \mid 1 \leq i \leq k\}$$

The semantics of a nu-calculus declaration is defined relative to a labelled transition system $(Pr, Act, \rightarrow)$ and to an assignment of the occurring recursion variables; the semantics of a formula F is the set of states satisfying the formula F. Given some $\rho : Rvar \rightarrow \mathbf{2}^{Pr}$, the semantics is defined inductively by

$$\begin{aligned}
[\![\mathit{tt}]\!]\rho &= Pr \\
[\![\mathit{ff}]\!]\rho &= \emptyset \\
[\![[a]F]\!]\rho &= \{p \mid \exists p' : p \xrightarrow{a} p', p' \in [\![F\rho]\!]\} \\
[\![\langle a\rangle F]\!]\rho &= \{p \mid p \xrightarrow{a} p' \Rightarrow p' \in [\![F]\!]\rho\} \\
[\![F_1 \wedge F_2]\!] &= [\![F_1]\!]\rho \cap [\![F_1]\!]\rho \\
[\![X]\!]\rho &= \rho(X) \\
[\![\neg F]\!]\rho &= Pr \setminus [\![F]\!]
\end{aligned}$$

and by the condition that the semantics of any recursion variable X_i is the largest set of processes S such that $[\![X_i]\!]\rho[X_i \mapsto S] = [\![F_i]\!]\rho[X_i \mapsto S]$. The existence of this set is guaranteed by Theorem 3.2, provided all recursion variables occur within the scope of an even number of negation signs on any right-hand side of a defining equation.

A characteristic formula for the relation R and the state p is then a mu-calculus formula F_p such that qRp iff q satisfies F_p. Thus, this approach reduces the equivalence problem to that of *model checking*. So, as the semantics of formulae involve fixed-points, the characteristic formula approach again (albeit somewhat indirectly) decides membership of a relation by means of computing fixed-points.

EXAMPLE 3.2. (Ready simulation) Given a finite labelled transition system $(Pr, Act, \rightarrow)$, the characteristic formulae for the ready simulation preorder for all states are collectively given by the declaration

$$\{X_p \stackrel{\text{def}}{=} \bigvee_{\{(a,q) \mid p \xrightarrow{a} q\}} \langle a \rangle X_q \wedge \bigvee_{\{b \mid p \stackrel{b}{\not\rightarrow}\}} [b]\mathit{ff} \mid p \in Pr\}$$

The first conjuncts of the right-hand side for X_p in the above declaration describe that any process satisfying X_p must be able to perform the same transitions as p in such a way that the resulting processes are again related. The final conjunct describes the requirement that any process satisfying X_p cannot have other initial actions than those of p. Taken together, these are precisely the requirements of the definition of ready simulation. $\Box$

In [CS91; And93] it was shown how one can construct characteristic formulae for a number of equivalences and preorders within the nu-calculus. As the model checking problem for the nu-calculus is decidable in time $O(n \cdot m)$, where m is the size of the declaration and n is the size of the transition system, and as the declaration constructed essentially describes the transitions of the transition system and therefore is of size $O(n)$, this shows that the simulation-like equivalences considered in [CS91; And93] are polynomial-time decidable.

3.3.3 *Horn clauses.* A third approach, which is closely related to the characteristic formula approach, builds on the approach used in giving polynomial-time algorithms for the rest of the simulation-like relations first presented in [SRHS96]. The simulation-like relations can be reduced to the satisfiability problem for weakly negative Horn formulas [Sch78], known as the NHORNSAT problem. Since there the NHORNSAT problem is decidable in linear time [DG84; AI91], this shows that all simulation-like equivalences are decidable in polynomial time.

Given a type of simulation relation R, the method in [SRHS96] entails a *top-down* construction of a propositional formula f in CNF. The variables in the formula f are $X_{p,q}$ where p and q are states in the two transition systems. Intuitively, $X_{p,q}$ is true iff p and q are related by R. The clauses in the formula f are of the following three types.

(1) A single positive literal $X_{p,q}$. When we want (p, q) to be in the simulation relation we construct this type of clause.
(2) A single negated literal $\overline{X_{p,q}}$. Such a clause is constructed when (p, q) cannot be in any simulation relation of the given type.
(3) Implication clauses of the form $X_{p,q} \Rightarrow \bigvee_{i,j} X_{i,j}$. A clause of this form is constructed when, for (p, q) to be in the simulation relation, one of the (i, j)'s must also be in the simulation relation.

The details of the construction of the clauses depends on the actual properties that must be satisfied by R. The effectiveness of the reduction relies on the property that if we generate a clause of the form $\overline{X_{s,t}}$, then it is guaranteed that no relation satisfying the properties of that particular relation can contain the pair (s,t).

The resulting CNF formulae are so-called weakly negative Horn clauses[2] [Sch78]. The satisfiability problem for such formulae is called NHORNSAT. It is easy to show that NHORNSAT is decidable in linear time [DG84; AI91]. Moreover, from [AI91], it is easy to construct an algorithm for NHORNSAT which is *incremental* or *on-line*. As the size of the formula is $O(n^2)$ where n is the size of the transition systems, we get

THEOREM 3.8. *Let* $(Pr, Act, \rightarrow)$*be a labelled transition system of size* n. *Any simulation-like equivalence relation on* n *is decidable in time* $O(|\rightarrow|^2)$.

EXAMPLE 3.3. (Ready simulation) We give a polynomial time algorithm that takes $(Pr, Act, \rightarrow)$and two states $s,t \in Pr$ as input and outputs an instance h of $NHORNSAT$ such that h is satisfiable if and only if $s \subsetneq_r t$. In the instance h the number of variables is $\leq |Pr|^2$ and the size of the instance is $O(|\rightarrow|^2)$.

