DUNG T. HUYNH

LU TIAN

On deciding some equivalences for concurrent processes

RAIRO. Informatique théorique et applications, tome 28, nº 1 (1994), p. 51-71

<http://www.numdam.org/item?id=ITA_1994__28_1_51_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 28, n° 1, 1994, p. 51 à 71)

ON DECIDING SOME EQUIVALENCES FOR CONCURRENT PROCESSES (*)

by DUNG T. HUYNH ([1]) and LU TIAN ([1])

Communicated by J. GABARRO

Abstract. – *In this paper, we study the complexity of some equivalences for finite processes and for normed context-free processes. The results are as follows.* (1) *For normed context-free processes, the determinacy problem with respect to any equivalence between bisimulation and failure equivalence is decidable. Determining whether a given (general) normed context-free process is equivalent to a given normed determinate context-free process with respect to any equivalence between bisimulation and failure equivalences is decidable. In fact, these two types of problems are both in* Σ_2^p. *However, they are undecidable with respect to simulation, trace and finite trace equivalences.* (2) *For finite processes, the determinacy problem with respect to any equivalence between bisimulation and failure equivalences is* **NL**-*complete. Determining the equivalence of a (general) finite process and a determinate finite process with respect to any equivalence between bisimulation and failure equivalences is* **NL**-*complete. For determinate finite processes, all equivalences are* **NL**-*complete.* (3) *For finite processes, simulation and ready-simulation equivalences are in* **P**, *whereas ready-trace and failure-trace equivalences are* **PSPACE**-*complete.*

Résumé. – *Dans cet article, on étudie la complexité de certaines équivalences pour les processus finis et pour les processus algébriques normés. Les résultats sont les suivants :* (1) *Pour les processus algébriques normés, le problème de la désambiguation (par quotient) est décidable pour toute équivalence entre la bisimulation et l'équivalence par échec. De plus, on peut décider si un processus algébrique normé est équivalent à un processus algébrique normé déterministe, où l'équivalence est située entre la bisimulation et l'équivalence par échec. En fait, ces deux problèmes sont dans* Σ_2^p. *Cependant, ils sont indécidables pour la simulation et les équivalences de traces et de traces finies.* (2) *Pour les processus finis, le problème de la désambiguation pour toute équivalence entre la bisimulation et l'équivalence par échec est* **NL**-*complet. Il en est de même du problème de l'équivalence entre un processus fini déterministe. Pour les processus finis déterministes, toutes ces équivalences sont* **NL**-*complet.* (3) *Pour les processus finis, la simulation et la simulation "ready" sont dans* **P** *tandis que les équivalences de traces "ready" et "failure" sont* **PSPACE**-*complet.*

1. INTRODUCTION

Process equivalences are introduced in various theories of communication and concurrency in order to verify the behaviors of concurrent systems.

(*) Received May 1992, acepted April 1993.
([1]) Computer Science Program, University of Texas at Dallas, Richardson, TX 75083.

In [12] one finds the linear/branching time hierarchy of process semantics in which the coarsest one is trace equivalence ($\sim_{tr}$) and the finest one is bisimulation equivalence ($\sim_{bis}$) [31]. Between them are failure equivalence ($\sim_{f}$) [4], failure-trace equivalence (refusal testing) ($\sim_{fat}$) [32], readiness equivalence ($\sim_{r}$) [30, 6], ready-trace equivalence ($\sim_{rt}$) (*cf.* [3]), simulation equivalence ($\sim_{sim}$) [31], ready-simulation (2/3-bisimulation) equivalence ($\sim_{rsim}$) [5, 26] and so on.

A fundamental research issue is to study these process equivalences from the complexity-theoretic view point. Thus, the basic question is how hard it is to determine whether one process is equivalent to another. Unfortunately, all these process equivalences are undecidable for processes specified by expressions in the general framwork of any algebraic theory of concurrency. However, for finite processes, it is obvious that all these equivalences are decidable. Indeed, Kanellakis and Smolka showed in [27] that failure equivalence is **PSPACE**-complete, and Huynh and Tian showed in [18] that readiness equivalence is **PSACE**-complete, whereas Alvarez, Balcázar, Cabarró and Santha showed in [1] that bisimulation equivalence is **P**-complete.

For normed context-free processes, Baeten, Bergstra and Klop showed in [2] that bisimulation equivalence is decidable. Recently, Huynh and Tian showed in [21] that bisimulation equivalence for normed context-free processes is in $\Sigma_2^{\mathbf{p}}$, the second level of the polynomial time hierarchy, thereby improving the results in [7] and [13]. In [2], Baeten, Bergstra and Klop raised as an open question the decidability of readiness and failure equivalences. Unfortunately, Huynh and Tian showed in [18] that readiness and failure equivalences are undecidable for normed context-free processes. Subsequently, Groote and Hüttel showed in [14] that, except bisimulation equivalence, none of the other process equivalences in the linear/branching time hierarchy [12] is decidable for normed context-free processes.

To obtain a more complete answer to the open question in [2], we will show in this paper that for two normed context-free process graphs, if one of them is determinate, then all process equivalences between bisimulation and failure equivalence coincide and are therefore decidable in $\Sigma_2^{\mathbf{p}}$. This should be contrasted with a well known result in the theory of formal languages that it is undecidable whether an arbitrary context-free grammar is equivalent to a simple grammar. (Note that context-free grammars in Greibach normal form correspond to context-free processes and simple grammars to deterministic context-free processes [*cf.* Section 5].

Tables 1 and 2 contain main existing results and new results obtained in this paper. We hope the our results provide a more complete complexity classification of equivalences for concurrent processes.

TABLE I

Complexity of process equivalences for normed context-free processes.

Normed CFPs M and N	M and N are determinate	One of M and N is determinate	M and N are general
$\sim_{bis}$	Σ_2^p		
$\sim_{rsim}$, $\sim_{rt}$, $\sim_r$, $\sim_{fat}$, $\sim_f$	Σ_2^p		undecidable
$\sim_{sim}$	Σ_2^p	undecidable	undecidable
$\sim_{tr}$, $\sim_{ftr}$	Σ_2^p	undecidable	undecidable

TABLE II

Complexity of process equivalences for finite processes.

