Decidability Issues for Processes with Infinitely Many States

Antonín Kučera

Ph.D. Thesis

Faculty of Informatics

Masaryk University

1997

Acknowledgements

First of all I want to thank my supervisor Mojmír Křetínský for continuous support, guidance and encouragement. It is difficult to express how much I owe to him; I am very grateful for his help I could always rely on.

My warm thanks go to Ivana Černá and Petr Jančar. I have learned much from our numerous discussions; it was a great pleasure to work with them.

Thanks are also due to Mogens Nielsen for his kind supervision during my stay at Aarhus University (BRICS). The results presented in Chapter 3 originated in Denmark.

Special thanks go to my mother for constant emotional and practical support during my studies, and to Hana for her company, love and understanding.

Declaration

I declare that this thesis was composed by myself, and all presented results are my own, unless otherwise stated.

Some of the material has been previously published as [Kuč96a], [Kuč96b], [ČKK96] and [Kuč97].

Antonín Kučera

Contents

List of Figures

Chapter 1

Introduction

The problem of program verification is nearly as old as computer science (see e.g., [Flo67]). Various models of computation and its semantics were proposed, emphasizing different aspects of computation. Using this mathematical theory, many interesting questions about programs can be exactly formulated and answered.

1.1 Computation and its Semantics

Denotational semantics, originated by Scott and Strachey in sixties (see e.g., [Sch86]) identifies each program $\mathcal{P}$ with its input-output behaviour. As input parameters of $\mathcal{P}$ are finite strings over a finite (or countably infinite) alphabet, each potential parameter can be uniquely and effectively represented by a natural number. The same applies to output values—the formal meaning of $\mathcal{P}$ can be thus defined by a partial function $f_{\mathcal{P}} : N \to N$. This approach is based on two implicit assumptions:

- There is no interaction with programs except passing input parameters and fetching output values.

- Infinite runs of programs are completely uninteresting (they do no produce any output).

However, reality is considerably different today. Computers are used for wide variety of applications—they control airports, power stations, stoc exchanges and even nuclear weapons. Such systems are usually not de signed to terminate (it would be a disaster in most cases) and their inpu output behaviour is "distributed" over many single acts of communicatio with the surrounding world. To understand and verify properties of thes concurrent systems, three elementary questions must be answered:

- How to model concurrent systems?
- What is their semantics?
- What systems are semantically equivalent?

There are two main approaches to mentioned problems, based on two di ferent classical notions of computability. Ideas around Turing machine and automata lead to the model of Petri net (see [Pet81] or [Rei85] for ger eral introduction). Petri nets can be seen as automata with distribute control units. Semantics is given in terms of partial orders which reflec causal dependencies between actions.

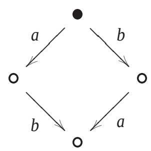
The model of process algebras (such as CCS [Mil89], CSP [Hoa85] o ACP [BW90]) has grown out of concepts in λ-calculus and structured pro gramming. It has so-called interleaving semantics, based on transition sys tems.

The difference between partial order semantics and interleaving se mantics can be well illustrated by the following example. Assume we hav two processes X, Y where

$$X \stackrel{def}{=} a \| b \qquad\qquad Y \stackrel{def}{=} a.b + b.a$$

In other words, the process X can run actions a, b independently in parallel, while the process Y can do either the sequence $a.b$ or the sequence $b.a$ ('$+$' stands for nondeterminism and '.' for sequencing). Here we in fact used the notation of process algebras, but those behaviours could be easily described by labelled Petri nets too.

Interleaving semantics does not distinguish between X and Y; the "real" concurrency of X can be equivalently expressed by sequencing and nondeterminism of Y. Associated transition systems are even isomorphic:



Partial order semantics models concurrency explicitly—the process X is associated with a set of two events (labelled with 'a' and 'b') which are causally independent, hence the ordering is empty. The process Y determines a set of four events. The ordering is indicated by arrows in the picture below. Dotted lines represent the symmetric conflict relation which models the phenomenon of nondeterminism.

$X \stackrel{def}{=} a||b$ $\qquad$ $Y \stackrel{def}{=} a.b + b.a$

Obtained structures are rather different, hence X and Y are not considered as equivalent in the sense of partial order semantics.

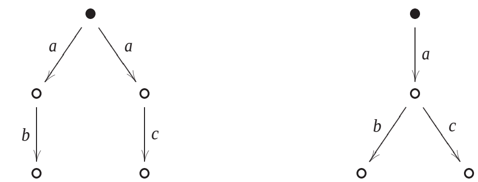
Mentioned approaches to process semantics are naturally independent of a concrete model. They express general ideas which can be "mapped" on a concrete syntax.

The interleaving semantics actually describes processes from the point of view of an external observer who cannot detect causality between ac-

tions by means of experimentation (see [Mil89]). This approach is adopte also in this thesis.

1.2 Verification of Concurrent Systems

Process semantics is formally defined by its associated transition systen It remains to clarify what processes should be taken as semantically equiv alent. Consider the following transition systems:



T_1 $\qquad$ T_2

The systems T_1, T_2 have the same sets of completed traces[1], i.e., $\{ab, ac$ T_1, T_2 can thus be taken as language equivalent in the sense of classica automata theory. However, language equivalence is obviously not the de sired notion of "sameness" in this case—the system T_1 can emit an actio 'a' and enter a state where it can do either 'b' or 'c' (i.e., one of these actior is "blocked" and this is clearly observable). On the other hand, the systen T_2 can choose between 'b' and 'c' after emitting 'a', hence its behaviour *different* from the behaviour of T_1.

This example indicates that branching structure of transition system must be taken into account. In Section 2.2 we present van Glabbeek's hie archy of behavioural equivalences which gives a nice survey and compa ison of existing approaches.

[1]A completed trace is a sequence of labels associated with a path from the root to leaf.

Behavioural equivalences can be used for verification of concurrent systems. For example, correctness a network transport protocol can be proved as follows:

1. *Describe the specification (the intended behaviour).* This is rather trivial in this case, because a reliable transport protocol behaves like a queue—it delivers everything what it receives, preserving original order.
2. *Describe the implementation.* The protocol is essentially performed by three individual cooperating components—Sender sends messages to Medium which passes them to Receiver. Naturally, a suitable level of abstraction must be chosen before a detailed analysis is carried out.
3. *Prove that specification and implementation are equivalent.* Different behavioural equivalences preserve different features (e.g., deadlock or liveness properties). The choice should be based on a careful consideration.

Naturally, computers can assist at this work—and especially at the last task. Although all reasonable behavioural equivalences are generally undecidable, there are interesting classes of transition systems where some of them become decidable. For example, if we restrict our attention to *finite-state* transition systems, each behavioural equivalence is decidable. In fact, the theory of finite-state systems and their equivalences can be said to be well-established today.

Some of those positive results can be even extended to certain classes of infinite-state transition systems. For example, Baeten, Bergstra and Klop showed in [BBK87, BBK93] that bisimilarity (see Definition 2.2) is decidable for processes generated by reduced context-free grammars in Greibach normal form.[2] It was the first result indicating that decidability

[2]Those processes are also known under the name "normed BPA"—see Section 2.3.1.

properties of behavioural equivalences can differ from decidability prop
erties of language equivalence.

Another important approach to verification of concurrent systems ut
lizes various program logics. Intended properties of a process can be ofte
expressed as formulae of certain modal or temporal logic. This leads t
the problem of *model checking*—given a formula F and a state s of a tran
sition system T, does s satisfy F? There are many positive answers fo
certain classes of formulae and transition systems; for example, Stirlin
and Walker gave in [SW89] a model checker for modal μ-calculus an
finite-state transition systems. Similar results exist also for some classe
of infinite-state transition systems. The problem of model checking is no
considered in this thesis, hence we refer to [Sti92] for further informatio
and references.

1.3 Layout of the Thesis

Each chapter (and most sections) begins with a discussion which aims t
give a reasonable motivation to the considered problem. Notes on relate
results and current state of knowledge are included either at the beginnin
or at the end of each section.

Chapter 2 contains definitions of basic notions which are used throug
the thesis. We formally introduce transition systems and various be
havioural equivalences over the class of transition systems. Then w
present several process classes such as BPA, BPP, BPA$_\tau$, BPP$_\tau$, P
and we also define normal forms for these processes. Finally, we in
troduce the condition of normedness which specifies important sub
classes of mentioned algebras and we explain what is meant by th
notion of regularity.

Chapter 3 is devoted to the regularity problem. We prove that regularity of normed PA processes is decidable in polynomial time. Moreover, a bisimilar finite-state process in normal form can be effectively constructed. This implies decidability of bisimilarity for any pair of processes such that one process of this pair is a normed PA process and the other has finitely many states. Obtained results also apply to normed subclasses of BPA, BPP, BPA_τ and BPP_τ and this fact simplifies many considerations in next chapters.

In the next section we examine regularity w.r.t. other equivalences from van Glabbeek's hierarchy. We suggest new notions of finite characterization and strong regularity and we explain their advantages. Then we study the relationship between regularity and strong regularity. We show that the two conditions may coincide w.r.t. certain equivalences, but in case of all equivalences from van Glabbeek's hierarchy except bisimilarity they express different features.

Finally, we demonstrate that regularity and strong regularity w.r.t. any equivalence from van Glabbeek's hierarchy are undecidable for PAPDA processes (this class of processes is obtained from PA by adding a finite-state control unit). This is essentially caused by the fact that PAPDA processes can correctly simulate an arbitrary Minsky machine; in other words, PAPDA is a calculus with full Turing power.

Chapter 4 gives a complete characterization of all processes which can be equivalently defined by the syntax of normed BPA_τ and normed BPP_τ processes. BPA_τ processes are in fact primitive sequential programs, while BPP_τ can be seen as simple parallel programs. Hence we actually characterize all normed behaviours which can be considered as purely sequential as well as purely parallel. This characterization is formulated in terms of special normal forms for BPA_τ and

BPP_τ processes, denoted $\mathrm{INF}_{\mathrm{BPA}}$ and $\mathrm{INF}_{\mathrm{BPP}}$, respectively.

Next we show that any normed BPA_τ or BPP_τ process which belong to the "semantical intersection" of BPA_τ and BPP_τ can be effectivel transformed to $\mathrm{INF}_{\mathrm{BPA}}$ or $\mathrm{INF}_{\mathrm{BPP}}$, respectively. As a consequence w obtain decidability of bisimilarity in the union of normed BPA_τ an normed BPP_τ processes.

We also show that mentioned results can be simplified in case o normed BPA and BPP processes.

Chapter 5 contains results on effective parallelization of normed BPA pro cesses. A normed BPA process is said to be prime if it cannot be de composed into a parallel product of two nontrivial processes. W characterize all normed BPA processes which are not prime togethe with their decompositions in terms of normal forms.

Moreover, we prove that normed BPA processes can be decompose (parallelized) effectively. From this we derive other positive decid ability results—namely decidability of bisimilarity in a natural sub class of normed PA processes, denoted sPA (the sPA class is com posed of all processes of the form $\Delta_1 \| \cdots \| \Delta_n$ where $n \in N$ and Δ_i i a normed BPA or BPP process for each $1 \leq i \leq n$).

Chapter 6 summarizes main results achieved in this thesis and suggest possible directions of future research.

Chapter 2

Basic Definitions

In this chapter we present all the definitions which are necessary for understanding this thesis.

2.1 Transition Systems

Transition systems are widely accepted as a structure which can exactly define operational semantics of programs by means of structural rules (see [Plo81]). This approach is especially advantageous in case of interactive concurrent systems which usually have quite complex input-output behaviour.

Definition 2.1. *A transition system is a tuple* $(S, Act, \rightarrow, r)$ *consisting of a set of* states S, *a set of* actions *(or* labels*)* Act, *a* transition relation $\rightarrow \subseteq S \times Act \times S$ *and a distinguished element* $r \in S$ *called* root.

The reflexive and transitive closure of '$\rightarrow$' is denoted by '$\rightarrow^*$'. As usual, we write $A \xrightarrow{a} B$ instead of $(A, a, B) \in \rightarrow$ and this notation is also extended to elements of Act^* in an obvious way. Moreover, we often write $A \rightarrow^* B$ instead of $A \xrightarrow{w} B$ if $w \in Act^*$ is irrelevant.

Given two states u, v of a transition system T, we say that v is *reachab* from u if $u \rightarrow^* v$. States of T which are reachable from the root of T ar said to be *reachable*.

2.2 Behavioural Equivalences

Before we start to deal with verification of concurrent systems we mus clarify the question what processes should be considered as "semanticall equivalent". As we already know, formal semantics of a concurrent sys tem is given by its corresponding transition system—but what transitio systems do exhibit the same behaviour? The answer is not easy; there ar many different approaches and consequently there are also many differer equivalences over the class of transition systems which deserve the adje tive "behavioural". R. van Glabbeek presented in [vG90] various equiva lences in a uniform way, relating them w.r.t. their *coarseness*, i.e., how man identifications they make. The resulting lattice is presented in Figure 2. The order is determined by the relation "makes strictly more identifica tions than".