The algorithm is given in Figure 4. The algorithm relies on three auxiliary functions that together code up the conditions of the definition of ready simulation. $\delta_r(a,q,p)$ is the set of all states that are reachable from state q by executing an a action and have the same initial actions as p.

$$\delta_r(a,q,p') = \{q' | q \xrightarrow{a} q' \wedge init(p') = init(q')\}$$

Whenever we consider the pair (p,q), we want to represent the conditions for their inclusion in a ready simulation relation. Given a transition $p \xrightarrow{a} p'$ there must be a transition $q \xrightarrow{a} q'$ for which (p',q') is in the ready simulation relation. $\mathcal{C}$ computes clauses expressing this fact.

$$\mathcal{C}(p_i, a, p_i^{'}, q_j) = \begin{cases} \bigvee_{q_j' \in \delta_r(a,q_j,p_i')} X_{p_i',q_j'} & \text{if } \delta_r(a,q_j,p_i^{'}) \neq \emptyset \\ \mathit{false} & \text{otherwise} \end{cases}$$

Finally, we need to keep track of the variable occurrences in a newly created condition clause as these correspond to the pairs of processes that need to be included in a ready simulation. This is expressed using $\mathcal{V}$.

$$\mathcal{V}(p_i, a, p_i^{'}, q_j) = \begin{cases} \{X_{p_i',q_j'} \mid q_j^{'} \in \delta_r(a,q_j,p_i^{'})\} & \text{if } \delta_r(a,q_j,p_i^{'}) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

The constructed $NHORNSAT$ instance h has the property that given any satisfying truth-assignment v, the relation R defined by $R = \{(p,q) \,|\, v(X_{p,q}) = 1\}$ is a ready simulation. Conversely, if $s \subsetneq_r t$ then sRt for some ready simulation and we can define a truth-assignment v by $v(X_{p,q}) = 1$ iff $(p,q) \in R$. This truth assignment satisfies h.

□

[2]A weakly negative clause is a clause which contains at most one negative literal.

Let C be the set of clauses initially empty. Let V be the set of variables initially empty.

(1) If $(init(s) \neq init(t))$ then return an unsatisfiable formula of the form $X_{s,t} \wedge \overline{X_{s,t}}$ and terminate.
(2) $C := C \cup \{X_{s_1,t_1}\}$; $V := \{X_{s_1,t_1}\}$
(3) $p_i := s_1; q_j := t_1$;
(4) Do until V is empty.
 (a) $V := V - \{X_{p_i,q_j}\}$
 (b) For each $t \in D_1$
  such that $t = (p_i, a, p_i^{'})$
   for some $a \in Act$
    $C := C \cup \{\overline{X_{p_i,q_j}} \vee \mathcal{C}(p_i, a, p_i^{'}, q_j)$
     $V := V \cup \mathcal{V}(p_i, a, p_i^{'}, q_j)$
(5) Let $X_{p,q}$ be the one element in V. Then $p_i := p$ and $q_j := q$; go to step 3.
(6) Output C.

Fig. 4. Algorithm for reducing an instance of the ready simulation problem to an instance of $NHORNSAT$

Notice the similarity to the corresponding characteristic formula for ready simulation given in Example 3.2.

3.3.4 *Game-theoretic characterizations.* The fourth and final approach to obtaining complexity bounds gives a game-theoretic characterization of behavioural relations.

In [Sti93], Colin Stirling introduced the notion of a *characteristic game* for bisimulation equivalence. In [SHR95; SHR96] a general class of games, called the *Stirling* class of games, was defined and shown to characterize all equivalences and preorders in the linear/branching time hierarchy.

A game in the Stirling class has two players. One player is called the *prover* and the other is called the *disprover*. The game starts in a position $\langle s, t\rangle \in \Sigma$. A *play* of the game is a finite or infinite length sequence of the form $\langle s_0^1, s_0^2\rangle, ..., \langle s_i^1, s_i^2\rangle, ...$ The *disprover* wants to show that there is a *difference* between the two transition systems. The *prover* wants to show that such a distinction is not possible.

A *partial play* in a game is a prefix of a *play* of the game. Let π_j be a *partial play* $\langle s_0^1, s_0^2\rangle, ..., \langle s_j^1, s_j^2\rangle$. The next pair $\langle s_{j+1}^1, s_{j+1}^2\rangle$ is determined by the following move rule:

—The disprover picks a triple $\langle i, x, u\rangle$ such that $i \in M$ and $x \in R_i$ and $s_j^i \xrightarrow{x}_i u$. and $u = s_{j+1}^i$. (Note that $\rightarrow_i$ denotes an extended step in the transition system T_i).

—Let the choice of the disprover in the move be $\langle i, x, u\rangle$ and let $i' \neq i$. Then the prover picks a pair $\langle y, u'\rangle$ such that $(x, y) \in m_{i'}$ and $s_j^{i'} \xrightarrow{y}_{i'} u'$ and $u' = s_{j+1}^{i'}$.

This constitutes a *round* of the game. If in a round, after the disprover has made its move, the prover can also make a move according to the moves described above, then we say that the prover has a *matching move* in that round.

The game continues until one of the players wins. The prover wins the game if either in the last position of the play, no player can move, or there is no further allowable move by the disprover. The prover also wins, if in the play a position is repeated. In both cases, the disprover has failed to expose a distinction between the transition systems.

The disprover wins, if in the last position of the play is not a winning position which means the disprover has been able to force the prover to a non winning position of the game or if in the last position, the disprover has an allowable move but the prover does not have a matching move.

A *strategy* for a player is a set of rules which tells how to make a move depending on the partial play and the previous moves of the opponent so far. A strategy is said to be *history-free* if it only depends on the most recent move. A strategy is a *winning strategy* for a player if, for any strategy by the opponent, the strategy always causes the player to win.