Finite Processes M and N	M and N are determinate	One of M and N is determinate	M and N are general
$\sim_{bis}$	**NL**-complete	**NL**-complete	**P**-complete
$\sim_{rsim}$	**NL**-complete	**NL**-complete	**P**
$\sim_{sim}$	**NL**-complete	**P**	**P**
$\sim_{rt}$, $\sim_r$, $\sim_{fat}$, $\sim_f$	**NL**-complete	**NL**-complete	**PSPACE**-complete
$\sim_{tr}$, $\sim_{ftr}$	**NL**-complete	**PSPACE**-complete	**PSPACE**-complete

We assume familiarity with basic notions in formal language theory and complexity theory. The reader is referred to [16], [24] and [11] for further details. This paper is a continuation of the study in [18], [19] and [21]. It is organized as follows. In Section 2, we define formally the notion of a process graph and various process equivalences, and for some interesting classes of process graphs we show the relationship between those equivalences. In Section 3, we show that simulation and ready-simulation is in **P** for finite process graphs. In Section 4, we show that, for finite process graphs, the determinacy problem with respect to any process equivalence between bisimulation and failure equivalences is **NL**-complete. For two finite process graphs, if one them is determinate, then any process equivalence between bisimulation and failure equivalences is **NL**-complete. For determinate finite process graphs, all process equivalences (between bisimulation and trace equivalences) are **NL**-complete. We also show that for (locally unary) finite process graphs, ready-trace and failure trace equivalences are **PSPACE**-complete. In Section 5, we show that, for normed context-free process graphs, the determinacy problem with respect to any process equivalence between bisimulation and failure equivalences is in Σ_2^p, while it is undecidable with respect to simulation and trace equivalences. For two normed context-free process graphs, if one of them is determinate, then any process between bisimulation and failure equivalences is in Σ_2^p. In Section 6, we make some concluding remarks.

2. BASIC DEFINITIONS AND FACTS

In this section, we introduce some basic definitions and prove some technical results that will be used in subsequent sections. The notion of a process graph will be formally defined and various classes of process graphs will be introduced. A process graph is essentially a nondeterministic (possibly infinite) automaton.

DEFINITION 2.1: A *process graph* M is a 4-tuple $M=(P, Act, \delta, p_0)$ where

1. P is a set of *processes*.
2. Act is a finite set of (*observable*) *actions*.
3. $\delta \subseteq P \times Act \times P$ is the transition relation.
4. $p_0 \in P$ is the *initial process*.

We say that there is a transition labeded by action a from process p to processes q, denoted by $p \rightarrow^a q$, if $(p, a, q) \in \delta$. We write $p \rightarrow^a$, (resp. $p \rightarrow q$) if $p \rightarrow^a q$ for some process q (resp. action a). If

$$q_0 \rightarrow^{a_1} q_1 \rightarrow^{a_2} q_2 \ldots \rightarrow^{a_n} q_n,$$

then we call this a (*transition*) *path* labeled by action string $a_1 a_2 \ldots a_n$ from process q_0 to process q_n. *We write* $q_0 \Rightarrow^{a_1 \cdots a_n} q_n$ if such a path exists. We write $p \Rightarrow^x$ (resp. $p \Rightarrow q$) if $p \Rightarrow^x q$ for some process q (resp. action string x).

DEFINITION 2.2: Let $M=(P, Act, \delta, p_0)$ be a process graph and p be a process in P.

The *initial set* of p is defined by $init(p)=\{a \in Act \mid p \rightarrow^a\}$.

The *next set* of p is defined by $next(p)=\{q \in P \mid p \rightarrow q\}$.

p is said to be *terminal process*, denoted by $p\downarrow$, if $init(p)=\emptyset$.

The *norm* of p is defined by $norm(p)=min\{length(x) \mid \exists q \in P: p \Rightarrow^x q\downarrow\}$.

We now introduce various classes of process graphs.

DEFINITION 2.3: Let $M=(P, Act, \delta, p_0)$ be a process graph. M is said to be *finitely branching* if for each $p \in P$, $init(p)$ and $next(p)$ are both finite, *deterministic* if δ is a (partial) function from $P \times Act$ into P, *normed* if for each $p \in P$, $norm(p) < +\infty$,

finite if P and Act are both finite,

locally unary if for each $p \in P$, $init(p) \leq 1$, *i.e.*, there is at most one action $a \in Act$ such that $p \to^a$,

unary if Act contains exactly one action.

We say that processes in P are *finitely branching* (*deterministic*, *normed*, *finite*, *locally unary*, *unary*) if M is finitely branching (deterministic, normed, finite, locally unary, unary).

Note that the notion of a normed process was introduced in [2], whereas locally unary processes were considered in [18]. In the following, we reproduce the definitions of various process equivalences, which can be found in the literature (*cf.* e. g. [12]). The notion of bisimulation equivalence was introduced in [31] and used in the *Calculus of Communicating Systems* (*CCS*) [28]. The notion of failure equivalence was introduced and studied in [4], which coincides essentially with the notion of testing equivalence defined in [29]. The notion of readiness equivalence was introduced in [30] and further investigated in [6]. Furthermore, one finds other process equivalences such as simulation equivalence [31], ready simulation (2/3-bisimulation) equivalence [5, 26], ready trace equivalence (*cf.* [3]), failure trace equivalence (refusal testing) [32] and so on. Now, let $M = (P, Act, \delta, p_0)$ and $N = (Q, Act, \theta, q_0)$ be process graphs.

DEFINITION 2.4: Let R be a relation over $(P \cup Q) \times (P \cup Q)$. R is said to be

a *simulation* if, for each $(p, q) \in R$, $p \to^a p'$ implies that there is q' such that $q \to^a q'$ and $(p', q') \in R$,

a *bisimulation* if R and R^{-1} are both simulations,

a *ready-simulation* if R is a simulation and for each $(p, q) \in R$, $init(p) = init(q)$.

DEFINITION 2.5: Let p and q be processes in $(P \cup Q)$.

p and q are said to be *bisimulation equivalent*, denoted by $p \sim_{bis} q$, iff there is a bisimulation R with $(p, q) \in R$.

p is said to be *simulated by* q, denoted by $p \preccurlyeq_{sim} q$, iff there is a simulation R with $(p, q) \in R$. p and q are said to be *simulation equivalent*, denoted by $p \sim_{sim} q$, iff they are simulated by each other, *i.e.* $p \preccurlyeq_{sim} q$ and $q \preccurlyeq_{sim} p$.

p is said to be *ready-simulated by* q, denoted by $p \preccurlyeq_{rsim} q$, iff there is a ready-simulation R with $(p, q) \in R$. p and q are said to be *ready-simulation* (2/3-*bisimulation*) *equivalent*, denoted by $p \sim_{rsim} q$, iff they are ready-simulated by each other, *i.e.* $p \preccurlyeq_{rsim} q$ and $q \preccurlyeq_{rsim} p$.

DEFINITION 2.6: Let p be a process in P.

The *trace set* of p is defined by $traces(p) = \{x \in Act^* \mid p \Rightarrow^x\}$.

The *finite trace set* of p is defined by $ftr(p) = \{x \in Act^* \mid \exists q \in P : p \Rightarrow^x q \downarrow\}$.