The finest equivalence in this hierarchy is *bisimilarity* [Par81], define as follows:

Definition 2.2 (bisimilarity). *Let* $T_1 = (S_1, Act_1, \rightarrow_1, r_1)$, $T_2 = (S_2, Act_2, \rightarrow$ $r_2)$ *be transition systems. A binary relation* $R \subseteq S_1 \times S_2$ *is a* bisimulation *whenever* $(s, t) \in R$ *then for each* $a \in Act_1 \cup Act_2$

- *if* $s \xrightarrow{a}_1 s'$, *then* $t \xrightarrow{a}_2 t'$ *for some* t' *such that* $(s', t') \in R$
- *if* $t \xrightarrow{a}_2 t'$, *then* $s \xrightarrow{a}_1 s'$ *for some* s' *such that* $(s', t') \in R$

Transition systems T_1, T_2 *are* bisimilar, *written* $T_1 \sim T_2$, *if their roots are relate by some bisimulation.*

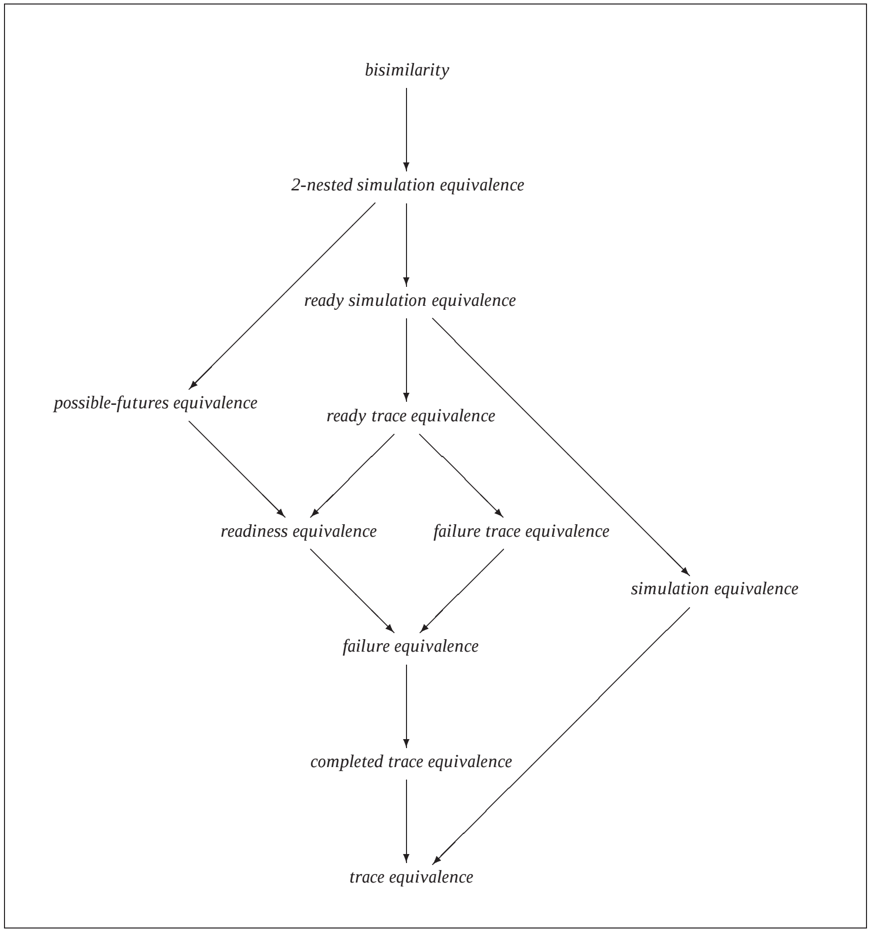


Figure 2.1: van Glabbeek's hierarchy of behavioural equivalences

Bisimilarity has many features which indicate that this equivalence is re ally something special. It is probably the most advantageous way ho to define "sameness" of two concurrent systems. Definitions of the othe equivalences from van Glabbeek's hierarchy are moved to Appendix A.

2.3 Process Algebras

The basic idea which stands behind the formalism of process algebras i that it is possible to define complicated behaviours from simple ones usin certain operators (e.g., parallel or sequential composition). In other word processes can carry an algebraic structure.

Many process algebras were proposed in the literature. They adop various sets of operators (the major difference is the kind of parallel ope ator and the way how to force cooperation between parallel components but those sets of operators usually have sufficient expressive power to sim ulate an arbitrary Turing machine—and therefore many interesting prob lems are generally undecidable. As examples of popular process algebra we can mention CCS [Mil89], ACP [BW90], or CSP [Hoa85].

In this thesis we present several positive decidability results about cer tain process algebras. It is thus clear that those process algebras canno have full Turing power—they are obtained as natural subclasses of pro cess algebras mentioned above.

2.3.1 BPA, BPP, BPA$_\tau$, BPP$_\tau$, PA — Subclasses of CCS an ACP

Let $\Lambda = \{a, b, c, \ldots\}$ be a countably infinite set of *atomic actions* such tha for each $a \in \Lambda$ there is a corresponding *dual* action $\overline{a}$ with the conventio that $\overline{\overline{a}} = a$. Let $Act = \Lambda \cup \{\tau\}$ where $\tau \notin \Lambda$ is a special (silent) actio Let $Var = \{X, Y, Z, \ldots\}$ be a countably infinite set of *variables* such tha

$Var \cap Act = \emptyset$. The classes of BPA, BPP, BPA$_\tau$, BPP$_\tau$, and PA expressions are defined by the following abstract syntax equations:

$$
\begin{array}{llllllllllllll}
E_{\text{BPA}} & ::= & \epsilon & | & b & | & bE_{\text{BPA}} & | & X & | & E_{\text{BPA}}.E_{\text{BPA}} & | & E_{\text{BPA}} + E_{\text{BPA}} \\
E_{\text{BPP}} & ::= & \epsilon & | & b & | & bE_{\text{BPP}} & | & X & | & E_{\text{BPP}}\|E_{\text{BPP}} & | & E_{\text{BPP}} + E_{\text{BPP}} \\
E_{\text{BPA}_\tau} & ::= & \epsilon & | & a & | & aE_{\text{BPA}_\tau} & | & X & | & E_{\text{BPA}_\tau}.E_{\text{BPA}_\tau} & | & E_{\text{BPA}_\tau} + E_{\text{BPA}_\tau} \\
E_{\text{BPP}_\tau} & ::= & \epsilon & | & a & | & aE_{\text{BPP}_\tau} & | & X & | & E_{\text{BPP}_\tau}|E_{\text{BPP}_\tau} & | & E_{\text{BPP}_\tau} + E_{\text{BPP}_\tau} \\
E_{\text{PA}} & ::= & \epsilon & | & b & | & bE_{\text{PA}} & | & X & | & E_{\text{PA}}.E_{\text{PA}} \;|\; E_{\text{PA}}\|E_{\text{PA}} \;|\; E_{\text{PA}}\lfloor\!\lfloor E_{\text{PA}} & | & E_{\text{PA}} + E_{\text{PA}}
\end{array}
$$

Here b ranges over Λ, a ranges over Act, and X ranges over Var. The symbol 'ϵ' denotes the empty expression.

As usual, we restrict our attention to guarded expressions. A BPA, BPP, BPA$_\tau$, BPP$_\tau$, or PA expression E is *guarded* if every variable occurrence in E is within the scope of an atomic action.

A *guarded BPA, BPP, BPA$_\tau$, BPP$_\tau$, or PA process* is defined by a finite family Δ of recursive process equations

$$\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid 1 \le i \le n\}$$

where X_i are distinct elements of Var and E_i are guarded BPA, BPP, BPA$_\tau$, BPP$_\tau$, or PA expressions, respectively, containing variables of $\{X_1, \ldots, X_n\}$. The set of variables which appear in Δ is denoted by $Var(\Delta)$.

The variable X_1 plays a special role (X_1 is sometimes called the *leading* variable)—it is a root of a labelled transition system, defined by the process Δ and SOS rules of Figure 2.2.

Presented rules should be considered modulo *structural congruence*, defined as follows:

Definition 2.3. *Let $\equiv$ be the smallest congruence relation over process expressions such that the following laws hold:*

- *associativity and 'ϵ' as a unit for sequential composition (the '.' operator).*

$$
\begin{array}{cccc}
\dfrac{}{aE \xrightarrow{a} E} & \dfrac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F} & \dfrac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'} & \dfrac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'} \\[2ex]
\dfrac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F} & \dfrac{F \xrightarrow{a} F'}{E\|F \xrightarrow{a} E\|F'} & \dfrac{E \xrightarrow{a} E'}{E\lfloor\!\lfloor F \xrightarrow{a} E'\|F} & \dfrac{E \xrightarrow{a} E'}{E|F \xrightarrow{a} E'|F} \\[2ex]
\dfrac{F \xrightarrow{a} F'}{E|F \xrightarrow{a} E|F'} & \dfrac{E \xrightarrow{b} E' \quad F \xrightarrow{\bar{b}} F'}{E|F \xrightarrow{\tau} E'|F'}\ (b \neq \tau) & & \dfrac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'}\ (X \stackrel{\text{def}}{=} E \in \Delta)
\end{array}
$$

Figure 2.2: SOS rules

- *associativity, commutativity, and 'ϵ' as a unit for pure merge (the '$\|$' oper ator).*
- *'ϵ' as a unit for left merge (the '$\lfloor\!\lfloor$' operator).*
- *associativity, commutativity, and 'ϵ' as a unit for CCS parallel compositio (the '$|$' operator).*
- *associativity, commutativity, and 'ϵ' as a unit for nondeterministic choic (the '$+$' operator).*
- *$a\epsilon = a$.*

States of the transition system generated by Δ are BPA, BPP, BPA$_\tau$, BPP$_\tau$ or PA expressions, which are also called *states of* Δ, or just "states" whe Δ is understood from the context.

Remark 2.4. *As each process determines a unique transition system, all notion which were originally defined for transition systems (see Section 2.1) can be use for processes too.*

Remark 2.5. *Guarded processes generate finitely branching transition system i.e., the set $\{F \mid E \xrightarrow{a} F,\ a \in Act\}$ is finite for each state E. It is easy to see that* *would not be true if we allowed unguarded expressions (assume e.g., the process $X \overset{def}{=} a \| X$).*

Remark 2.6. *Processes are often identified with their leading variables. Furthermore, if we assume a fixed process Δ, we can view any process expression E (not necessarily guarded) whose variables are defined in Δ as a process too; we simply add a new leading equation $X \overset{def}{=} E'$ to Δ, where X is a variable from Var such that $X \notin Var(\Delta)$ and E' is a process expression which is obtained from E by substituting each variable in E with the right-hand side of its corresponding defining equation in Δ (E' must be guarded now). All notions originally defined for processes can be used for process expressions in this sense too.*

2.3.2 Normal Forms

Many definitions and proofs in this thesis take advantage of the fact that BPA, BPP, BPA$_\tau$, BPP$_\tau$, and PA processes can be equivalently (up to bisimilarity) represented in special normal forms. Moreover, those normal forms can be effectively constructed.

Definition 2.7 (GNF for BPA and BPA$_\tau$ processes). *A BPA (or BPA$_\tau$) process Δ is said to be in* Greibach normal form (GNF) *if all its defining equations are of the form*

$$X \overset{def}{=} \sum_{j=1}^{n} a_j \alpha_j$$

where $n \in N$, $a_j \in \Lambda$ (or $a_j \in Act$) and $\alpha_j \in Var(\Delta)^*$. If length$(\alpha_j) \leq 2$ for each j, $1 \leq j \leq n$, then Δ is said to be in 3-GNF. Moreover, we also require that for each $Y \in Var(\Delta)$ there is a reachable state $\alpha \in Var(\Delta)^*$ such that α begins with Y.

Any BPA or BPA$_\tau$ process can be effectively transformed into 3-GNF (see [BBK87]). A similar normal form exists also for BPP and BPP$_\tau$ processes (see [Chr93]). Before the definition we need to introduce the set $Var(\Delta)^\otimes$

of all finite multisets over $Var(\Delta)$. Each multiset of $Var(\Delta)^\otimes$ denotes a BP (or BPP$_\tau$) expression by combining its elements in parallel using the '‖ operator (or the '|' operator).