A game G in the Stirling class is called a *characteristic game* for a relation R between two finite-state processes, if the following condition holds: Whenever the game G be played on two transition systems T_1 and T_2 with start position $\langle s, t\rangle$, then the prover has a history-free winning strategy if and only if sRt.

The following was shown in [SHR96]:

THEOREM 3.9. *All equivalences in the linear-branching time hierarchy have characteristic games.*

EXAMPLE 3.4. (**A characteristic game for ready simulation**) $Rsim-game$ is a game in the Stirling class with the following parameters: $R_1 = R_2 = A$, $m_1, m_2 = \iota$, $\Gamma = \{\langle s,t\rangle \mid s \in S_1, t \in S_2 \wedge init(s) = init(t)\}$, $\Sigma = \{\langle s_1, s_2\rangle\}$, $M = \{1\}$, $r = \mid S_1 \mid * \mid S_2 \mid +1$.
□

For certain games in the Stirling class, the problem whether the prover has a winning strategy is directly reducible to the NHORNSAT problem. Hence, for any behavioural relation R, whose characteristic game is in this subclass, the decision problem for R is reducible to the NHORNSAT problem. This immediately leads to a polynomial time algorithm for the problem of checking that relation, provided one can create an instance of the game from the instance of the relational problem in polynomial time. For all the games in the Stirling class, such a transformation to the game instance can be shown to done in polynomial time, provided that the winning positions can be decided in polynomial time. Hence, we get a sufficiency condition as to under what condition a behavioural relation between finite state processes is polynomial time decidable.

THEOREM 3.10. *Whenever a game G in the Stirling class satisfies the following conditions:*

—*The game languages R_1 and R_2 are finite and explicitly enumerated. For example, in ready simulation game $R_1 = R_2 = A$, where A is the set of action symbols.*

—*The representation of the set of winning positions is either by an explicit listing or such that determining if a position of the game is a winning position is polynomial time decidable.*

then it is polynomial-time decidable whether the prover has a winning strategy for G.

An immediate corollary is

Corollary 3.1. *Any behavioural relation between two finite state transition systems, whose characteristic game satisfies the conditions listed above, is decidable in polynomial time.*

It can be shown that all simulation-like equivalences satisfy the conditions of Theorem 3.10, and this again shows Theorem 3.6.

3.4 Trace-like equivalences and preorders

The **trace-like** equivalences and preorders are defined in terms of the behaviours of the processes on unbounded sequences of actions (traces). The trace-like equivalences are *trace equivalence*, *completed trace equivalence*, *failure-trace equivalence*, *ready-trace equivalence*, *failure equivalence* , *readiness equivalence*, *n-bounded bisimulation equivalences.*

One can give characteristic characterizations of all trace-like equivalences (cf. the previous section) and show that the existence of a winning strategy can be decided in PSPACE for any such game; this shows the following:

Theorem 3.11. *All trace-like equivalences are in PSPACE for the case of finite-state processes.*

However, we can say much more. The main result of this section is that

Theorem 3.12. *All trace-like equivalences are PSPACE-complete for the case of finite-state processes.*

3.4.1 *Completed trace equivalence.* Given an arbitrary labelled transition system $(Pr, Act, \rightarrow)$, we can define the notion of completed traces as follows:

Definition 3.11. *Let a labelled transition system $(Pr, Act, \rightarrow)$be given. The set of* completed traces *of a state $p \in Pr$ is defined by*

$$\mathrm{ctraces}(p) = \{w \in Act^* \mid p \xrightarrow{w} p' \textit{ where} p' \not\rightarrow \}$$

Two states p and q are completed trace equivalent, *written $\sim_{ctr}$, if* $\mathrm{ctraces}(p) = \mathrm{ctraces}(q)$.

Thus completed trace equivalence is in all essential the well-known language equivalence from automata theory. The following result is well-known and be found in e.g. [KS90]:

Theorem 3.13. *The completed trace equivalence problem is PSPACE-complete for the class of finite transition systems.*

Completed trace equivalence can be seen as the equivalence closure of the completed trace preorder:

Definition 3.12. *Given a labelled transition system the* completed trace preorder $\sqsubseteq_{ctr}$ *is defined as follows: $p \sqsubseteq_{ctr} q$ if* $\mathrm{ctraces}(p) \subseteq \mathrm{ctraces}(q)$.

The following is immediate:

Theorem 3.14. *The completed trace preorder problem is PSPACE-complete for the class of finite transition systems.*

PROOF. By Theorem 3.11, we see that the completed trace preorder problem is in PSPACE. Showing that the preorder problem is PSPACE-hard follows from the fact that

$$p + q \sim_{ctr} q \text{iff} p \sqsubseteq_{ctr} p$$

which immediately shows that the equivalence problem is polynomial-time reducible to the corresponding equivalence problem. In fact, for all trace-like preorders, $+$ acts as a least upper bound operator, so the above reduction applies to *any* trace-like preorder. □

3.4.2 *Trace equivalence.* The notion of trace equivalence considers *arbitrary* (incomplete) traces.

DEFINITION 3.13. *Let a labelled transition system* $(Pr, Act, \rightarrow)$*be given. The set of* traces *of a state* $p \in Pr$ *is defined by*

$$\text{traces}(p) = \{w \in Act^* \mid p \xrightarrow{w} p'\}$$

Two states p *and* q *are* trace equivalent*, written* $\sim_{tr}$*, if* $\text{traces}(p) = \text{traces}(q)$.