The *ready set* of p is defined by

$$readies(p) = \{(x, Z) \in Act^* \times 2^{Act} \mid \exists q \in P : p \Rightarrow^x q \text{ and } init(q) = Z\}.$$

The *ready-trace set* of p is defined by

$$ready\text{-}traces(p) = \{A_0 a_0 A_1 \ldots a_{m-1} A_m \mid m \geqq 0, \exists p_1, p_2, \ldots, p_m \in P : p = p_0 \rightarrow^{a_0} p_1 \rightarrow^{a_1} \ldots p_{m-1} \rightarrow^{a_{m-1}} p_m \text{ and } init(p_i) = A_i, 0 \leqq i \leqq m\}.$$

The *failure set* of p is defined by

$$failures(M) = \{(x, Y) \in Act^* \times 2^{Act} \mid \exists q \in P : p \Rightarrow^x q \text{ and } init(q) \cap Y = \emptyset\}.$$

The *failure-trace set* of p is defined by

$$failure\text{-}traces = \{x_0 A_1 x_1 \ldots A_m x_m \mid m \geqq 0, \exists p_1 . p_2, \ldots, p_m \in P : p \Rightarrow^{x_0} p_1 \Rightarrow_{x_1} \ldots p_m \Rightarrow^{x_m} \text{ and } init(q_i) \cap A_i = \emptyset, 1 \leqq i \leqq m\}.$$

DEFINITION 2.7: Let $p \in P$ and $q \in Q$ be processes. p and q are said to be

trace equivalent, denoted by $p \sim_{tr} q$, iff $traces(p) = traces(q)$,

finite trace equivalent, denoted by $p \sim_{ftr} q$, iff $ftr(p) = ftr(q)$,

readiness equivalent, denoted by $p \sim_r q$, iff $readies(p) = readies(q)$,

ready-trace equivalent, denoted by $p \sim_{rt} q$, iff $ready\text{-}traces(p) = ready\text{-}traces(q)$,

failure equivalent, denoted by $p \sim_f q$, iff $failures(p) = failures(q)$,

failure-trace equivalent, denoted by $p \sim_{fat} q$, iff $failure\text{-}traces(p) = failure\text{-}traces(q)$.

For an equivalence $\sim$, we say that $M \sim N$ iff $p_0 \sim q_0$.

The general inclusion relationship between equivalences for process graphs is shown in Figure 1 (*see* also [12]).

Clearly, for normed processes, $\sim_{ftr}$ is finer than $\sim_{tr}$.

PROPOSITION 2.8: *For normed processes, it holds that* $\sim_{ftr} \subset \sim_{tr}$.

We now consider the notion of determinate process graphs which was introduced in [28].

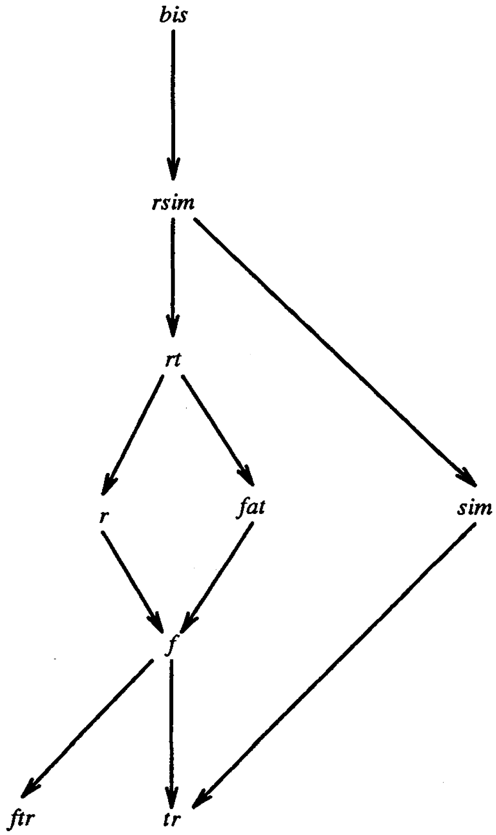

Figure 1.

DEFINITION 2.9: Let $M=(P, Act, \delta, p_0)$ be a process graph. M is said to be *determinate*, if for all $p \in P$ and $a \in Act$, whenever $p \to^a p_1$ and $p \to^a p_2$ then $p_1 \sim_{bis} p_2$.

Obviously, every deterministic process graph is determinate, but not vice versa. A characterization of determinate process graphs is contained in the following proposition whose proof is straightforward.

PROPOSITION 2.10: *Let $M=(P, Act, \delta, p_0)$ be a process graph. Then the following are equivalent:*

1. *M is a determinate process graph.*
2. *For all $p \in P$ and $x \in Act^*$, whenever $p \Rightarrow^x p_1$ and $p \Rightarrow^x p_2$ then $p_1 \sim_{bis} p_2$.*

3. *$M \sim_{bis} N$ for some deterministic process graph N.*

The following proposition was shown in [9] and [28].

PROPOSITION 2.11: *Let M and N be determinate process graphs. Then $M \sim_{bis} N$ iff $M \sim_{tr} N$.*

Proposition 2.11 implies that all equivalences between bisimulation and trace equivalences coincide for determinate process graphs. Moreover, from Proposition 2.8, for normed determinate process graphs, finite trace equivalence coincides with all other equivalences.

Engelfriet showed in [9] that determinacy is preserved under failure equivalence as stated in the following

LEMMA 2.12: *Let M be a (general) process graph and N be a determinate process graph with p_0 and q_0 as the initial processes, respectively. Then failure $(p_0) \subseteq failure(q_0)$ implies that $M \sim_{bis} N$.*

From Lemma 2.12, we have immediately as a corollary

PROPOSITION 2.13: *Let M be a (general) process graph and N a determinate process graph. Then*

$$M \sim_{bis} N \text{ iff } M \sim_{rsim} N \text{ iff } M \sim_{rt} N \text{ iff } M \sim_{r} N \text{ iff } M \sim_{fat} \text{ iff } M \sim_{f} N.$$

Proposition 2.13 says that for two process graphs, if any of them is determinate, then all equivalences between bisimulation equivalence and failure equivalence coincide. From Propositions 2.10 and 2.13 we have

COROLLARY 2.14: *Let M be a process graph. Then $M \sim_{bis} N$ for some deterministic process graph N iff $M \sim_{r} (\sim_{rsim}, \sim_{rt}, \sim_{r}, \sim_{fat}, \sim_{f}) N$ for some deterministic process graph N.*

For normed and locally unary processes, Huynh and Tian first showed in [18] that readiness, failure and finite trace equivalences coincide. Subsequently, Groote and Hüttel showed in [14] that they coincide with ready-trace and failure-trace equivalences.