Definition 2.8 (GNF for BPP and BPP$_\tau$ processes). *A BPP (or BPP$_\tau$) pro cess Δ is said to be in* Greibach normal form (GNF) *if all its defining equatior are of the form*

$$X \overset{def}{=} \sum_{j=1}^{n} a_j \alpha_j$$

where $n \in N$, $a_j \in \Lambda$ (or $a_j \in Act$) and $\alpha_j \in Var(\Delta)^\otimes$. If card$(\alpha_j) \leq 2$ *for each $1 \leq j \leq n$, then Δ is said to be in 3-GNF. Moreover, we also require that for eac $Y \in Var(\Delta)$ there is a reachable state $\alpha \in Var(\Delta)^\otimes$ such that $Y \in \alpha$.*

A normal form for PA processes is a generalization of Greibach norma form. First we need to define the set of *VPA* expressions.

1. The empty expression 'ϵ' is a *VPA* expression.
2. Each variable $X \in Var(\Delta)$ is a *VPA* expression.
3. If α, β are nonempty *VPA* expressions, then $\alpha.\beta$, $\alpha \| \beta$, and $\alpha \lfloor\!\lfloor \beta$ ar *VPA* expressions.
4. Each *VPA* expression can be constructed using the rules 1, 2 and 3 i a finite number of steps.

The set of *VPA* expressions which contain only variables from $Var(\Delta$ where Δ is a PA process, is denoted $VPA(\Delta)$. Finally, the set of variable which appear in a *VPA* expression α is denoted $Var(\alpha)$.

Definition 2.9 (normal form for PA processes). *A PA process Δ is said to b in normal form if all its equations are of the form*

$$X \overset{def}{=} \sum_{j=1}^{n} a_j \alpha_j$$

where $n \in N$, $a_j \in \Lambda$ and $\alpha_j \in VPA(\Delta)$. Moreover, we also require that for each $Y \in Var(\Delta)$ there is a reachable state $\alpha \in VPA(\Delta)$ such that $Y \in Var(\alpha)$.

Any PA process can be effectively presented in the normal form just defined (see [BEH95]).

From now on we assume that all BPA, BPP, BPA$_\tau$, BPP$_\tau$, and PA processes we are working with are presented in corresponding normal forms. This justifies also the assumption that reachable states of a BPA, BPP, BPA$_\tau$, BPP$_\tau$, or PA process Δ are elements of $Var(\Delta)^*$, $Var(\Delta)^\otimes$, $Var(\Delta)^*$, $Var(\Delta)^\otimes$, or $VPA(\Delta)$, respectively.

The following overloaded function is needed in some proofs of this thesis:

Definition 2.10 (*Length* function). *The function Length is defined for VPA expressions and elements of Act^*. In the first case it returns the number of variables which are contained in its argument, distinguishing multiple occurrence of the same variable. In the latter case it returns the length of its argument. For example, $Length(X.(Y\|X)) = 3$ and $Length(aabac) = 5$.*

2.3.3 Normed Processes

Important subclasses of BPA, BPP, BPA$_\tau$, BPP$_\tau$, and PA processes can be obtained by an extra restriction of *normedness*. A variable $X \in Var(\Delta)$ is *normed* if there is $w \in Act^*$ such that $X \xrightarrow{w} \epsilon$. In that case we define the *norm* of X, written $|X|$, to be the length of the shortest such w. In case of BPP$_\tau$ processes we also require that no τ action which appears in w is a result of a communication on dual actions in the sense of operational semantics given in Figure 2.2. This is necessary if we want the norm to be additive over the '$|$' operator (τ may still occur in w—it can be used as an action prefix). A process Δ is *normed* if all variables of $Var(\Delta)$ are normed. The norm of Δ is then defined to be the norm of its leading variable X_1.

Remark 2.11. *As normed processes are intensively studied in this thesis, we emphasize some properties of the norm:*

- *The norm of a normed process is easily computed by the following rules:*
  - $|a| = 1$
  - $|E + F| = \min\{|E|, |F|\}$
  - $|E.F| = |E| + |F|$
  - $|E\|F| = |E| + |F|$
  - $|E \lfloor\!\lfloor F| = |E| + |F|$
  - $|E|F| = |E| + |F|$
  - *if $X_i \stackrel{def}{=} E_i$ and $|E_i| = n$, then $|X_i| = n$.*
- *Bisimilar processes must have the same norm.*

In the rest of this thesis we use the prefix 'n' for denoting the norme subclass, writing e.g., 'nBPA' instead of 'normed BPA'.

2.3.4 Regular Processes

One of the problems considered in this thesis is decidability of regularit for certain process classes. The next definition explains what is meant b the notion of regularity.

Definition 2.12 (regularity). *Let $\leftrightarrow$ be an equivalence over the class of trans tion systems. A process Δ is* regular w.r.t. $\leftrightarrow$ *if there is a process Δ' with finitel many states such that $\Delta \leftrightarrow \Delta'$.*

In [Mil89] it is shown that finite-state processes (and hence also processe which are regular w.r.t. bisimilarity) can be represented in the followin normal form:

Definition 2.13 (normal form for finite-state processes). *A finite-state process Δ is said to be in normal form if all its equations are of the form*

$$X \stackrel{def}{=} \sum_{j=1}^{n} a_j[X_j]$$

where $n \in N$, $a_j \in Act$ and $X_j \in Var(\Delta)$ (square brackets indicate optional occurrence).

Chapter 3

Deciding Regularity in Process Algebras

Process algebras provide us with a very powerful syntax which can describe concurrent systems with finitely as well as infinitely many states. Since the very beginning people have concentrated on finite-state processes. Consequently, the theory of finite-state processes is well established today and it is also applied—there are many automated tools which can answer plenty of interesting questions about finite-state processes.

Now we can ask whether it is possible to extend those nice results to process classes which contain also processes with infinitely many states. This is problematic of course—many problems become undecidable and even if some property remains decidable, the algorithm is often not interesting from the practical point of view due to its complexity. If we want to examine features of some process Δ with infinitely many states, a good idea is to ask whether there is an *equivalent* finite-state process Δ' which could be analyzed instead of Δ—and this is exactly what we mean by the *regularity problem*. Naturally, we can also ask whether such a process Δ' can be effectively constructed.

This chapter is devoted to the regularity problem. In Section 3.1 we

prove that regularity (w.r.t. bisimilarity) is decidable for nPA processes i polynomial time. Moreover, if a nPA process Δ is regular, then it is als possible to construct a bisimilar finite-state process Δ' in normal form (se Definition 2.13). These results have been previously published in [Kuč96a and [Kuč96b].

In Section 3.2 we discuss the regularity problem w.r.t. other behaviour equivalences. We design and justify new notions of *finite characterizatio* and *strong regularity* and we study their relationship. This section is base on [Kuč95].

Section 3.3 contains some negative (undecidability) results. We explor a calculus PAPDA obtained from PA by adding a finite-state control uni We show that an arbitrary Minsky machine [Min67] can be simulated by (normed) PAPDA process which is effectively constructible. This implie undecidability of regularity and strong regularity w.r.t. any equivalence c van Glabbeek's hierarchy.

In Section 3.4 we summarize related results which are known at th time of writing this thesis and we also mention major open problems.

3.1 Regularity of Normed PA Processes

In this section we show that regularity w.r.t. bisimilarity (Definition 2.12 is decidable for nPA processes in polynomial time (we speak just abou "regularity" for short). The basic idea is quite simple—reachable state of a nPA process Δ are elements of *VPA*(Δ) (see Definition 2.9). As Δ is normed, each of its reachable states has a finite norm. As the norm i additive over '$\|$', '$\lfloor\!\lfloor$' and '.' operators (see Remark 2.11), there are onl finitely many elements of *VPA*(Δ) with a given finite norm. Hence Δ ca reach infinitely many states up to bisimilarity iff it can reach a state of a arbitrary norm. As we shall see, this condition can be easily verified i polynomial time.

We also show that if a nPA process Δ is regular, then it is possible to construct a bisimilar finite-state process Δ' in normal form (see Definition 2.13). However, this algorithm is of exponential space complexity, because a regular nPA process with n variables can generally reach exponentially many pairwise non-bisimilar states and each such state requires a special variable.

Lemma 3.1. *A process Δ is not regular iff there is an infinite path $X_1 = \alpha_0 \stackrel{a_0}{\to} \alpha_1 \stackrel{a_1}{\to} \alpha_2 \stackrel{a_2}{\to} \cdots$ such that $\alpha_i \not\sim \alpha_j$ for $i \neq j$.*

Proof:
"$\Leftarrow$" Obvious—Δ can reach infinitely many pairwise non-bisimilar states.
"$\Rightarrow$" Let $T = (S, Act, \to, r)$ be the transition system generated by Δ. If we identify bisimilar states of T, we obtain a transition system $T' = (S', Act, \to', r')$ where

- S' contains equivalence classes of $S/\!\sim$ (the equivalence class which contains $E \in S$ is denoted by $[E]$)
- the relation $\to'$ is determined by the rule $E \stackrel{a}{\to} F \;\Rightarrow\; [E] \stackrel{a}{\to}' [F]$
- $r' = [r]$

Clearly $T \sim T'$. Moreover, T' is infinite but finitely branching (see Remark 2.5), hence due to König's lemma there must be an infinite path $[X_1] \stackrel{a_0}{\to}' [E_1] \stackrel{a_1}{\to}' [E_2] \stackrel{a_2}{\to}' [E_3] \stackrel{a_3}{\to}' \cdots$, where X_1 is the leading variable of Δ. If $F \in [E_i]$, then $F \stackrel{a_i}{\to} G$ for some $G \in [E_{i+1}]$ (it follows directly from the definition of bisimulation—see Definition 2.2). Hence the required path in T can be constructed just by taking suitable representatives of $[E_i]$ for each $i \in N$. □

3.1.1 The Inheritance Tree

Let Δ be a nPA process. The aim of the following definition is to describe all variables in a state $\alpha \in VPA(\Delta)$ which can potentially emit an action:

Definition 3.2 (*FIRE* set). *Let Δ be a nPA process. For each $\alpha \in VPA(\Delta)$ w define the set $FIRE(\alpha)$ in the following way:*

$$FIRE(\alpha) = \begin{cases} \emptyset & \text{if } \alpha = \epsilon \\ \{X\} & \text{if } \alpha = X \\ FIRE(\beta_1) & \text{if } \alpha = \beta_1.\beta_2 \text{ or } \alpha = \beta_1 \lfloor\!\lfloor \beta_2 \\ FIRE(\beta_1) \cup FIRE(\beta_2) & \text{if } \alpha = \beta_1 \| \beta_2 \end{cases}$$

Lemma 3.3. *Let Δ be a nPA process, $\alpha \in VPA(\Delta)$. Then for each $X \in Var(\alpha$ there is $\beta \in VPA(\Delta)$ such that $\alpha \to^* \beta$ and $X \in FIRE(\beta)$.*

Proof: By induction on the structure of α:

- $\alpha = \mathbf{X}$: Obvious.
- **induction step:** The expression α can be of three forms: $\alpha = \gamma.$ $\alpha = \gamma \| \delta$ or $\alpha = \gamma \lfloor\!\lfloor \delta$. Furthermore, there are two possibilities:
  1. X appears within γ. Then (by ind. hypothesis) $\gamma \to^* \gamma'$ for som γ' such that $X \in FIRE(\gamma')$. Hence $\alpha \to^* \gamma'.\delta$, $\alpha \to^* \gamma' \| \delta$, o $\alpha \to^* \gamma' \lfloor\!\lfloor \delta$, respectively. Clearly $X \in FIRE(\gamma'.\delta)$, $X \in FIRE(\gamma' \| \delta$ or $X \in FIRE(\gamma' \lfloor\!\lfloor \delta)$, respectively.
  2. X appears within δ. Then (by ind. hypothesis) $\delta \to^* \delta'$ for som δ' such that $X \in FIRE(\delta')$. Moreover, $\alpha \to^* \delta$, hence $\alpha \to^* \delta'$ an the proof is finished. □

The following concept stands behind many constructions of this section:

Definition 3.4 (*Tail* set). *For each $\alpha \in VPA$ we define the set $Tail(\alpha) \subseteq Var$ i the following way:*

$$Tail(\alpha) = \begin{cases} \{X\} & \text{if } \alpha = X \\ \emptyset & \text{if } \alpha = \epsilon \text{ or } \alpha = \beta \| \gamma \text{ where } \beta \neq \epsilon \neq \gamma \\ Tail(\gamma) - Var(\beta) & \text{if } \alpha = \beta.\gamma \text{ or } \alpha = \beta \lfloor\!\lfloor \gamma \text{ where } \beta \neq \epsilon \neq \gamma \end{cases}$$

Remark 3.5. *The set $Tail(\alpha)$ provides two important pieces of information:*

1. *If $X \in Var(\alpha)$ such that $X \notin Tail(\alpha)$, then there is α' such that $\alpha \rightarrow^* \alpha'$, $X \in FIRE(\alpha')$ and $Length(\alpha') \geq 2$.*

2. *If $X \in Tail(\alpha)$, then the only occurrence of X in α can become active (i.e., X can emit an action) after all other variables disappear.*

Definition 3.6 (growing variable). *Let Δ be a nPA process. A variable $X \in Var(\Delta)$ is* growing *if there is $\alpha \in VPA(\Delta)$ such that $X \rightarrow^* \alpha$, $X \in FIRE(\alpha)$ and $Length(\alpha) \geq 2$.*

Lemma 3.7. *Let Δ be a nPA process. The problem whether $Var(\Delta)$ contains a growing variable is decidable in polynomial time.*

Proof: We define the binary relation $GROW$ on $Var(\Delta)$ in the following way:

$$(X, Y) \in GROW \overset{def}{\iff} \exists \beta \in VPA(\Delta) \text{ such that } X \rightarrow^* \beta \text{ where } Length(\beta) \geq 2 \text{ and } Y \in FIRE(\beta).$$

Clearly $Var(\Delta)$ contains a growing variable iff there is $X \in Var(\Delta)$ such that $(X, X) \in GROW$. We show that the relation $GROW$ can be effectively constructed in polynomial time. We need two auxiliary binary relations on $Var(\Delta)$:

$X \rightsquigarrow Y \quad \overset{def}{\iff}$ there is a summand $a\alpha$ in the defining equation for X in Δ such that $Length(\alpha) \geq 2$, $Y \in Var(\Delta)$ and $Y \notin Tail(\alpha)$

$X \hookrightarrow Y \quad \overset{def}{\iff}$ there is a summand $a\alpha$ in the defining equation for X in Δ such that $Y \in Var(\alpha)$.