The following was shown by Kanellakis and Smolka [KS90]:

THEOREM 3.15. *[KS90] Trace equivalence is PSPACE-complete for finite labelled transition systems.*

DEFINITION 3.14. *Given a labelled transition system the* trace preorder $\sqsubseteq_{ctr}$ *is defined as follows:* $p \sqsubseteq_{tr} q$ *if* $\text{traces}(p) = \text{traces}(q)$.

THEOREM 3.16. *The trace preorder problem is PSPACE-complete for the class of finite transition systems.*

PROOF. Along the same lines as the proof of Theorem 3.14. □

3.5 n-bounded-tr-bisimulation

We next consider n–bounded-tr-bisimulation. This equivalence is a generalisation of *trace equivalence* and *possible futures equivalence*, in that 1-bounded-tr-bisimulation corresponds to *trace equivalence* and 2-bounded-tr-bisimulation is the *possible futures equivalence* of [RB81].

DEFINITION 3.15. *We define* n-bounded-tr-bisimulation*, written* $\sim_{tr}^{n}$*, inductively as follows.*

—$p \sim_{tr}^{0} q$ *for all processes* p *and* q*,*
—$p \sim_{tr}^{n+1} q$ *iff*
 —*if* $p \xrightarrow{w} p'$ *then* $\exists q'$ *such that* $q \xrightarrow{w} q'$ *and* $p' \sim_{tr}^{n} q'$ *and*
 —*if* $q \xrightarrow{w} q'$ *then* $\exists p'$ *such that* $p \xrightarrow{w} p'$ *and* $p' \sim_{tr}^{n} q'$*.*

This notion of equivalence also arises naturally as the consecutive approximations of bisimulation equivalence [Mil80; Mil89].

Kanellakis and Smolka have shown that the n-bounded-tr-bisimulation problem is PSPACE-complete for finite transition systems.

THEOREM 3.17. *[KS90] For alle* $n > 0$*, the* n*-bounded-tr-bisimulation problem is PSPACE-complete for finite transition systems.*

For finitely branching transition graphs, and therefore for finite processes, the limit of the n-bounded-tr-bisimulations for $n \to \omega$ is bisimulation equivalence:

THEOREM 3.18. [Mil89] *For any finitely branching labelled transition graph we have*

$$\sim \; = \bigcap_{n=0}^{\omega} \sim_{tr}^{n}$$

3.6 Failures, readiness, failure-trace and ready-trace equivalences

Failures equivalence was suggested by Hoare etal in [BHR84; Hoa84]; for finite transition systems it coincides with the notion of *testing equivalence* proposed by Hennessy and de Nicola.

The notion of readiness equivalence can be seen as the dual of failures equivalence and was originally put forward by Bergstra, Klop, and Olderog [BKO88]

DEFINITION 3.16. *For any process p, define*

$$\text{failures}(p) = \{(w, X) \mid \exists p' : p \xrightarrow{w} p', \forall a \in X : p' \not\xrightarrow{a} \},$$

$$\text{readies}(p) = \{(w, X) \mid \exists p' : p \xrightarrow{w} p', p' \xrightarrow{a} \iff a \in X\}.$$

Processes p and q are failures equivalent*, written $p \sim_f q$, iff* $\text{failures}(p) = \text{failures}(q)$. *Processes p and q are* readiness equivalent*, written $p \sim_r q$, iff* $\text{readies}(p) = \text{readies}(q)$.

These equivalences could also be defined via the associated preorders:

DEFINITION 3.17. *Processes p and q are related by the* failures preorder*, written $p \sqsubseteq_f q$, iff* $\text{failures}(p) \subseteq \text{failures}(q)$. *Processes p and q are related by the* readiness preorder*, written $p \sqsubseteq_r q$, iff* $\text{readies}(p) \subseteq \text{readies}(q)$.

To show that the equivalences defined in this section are PSPACE-hard, we shall employ a class of processes introduced by Huynh and Tian [HT90], called *locally unary* processes, for which failures equivalence and readiness equivalence coincide with completed trace equivalence.

DEFINITION 3.18. *[HT90] A process p is* locally unary *iff for each p' with $p \xrightarrow{w} p'$ there is at most one $a \in Act$ such that $p' \xrightarrow{a}$.*

LEMMA 3.2. [HT90] *If p and q are locally unary normed processes then*

$$p \sim_r q \quad \textit{iff} \quad p \sim_f q \quad \textit{iff} \quad \textit{traces}(p) = \textit{traces}(q).$$

The idea is now, given a Δ to construct a locally unary Δ' containing the variables of Δ such that $\text{traces}(X) = \text{traces}(Y)$ in Δ if and only if $\text{traces}(X) = \text{traces}(Y)$ in Δ'. The following construction accomplishes this. The idea is simply to precede any action by a # that indicates that a nondeterministic choice has been made.

DEFINITION 3.19. *Given a system of regular process equations Δ let Δ' have the action set $Act \cup \{\#\}$ (where # is a new action) and process variables Var. For every process equation in Δ*

$$X_i \stackrel{\text{def}}{=} \sum a_j p + \sum b_k.\mathbf{0}$$

create the new equation

$$X_i \stackrel{\text{def}}{=} \sum \#.a_j.p + \sum \#.b_k.\mathbf{0}$$

in the new system Δ'.

It is obvious that Δ' is a system of regular process equations iff Δ and that the construction of Δ' can be accomplished in polynomial time w.r.t the size of Δ.

We immediately see that the resulting process is locally unary.

PROPOSITION 3.2. Δ' *is locally unary.*

The following is now obvious from the definition of Δ'.

PROPOSITION 3.3. *For* $X \in Var$ *we have* $X \xrightarrow{a}_\Delta p'$ *iff* $X \xrightarrow{\#}\xrightarrow{a}_{\Delta'} p'$.