PROPOSITION 2.15: *For normed and locally unary processes, it holds that*

$$\sim_{rt} = \sim_{r} = \sim_{fat} = \sim_{f} = \sim_{ftr}.$$

Proof: Let p and q be normed and locally unary processes. It suffices to show that $ftr(p) = ftr(q)$ implies that $ready\text{-}trace(p) = ready\text{-}trace(q)$. To this end, let $x = A_0 a_0 A_1 \ldots a_{m-1} A_m \in ready\text{-}traces(p)$. Since processes are locally unary, we have that $A_i = \{a_i\}$ for $0 \leq i < m$. If $A_m = \emptyset$, $a_0 a_1 \ldots a_{m-1} \in ftr(p)$. Since $ftr(p) = ftr(q)$, it follows that $x \in ready\text{-}traces(q)$. In the case that

$A_m \neq \emptyset$, $A_m = \{a\}$ for some $a \in Act$ since processes are locally unary. From the fact that processes are normed, there exists $y \in Act^*$ such that $a_0 a_1 \ldots a_{m-1} ay \in ftr(p)$. Since $ftr(p) = ftr(q)$, it follows that $a_0 a_1 \ldots a_{m-1} ay \in ftr(q)$ Thus, we have that $x \in$ *ready-traces* (q). By symmetry, we also have that $x \in$ *ready-traces* (q) implies $x \in$ *ready-traces* (p). Hence, *ready-traces* (p) = *ready-traces* (q). This completes the proof of Proposition 2.15. □

As unary processes are locally unary, we obtain immediately

COROLLARY 2.16: *For normed unary processes, it holds that*

$$\sim_{rt} = \sim_r = \sim_{fat} = \sim_f = \sim_{ftr}.$$

LEMMA 2.17: *Let* $p_1, p_2, \ldots, p_m$ *be unary processes. Then for some* $1 \leqq i \leqq m$, $traces(p_i) = \bigcup_{j=1}^{m} traces(p_j)$.

Proof: By induction on m. □

PROPOSITION 2.18: *For unary processes, it holds that* $\sim_{sim} = \sim_{tr}$.

Proof: Let p and q be unary processes. We show that $p \preccurlyeq_{sim} q$ iff $traces(p) \subseteqq traces(q)$. The "only if" part is obvious. For the "if" part, let $M = (P, \{a\}, \delta, p_0)$ and $N = (Q, \{a\}, \theta, q_0)$. Consider relation $R = \{(p, q) \in P \times Q \mid traces(p) \subseteqq traces(q)\}$. We show that R is simulation.

To this end, let $(p, q) \in R$ and $p \to^a p'$. Then $\{a\}\, traces(p') \subseteqq traces(p)$. Since $traces(p) \subseteqq traces(q)$, $\{a\}\, traces(p') \subseteqq traces(q)$. Note that $traces(q) = \{\varepsilon\} \cup \{a\} (\bigcup_{q \to^a q'} traces(q'))$. From Lemma 2.17, there is $q \to^a q'$ such that $traces(q) = \{\varepsilon\} \cup \{a\}\, traces(q')$. Thus, we have that $\{a\}\, traces(p') \subseteqq \{a\}\, traces(q')$, *i.e.* $traces(p') \subseteqq traces(q')$, which implies that $(p', q') \in R$. This completes the proof of Proposition 2.18. □

3. COMPLEXITY OF SIMULATION EQUIVALENCES FOR FINITE PROCESSES

Bisimulation equivalence for finite processes was shown to be decidable in polynomial time in [27] and an efficient algorithm for deciding bisimulation equivalence for finite processes can be in [33]. It was shown to be **P**-complete in [1]. In this section, we show that deciding simulation and ready-simulation equivalences for finite processes are decidable in polynomial time. Let $M = (P, Act, \delta, p_0)$ and $N = (Q, Act, \theta, q_0)$ be process graphs and U be a transitive subset of $(P \cup Q) \times (P \cup Q)$.

DEFINITION 3.1: A simulation R is said to be a *U-simulation* if $R \subseteq U$. Process p is said to be *U-simulated by* process q, denoted by $p \preccurlyeq_{sim}^{U} q$, iff there is a U-simulation R with $(p, q) \in R$. p and q are said to be *U-simulation equivalent*, denoted by $p \sim_{sim}^{U} q$, iff they are U-simulated by each other, *i.e.* $p \preccurlyeq_{sim}^{U} q$ and $q \preccurlyeq_{sim}^{U} p$.

Let $I = \{(p, q) \in (P \cup Q) \times (P \cup Q) \mid init(p) = init(q)\}$. By definition, simulation equivalence is just the $(P \cup Q)$-simulation equivalence and ready-simulation equivalence is just the I-simulation equivalence. Moreover, if U and U' are transitive subsets of $(P \cup Q) \times (P \cup Q)$ and $U \subseteq U'$, then $p \sim_{sim}^{U} q$ implies $p \sim_{sim}^{U'} q$. To show that simulation and ready-simulation equivalences are in **P**, it suffices to show that U-simulation equivalence is decidable in polynomial time for any transitive subset $U \subseteq (P \cup Q) \times (P \cup Q)$. Indeed, for finite processes, U-simulation may be regarded as the intersection of a sequence of polynomially many approximate simulation refinements.

DEFINITION 3.2: 1. $p \leqq_{0}^{U} q$ for all $(p, q) \in U$.

2. $p \leqq_{i+1}^{U} q$ iff for all $(p, q) \in U$, $p \to^{a} p'$ implies that there is $q \to^{a} q'$ and $p' \leqq_{i}^{U} q'$.

PROPOSITION 3.3: *For finitely branching processes, it holds that* $\preccurlyeq_{sim}^{U} = \bigcap_{i=0}^{+\infty} \leqq_{i}^{U}$.

Proof: We need to show that (1) $\preccurlyeq_{sim}^{U} \subseteq \leqq_{i}^{U}$ for all $i \geqq 0$ and that (2) $\bigcap_{i=0}^{+\infty} \leqq_{i}^{U}$ is a U-simulation.

Part (1) can be shown by induction on i. The basis is obvious. For induction let $p \preccurlyeq_{sim}^{U} q$ and $p \to^{a} p'$. Then, there is $q \to^{a} q'$ with $p' \preccurlyeq_{sim}^{U} q'$. By induction hypothesis, $p' \preccurlyeq_{i}^{U} q'$. Thus, $p \leqq_{i+1}^{U} q$.

To show Part (2), let $R = \bigcap_{i=0}^{+\infty} \leqq_{i}^{U}$ and $(p, q) \in R$. Let $next = next(p) \cup next(q)$. From the fact that processes are finitely branching, we have that $next$ is finite. Therefore, there exists $m \geqq 0$ such that, for all $s, t \in next$, $(s, t) \in R$ iff $s \preccurlyeq_{m}^{U} t$.

Now, let m is as stated in the claim. Since $(p, q) \in R$, we have that $p \leqq_{m+1}^{U} q$. Then, $p \to^{a} p'$ implies that there is $q \to^{a} q'$ with $p' \leqq_{m}^{U} q'$. Because $(p', q') \in next$, $(p', q') \in R$. Therefore, R is a U-simulation. □

For finite processes, let $n = Card(U)$. As finite processes are also finitely branching, it holds that $\preccurlyeq_{sim}^{U} = \leqq_{n^2}^{U}$.