It is easy to prove that $GROW = \hookrightarrow^* \,.\, \rightsquigarrow \,.\, \hookrightarrow^*$ where $\hookrightarrow^*$ denotes the reflexive and transitive closure of $\hookrightarrow$. Moreover, the composition $\hookrightarrow^* \,.\, \rightsquigarrow \,.\, \hookrightarrow^*$ can be constructed in polynomial time. □

Let Δ be a nPA process. If Δ is not regular then there is (due to Lemma 3.1 an infinite path $\mathcal{P}$ of the form $X_1 = \alpha_0 \xrightarrow{a_0} \alpha_1 \xrightarrow{a_1} \alpha_2 \xrightarrow{a_2} \cdots$ such tha $\alpha_i \not\sim \alpha_j$ for $i \neq j$. To be able to examine properties of $\mathcal{P}$ in a detail, w define for $\mathcal{P}$ the corresponding *inheritance tree*, denoted $IT_{\mathcal{P}}$. The aim o this construction is to describe the relationship between variables whic are located in successive states of $\mathcal{P}$. The way how $IT_{\mathcal{P}}$ is constructed i similar to the construction of a derivation tree for a word $w \in L(G)$ wher $L(G)$ is a language generated by a context-free grammar G. We start wit an example which shows how $IT_{\mathcal{P}}$ looks for a given prefix of $\mathcal{P}$.

Example 3.8. *Let Δ be a nPA process given by the following set of equations:*

$$X \overset{def}{=} b + a(Y.(Z\|Y))$$
$$Y \overset{def}{=} c + b(Y.Z.X)$$
$$Z \overset{def}{=} a + a((Z\|Y).X)$$

Let $\mathcal{P} = X \xrightarrow{a} Y.(Z\|Y) \xrightarrow{c} Z\|Y \xrightarrow{a} ((Z\|Y).X)\|Y \xrightarrow{b} ((Z\|Y).X)\|(Y.Z.X) \cdots$. we draw a fragment of $IT_{\mathcal{P}}$, we get the tree of Figure 3.1.

Nodes of $IT_{\mathcal{P}}$ are labelled with variables of $Var(\Delta)$. The state α_i, $i \in N \cup \{0$ of $\mathcal{P}$ corresponds to the set of nodes in $IT_{\mathcal{P}}$ which have the distance i fron the root of $IT_{\mathcal{P}}$ (the root itself has the distance 0). This set of nodes is calle the i^{th} *Level* of $IT_{\mathcal{P}}$. Each transition $\alpha_i \xrightarrow{a_i} \alpha_{i+1}$ is due to a single variabl $A \in Var(\alpha_i)$ and a transition $A \xrightarrow{a_i} \gamma$ where the expression $a_i\gamma$ is a summan in the defining equation for A in Δ (see Definition 2.9). Moreover, α_{i+} can be obtained from α_i by replacing one occurrence of A with γ (here w must distinguish between multiple occurrence of the variable A within th state α_i). We call the variable A the *active variable* of α_i and the transitio $A \xrightarrow{a_i} \gamma$ the *step* of α_i. The nodes of $IT_{\mathcal{P}}$ which correspond to active variable are called *active*. Each active node is placed within a box in the tree o Figure 3.1.

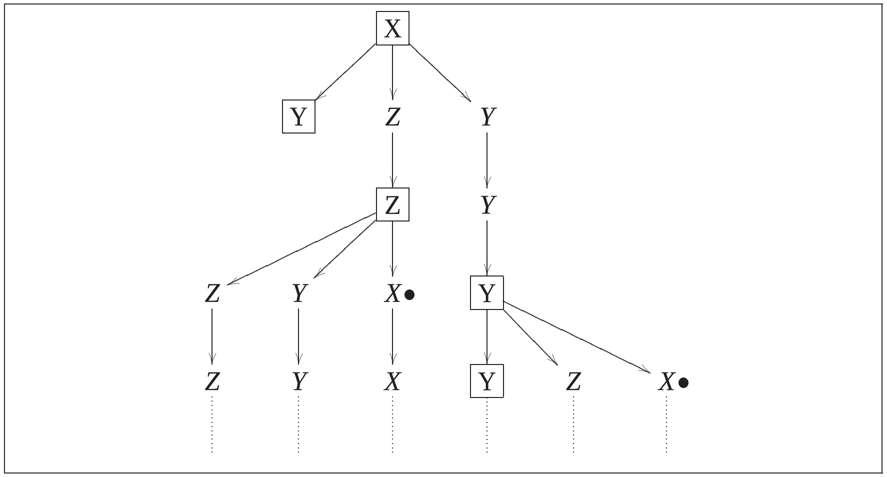


Figure 3.1: The inheritance tree associated with the path $\mathcal{P}$.

Nodes and edges of $IT_{\mathcal{P}}$ are defined inductively—we define all nodes in *Level* $i+1$ together with their labels, using the nodes from *Level* i. Moreover, we also define all edges between nodes in these two levels.

1. **i=0:** There is just one node N in *Level* 0 — the root, labelled X_1.
2. **induction step:** Let us suppose that nodes of *Level* i have been already defined. For each node U of *Level* i we define its immediate successors. There are two possibilities:
   - **U is not active:** Then U has just one immediate successor whose label is the same as the label of U.
   - **U is active:** Let $A \xrightarrow{a_i} \gamma$ be the step of α_i and let $n = Length(\gamma)$. The node U (whose label is A) has n immediate successors (if $n = 0$ then U is a leaf). The l^{th} immediate successor of U is labelled by the l^{th} variable from γ, reading γ from left to right. Here l ranges from 1 to n. As we cannot afford to lose the information about the structure of γ completely, we distinguish

the case when $Tail(\gamma) = \{B\}$ where $B \in Var(\Delta)$. Then we sa that the last successor of U is a *tail* of U. All tails in the tree c Figure 3.1 are marked with a black dot.

A node of $IT_{\mathcal{P}}$ which has at least two immediate successors is called *branching node*. Branching nodes are especially important because the labels are potential candidates to be growing variables. This is the basi idea which stands behind the notion of *Allow* set.

Definition 3.9 (*Allow* set). *For each node U of $IT_{\mathcal{P}}$ we define the set $Allow(U)$ $Var(\Delta)$ in the following way:*

- *If U is the root of $IT_{\mathcal{P}}$, then $Allow(U) = Var(\Delta)$.*
- *If U is an immediate successor of a node V, then*
  - *If V is not branching, then $Allow(U) = Allow(V)$.*
  - *If V is branching and U is not a tail of V, then $Allow(U) = Allow(V)$ $\{Label(V)\}$.*
  - *If V is branching and U is a tail of V, then $Allow(U) = Allow(V)$.*

The next lemma explains what is the relationship between a node U an its associated set $Allow(U)$:

Lemma 3.10. *Let U be a node of $IT_{\mathcal{P}}$. If $Label(U) \notin Allow(U)$ then $Label(U)$ a growing variable.*

Proof: Let $A = Label(U)$. As $A \notin Allow(U)$, the node U has an ancesto V such that $Label(V) = A$, V is branching and U is a descendant of a immediate successor V' of V which is not a tail of V. Let $B = Label(V'$ As V is branching, it is also active and hence it corresponds to some ste $A \xrightarrow{a} \gamma$ where $B \in Var(\gamma)$ and $B \notin Tail(\gamma)$. Moreover, $\gamma \to^* \gamma'$ for some such that $B \in FIRE(\gamma')$ and $Length(\gamma') \geq 2$ (see Remark 3.5). Furthermore as U is a descendant of V', $B \to^* \delta$ where A is contained in δ. Due t

Lemma 3.3 there is δ' such that $\delta \to^* \delta'$ and $A \in FIRE(\delta')$. To sum up, we have $A \to^* \gamma' \to^* \eta$ where η is obtained from γ' by substituting B with δ'. Clearly $Length(\eta) \geq 2$ and $A \in FIRE(\eta)$, hence A is growing as required. □

Now we prove the first main theorem of this chapter:

Theorem 3.11. *A nPA process Δ is regular iff $Var(\Delta)$ does not contain any growing variable.*

Proof:
"$\Rightarrow$" If $Var(\Delta)$ contains a growing variable X, then Δ is non-regular as it can reach a state of an arbitrary norm. To see this, it suffices to realize that $X \to^* \gamma$ where $Length(\gamma) \geq 2$ and $X \in FIRE(\gamma)$. Moreover, there is a reachable state α of Δ such that $X \in FIRE(\alpha)$. Now we can repeatedly substitute X by γ within α, producing a reachable state of an arbitrary *Length* (and hence also norm).

"$\Leftarrow$" This part of the proof is more complicated. The basic scheme is similar to the method which was used by Mauw and Mulder in [MM94] and can be described in the following way: We need to show that if Δ is not regular then there is a growing variable $X \in Var(\Delta)$. As Δ is not regular, there is (due to Lemma 3.1) an infinite path $\mathcal{P}$ of the form $X_1 = \alpha_0 \xrightarrow{a_0} \alpha_1 \xrightarrow{a_1} \alpha_2 \xrightarrow{a_2} \cdots$ such that $\alpha_i \not\sim \alpha_j$ for $i \neq j$. We show that if $Var(\Delta)$ does not contain any growing variable, then there are $i \neq j$ such that $\alpha_i \sim \alpha_j$. It contradicts the assumption above—hence $Var(\Delta)$ contains at least one growing variable.

Let $IT_{\mathcal{P}}$ be the inheritance tree for the path $\mathcal{P}$. To complete the proof we need to divide $IT_{\mathcal{P}}$ into more manageable units called *blocks*. Levels of $IT_{\mathcal{P}}$ which contain just one node are called *delimiters* of $IT_{\mathcal{P}}$. A *block* of $IT_{\mathcal{P}}$ is a subgraph S of $IT_{\mathcal{P}}$ composed of:

1. all nodes and edges between two successive delimiters i and j where $i < j$. The only node of *Level* i is called the *opening* node of S and the only node of *Level* j is called the *closing* node of S. Out-going edge of the closing node and in-going edges of the opening node are not part of S.

2. all nodes below the delimiter i (including *Level* i), if there is no de limiter j with $j > i$. The only node of *Level* i is called the *opening* nod of S. In-going edges of the opening node are not a part of S.

As *Level* 0 is a delimiter of $IT_{\mathcal{P}}$, we can view $IT_{\mathcal{P}}$ as a vertical sequenc of blocks. The *width* of $IT_{\mathcal{P}}$ is defined to be the least $n \in N$ such tha cardinality of the i^{th} *Level* of $IT_{\mathcal{P}}$ is less or equal n for each $i \in N \cup \{0\}$. there is no such n, the width of $IT_{\mathcal{P}}$ is defined to be ∞. Similarly, if S i a block of $IT_{\mathcal{P}}$, the *width* of S is the least $n \in N$ such that the cardinalit of each *Level* which is a part of S is less or equal n. If there is no such the width of S is ∞. Furthermore, we define the *branching degree* of $IT_{\mathcal{P}}$ t be the least $n \in N$ such that each node U of $IT_{\mathcal{P}}$ has at most n immediat successors. The branching degree of $IT_{\mathcal{P}}$ is always finite—it is denote by $\mathcal{D}$ in the rest of this proof. Each node U of $IT_{\mathcal{P}}$ defines its associate subtree, rooted by U. This subtree is denoted $Subtree(U)$. Although th notions of block, width, tail, branching node, etc. were originally define for $IT_{\mathcal{P}}$, they can be used also for any $Subtree(U)$ of $IT_{\mathcal{P}}$ in an obviou way. We prove that if $Var(\Delta)$ does not contain any growing variable, the for each node U of $IT_{\mathcal{P}}$ the $Subtree(U)$ has the width at most $\mathcal{D}^{n-1}$, wher $n = \mathrm{card}(Allow(U))$. We proceed by induction on $n = \mathrm{card}(Allow(U))$.