We therefore also see that

PROPOSITION 3.4. *Let* Δ *be a system of equation in normal form. For* $X \in Var$ $b_1 b_2 ... b_n \in traces(X)$ *relative to* Δ *iff* $\#b_1\#b_2...\#b_n \in traces(X)$ *relative to* Δ'.

THEOREM 3.19. *[HT90] Failure and ready equivalence are PSPACE-hard for locally unary regular processes.*

PROOF. From Proposition 3.4 we get a polynomial-time reduction from language equivalence to language equivalence for locally unary normed processes and the theorem now follows from Lemma 3.2. □

The above ideas can also be used to prove that failure trace and ready trace equivalence are PSPACE-hard. For finite transition systems, failure trace equivalence [vG90a] coincides with the notion of refusal testing [Phi87].

DEFINITION 3.20. *The* refusal relation $\xrightarrow{A}$ *for* $A \subseteq Act$ *is defined for any processes* p, q *by* $p \xrightarrow{A} q$ *iff* $p = q$ *and whenever* $a \in A$, $p \not\xrightarrow{a}$. *The* failure trace relations $\xrightarrow{u}$ *for* $u \in (Act \cup \mathcal{P}(Act))^*$ *are defined as the reflexive and transitive closure of the refusal and transition relations. Define*

$$\text{failure-traces}(p) = \{u \in (Act \cup \mathcal{P}(Act))^* \mid \exists p' : p \xrightarrow{u} p'\}.$$

Two processes p *and* q *are* failure-trace equivalent, *written* $p \sim_{\text{ftr}} q$ *iff* failure-traces(p) = failure-traces(q).

LEMMA 3.3. *If* p *and* q *are locally normed unary processes then* $p \sim_{\text{ftr}} q$ *iff* $traces(p) = traces(q)$.

COROLLARY 3.2. *The failure trace equivalence problem is PSPACE-hard for locally unary regular processes.*

The definition of ready trace equivalence, that we shall use here, is the characterisation given by van Glabbeek [vG90a].

Definition 3.21. *Define*

$$\text{ready-trace}(p) = \{A_0 a_1 A_1 \dots a_n A_n \mid$$

$$\exists p_0, \dots, p_n : p = p_0 \xrightarrow{a_1} p_1 \cdots \xrightarrow{a_n} p_n,\ p_i \xrightarrow{a} \iff a \in A_i, 0 \leq i \leq n\}.$$

Two processes p *and* q *are* ready trace equivalent, *written* $p \sim_{rtr} q$, *iff* $\text{ready-trace}(p) = \text{ready-trace}(q)$.

Lemma 3.4. *If* p *and* q *are locally unary processes then* $p \sim_{rtr} q$ *iff* $L(p) = L(q)$.

Corollary 3.3. *The ready trace equivalence problem is PSPACE-complete for finite transition systems.*

3.6.1 *Subclasses of regular processes.* What happens if one considers only certain classes of finite-state transition systems ? This section summarizes the known results.

Tree processes. A regular process is called a *tree process* if its associated transition system can be unfolded into a finite tree, or equivalently, if its definition does not use recursion. Huynh and Tian showed that for tree processes, bisimulation is in NC, the class of problems decidable by non-uniform boolean circuits [HT90]. Hence, the bisimulation algorithm for tree processes is seen to be efficiently parallelizable.

Unary and locally unary processes. The proof of P-completeness of bisimulation due to Alvarez et al. consists of providing a log-space-reduction from the Alternating Monotone Fanin 2, Fanout 2 Circuit Value problem (AM2CVP) which is a well known P-complete problem [GHR95]. Given an instance of AM2CVP, the reduction constructs two unary processes such that the two processes are bisimilar if and only if the AM2CVP instance has output 1. As the processes constructed are tree processes, we immediately get that bisimulation equivalence for unary tree processes is P-complete.

Deterministic processes. A process is called *deterministic* if its associated transition system $(Pr, Act, \rightarrow)$ is deterministic in the sense that for any $p \in Pr$ and $a \in Act$ there is at most one p' such that $p \xrightarrow{a} p'$. One should note that for deterministic processes, trace equivalence and bisimulation equivalence coincide [Eng85] (where $\sim$ denotes strong bisimulation equivalence):

Proposition 3.5. *If* p *and* q *are deterministic processes, then* $Tr(p) = Tr(q)$ *iff* $p \sim q$.

Proof. $\{(p, q) \mid Tr(p) = Tr(q)\}$ is a bisimulation. □

Consequently, in the deterministic case the linear/branching time hierarchy collapses, and in this case *all* equivalences are in P.

More can be said, though. For deterministic transition systems, the bisimulation equivalence problem is in **NC**. By the above proposition, all equivalences are in **NC** for deterministic transition systems. In [HT94], it was proved that all these relations are in **NL**, which also implies that they are in **NC**.

4. Infinite transition systems

All equivalences are undecidable for sufficiently rich classes of labelled transition systems. For instance, it is well-known that bisimulation equivalence is undecidable for the full CCS calculus. In this section we shall briefly consider two extensions of the class of regular processes which have a decidable equivalence problem and survey the known decidability results. For a survey of known decidability results and their underlying proof techniques, the reader is referred to [HiM94].

4.1 Basic Process Algebra

The class of BPA (Basic Process Algebra) was defined by Bergstra and Klop in [BK84b]. The abstract syntax of BPA is given by

$$p ::= a \mid p_1 \cdot p_2 \mid p_1 + p_2 \mid X$$

Again, a ranges over a set Act of atomic actions, and X over a set Var of variables. As before, the symbol $+$ is the non-deterministic choice while $p_1 \cdot_2$ represents the sequential composition of p_1 and p_2 (one usually omits the '$\cdot$').