THEOREM 3.4: *U-simulation equivalence problem for finite processes is decidable in polynomial time.*

Proof: A straightforward polynomial time algorithm can be obtained by verifying, for all $q, p \in (P \cup Q)$, whether $p \leqq_i^U q$ for $i=0, 1, 2, \ldots, n^2$. Thus, $p \sim_{sim}^U q$ if $p \leqq_{n^2}^U q$ and $q \leqq_{n^2}^U$. □

Thus, we obtain the following theorem.

THEOREM 3.5: *Simulation and ready-simulation* (2/3-*bisimultion*) *equivalences for finite processes are decidable in polynomial time.*

Not that our result implies that *i*-nested simulation equivalence [15] for finite processes is decidable in polynomial time.

In [1] it was shown that bisimulation equivalence problem for (unary) finite processes is **P**-complete. In contrast to this, we show the **NL**-completeness of simulation equivalence for unary finite processes.

THEOREM 3.6: *Simulation equivalence for unary finite processes is* **NL**-*complete.*

Proof: In [19] Huynh and Tian showed that trace equivalence for unary finite processes is **NL**-complete. Thus, the theorem follows from Proposition 2.18. □

Note that, if unary finite process graph is tree-like, simulation equivalence becomes **L**-complete [19].

4. COMPLEXITY OF TRACE EQUIVALENCES FOR FINITE PROCESSES

Trace, finite trace, readiness and failure equivalences for finite processes were shown to be **PSPACE**-complete in [27], [18] and [20]. In this section we show that all equivalence problems for deterministic finite processes and for determinate finite processes are **NL**-complete and that ready-trace and failure-trace are **PSPACE**-complete. Furthermore, we show that, given a (general) finite process graph and a deterministic finite process graph, deciding bisimulation, ready-simulation, ready-trace, readiness, failure-trace and failure equivalences are all **NL**-complete.

It is well known that language equivalence for deterministic finite automata is **NL**-complete. However, this does not imply the **NL**-hardness of trace equivalence problem for deterministic finite processes because transition functions of processes are partial functions and trace sets of processes have the prefix property, *i. e.*, for any process p, if $x \in traces(p)$, then $y \in traces(p)$ for each prefix y of x. We give our own proof here.

THEOREM 4.1: 1. *All equivalences (defined in this paper) for (normed) deterministic (or determinate) finite processes are* **NL**-*complete.*

2. *Bisimulation, ready-simulation, ready-trace, readiness, failure-trace and failure equivalences for a (general) finite process graph and a determinate finite process graph are* **NL**-*complete.*

Proof: To show the **NL**-hardness, we need only to show that trace equivalence for deterministic finite processes is **NL**-hard. Consider the graph accessibility problem (GAP), a problem well known to be **NL**-complete. It is the problem of deciding for a directed graph G and two vertices s and g whether there is a directed path from s to g. When the graphs in the input are acyclic with vertices having outdegree $\leqq 2$, then we have a special case of GAP denoted by 2-AGAP, which is still **NL**-complete. We reduce 2-AGAP to the trace equivalence problem for normed finite processes. To this end, let $G=(V, E)$ be an acyclic directed graph whose vertices have outdegree at most 2, and s and g be two vertices in G. Without loss of generality we assume that $outdegree(g)=0$. We construct a normed finite process graph $M=(V \cup \{q\}, \{0, 1\}, \delta, s)$ as follows.

The transition function δ is defined by, for each $v \in V$,

1. *If* $outdegree(v)=2$, then $\delta(v, 0):=v_0$ and $\delta(v, 1):=v_1$ where v_0 and v_1 are two different vertices adjacent to v.

2. If $outdegree(v)=1$, then $\delta(v, 0):=v'$ where v' is the vertex adjacent to v.

3. $\delta(g, 1):=q$.

4. There are no other transitions in δ.

A finite process graph N can be obtained from M by replacing $\delta(g, 1):=q$ by $\delta(g, 0):=q$. From the constructions of M and N, we have that there is a directed path from s to g in G iff $traces(M) \neq traces(N)$.

To show the **NL** upper bound, let $M=(P, Act, \delta, p_0)$ be a finite process graph and $N=(Q, Act, \theta, q_0)$ be a determinate finite process graph. From Propositions 2.11 and 2.13, it suffices to show that deciding $M \sim_r N$ is in **NL**. In fact, from the initial processes of M and N, we can simultaneously guess paths labeled by same string in Act^* such that M and N reach different initial sets. Clearly, this can be done in **NL**. As **NL** is closed under complement [25, 37], we obtain the **NL** upper bound. This completes the proof of Theorem 4.1. □

We note that for a (general) finite process graph and a deterministic (or determinate) finite process graph, trace and finite trace equivalences are **PSPACE**-complete. This follows from the fact that the problem of deciding whether a nondeterministic finite automaton with input alphabet Σ accepts language $\Sigma^*\#$ where $\#$ is a terminal symbol not in Σ. We now consider the determinacy problem for finite process graphs. The determinacy problem with respect to a process equivalence $\sim$ is to determine for a given process graph M whether $M \sim N$ for some deterministic process graph N.

THEOREM 4.2: *The determinacy problem with respect to* $\sim_{bis}$ ($\sim_{rsim}$, $\sim_{rt}$, $\sim_r$, $\sim_{fat}$, $\sim_f$, $\sim_{ftr}$) *for finite process graphs is* **NL**-*complete.*

Proof: To show the **NL**-hardness, let $M=(P, Act, \delta, p_0)$ and $N=(Q, Act, \theta, q_0)$ be deterministic finite process graphs as constructed from the graph G in Theorem 4.1 with $P \cap Q = \emptyset$. We construct a finite process graph M' by identifying p_0 as a single initial process. Obviously, we have that $M \sim_{bis} N$ iff $M \sim_{ftr} N$ iff $M' \sim_{bis} N'$ for some deterministic finite process graph N' iff $M' \sim_{sim} N'$ for some deterministic finite process graph N'.

We next show the **NL** upper bound for finite trace equivalence. Let $M=(P, Act, \delta, p_0)$ be a finite process graph. Note that there is no deterministic finite process graph N such that $M \sim_{ftr} N$ iff there are $x, y \in ftr(M)$ such that x is a proper prefix of y. Clearly, this condition can be checked in **NL**. From the fact that **NL** is closed under complement, it follows that deciding whether $M \sim_{ftr} N$ for some deterministic finite process graph N is in **NL**.

To prove the **NL** upper bound for the other equivalences, from Corollary 2.14, it suffices to prove that determinacy problem with respect to $\sim_r$ for finite process graphs is in **NL**. To this end, let M be a finite process graph. From the initial process of M, we simultaneously guess two different paths labeled by the same string in Act^* such that M reaches two different initial sets. Again, from the fact that **NL** is closed complement, it follows that deciding whether $M \sim_r N$ for some deterministic finite process graph N is in **NL**.