1. **n=0:** If $Allow(U) = \emptyset$, then clearly $Label(U) \notin Allow(U)$ and henc $Label(U)$ is growing due to Lemma 3.10. So the implication triviall holds.

2. **induction step:** Let $\mathrm{card}(Allow(U)) = n$. We prove that each block c $Subtree(U)$ has the width at most $\mathcal{D}^{n-1}$. Let S be a block of $Subtree(U$ and let V be its opening node. Clearly $\mathrm{card}(Allow(V)) \leq n$. If V ha no successors then the width of S is 1. If V is not branching the

the only immediate successor of V is the closing node of S, thus the width of S equals 1. If V is branching, there are two possibilities:

- V does not have a tail. Then each immediate successor V' of V has the property $\mathrm{card}(\mathit{Allow}(V')) \leq n-1$. By the inductive hypothesis, the width of $\mathit{Subtree}(V')$ is at most $\mathcal{D}^{n-2}$. As V can have at most $\mathcal{D}$ immediate successors, the width of $\mathit{Subtree}(V)$ is at most $\mathcal{D}.\mathcal{D}^{n-2} = \mathcal{D}^{n-1}$. Thus the width of S is also at most $\mathcal{D}^{n-1}$.
- V has a tail T. Each immediate successor V' of V which is different from T has the property $\mathrm{card}(\mathit{Allow}(V')) \leq n-1$. Hence we can use the inductive hypothesis for each such V'. The only problem is the node T. However, it suffices to realize that if T has a branching successor, then the first active successor of T is the closing node of S (see Remark 3.5). Hence the width of S is at most $(\mathcal{D}-1).\mathcal{D}^{n-2}+1$.

We have just proved that if $\mathit{Var}(\Delta)$ does not contain any growing variable then the width of $IT_{\mathcal{P}}$ is at most $\mathcal{D}^{\mathrm{card}(\mathit{Var}(\Delta))-1}$. As $\mathit{Var}(\Delta)$ is finite, there are only finitely many $\mathit{VPA}(\Delta)$ expressions with this bounded *Length*. Hence $\alpha_i = \alpha_j$ for some $i \neq j$ and thus $\alpha_i \sim \alpha_j$. □

3.1.2 A Construction of the Process Δ' in Normal Form

In this section we show that if a given nPA process Δ is regular, then Δ can be effectively transformed into a finite-state process Δ' in normal form such that $\Delta \sim \Delta'$. Our algorithm is based on the following fact (see Definition 2.3):

Lemma 3.12. *A nPA process Δ is regular iff Δ can reach only finitely many states up to $\equiv$.*

Our algorithm finds all reachable states $\alpha \in \mathit{VPA}(\Delta)$ of Δ up to $\equiv$. Fo each such α a new variable and corresponding defining equation is adde to Δ'.

The relationship between variables of Δ' and reachable states of Δ i described by the set $\mathit{MEM} \subseteq \mathit{Var} \times \mathit{VPA}(\Delta)$. This set is initialized t $\mathit{MEM} = \{(Y_1, X_1)\}$ where X_1 is the leading variable of Δ and Y_1 is th leading variable of Δ'.

An element (Y, α) of *MEM* is said to be *undefined* if there is no definin equation for Y in Δ'. The algorithm chooses any undefined element o *MEM* and adds a new defining equation for Y to Δ', possibly producin new undefined elements of *MEM*. The algorithm halts when *MEM* doe not contain any undefined elements.

Let (Y, α) be an undefined element of *MEM*. To obtain the definin equation for Y, the expression α must be first *unfolded*. The function *Unfol* is defined as follows:

$$\mathit{Unfold}(\alpha) = \begin{cases} \sum a_{ij}\alpha_{ij} & \text{if } \alpha = X_j \text{ and } X_j \stackrel{\mathit{def}}{=} \sum a_{ij}\alpha_{ij} \in \Delta \\ \mathit{Distr}(\mathit{Unfold}(\beta_1), \beta_2) & \text{if } \alpha = \beta_1.\beta_2 \\ \mathit{Expand1}(\beta_1, \beta_2) & \text{if } \alpha = \beta_1 \| \beta_2 \\ \mathit{Expand2}((\beta_1, \beta_2) & \text{if } \alpha = \beta_1 \lfloor\!\lfloor \beta_2 \end{cases}$$

where *Expand1*, *Expand2* and *Distr* are defined as follows (the function *Expand1* and *Expand2* are instances of the CCS expansion law (see [Mil89 and the function *Distr* is a form of the right distributivity law (see [BW90])

$$\begin{aligned} \mathit{Expand1}(\beta_1, \beta_2) &= \sum\{\, a(\beta_1' \| \beta_2) \mid \beta_1 \xrightarrow{a} \beta_1', a \in \mathit{Act}\} \\ &+ \sum\{\, a(\beta_1 \| \beta_2') \mid \beta_2 \xrightarrow{a} \beta_2', a \in \mathit{Act}\} \\ \mathit{Expand2}(\beta_1, \beta_2) &= \sum\{\, a(\beta_1' \| \beta_2) \mid \beta_1 \xrightarrow{a} \beta_1', a \in \mathit{Act}\} \\ \mathit{Distr}(\sum a_{ij}\alpha_{ij}, \beta) &= \sum a_{ij}(\alpha_{ij}.\beta) \end{aligned}$$

The function *Unfold* returns an expression of the form

$$\sum_{i=1}^{n} a_i \alpha_i$$

where $n \in N$, $a_i \in Act$ and $\alpha_i \in VPA(\Delta)$. Now the algorithm replaces each α_i with a single variable. There are two possibilities: if the set *MEM* contains an element (Z, β) such that $\alpha_i \equiv \beta$, then the expression α_i is replaced with Z. Otherwise, the expression α_i is replaced with a new variable W and the pair (W, α_i) is added to *MEM*. After the replacement of each α_i the defining equation for Y is obtained and it is added to Δ'.

It is easy to see that each variable of Δ' corresponds to a reachable state of the process Δ'. Hence the algorithm has to terminate (due to Lemma 3.12).

Example 3.13. *Let Δ be a nPA process given by the following set of equations:*

$$\begin{aligned} X &\stackrel{def}{=} b + a(Y \| Z).X \\ Y &\stackrel{def}{=} c + a(Z \| (Z.Z)) \\ Z &\stackrel{def}{=} c \end{aligned}$$

The process Δ' is constructed in the following way (the first two elements of each line constitute an element of *MEM*, the third element is a result of *Unfold* and the last element is the defining equation):

A	$=$	X	$=$	$b + a(Y\|Z).X$	$=$	$b + aB$
B	$=$	$(Y\|Z).X$	$=$	$a(Z\|(Z.Z)\|Z).X + c(Z.X) + c(Y.X)$	$=$	$aC + cD + cE$
C	$=$	$(Z\|(Z.Z)\|Z).X$	$=$	$c((Z\|Z\|Z).X) + c((Z\|(Z.Z)).X)$	$=$	$cF + cG$
D	$=$	$Z.X$	$=$	cX	$=$	cA
E	$=$	$Y.X$	$=$	$cX + a((Z\|(Z.Z)).X)$	$=$	$cA + aG$
F	$=$	$(Z\|Z\|Z).X$	$=$	$c((Z\|Z).X)$	$=$	cH
G	$=$	$(Z\|(Z.Z)).X$	$=$	$c(Z.Z.X) + c((Z\|Z).X)$	$=$	$cI + cH$
H	$=$	$(Z\|Z).X$	$=$	$c(Z.X)$	$=$	cD
I	$=$	$(Z.Z.X)$	$=$	$c(Z.X)$	$=$	cD

Using this algorithm it is possible to decide bisimilarity for any pair of pro cesses (Δ_1, Δ_2), where Δ_1 is a nPA process and Δ_2 has finitely many state First, we check whether Δ_1 is regular. If not, then $\Delta_1 \not\sim \Delta_2$. Otherwise, w construct a finite-state process Δ_1' in normal form such that $\Delta_1 \sim \Delta_1'$ an check whether $\Delta_1' \sim \Delta_2$.

Theorem 3.14. *Bisimulation equivalence is decidable for any pair of process such that one process of this pair is a nPA process and the other process has finitel many states.*

3.1.3 Possible Generalization

We already mentioned that the major difference between various proces algebras is the kind of parallel operator they are equipped with. For ex ample, CCS has the '|' operator which allows synchronizations on comple mentary actions (see Section 2.3.1). An obvious question is, whether it i possible to replace the '‖' operator with the '|' operator in the definition o nPA processes without the loss of decidability of regularity. In this particu lar case the answer is positive. All constructions used in previous section are still valid. This is basically due to the fact that synchronizations car not be *forced*—each 'τ' action which is a result of some synchronization ca be "decomposed" into a sequence of two transitions with complementar labels. Consequently, we can "decompose" an arbitrary sequence of trar sitions in such a way that each transition is due to a *single* variable. Ou result on nPA processes can be thus applied to nBPA, nBPP, nBPA$_\tau$, an nBPP$_\tau$ processes as follows:

Definition 3.15. *Let Δ be a nBPA, nBPP, nBPA$_\tau$, or nBPP$_\tau$ process. A variab $X \in Var(\Delta)$ is* growing *if $X \to^* X.\alpha$, $X \to^* X\|\alpha$, $X \to^* X.\alpha$, or $X \to^* X|$ where α is a nonempty expression, respectively.*

Proposition 3.16. *A nBPA, nBPP, nBPA$_\tau$, or nBPP$_\tau$ process Δ is regular i $Var(\Delta)$ does not contain any growing variable.*

Naturally, there are also well-known parallel operators which cannot be plugged into nPA syntax so painlessly—if we choose e.g., the '$\|_A$' operator of CSP (see [Hoa85]) which forces synchronizations on actions from A, regularity becomes undecidable. This basically due to the fact that *counters* can be simulated using the '$\|_A$' operator. Those counters can be combined in parallel with a finite-state control unit and forced to cooperate with it. In other words, using this operator it is possible to simulate an arbitrary Minsky machine (see [Min67]). Undecidability of regularity follows from a simple reduction of the halting problem of the Minsky machine. Details are discussed in Section 3.3.

Another possible generalization of PA syntax is to add a finite-state control unit to PA processes. This class of processes is formally introduced in Section 3.3 where we prove that an arbitrary Minsky machine can be simulated by a PA process with finite-state control unit (even by a normed one). Regularity is thus undecidable again.

An obvious question we have not addressed so far is whether regularity is decidable for *all* (not necessarily normed) PA processes. This problem is open at the time of writing this thesis—however, P. Jančar recently observed that this problem is at least semi-decidable. Further information can be found in Section 3.4.

3.2 Regularity w.r.t. Other Equivalences

Bisimilarity is not the only behavioural equivalence which appeared in the literature. In Section 2.2 we presented van Glabbeek's hierarchy of behavioural equivalences, whose definitions can be found in Appendix A. The notion of regularity can be defined w.r.t. those equivalences in the same way as in case of bisimilarity (see Definition 2.12). However, there is a notable difference: if we have bisimilar transition systems T_1, T_2 such that T_2 has finitely many states, then for each reachable state p of T_1 there

is a reachable state q of T_2 such that $p \sim q$. In other words, T_2 gives complete characterization of *all* reachable states of T_1. This is no more tru for the other equivalences; if we take e.g., trace equivalence and two tran sition systems T_1, T_2 such that $T_1 =_{tr} T_2$ and T_2 has finitely many states then the only thing we can say about T_1 and T_2 is that their *roots* have th same sets of traces—but if we take a reachable state p of T_1 (which is no the root of T_1), it need not be trace equivalent to any reachable state of T If we want to check some temporal property (e.g., something bad neve happens) of T_1, then we are usually interested in *all* reachable states of T it is thus sensible to ask whether there is a finite transition system T_3 suc that *each* reachable state of T_1 is equivalent to some state of T_3. If so, w can examine features of T_3 instead of T_1 and as T_3 is finite, it should b easier. This is the basic idea which leads to the notions of finite character zation and strong regularity. In this section we present some basic result which describe the relationship between regularity and strong regularit and between finite representations and finite characterizations.

As we want to keep this section general, we abstract from the concret model of process algebras and we define all notions in terms of transitio systems (we adopt the definition of transition system from Section 2.1 The class of all transition systems is denoted by $\mathcal{T}$.