We say that a process expression is *guarded* iff every variable occurrence in p occurs in a subexpression aq of p. As in the case of regular processes, BPA processes are defined by a declaration, a finite set Δ of guarded equations

$$\Delta = \{X_i \stackrel{\text{def}}{=} p_i \mid 1 \leq i \leq k\}$$

– only now the p_i are guarded BPA expressions with free variables in $Var(\Delta) = \{X_1, \ldots, X_k\}$. The definition conventions of regular processes still apply.

The operational semantics of a BPA process expression, given a finite system of guarded equations Δ, is given by a labelled transition system $(Pr, Act, \rightarrow)$where Pr is the set of processes with variables being the variables of Δ and the transition relation $\rightarrow$ defined by the following rules (ϵ denotes the empty process with the convention that ϵq is q):

$$\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$\frac{p \xrightarrow{a} p'}{pq \xrightarrow{a} p'q} \qquad a \xrightarrow{a} \epsilon \quad a \in Act$$

$$\frac{p \xrightarrow{a} p'}{X \xrightarrow{a} p'} \qquad X \stackrel{\text{def}}{=} p \in \Delta$$

BPA processes are also known as *context-free processes*, as it can be shown (cf. Section 3.4.2) that a language L over the alphabet A is context-free iff $L =$ traces(X_1) for some BPA process Δ with root X_1 and action set A.

DEFINITION 4.1. *The* norm *of a process p is defined by*

$$|p| = \min\{length(w) \mid p \xrightarrow{w} \epsilon\}.$$

A finite set Δ of guarded equations is normed if for all $X \in Var$ it holds that $|X|$ is finite. A BPA process is called normed, if it has been generated via a normed set of guarded equations.

Note that the class of normed BPA processes does not include all the regular processes (such as $X \stackrel{\mathrm{def}}{=} aX$). Still, it is a very rich family, including processes with infinitely many states.

THEOREM 4.1. *Bisimulation equivalence is decidable for BPA processes.*

This result was originally first obtained for normed processes [BBK87] and can most easily be obtained via a finite representation theorem [Cau88]. This theorem states that the maximal bisimulation of any normed BPA transition graph is the congruence closure (under sequential composition) of a *bisimulation base*, a finite relation whose congruence closure has a decidable membership problem. The existence of a search procedure for such a bisimulation base is established by means of *unique factorization theorem*; Hirshfeld et al. have shown that this search can be done in time polynomial w.r.t. the size of the BPA process declaration.

THEOREM 4.2. *[HJM94] Bisimulation equivalence is decidable in time polynomial in the size of Δ for any normed BPA process.*

The general result can be shown using a more general notion of bisimulation base, which only requires semi-decidability of the congruence closure of the bisimulation base. The original decision procedure for bisimilarity for BPA processes relied on the conjunction of two semi-decision procedures [CHS92], one searching for a bisimulation base and another searching for a bisimulation error. It has recently been shown by Burkart, Caucal and Steffen [BCS95] that the search for a bisimulation base can be bounded, giving an elementary complexity bound.

THEOREM 4.3. *Bisimulation equivalence is decidable in elementary time in the size of Δ for any BPA process.*

However, no other equivalence or preorder is decidable, as was shown by Huynh and Tian and Groote and Hüttel.

THEOREM 4.4. *[GH94; HT90] All other equivalences and corresponding preorders of the linear-branching time hierarchy are undecidable.*

The undecidability proofs for the preorders all proceed by reductions from the trace inclusion problem for simple grammars, which was shown undecidable by Friedman in [Fri76]. The undecidability of the corresponding equivalences proceed either by reductions from the preorder problem, using the fact that $+$ acts as a least upper bound operator w.r.t. to the preorder (cf. the proof of Theorem 3.14) or by reductions from the language equivalence problem for context-free grammars.

In the deterministic case, however, the linear/branching time hierarchy collapses so all equivalences are in P in the normed case [Cau89] and elementary-time decidable in the general case [BCS95]. But as deterministic BPA processes correspond exactly to the class of simple grammars, all preorder problems remain undecidable.

$$\frac{p \xrightarrow{a} p'}{p \parallel q \xrightarrow{a} p' \parallel q} \qquad \frac{q \xrightarrow{a} q'}{p \parallel q \xrightarrow{a} p \parallel q'}$$

Table i. Additional transition rules for the merge operator

4.2 Basic Parallel Processes

Another extension of the class of regular processes is the class of BPP (Basic Parallel Processes), first considered by Christensen [Chr94]. In the case of BPP, the non-communicating parallel (merge) operator has been added to the syntax. The class of BPP processes can be shown to correspond to the class of communication-free Petri nets [HiM94].

The abstract syntax of BPP is given by

$$p ::= a \mid p_1 \parallel p_2 \mid p_1 + p_2 \mid X$$

As for BPA, processes are defined by declarations of the form $\{X_i \stackrel{\text{def}}{=} p_i \,|\, 1 \leq i \leq k\}$, where the p_i now are BPP process terms. The operationel semantics of BPP extends the semantics of regular processes with rules for the parallel operator; the transition rules are found in Table 4.2.

The following result was shown by Hirshfeld, Jerrum and Moller [HJM96] by appealing to a finite characterization theorem similar to that applied to the BPA case of the previous section.

THEOREM 4.5. *Bisimilarity is in* **P** *for normed BPP processes.*

It is also known that bisimilarity is decidable for the full BPP calculus [CHM93]; however, at the time of writing, the complexity bound for the bisimulation problem in this case remains an open problem.

ACKNOWLEDGMENTS

The authors would like to thank Rob van Glabbeek for comments on the linear-branching time hierarchy.