Finally, for the correctness of the algorithm, one simply proves that, if for all $x \in Act^*$ there exists at most one $Z \subseteq Act$ such that $(x, Z) \in readies(M)$, then

$$R = \{ (p, q) \in P \times P \mid \exists x \in Act^* : p_0 \Rightarrow^x p \text{ and } p_0 \Rightarrow^x q \}$$

is a bisimulation. We leave this to the reader. Thus, Theorem 4.2 follows. □

From Proposition 2.10, we obtain as a corollary

THEOREM 4.3: *The problem of determining whether a finite process graph is determinate is* **NL**-*complete.*

The proof of Theorem 4.2 implies that, for finite process graphs, the determinacy problem with respect to any process equivalence between bisimulation and failure equivalences is **NL**-complete. However, for any finite process graph, there exists always an equivalent deterministic finite process graph in the trace semantics. Instead of trace and finite trace equivalences, maximal trace equivalence is more used in concurrency theories. Two processes are said to be *maximal trace equivalent* iff they are trace equivalent and finite trace equivalent. One can easily show that all **NL**-completeness results remain true for maximal trace equivalence for finite processes (finite process graphs).

Now we turn our attention to (general) finite processes. Huynh and Tian showed in [18] that finite trace equivalence problem is **PSPACE**-complete for normed locally unary finite processes. As for normed locally unary processes ready-trace and failure-trace equivalences coincide with finite trace equivalence (Proposition 2.15), we have immediately.

THEOREM 4.4: *Ready-trace and failure-trace equivalences for normed and locally unary finite processes are* **PSPACE**-*complete.*

Now we show that ready-trace and failure-trace equivalence problems for (general) finite processes are **PSPACE**-complete.

THEOREM 4.5: *Ready-trace equivalence for finite processes is* **PSPACE**-*complete.*

Proof: We show the **PSPACE** upper bound only. We do so by reducing ready-trace equivalence problem for finite process to trace equivalence for finite processes. To this end, let $M=(P, Act, \delta, p_0)$ be a finite process graph. From M we construct a finite process graph $M'=(P', Act', \delta', p_0)$ as follows.

$P' := P \cup \{[p] \mid p \in P\}$.

$Act' := Act \cup \{init(p) \mid p \in P\}$.

The transition function δ' is defined by, for all $p \in P$ and $a \in Act$,

(1) $\delta'(p, init(p) := \{[q]\}$.

(2) $\delta'([q], a) := \delta(q, a)$.

Similarly, one can construct from finite process graph N a finite process graph N'. It is easy easy to see that *ready-traces*(M) = *ready-traces*(N) iff *traces*(M') = *traces*(N'). This completes the proof of Theorem 4.5. □

THEOREM 4.6: *Failure-trace equivalence for finite processes is* **PSPACE**-*complete.*

Proof: We show the **PSPACE** upper bound only. We do so by reducing failure-trace equivalence for finite processes to trace equivalence for finite processes. Now let $M=(P, Act, \delta, p_0)$ be a finite process graph. From M we construct a finite process graph $M'=(P, Act', \delta, p_0)$ as follows.

$Act' := Act \cup \{[a] \mid a \in Act\}$.

The transition function δ' is obtained by adding to δ the following transitions: for all $p \in P$ and $a \in Act$, $\delta'(p, [a]) := p$ if $a \notin init(p)$.

Similarly, one can construct from a finite process graph N a finite process graph N'. It is easy to see that *failure-traces* (M)=*failures-traces* (N) iff *traces* (M') = *traces* (N'). This completes the proof of Theorem 4.6. □

Note that readiness and failure equivalences for (normed and locally unary) finite processes were shown to be **PSPACE**-complete in [18]. In [18] it was also shown that readiness (failure) equivalence for unary finite processes is co-**NP**-complete. Thus, from Proposition 2.16, we obtain as a corollary

THEOREM 4.7: *Ready-trace and failure-trace equivalences for unary finite processes are co-***NP***-complete.*

5. COMPLEXITY OF NORMED CONTEXT-FREE PROCESSES

In this section, we show that, for a normed context-free process and a determinate normed context-free process, all equivalences between bisimulation and failure equivalences are in Σ_2^p. We also show that for normed context-free processes, the determinacy problem with respect to any equivalence between bisimulation and failure equivalences in Σ_2^p, while these problems with respect to simulation and trace equivalences are undecidable.

DEFINITION 5.1: Let $G=(Var, Act, \Pi, S)$ be a context-free grammar (CFG), where Var is the set of nonterminals, Act is the set of terminals, Π is the set of productions and S is the start symbol in Var.

1. G is said to be in *Greibach normal form* (*GNF*) if each production in Π is of the fomr $A \to a\alpha$, where a is a terminal in Act and α is a nonterminal string in Var^*.

2. A GNF CFG G is said to be *normed* if each nonterminal $A \in Var$ can generate a terminal string.

3. A GNF CFG G is said to be *simple* if for each $A \in Var$ and each $a \in Act$ there is at most one production of the form $A \to a\alpha$ in Π.

4. We use $L(G)$ to denote the context-free language (CFL) generated by G and *prefix* $(L(G))$ to denote the set $\{x \mid \exists y : xy \in L(G)\}$.

5. A CFL is said to be *deterministic* if it can be accepted by a deterministic push-down automaton [23, 24].

The following lemma is well-known in the theory of formal languages [23, 24].

LEMMA 5.2: 1. *Each CFL L without ε (the empty string) is generated by a normed GNF CFG which can be effectively constructed from any CFG generating L.*

2. *If R is a regular set, then $R\#$ is a simple CFL.*

3. *Any simple CFL is a deterministic CFL.*

4. *If L is a deterministic CFL and R is a regular set, then L/R is a deterministic CFL.*

5. *If L is a deterministic CFL, then so is $\bar{L}$.*

Every GNF CFG may be regarded as a guarded system of recursion equation over the basic process algebra (BPA). Therefore, every GNF CFG can be associated with a process graph as follows.

DEFINITION 5.3: Let $G=(Var, Act, \Pi, S)$ be a GNF CFG. The *(context-free) process graph* M_G specified by G is $M_G=(Var^*, Act, \delta, S)$ where δ is defined as follows: For each *(context-free) process* in Var^* of the form $A\beta$, if $A \to a\alpha$ is a production in Π where $a \in Act$, then $A\beta \to^a \alpha\beta$ is a transition in δ.

Note that M_G is a normed process iff for all $A \in Var$, $norm(A) < +\infty$ iff G is normed CFG. In this case, ε (the empty string) is the unique terminal process. Note also that, if G is normed, $ftr(M_G)=L(G)$ and $traces(M_G)=prefix(L(G))$ and that, if G is simple, M_G is a deterministic processes.