Remark 3.17. *Each state p of a transition system $T = (S, Act, \to, r)$ determine a unique transition system $T(p) = (S, Act, \to, p)$. All notions originally define for transition systems can be used for their states in this sense too.*

Definition 3.18 (finite representation). *Let T be a transition system and l $\leftrightarrow$ be an equivalence over $\mathcal{T}$. A finite-state transition system T' is said to be* finite representation *of T w.r.t. $\leftrightarrow$ if $T \leftrightarrow T'$.*

A finite representation T' of T represents the behaviour of the proces which is associated with the root of T. As we shall see, representation generally do not say much about behaviours associated with reachabl states of T. We need another notion:

Definition 3.19 (finite characterization). *Let T be a transition system and let $\leftrightarrow$ be an equivalence over $\mathcal{T}$. A finite-state transition system T' is* a finite characterization *of T w.r.t. $\leftrightarrow$ if the following conditions hold:*

- $T \leftrightarrow T'$
- *States of T' are pairwise nonequivalent w.r.t. $\leftrightarrow$.*
- *For each reachable state p of T there is a reachable state q of T' with $p \leftrightarrow q$.*

A finite characterization T' of T describes the whole system T—for each reachable state of T there is its finite characterization within T' (in the sense of Remark 3.17).

Now we examine the question when finite characterizations exist and what is their relationship with representations. First we need to introduce further notions:

Definition 3.20 (quotients). *Let $\leftrightarrow$ be an equivalence over $\mathcal{T}$. For each transition system $T = (S, Act, \rightarrow, r)$ we define the transition system $T/_{\leftrightarrow} = (S', Act, \rightarrow', r')$ in the following way:*

- *S' contains equivalence classes of $S/_{\leftrightarrow}$ (the equivalence class containing $p \in S$ is denoted by $[p]$).*
- *The relation $\rightarrow'$ is determined by the rule $p \xrightarrow{a} q \;\Rightarrow\; [p] \xrightarrow{a}' [q]$.*
- $r' = [r]$

The equivalence $\leftrightarrow$ is said to have quotients *if for any $T \in \mathcal{T}$ the natural projection $p : T \longrightarrow T/_{\leftrightarrow}$, assigning to each state q of T the state $[q]$ of $T/_{\leftrightarrow}$, is a part of $\leftrightarrow$ (i.e., $q \leftrightarrow [q]$ for each state q of T in the sense of Remark 3.17).*

The notion of finite characterization is naturally motivated. Now we can ask what features of a transition system T guarantee an existence of a finite characterization of T. This is the aim of the following definition:

Definition 3.21 (strong regularity). *Let $\leftrightarrow$ be an equivalence over $\mathcal{T}$. A tran sition system T is* strongly regular *w.r.t. $\leftrightarrow$ if T can reach only finitely man states up to $\leftrightarrow$.*

The next lemma says when the condition of strong regularity guarantee an existence of a finite characterization.

Lemma 3.22. *Let $\leftrightarrow$ be an equivalence over $\mathcal{T}$ which has quotients. Then T h a finite characterization w.r.t. $\leftrightarrow$ iff T is strongly regular w.r.t. $\leftrightarrow$.*

Proof:
"$\Rightarrow$" Obvious.
"$\Leftarrow$" As T is strongly regular w.r.t. $\leftrightarrow$ and $\leftrightarrow$ has quotients, the transitio system $T/_{\leftrightarrow}$ is a finite characterization of T. □

Now we prove that the requirement of "having quotients" from the prev ous lemma is not too restrictive in fact. There are many reasonable equiv alences which satisfy this condition.

Lemma 3.23. *The equivalences $=_{tr}, =_{ct}, =_{f}, =_{r}, =_{ft}, =_{rt}, =_{pf}$ have quotients.*

Proof: We will not give a separate proof for each of those equivalence because the main idea is always the same. The crucial thing is to rea ize that equivalent states always have the same sets of initial actions (se Appendix A). Here we present a full proof for failure equivalence.

Let $T = (S, Act, \rightarrow, r)$ be a transition system and let $p \in S$ be a state o T. We show that $F(p) = F([p])$ where $[p]$ denotes the equivalence class o $S/_{=_f}$ containing the state p:

"$\subseteq$": Let $(w, \Phi) \in Act^* \times \mathcal{P}(Act)$ be a failure pair of p (see Appendix A). B definition, there is a state $p' \in S$ such that $p \xrightarrow{w} p'$ and $I(p') \cap \Phi = \emptyset$. Bu then also $[p] \xrightarrow{w} [p']$. The set $I([p'])$ is the union of all $I(q)$ such that $q \in [p'$ As $u =_f v$ implies $I(u) = I(v)$, we can conclude that $I([p']) = I(p')$, henc $I([p']) \cap \Phi = \emptyset$, thus $(w, \Phi) \in F([p])$.

"$\supseteq$": Let $(w, \Phi) \in Act^* \times \mathcal{P}(Act)$ be a failure pair of $[p]$ and let $w = a_k \ldots a_1$. By definition, there is a sequence of transitions $[p_k] \xrightarrow{a_k} [p_{k-1}] \xrightarrow{a_{k-1}} \ldots \xrightarrow{a_1} [p_0]$ in $T/_{=_f}$ such that $p \in [p_k]$ and $I([p_0]) \cap \Phi = \emptyset$. We show that for each state q of T such that $q \in [p_i]$, where $i \in \{0, \ldots, k\}$, the pair $(a_i \ldots a_1, \Phi)$ belongs to $F(q)$. We proceed by induction on i:

- $\mathbf{i = 0}$: as $I(q) = I([p_0])$, we have $(\epsilon, \Phi) \in F(q)$.
- **induction step:** as $[p_i] \xrightarrow{a_i} [p_{i-1}]$, there are states u, v of T such that $u \xrightarrow{a_i} v$, $u \in [p_i]$ and $v \in [p_{i-1}]$. By the inductive hypothesis we have $(a_{i-1} \ldots a_1, \Phi) \in F(v)$, hence $(a_i \ldots a_1, \Phi) \in F(u)$. As $q =_f u$, the pair $(a_i \ldots a_1, \Phi)$ belongs to $F(q)$. $\square$

Lemma 3.24. *Simulation equivalence, ready simulation equivalence and 2-nested simulation equivalence have quotients.*

Proof: Let $T = (S, Act, \rightarrow, r)$ be a transition system and let $p \in S$ be a state of T. First we show that $p =_s [p]$ where $[p]$ denotes the equivalence class of $S/_{=_s}$ containing the node p. By definition, we must show an existence of two simulations P, R such that $(p, [p]) \in P$ and $([p], p) \in R$. The simulation P is exactly the natural projection $p : T \rightarrow T/_{=_s}$:

$$P = \{(q, [q]) \mid q \in S\}$$

It is easy to check that P is indeed a simulation. The way how R is defined is more complicated:

$$([u], v) \in R \textit{ iff there exists a } \text{derivation scheme } \textit{for } ([u], v).$$

A derivation scheme for $([u], v)$ of depth $k \geq 0$ consists of:

- a path $[m_0] \xrightarrow{a_1} [m_1] \xrightarrow{a_2} \ldots \xrightarrow{a_k} [m_k]$ in $T/_{=_s}$,
- a set of nodes $\{q_{i,j} \mid 0 \leq i \leq k,\ i \leq j \leq k\} \subseteq S$
- a set of states $\{r_0, \ldots, r_{k-1}\} \subseteq S$, *if* $k > 0$

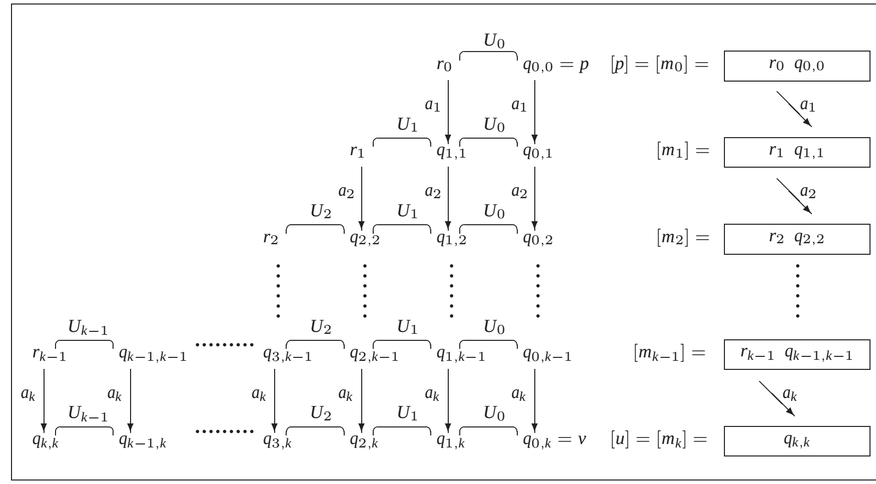


Figure 3.2: The structure of a derivation schema for $([u], v)$.

- a set of simulations $\{U_0, \ldots, U_{k-1}\}$, *if* $k > 0$

such that:

- $p = q_{0,0}$, $u \in [m_k]$, $v = q_{0,k}$
- $r_i \in [m_i]$ for $0 \leq i < k$, $q_{i,i} \in [m_i]$ for $0 \leq i \leq k$
- $r_i \xrightarrow{a_{i+1}} q_{i+1,i+1}$ for $0 \leq i < k$
- $q_{i,j} \xrightarrow{a_{j+1}} q_{i,j+1}$ for $0 \leq j < k$, $0 \leq i \leq j$
- $(r_i, q_{i,i}) \in U_i$ for $0 \leq i < k$
- $(q_{i+1,j}, q_{i,j}) \in U_i$ for $0 \leq i < k$, $i < j \leq k$

The structure of a derivation scheme for $([u], v)$ is shown in Figure 3.2.

The relation R is a simulation—whenever $([u], v) \in R$ and $[u] \xrightarrow{a} [u'$
then there is a state v' such that $v \xrightarrow{a} v'$ and $([u'], v') \in R$. This is due to a
existence of a derivation scheme for the pair $([u], v)$. We can simply add

new "layer" to the scheme and construct a derivation scheme for the pair $([u'], v')$. The way how it is done is obvious. Moreover, R contains the pair $([p], p)$ because this pair has a derivation scheme of depth 0.

This construction can be also used for ready simulation equivalence. The simulation P becomes a ready simulation. It follows directly from the fact that two states which are ready simulation equivalent have the same sets of initial actions. The notion of derivation scheme has to be modified slightly—we now require that $\{U_0, \ldots, U_{k-1}\}$ is a set of ready simulations. Then R is also a ready simulation: assume that $([u], v) \in R$. Then $I([u]) = I(v)$ because $q_{k,k} \in [u]$ and $U_0, \ldots, U_{k-1}$ are ready simulations now.

In case of 2-nested simulation equivalence the construction can be used too. The simulation P becomes a 2-nested simulation because we can easily prove that $p =_s [p]$ for each state p of T. The notion of derivation scheme has to be modified again—$\{U_0, \ldots, U_{k-1}\}$ must be a set of 2-nested simulations now. We prove that R is a 2-nested simulation. Let $([u], v) \in R$. We need to show that $[u] =_s v$. By definition, two simulations Q, V such that $([u], v) \in Q$ and $(v, [u]) \in V$ have to be constructed. Clearly R is a simulation which contains the pair $([u], v)$, hence we can choose $Q = R$. The construction of V is slightly more complicated. As $([u], v) \in R$, there is a derivation scheme for $([u], v)$. $U_0, \ldots, U_{k-1}$ are 2-nested simulations, hence $q_{k,k} =_s v$. Therefore there is a simulation K containing the pair $(v, q_{k,k})$. It is easy to check that $V = \{(e, [f]) \mid (e, f) \in K\}$ is a simulation. Moreover, $(v, [u]) \in V$ because $q_{k,k} \in [u]$. □

We have just proved the following theorem:

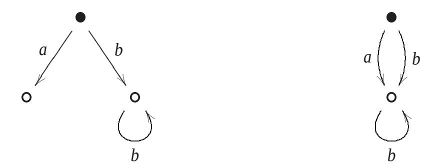
Theorem 3.25. *Each equivalence in van Glabbeek's hierarchy has quotients.*

There are also other well-known equivalences which have quotients, e.g., weak bisimilarity (see [Mil89]) or branching bisimilarity (see [vGW89]). But this property is naturally not general—there are also equivalences

which do not have quotients. A simple example is *language equivalenc* (denoted by '$=_L$'). Two transition systems are language equivalent if the roots have the same sets of completed traces (realize that language equiv alence is different from completed trace equivalence and it is even incom parable with trace equivalence—see Appendix A). As a counterexampl we can choose e.g., the transition system $T = (S, Act, \rightarrow, r)$ where

$$\begin{array}{rcl} S & = & \{r, p, q\} \\ Act & = & \{a, b\} \\ \rightarrow & = & \{(r, a, p),\ (r, b, q),\ (q, b, q)\} \end{array}$$

Transition systems T and $T/_{=_L}$ look as follows:



Clearly $r \neq_L [r]$ because $ct(r) = \{a\}$ and $ct([r]) = \emptyset$.