REFERENCES

C. Alvarez, J.L Balcazar, J. Gabarro, and M Santha, *Parallel complexity in the design and analysis of concurrent systems*, PARLE91, Lecture Notes in Computer Science 505 Springer-Verlag, 1991.

G. Ausiello and G. F. Italiano, *On-line algorithms for polynomially solvable satisfiability problems*, Journal of Logic Programming **10** (1991), 69–90.

S. Abramsky. Observational equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.

H. R. Andersen, *Verification of technical properties of concurrent systems*, Tech. Report DAIMI PB-445, Computer Science Department, Aarhus University, Aarhus University, Denmark, 1993.

H. R. Andersen, *Model checking and boolean graphs*, Theoretical Computer Science **126** (1994), no. 1, 3–30.

J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, *Symbolic model checking:* $10^2 0$ *states and beyond*, Information and Computation **98** (1992), no. 2, 142–170.

J.A. Bergstra and J.W. Klop Process algebra for synchronous communication *Information and Computation*, 60:109–137, 1984.

J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings PARLE conference,* Eindhoven*, Vol. II (Parallel Languages)*, volume 259 of *Lecture Notes in Computer Science*, pages 94–113. Springer-Verlag, 1987.

O. Burkart, D. Caucal, B. Steffen. An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes In Jirí Wiedermann and Petr Hájek, editors: *Proceedings of Mathematical Foundations of Computer Science 1995, 20th International Symposium* Volume 969 of *Lecture Notes in Computer Science*, Springer-Verlag 1995.

J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Readies and failures in the algebra of communicating processes. *SIAM J. on Comput.*, 17:1134–1177, 1988.

B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990.

S.D. Brookes, C.A.R. Hoare, and W. Roscoe. A theory of communicating sequential processes. *JACM*, 31:560–599, 1984.

B. Bloom and R. Paige, *Transformational design and implementation of a new efficient solution to the ready simulation problem*, Draft (1994).

D. Caucal. A Fast Algorithm to Decide on Simple Grammars Equivalence. In H. Djidjev, editor: *Proceedings of the International Symposium on Optimal Algorithms* Volume 401 of *Lecture Notes in Computer Science*, pages 66–85. Springer-Verlag, 1989.

D. Caucal. Graphes canoniques de graphes algébriques. *Theoretical Informatics and Applications* 24:339–352, 1990.

U. Celikkan and R. Cleaveland, *Generating diagnostic information for behavioural preorders*, Proceedings of Computer Aided Verification: 1992, Lecture Notes in Computer Science 663, 1992, pp. 370–383.

R. Cleveland and M. Hennessy, *Testing equivalence as a bisimulation equivalence*, Formal Aspects of Computing **3** (1992).

S. Christensen. Decidability and Decomposition in Process Algebras, Ph.D. thesis, University of Edinburgh 1994.

S. Christensen, Y. Hirschfeld, F. Moller. Decomposability, decidability, and axiomatisability for bisimulation equivalence on basic parallel processes. To appear in *Proceedings* 8^{th} *Annual Symposium on Logic in Computer Science,* Montreal, Canada. IEEE, 1993. Also published as LFCS Report ECS-LFCS-92-244, University of Edinburgh, 1992.

S. Christensen, H. Hüttel, C.P. Stirling. Bisimilarity is decidable for all context-free processes. In R. Cleaveland, editor, *Proceedings of CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 138–147, 1992.

R. Cleveland and B. Steffen, *Linear time model checking algorithm for alternation-free modal mu calculus*, Formal Methods of Software Design **2** (1993), 127–147.

W.F. Dowling and J.H. Gallier, *Linear time algorithm for testing the satisfiability of propositional horn formulae*, Journal of Logic Programming **3** (1984), 267–284.

J. Engelfriet Determinacy $\rightarrow$ (observation equivalence = trace equivalence) *Theoretical Computer Science* 36:21–25, 1985.

J. C. Fernandez and L. Mounier, *On the fly verification of behavioural equivalences and preorders*, The 3rd International Workshop on Computer Aided Verification 1991, Lecture Notes in Computer Science 575, 1991, pp. 181–191.

E.P. Friedman. The inclusion problem for simple languages. *Theoretical Computer Science*, 1:297–316, 1976.

R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.

R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to parallel computation: P-completeness theory*, Oxford University Press, 1995.

J.F. Groote. Transition system specifications with negative premises. Report CS-R8950, CWI, 1989. An extended abstract appeared in J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90*, Amsterdam, LNCS 458, pages 332–341. Springer-Verlag, 1990.

J.F. Groote. A short proof of the decidibility of bisimulation for normed BPA-processes. *Information Processing Letters*, 42:167–171, 1992.

J.G. Groote and H. Hüttel. Undecidable Equivalences for Basic Process Algebra *Information and Computation*, 115:354–371, 1994.

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 Del la Rocca, editors, *Proceedings of ICALP89*, volume 372 of *LNCS*, pages 423–438. Springer-Verlag, 1989. Full version to appear in *Information and Computation*.

J.F. Groote and F.W. Vaandrager, *Structured operational semantics and bisimulation as a congruence*, Information and Computation **100** (1992), no. 2, 202–260.

M. Hennessy. *Algebraic Theory of Processes*. MIT Press, Cambridge, Massachusetts, 1988.

M. R. Henzinger, T. Henzinger, and P. W. Kopke, *Computing simulations on finite and infinite graphs*, Proceedings of IEEE Conference on Foundations of Computer Science, 1995.

Y. Hirshfeld and M. Jerrum and F. Moller. A Polynomial-Time Algorithm for Deciding Equivalence of Normed Context-Free Processes In Shafi Goldwasser, editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 623–631, IEEE Computer Society Press, 1994.