THEOREM 5.4: *For normed context-free processes,*

1. *bisimulation equivalence is decidable* [2, 7, 17, 13],
2. *bisimulation equivalence is in* Σ_2^p [21],
3. *readiness and failure equivalences are undecidable* [18],
4. *all other equivalences in* [12] *are undecidable* [14],

We now show the main results of this section.

THEOREM 5.5: *Bisimulation, ready simulation, ready trace, readiness, failure-trace and failure equivalences for a normed context-free process graph and a normed determinate context-free process graph are in* Σ_2^p.

Proof: In [21] it was shown that bisimulation equivalence for normed context-free processes is in Σ_2^p. Thus, the theorem follows from Proposition 2.13. □

The proof of Theorem 5.5 implies that any equivalence between bisimulation and failure equivalences for a normed context-free process graph and a determinate normed context-free process graph is in Σ_2^p. However, we have

THEOREM 5.6: *Simulation and (finite) trace equivalences for a normed context-free process graph and a deterministic normed context-free process graph are undecidable.*

Proof: The undecidability of simulation equivalence was shown in [14]. The undecidability of (finite) trace equivalence follows from the fact the problem of deciding whether a given context-free grammar with terminal alphabet Σ generates language $\Sigma^*\#$ is undecidable where $\#$ is a new terminal symbol not in Σ (see the proof of Theorem 5.7). □

We now consider the determinacy problems with respect to process equivalences for normed context-free process graphs.

THEOREM 5.7: *For normed context-free process graphs, the determinacy problem with respect to (finite) trace equivalence is undecidable.*

Proof: It is well known that the problems of deciding for a CFL L whether L is a regular set and whether $\bar{L}$ is a CFL are undecidable. The proof [23] of the undecidability of these two problems consists of constructing a CFL L from an instance (A, B) of the Post's Correspondence Problem, a well known undecidable problem, such that (A, B) has no solution iff $\bar{L}$ is a CFL iff $L=\Sigma^*$ where Σ is a suitable alphabet. Furthermore, for this CFL L, letting $\#$ be a new terminal symbol not in Σ, we have the following equivalent statements from Lemma 5.2:

1. *prefix* $(L\#)=$ *prefix* $(\Sigma^*\#)$.
2. *prefix* $(L\#)=$ *prefix* $(L(G))$ for some normed simple CFG G.
3. $L\#$ is a simple CFL (*i.e.* $L\#=L(G)$ for some normed simple CFG G).
4. $L\#$ is a deterministic CFL.
5. L is a deterministic CFL.
6. $\bar{L}$ is a deterministic CFL.
7. $\bar{L}$ is a CFL.
8. $L=\Sigma^*$.
9. $L\#=\Sigma^*\#$.

This completes the proof of Theorem 5.7. □

THEOREM 5.8: *For normed context-free process graphs, the determinacy problem with respect to simulation equivalence is undecidable.*

Proof: In the theory of formal languages, it is known that the language inclusion problem for simple CFGs is undecidable [10]. We reduce this problem into the determinacy problem with respect to simulation equivalence for normed context-free process graphs. To this end, let $G=(V, Act, \Pi_G, R)$ and $H=(U, Act, \Pi_H, T)$ be two normed simple CFGs. Let $\#$, a and b be three new distinct terminal symbols not in Act. We construct normed contex-free grammars $G_a=(V_a, Act \cup \{\#, a\}, \Pi_a, R_a)$, $G_b=(V_b, Act \cup \{\#, b\}, \Pi_b, R_b)$, $H_{a+b}=(U_{a+b}, Act \cup \{\#, a, b\}, \Pi_{a+b}, T_{a+b})$ and $F=(Var, Act', \Pi, S)$ as follows.

For G_a, $V_a := V \cup \{R_a, A\}$ where A and R_a are new nonterminals. Π_a contains all productions in Π_G as well as productions $R_a \to \# RA$ and $A \to a$. G_b can be obtained from G_a by replacing every a by b and every A by B. For H_{a+b}, $U_{a+b} := U \cup \{T_{a+b}, C\}$ where C and T_{a+b} are new nonterminals. Π_{a+b} contains all productions in Π_H was well as productions $T_{a+b} \to \# TC$ and $C \to a \mid b$. F is obtained from G_a, G_b and H_{a+b} by identifying R_a, R_b and T_{a+b} as a single initial symbol. Note that G_a, G_b and H_{a+b} are simple CFGs.

It is not hard to verify that $L(G) \subseteq L(H)$ iff $M_F \sim_{sim} N$ for some normed deterministic context-free process graph N. In fact, if $L(G) \subseteq L(H)$, then $M_{G_a} \preccurlyeq_{sim} M_{H_{a+b}}$ and $M_{G_b} \preccurlyeq_{sim} M_{H_{a+b}}$. Hence, $M_F \sim_{sim} M_{H_{a+b}}$. On the other hand, if $M_F \sim_{sim} N$, for some normed deterministic context-free process graph N, then $M_{H_{a+b}} \preccurlyeq_{sim} N$ and $N \preccurlyeq_{sim} M_{H_{a+b}}$. Thus, we have that $M_{G_a} \preccurlyeq_{sim} N \sim_{bis} M_{H_{a+b}}$ and hence $L(G) \subseteq L(H)$. This completes the proof of Theorem 5.8. □

However, with respect to any equivalence between bisimulation and failure equivalences, the determinancy problem for normed context-free process graphs is decidable. As a matter of fact, we have

THEOREM 5.9: 1. *The problem of determining whether a normed context-free process graph is determinate is in* Σ_2^p.

2. *For normed context-free process graphs, the determinacy problems with respect to* $\sim_{bis}$ ($\sim_{rsim}$, $\sim_{rt}$, $\sim_{r}$, $\sim_{fat}$, $\sim_{f}$) *is in* Σ_2^p.

Proof: By Propositions 2.10 and 2.13, it suffices to show Part (1). Now let $G=(Var, Act, \Pi, S)$ be a normed GNF CFG. We have that M_G is determinate iff for all productions $A \to a\alpha \mid a\beta$ in Π, $\alpha \sim_{bis} \beta$.

The proof of Theorem 5.9 implies that, for normed context-free process graph, the determinacy problem with respect to any process equivalence between bisimulation and failure equivalences is in Σ_2^p. To end this section we state some results for normed unary context-free processes.

THEOREM 5.10: *For normed unary context-free processes,*

1. *bisimulation equivalence is in* Σ_2^p [21, 22],
2. *readiness, failure and finite trace equivalences are* Π_2^p*-complete* [18],
3. *ready-trace and failure-trace equivalences are* Π_2^p*-complete,*
4. *trace equivalence is in* **P** [19],
5. *simulation equivalence is in* **P**.