We have seen that if we restrict our attention to behavioural equiva lences which have quotients, then the condition of strong regularity be comes necessary and sufficient for an existence of a finite characterizatio An interesting question is, what is the exact relationship between cond tions of regularity and strong regularity. First, we already know that the are equivalences for which these two conditions coincide (e.g., bisimila ity). The following notion aims to cover further examples of such equiva lences:

Definition 3.26. *An equivalence $\leftrightarrow$ over $\mathcal{T}$ is* safe *if whenever $T \leftrightarrow T'$ then fo each reachable state p of T there is a reachable state p' of T' such that $p \leftrightarrow p'$.*

Lemma 3.27. *Let $\leftrightarrow$ be a safe equivalence over $\mathcal{T}$ which has quotients. Then is strongly regular w.r.t. $\leftrightarrow$ iff T is regular w.r.t. $\leftrightarrow$.*

Proof:
"$\Rightarrow$" Obvious.
"$\Leftarrow$" We prove that the transition system $T/_{\leftrightarrow}$ is a finite characterization of T. As $\leftrightarrow$ has quotients, $T \leftrightarrow T/_{\leftrightarrow}$. As $\leftrightarrow$ is safe, for each reachable state p of T there is a reachable state $[q]$ of $T/_{\leftrightarrow}$ such that $p \leftrightarrow [q]$. Moreover, states of $T/_{\leftrightarrow}$ are pairwise nonequivalent. □

In other words, if $\leftrightarrow$ is a safe equivalence over $\mathcal{T}$ which has quotients then each transition system T has a finite representation iff T has a finite characterization. We have already mentioned some examples—bisimilarity, weak bisimilarity and branching bisimilarity are safe and have quotients. But there are also equivalences for which conditions of regularity and strong regularity are really different.

Lemma 3.28. *For each behavioural equivalence $\leftrightarrow$ which lies under ready simulation equivalence in van Glabbeek's hierarchy (including this relation) there is a transition system T such that T is regular w.r.t. $\leftrightarrow$ and T is not strongly regular w.r.t. $\leftrightarrow$.*

Proof: Let $T_1 = (S_1, Act_1, \rightarrow_1, r_1)$, $T_2 = (S_2, Act_2, \rightarrow_2, r_2)$ be transition systems where:

$$\begin{aligned}
S_1 &= \bigcup_{i=0}^{\infty} \{(i,j) \mid i,j \in N \cup \{0\},\ 0 \le j \le i+1\} \\
Act_1 &= \{a\} \\
\rightarrow_1 &= \bigcup_{i=0}^{\infty} \{\ ((i,j), a, (i,j+1)) \mid 0 \le j \le i\} \ \cup\ \{((0,0), a, (0,0))\} \\
&\quad \cup \{\ ((i,0), a, (i+1,0)) \mid i \in N \cup \{0\}\} \\
r_1 &= (0,0)
\end{aligned}$$

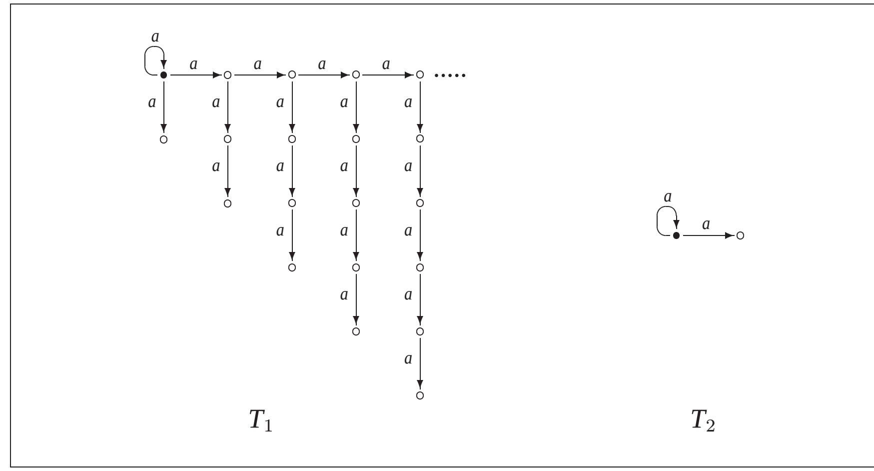


Figure 3.3: Transition systems from the proof of Lemma 3.28

$$\begin{aligned}
S_2 &= \{A, B\} \\
Act_2 &= \{a\} \\
\rightarrow_2 &= \{(A,a,A), (A,a,B)\} \\
r_2 &= A
\end{aligned}$$

If we draw these transition systems, we obtain pictures of Figure 3.3.

The transition system T_1 is not strongly regular w.r.t. trace equivalenc because $tr((i,1)) \subsetneq tr((i+1,1))$ for each $i \in N \cup \{0\}$, thus T_1 contain infinitely many states up to trace equivalence. Therefore T_1 is not strongl regular w.r.t. any equivalence in van Glabbeek's hierarchy.

Now we show that $T_1 =_{rs} T_2$. By definition, two ready simulations R, S such that $(r_1, r_2) \in R$ and $(r_2, r_1) \in S$ have to be constructed:

$$\begin{aligned}
R &= \bigcup_{i=0}^{\infty}\{\ ((i,j), A):\ 0 \le j \le i\} \ \cup\ \bigcup_{i=0}^{\infty}\{\ ((i,i+1), B)\ \} \\
S &= \{\ (A, (0,0)),\ (B, (0,1))\ \}
\end{aligned}$$

It is easy to check that R, S are ready simulations. Moreover, $((0,0), A) \in$

and $(A,(0,0)) \in S$.

As $T_1 =_{rs} T_2$, transition systems T_1, T_2 are equivalent w.r.t. any behavioural equivalence which lies under ready simulation equivalence in van Glabbeek's hierarchy. As T_2 is finite, the system T_1 is regular w.r.t. each of these equivalences. $\square$

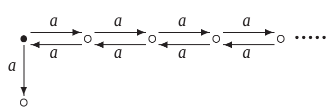
Lemma 3.29. *There is a transition system T such that T is regular w.r.t. possible-futures equivalence and 2-nested simulation equivalence, but T is not strongly regular w.r.t. these equivalences.*

Proof: Let $T_1 = (S_1, Act_1, \rightarrow_1, r_1)$, $T_2 = (S_2, Act_2, \rightarrow_2, r_2)$ be transition systems where:

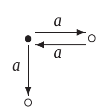
$$\begin{aligned} S_1 &= N \cup \{0\} \\ Act_1 &= \{a\} \\ \rightarrow_1 &= \{ (i, a, i+1) \mid i \in N\} \cup \{ (i, a, i-1) \mid i \in N\} \\ r_1 &= 1 \end{aligned}$$

$$\begin{aligned} S_2 &= \{A, B, C\} \\ Act_2 &= \{a\} \\ \rightarrow_2 &= \{ (A, a, B), (A, a, C), (C, a, A) \} \\ r_2 &= A \end{aligned}$$

Systems T_1, T_2 can be depicted as follows:



T_1



T_2

We show that T_1 has infinitely many states w.r.t. $=_{pf}$ and $=_2$. Let $i, j \in N$, $i < j$, be states of T_1. The state i has a possible future $(a^i, \emptyset)$. Clearly

$(a^i, \emptyset) \notin PF(j)$, hence $i \neq_{pf} j$. As 2-nested simulation equivalence is abov possible-futures equivalence in van Glabbeek's hierarchy, the system T has infinitely many states also w.r.t. $=_2$, thus T_1 is not strongly regul w.r.t. $=_2$ and $=_{pf}$.

It remains to prove that T_1 is regular w.r.t. $=_2$ and $=_{pf}$. We show tha $T_1 =_2 T_2$. First we have to realize which states of T_1 and T_2 are simulatio equivalent. Clearly $0 =_s B$. If $i \in N$ is odd then $i =_s A$ and if $i \in N$ is eve then $i =_s C$. Following relations are the required simulations:

$$\begin{aligned} R_i &= \{ (k, A) \mid k \in N \wedge k \textit{ is odd}\} \cup \{ (k, C) \mid k \in N \cup \{0\} \wedge k \textit{ is even}\} \\ S_i &= \{ (A, i), (B, i+1), (C, i+1) \} \end{aligned}$$

Now we can define two 2-nested simulations which relate roots of T_1 an T_2:

$$\begin{aligned} R &= \{ (i, A) \mid i \in N \wedge i \textit{ is odd}\} \cup \{ (i, C) \mid k \in N \wedge i \textit{ is even}\} \cup \{(0, B)\} \\ S &= \{ (A, 1), (B, 0), (C, 2) \} \end{aligned}$$

Elements of R, S are pairs of simulation equivalent states. Now it is eas to check that R, S are 2-nested simulations. As $(1, A) \in R$ and $(A, 1) \in S$ transition systems T_1, T_2 are 2-nested simulation equivalent.

As $T_1 =_2 T_2$ and possible-futures equivalence lies under 2-nested simu lation equivalence in van Glabbeek's hierarchy, systems T_1 and T_2 are als possible-futures equivalent. Thus T_1 is regular w.r.t. $=_2$ and $=_{pf}$. $\square$

We have just proved the following theorem:

Theorem 3.30. *Let $\leftrightarrow$ be an equivalence in van Glabbeek's hierarchy which li under bisimilarity. Then there is $T \in \mathcal{T}$ such that T is regular w.r.t. $\leftrightarrow$ and T not strongly regular w.r.t. $\leftrightarrow$.*

An open problem is whether the notions of regularity and strong regu larity have different decidability features. In the next section we prese some negative results, stating that both regularity and strong regularity can be undecidable in certain process algebras. From the practical point of view it would be much more interesting to obtain some positive results, but this area seems to be quite unexplored.

3.3 Negative Results

In this section we present some negative results, stating that regularity and strong regularity w.r.t. all equivalences of van Glabbeek's hierarchy are undecidable in the class of processes which is obtained from PA by adding a finite-state control unit. As we shall see, those problems are undecidable even for normed processes of that class. Our results are proved in a uniform way by a simple reduction of the halting problem of the Minsky machine. This technique can also be applied to other process algebras which are powerful enough to simulate an arbitrary Minsky machine.

3.3.1 The Minsky Machine

The Minsky machine (denoted here by $\mathcal{M}$) is equipped with two counters C_1, C_2 which can store nonnegative integers. The behaviour of $\mathcal{M}$ is determined by a finite-state program, composed of $m \in N$ labelled statements:

$$
\begin{array}{lcl}
l_1 & : & s_1 \\
l_2 & : & s_2 \\
\vdots & & \\
l_{m-1} & : & s_{m-1} \\
l_m & : & \mathtt{HALT}
\end{array}
$$

where for each i, $1 \leq i < m$ the statement s_i has one of the two forms:

$$
s_i = \begin{cases} C_j = C_j + 1;\ \mathtt{goto}\ l_k \\ \mathtt{if}\ C_j = 0\ \mathtt{then}\ \ \mathtt{goto}\ l_k\ \mathtt{else}\ C_j = C_j - 1;\ \mathtt{goto}\ l_n; \end{cases}
$$

where $j \in \{1, 2\}$. The machine $\mathcal{M}$ starts its execution (with given inpu values on C_1, C_2) from the command l_1. $\mathcal{M}$ *halts* if it reaches the comman 'HALT' in a finite number of steps, and *diverges* otherwise. Undecidabilit of the halting problem of the Minsky machine has been demostrated b Minsky in [Min67].

3.3.2 Extending PA Processes with a Finite-state Contro Unit

In this section we explore a calculus obtained by extending PA processe with a finite-state control unit. First we explain what happens if we ad a finite-state control unit to BPA and BPP processes, because these mode have been already studied by other researchers.

Any BPA process Δ in GNF can be viewed as a push-down proces (PDA; see e.g., [MS85]) whose control unit has just one state—we ca imagine that reachable states of Δ are stored on a stack with the lef most variable on the top. A well-known fact from the theory of forma languages and automata says that if we are interested in *language equiva lence*, then the expressive power of context-free grammars and push-dow automata coincide. As BPA processes can be seen as context-free gram mars in GNF, we can ask the same question for bisimilarity. The answer i surprising—there are PDA processes for which there are no bisimilar BP processes (this was demonstrated in [CM90]). In other words, if we ad a finite-state control unit to BPA, we get a strictly more expressive calcu lus. Stirling has recently shown in [Sti96] that bisimilarity is decidable fo normed PDA processes (he defines a PDA process to be normed if it ca always empty its stack). It is easy to see that regularity w.r.t. bisimilarity i also decidable for normed PDA processes—such a process is not regular i there is no bound on the length (or height) of the stack, and this is clearl decidable. Further decidability issues for PDA processes are discussed i

Section 3.4.