Y. Hirshfeld and M. Jerrum and F. Moller. A Polynomial-Time Algorithm for Deciding Bisimulation Equivalence of Normed Basic Parallel Processes *Mathematical Structures in Computer Science*, 6:251–259, 1996.

Y. Hirshfeld and F. Moller. Decidability Results in Automata and Process Theory In G. Birtwistle and F. Moller, editors, *Proceedings of Logics for Concurrency: Automata vs Structure. The VIII Banff Higher Order Workshop*, volume 1043 of *LNCS*, Springer-Verlag 1994.

M. Hennessy and R. Milner, *Algebraic Laws for Nondeterminism and Concurrency*, Journal of the ACM, **32**(1), 1985, pp. 137–161.

C. A. R. Hoare, *Communicating sequential processes*, Prentice Hall International, 1984.

H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, The Netherlands, pages 376–386. IEEE Computer Society Press, 1991.

D.T. Huynh and L. Tian. On deciding readiness and failure equivalences for processes. Technical Report UTDCS-31-90, University of Texas at Dallas, September 1990.

Dung T. Huynh and Lu Tian, *On deciding some equivalences for concurrent processes*, Theoretical Informatics and Applications **28** (1994), no. 1, 51–71.

J. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

H. Hüttel. Dedidability, Behavioural Equivalences and Infinite Transition Graphs Ph.D. thesis. Report ECS-LFCS-91-191, University of Edinburgh, 1991.

A.J. Korenjak and J.E. Hopcroft Simple Deterministic Languages In *Proc. Seventh Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46, 1966.

D. Kozen, *Results on the propositional mu-calculus*, Theoretical Computer Science **27** (1983).

Paris C Kanellakis and Scott A Smolka, *CCS expressions, finite state processes and three problems of equivalence*, Information and Computation **86** (1990), 43–68.

K. G. Larsen, *Efficient local correctness checking*, CAV 92, Lecture Notes in Computer Science 663, 1992, pp. 30–43.

K. G. Larsen, *Proof Systems for Hennessy Milner Logic with Recursion*, CAAP 88, Lecture Notes in Computer Science 299, 1988.

K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Proceedings* 16^{th} *ACM Symposium on Principles of Programming Languages,* Austin, Texas, pages 344–352, 1989.

X. Liu, *Specification and Decomposition in Concurrency*, PhD Thesis, Department of Mathematics and Computer Science, Aalborg University Center, Denmark, 1992.

Nancy Lynch and Frits Vaandrager, *Forward and backward simulation:untimed systems*, REX Workshop on Real Time systems, 1991.

Milner, R. A Calculus of Communicating Systems Springer-Verlag LNCS 92, 1980.

Milner, R. Communication and Concurrency Prentice-Hall International 1989.

Milner, R. A Complete Inference System for a Class of Regular Behaviours Journal of Computer and System Sciences, 28:439-466, 1984.

E.-R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, 23:9–66, 1986.

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

I.C.C. Philips. Refusal testing. *Theoretical Computer Science*, 50:241–284, 1987.

Gordon D. Plotkin, *A structural approach to operational semantics*, Tech. Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus University, Denmark, 1981.

Robert Paige and Robert E Tarjan, *Three partition refinement algorithms*, SIAM Journal Of Computing **16** (1987), 973–989.

W.S. Rounds and S.D. Brookes. Possible futures, acceptances, refusals and communicating processes. In *Proc. 22nd Annual Symposium on Foundations of Computer Science*, pages 140–149, New York, 1981. IEEE.

F. Moller and S.A. Smolka. On the Computational Complexity of Bisimulation. *ACM Computing Surveys*, 27:287-289, June 1995.

Thomas J. Schaefer, *The complexity of satisfiability problems*, Tenth Annual Symposium on Theory of Computing, 1978.

S. K. Shukla, H. B. Hunt III, and D. J. Rosenkrantz, *Hornsat, model checking, verification, and games*, Research Report TR-95-8, Department of Computer Science, SUNY Albany, 1995.

S. K. Shukla, H. B. Hunt III, and D. J. Rosenkrantz, *Hornsat, model checking, verification and games*, In Proceedings of CAV'96 (1996).

S. K. Shukla, D. J. Rosenkrantz, H. B. Hunt III, and R. E. Stearns *The Polynomial Time Decidability of Simulation Relations for Finite State Processes: A HORNSAT Based Approach*, Presented at the DIMACS Workshop on Satisfiability Problems March 1996. To be included in AMS DIMACS special volume.

S. K. Shukla, H. B. Hunt III, D. J. Rosenkrantz, and R. E. Stearns, *On the complexity of relational problems for finite state processes*, In Proceedings of ICALP 1996 (1996).

O. Sokolsky and S. A. Smolka, *Incremental model checking in the modal mu-calculus*, Proceedings of CAV'94, 1994.

J. D. Ullman, *Principles of database and knowledge base systems : Volume i*, Computer Science Press, Rockville, MD, 1988.

R.J. van Glabbeek, *The linear time - branching time spectrum*, Tech. Report CS-R9029, Computer Science Department, CWI, Centre for Mathematics and Computer Science, Netherlands, 1990.

D. Walker, *Bisimulation and divergence*, Proceedings of the Third Annual Symposium on Logic in Computer Science, 1988, pp. 186–192.

R. Cleveland and B. Steffen. Computing behavioural relations, logically. In *Proceedings of ICALP 91*, Springer LNCS, pages 127–138, 1991.

B. U. Steffen. Characteristic formulae for ccs with divergence. In *Proceedings of ICALP 89, LNCS 372*, pages 723–733, 1989.

C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

Colin Stirling. Modal and temporal logics for processes. In *Notes for Summer School in Logic Methods in Concurrency*, pages Department of Computer Science, Aarhus University, 1993.

C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

A. Tarski. A lattice theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.