Proof: (3) follows from Proposition 2.16 and (5) follows from Proposition 2.18. □

6. CONCLUDING REMARKS

In this paper, we characterize the complexity of some process equivalences. Specifically, we show that any process equivalence between bisimulation and failure equivalences is **NL**-complete for a (general) finite process graph and a deterministic finite process graph and is in Σ_2^p for a (general) normed context-free process graph and a deterministic normed context-free process graph, and that the determinacy problem with respect to any process equivalence between bisimulation and failure equivalences is **NL**-complete for finite process graphs and is in Σ_2^p for normed context-free process graphs.

However, some problems are still open. Simulation equivalence for finite process lies somewhere between **NL** and **P**. It would be interesting to show that this problem is in **NC** or **P**-complete. For normed context-free process graph, determinacy problem with respect to bisimulation equivalence lies between **NL** and Σ_2^p. It would be interesting to close this gap.

REFERENCES

1. C. ALVAREZ, J. L. BALCÁZAR, J. GABARRÓ and M. SANTHA, Parallel Complexity in the Design and Analysis of Concurrent Systems, *Lecture Notes in Computer Science*, 1991, *505*, pp. 288-303.
2. J. C. M. BAETEN, J. A. BERGSTRA and J. W. KLOP, Decidability of Bisimulation Equivalence for Process Generating Context-Free Languages, *Lecture Notes in Computer Science*, 1987, *259*, pp. 94-113.
3. J. C. M. BAETEN, J. A. BERGSTRA and J. M. KLOP, Ready-Trace Semantics for Concrete Process Algebra with the Priority Operator, *Computer Journal*, 1987, *30*, pp. 498-506.

4. S. D. Brookes, C. A. R. Hoare and A. W. Roscoe, A Theory of Communicating Sequential Processes, *J. Assoc. Comput. Mach.*, 1984, *31*, pp. 560-599.

5. B. Bloom, S. Istrail and A. R. Meyer, Bisimulation Can't Be Traced, *Proc. of the 15th ACM Symp. on Principles of Programming Languages*, 1988, pp. 229-239.

6. J. A. Bergstra, J. W. Klop and E. R. Olderog, Readies and Failures in the Algebra of Communicating Processes, *SIAM J. on Computing*, 1988, *17*, pp. 1134-1177.

7. D. Caucal, Graphes Canoniques de Graphes Algébriques, *Theoretical Informatics and Application*, 1990, *24*, pp. 339-352.

8. S. Cho and D. T. Huynh, The Parallel Complexity of Coarset Set Partition Problems, *Information Processing Letters*, 1992, *42*, pp. 89-94.

9. J. Engelfriet, Determinacy → (Observation Equivalence = Trace Equivalence), *Theoretical Computer Science*, 1985, *36*, pp. 21-25.

10 E. P. Friedman, The Inclusion Problem for Simple Languages, *Theoretical Computer Science*, 1976, *1*, pp. 297-316.

11. M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

12. R. J. Van Glabbeek, The Linear Time – Branching Time Spectrum, *Lecture Notes in Computer Science*, 1990, *458*, pp. 278-297.

13. J. F. Groote, A Short Proof of the Decidability of Bisimulation for Normed BPA-Processes, Technical Report CS-R9151, CWI, 1991, to appear in *Information Processing Letters*.

14. J. F. Groote and H. Hüttel, *Undecidable Equivalences for Basic Process Algebra*, Technical Report CS-R9137, CWI, 1991.

15. J. F. Groote and F. W. Vaandrager, Structured Operational Semantics and Bisimulation as a Congruence, *Lecture Notes in Computer Science*, 1989, *372*, pp. 423-438 (to appear in *Information and Computation*).

16. M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, 1978.

17. H. Hüttel and C. Stirling, Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes, *Proc. 6th Annual Symp. on Logic in Computer Science*, 1991, pp. 376-386.

18. D. T. Huynh and L. Tian, Complexity of Deciding Readiness and Failure Equivalences for Processes, *Proc. of the 3rd IEEE Symp. on Parallel and Distributed Processing*, 1991, pp. 738-745.

19. D. T. Huynh and L. Tian, On Deciding Trace Equivalences for Processes, Technical Report UTDCS-4-91, University of Texas at Dallas, 1991, to appear in *Information Sciences*.

20. D. T. Huynh and L. Tian, On Some Equivalence Relations for Probabilistic Processes, *Fundamenta Informaticae*, 1992, *17*, pp. 211-234.

21. D. T. Huynh and L. Tian, Deciding Bisimilarity of Normed Context-Free Processes Is in Σ_2^p. Technical Report UTDCS-1-92, University of Texas at Dallas, 1992, to appear in *Theoretical Computer Science*.

22. D. T. Huynh and L. Tian, *A Note on Complexity of Deciding Bisimilarity of Normed Unary Processes*, Technical Report UTDCS-2-92, University of Texas at Dallas, 1992.

23. J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley Publishing Co., 1969.

24. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Co., 1979.
25. N. Immerman, Nondeterministic Space is Closed under Complement, *SIAM J. on Computing*, 1988, *17*, pp. 935-938.
26. K. G. Larsen and A. Skou, Bisimulation through Probabilistic Testing, *Information and Computation*, 1991, *94*, pp. 1-28.
27. P. C. Kanellakis and S. A. Smolka, CCS Expressions, Finite State Processes and Three Problems of Equivalence, *Information and Computation*, 1990, *86*, pp. 43-68.
28. R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.
29. R. De Nicola and M. C. B. Hennessy, Testing Equivalences for Processes, *Theoretical Computer Science*, 1984, *34*, pp. 83-133.
30. E. R. Olderog and C. A. R. Hoare, Specification-Oriented Semantics for Communicating Processes, *Acta Informatica*, 1986, *23*, pp. 9-66.
31. D. Park, Concurrency and Automata on Infinite Sequences, *Lecture Notes in Computer Science*, 1981, *104*, pp. 168-183.
32. I. C. C. Phillips, Refusal Testing, *Theoretical Computer Science*, 1987, *50*, pp. 241-284.
33. R. Paige and R. E. Tarjan, Three Partition Refinement Algorithm, *SIAM J. on Computing*, 1987, *16*, pp. 937-989.
34. W. S. Rounds and S. D. Brooks, Possible Futures, Acceptances, Refusals and Communicating Processes, *Proc. 22-nd Annual Symp. on Foundations of Computer science*, 1981, pp. 140-149.
35. L. J. Stockmeyer, The Polynomial Time Hierarchy, 1977, *3*, pp. 1-22.
36. I. H. Sudborough, On Tape-Bounded Complexity Classes and Multi-Head Finite Automata, *Comput. Syst. Sci.*, 1975, *10*, pp. 62-76.
37. R. Szelepcsényi, The Method of Forced Enumeration for Nondeterministic Automata, *Acta Information*, 1988, *29*, pp. 279-284.