The idea of adding a finite-state control unit is quite general—the unit can be seen as a finite-state context for process variables which can behave differently under different contexts and which can change the context by emitting an action. If we extend BPP processes in this way, we obtain so-called parallel push-down processes (PPDA). The only difference between PDA and PPDA processes is that the "stack" of PPDA has random access capability (remember that reachable states of BPP processes are multisets of variables which are stored on the "stack" now). Moller demonstrated in [Mol96] that the expressive power of PPDA in strictly greater then the one of BPP. Decidability properties of PPDA were examined by Hirshfeld; he noticed that PPDA processes form a subclass of Petri nets and hence all positive decidability results on Petri nets also apply to PPDA (see Section 3.4). However, some negative results remain valid too—the most significant example is undecidability of bisimilarity for PPDA[1] (see [Mol96]).

Now we can ask what happens if we add a finite-state control unit to PA processes (we denote the resulting calculus PAPDA for short). We show that PAPDA processes are strictly more expressive than PA, PDA and PPDA. The reason is quite simple—PAPDA is a calculus with full Turing power. We show that an arbitrary Minsky machine can be simulated by an effectively constructible PAPDA process (even by a normed one). This fact brings other negative results on PAPDA processes, e.g., undecidability of regularity and strong regularity w.r.t. any equivalence of van Glabbeek's hierarchy.

Definition 3.31 (PAPDA processes). *A PAPDA process is formally defined as a tuple* (Q, V, Λ, P, R) *where*

- Q *is a finite set of* states.

[1] This result is due to Hirshfeld; it is obtained by utilizing Jančar's technique for showing undecidability of bisimilarity for labelled Petri nets [Jan94].

- V *is a finite set of* variables.
- Λ *is a finite set of* actions.
- $P \subseteq (Q \times V) \times \Lambda \times (Q \times VPA(V))$ *is a finite* transition relation *(*$VPA(V$ *denotes the set of all VPA expressions over V—see Section 2.3.2).*
- $R \in Q \times VPA(V)$ *is a distinguished pair called* root.

As usual, we will write $p\alpha$ instead of (p, α) where $(p, \alpha) \in Q \times VPA(V)$, an $pX \xrightarrow{a} q\alpha$ instead of $((p, X), a, (q, \alpha)) \in P$. Furthermore, we will identify a *VPA* expressions which are structurally congruent (see Definition 2.3).

To be able to extend the transition relation P to elements of $Q \times VPA(V$ we first need to introduce a predicate $Active(X, i, \alpha)$ which is true iff the i occurrence of the variable X within a *VPA* expression α (reading α fro left to right) can emit an action.

Definition 3.32 (*Active* predicate). *The predicate Active is defined inductivel on the structure of* α*:*

- $\alpha = Y$. *Then* $Active(X, i, \alpha)$ *is True if* $Y = X$ *and* $i = 1$, *and False other wise.*
- $\alpha = \beta.\gamma$ *or* $\alpha = \beta \lfloor\!\lfloor \gamma$. *Then* $Active(X, i, \alpha) = Active(X, i, \beta)$.
- $\alpha = \beta \| \gamma$. *Then* $Active(X, i, \alpha) = Active(X, i, \beta) \vee Active(X, i-k, \gamma)$, *wher* k *denotes the number of occurrences of* X *in* β.

The transition relation P is extended to elements of $Q \times VPA(V)$ in the fo lowing way: $p\alpha \xrightarrow{a} q\beta$ iff there is a transition $pX \xrightarrow{a} q\gamma$ in P and $i \in N$ suc that $Active(X, i, \alpha)$ is True, and β can be obtained from α by substitutin the i^{th} occurrence of X with γ. The way how PAPDA processes determin their associated transition systems is now obvious.

Now we show that an arbitrary Minsky machine $\mathcal{M}$ whose progra consists of m statements can be simulated by a PAPDA process which ca be effectively constructed. For simplicity, assume that $\mathcal{M}$ starts its execution with both counters initialized to 0 (we can afford this because the halting problem is clearly undecidable also for this subclass of Minsky machines). The simulating PAPDA process $\psi = (Q, V, \Lambda, P, R)$ looks as follows:

- $Q = \{q_1, \ldots, q_m\}$
- $V = \{I_1, I_2, Z_1, Z_2\}$
- $\Lambda = \{a\}$
- $R = q_1(Z_1 \| Z_2)$

The transition relation P is constructed using the following rules:

1. If the program of $\mathcal{M}$ contains an instruction of the form
   $l_i : \ C_j = C_j + 1; \ \texttt{goto}\ l_k$
   then P contains the elements $q_i Z_j \xrightarrow{a} q_k(I_j.Z_j)$ and $q_i I_j \xrightarrow{a} q_k(I_j.I_j)$.
2. If the program of $\mathcal{M}$ contains an instruction of the form
   $l_i : \ \texttt{if}\ C_j = 0\ \texttt{then goto}\ l_k\ \texttt{else}\ C_j = C_j - 1; \texttt{goto}\ l_n$
   then P contains the elements $q_i Z_j \xrightarrow{a} q_k Z_j$ and $q_i I_j \xrightarrow{a} q_n$.
3. Each element of P can be derived using the rule 1 or 2.

Intuitively, counters of $\mathcal{M}$ are simulated by two BPA processes which are combined in parallel on the "stack" and the program of $\mathcal{M}$ is simulated by the finite-state control unit of ψ . Each step of $\mathcal{M}$ is mimicked by ψ which emits the action a. Let Y be a process defined by $Y \stackrel{\text{def}}{=} aY$. If the machine $\mathcal{M}$ diverges then $\psi \sim Y$, hence $\psi \leftrightarrow Y$ for any equivalence $\leftrightarrow$ of van Glabbeek's hierarchy. If the machine $\mathcal{M}$ halts then $\psi \neq_{tr} Y$, because ψ emits the action a only finitely many times (note that $\mathcal{M}$ is deterministic). Hence $\psi \not\leftrightarrow Y$ for any equivalence $\leftrightarrow$ of van Glabbeek's hierarchy. This reduction proves the following theorem:

Theorem 3.33. *Let ψ be a PAPDA process, let Δ be a finite-state process an let $\leftrightarrow$ be an equivalence of van Glabbeek's hierarchy. It is undecidable whethe $\psi \leftrightarrow \Delta$.*

It is worth noting that $\mathcal{M}$ can be simulated even by a *normed*[2] PAPD process ψ' which can be obtained from ψ just by adding a special state t t Q and the following set of transitions to P:

$$\{q_i U \xrightarrow{a} t \mid U \in V, \ 1 \leq i < m\}$$

The only difference between ψ and ψ' is that ψ' can terminate in one ste at any moment (due to the deadlock in the state t). If $\mathcal{M}$ diverges, then has an infinite run—it is thus bisimilar to $Y' \stackrel{\text{def}}{=} aY' + a$. If $\mathcal{M}$ halts, then is not trace equivalent to Y'. Theorem 3.33 is thus valid also for norme PAPDA processes.

Theorem 3.34. *Let φ be a PAPDA process and let $\leftrightarrow$ be an equivalence of va Glabbeek's hierarchy. It is undecidable whether φ is (strongly) regular w.r.t. $\leftrightarrow$.*

Proof: We show that the halting problem of the Minsky machine can b reduced to both mentioned problems. Let $\mathcal{M}$ be an arbitrary Minsky ma chine and let ψ be the PAPDA process which simulates the execution o $\mathcal{M}$. Now we modify the process ψ slightly, producing a new PAPDA pro cess ϱ: we add a new state s to Q, two new variables B, C to V and th following set of transitions to P:

$$\{q_m U \xrightarrow{b} sB \mid U \in V\} \cup \{sB \xrightarrow{b} sBC, \ sB \xrightarrow{c} s, \ sC \xrightarrow{c} s\}$$

If $\mathcal{M}$ diverges, then $\varrho \sim \psi \sim Y$ where $Y \stackrel{\text{def}}{=} aY$, hence ϱ is (strongly) regula w.r.t. $\leftrightarrow$. Now we show that if $\mathcal{M}$ halts, then ϱ is not (strongly) regula w.r.t. $\leftrightarrow$.

[2] A PAPDA process is normed if its corresponding transition system has the featu that from any state it is possible to reach a state which does not have any successors.

As $\mathcal{M}$ halts, ϱ is normed because $\varrho \stackrel{a^k}{\to} q_m\alpha$ for some $k \in N \cup \{0\}$, $\alpha \in VPA(V)$ and $q_m\alpha$ is normed (realize that the first k steps of ϱ are completely deterministic). Traces of ϱ are thus exactly prefixes of completed traces of ϱ which look as follows:

$$ct(\varrho) = \{a^k b^i c^i \mid i \in N\}$$

Assume that ϱ is trace equivalent to some finite-state process E with n states. Then E has a trace $a^k b^n c^n$. As E has only n states, it had to pass through the same state twice before emitting the first c; there are three possibilities:

1. $E \stackrel{a^p}{\to} F \stackrel{a^q}{\to} F \stackrel{a^r b^n c^n}{\to} G$ where $p + q + r = k$, $q \geq 1$. But then also $a^{p+r} b^n c^n$ is a trace of E and as this sequence of actions is not a prefix of any element of $ct(\varrho)$, $\varrho \neq_{tr} E$ and we have a contradiction.

2. $E \stackrel{a^p}{\to} F \stackrel{a^q b^r}{\to} F \stackrel{b^s c^n}{\to} G$ where $p + q = k$, $r + s = n$, $r \geq 1$. But then $a^p b^s c^n$ is a trace of E which is not a trace of ϱ (because $s < n$).

3. $E \stackrel{a^k b^p}{\to} F \stackrel{b^q}{\to} F \stackrel{b^r c^n}{\to}$ where $p + q + r = n$, $q \geq 1$. Then $a^k b^{p+r} c^n$ is a trace of E which is not a trace of ϱ.

We just proved that if $\mathcal{M}$ halts, then ϱ is not regular w.r.t. trace equivalence. Hence ϱ is not regular w.r.t. $\leftrightarrow$. As $\leftrightarrow$ has quotients, strong regularity w.r.t. $\leftrightarrow$ implies regularity w.r.t. $\leftrightarrow$. Thus non-regularity of ϱ w.r.t. $\leftrightarrow$ implies that ϱ is not strongly regular w.r.t. $\leftrightarrow$. □

The previous theorem is valid also for normed PAPDA processes—we can use the same proof, replacing ψ with ψ'.

This technique also works for other process algebras which are sufficiently expressive to simulate any Minsky machine. We can mention e.g., BPP processes where the merge operator '$\|$' is replaced with the '$\|_A$' parallel operator of CSP (see [Hoa85]). This operator has the following seman-
tics (A is a set of actions):

$$\frac{E \stackrel{b}{\to} E'}{E\|_A F \stackrel{b}{\to} E'\|_A F}\ (b \notin A) \qquad \frac{F \stackrel{b}{\to} F'}{E\|_A F \stackrel{b}{\to} E\|_A F'}\ (b \notin A) \qquad \frac{E \stackrel{a}{\to} E' \quad F \stackrel{a}{\to} F'}{E\|_A F \stackrel{a}{\to} E'\|_A F'}\ (a \in A$$

The '$\|_A$' operator forces synchronizations on actions from the set A. Taub
ner proved in [Tau89] that using this operator it is possible to simulat
counters (and consequently an arbitrary Minsky machine—it suffices t
combine two counters in parallel with a finite-state process which simu
lates the control unit. The three components can be forced to cooperate).

Another example is BPP_τ algebra enhanced with the restriction opera
tor '$\backslash L$' (see [Mil89]) which can force synchronizations on complementar
actions (L is a set of actions such that $\tau \notin L$):

$$\frac{E \stackrel{a}{\to} E'}{E\backslash L \stackrel{a}{\to} E'\backslash L}\ (a, \overline{a} \notin L)$$

BPP_τ processes can simulate an arbitrary Minsky machine in a similar wa
as the previously mentioned ones. The crucial thing is the description o
counters which is due to Taubner [Tau89] again.

3.4 Related Work and Future Research

In this section we present further results which are related to the subject o
this chapter. Here we discuss the work of *other* researchers and therefor
we will not give full proofs of all theorems. Nevertheless, sometimes w
describe the basic idea of the proof or comment the technique briefly, be
cause it well illustrates the variety of possible approaches to the problem

The question whether for a given infinite-state behaviour there is a
equivalent finite-state one has been known from the theory of formal lan
guages for a long time. However, the problem is not too interesting in th

setting, because it becomes undecidable even for context-free grammars—it is folklore that the problem whether a given context-free grammar G generates regular language is undecidable.

The question was later "rediscovered" within the framework of concurrency theory (after new, well-motivated equivalences appeared). Taubner proved in his Ph.D. thesis (also published as [Tau89]) that regularity w.r.t. bisimilarity and trace equivalence is undecidable for certain process algebras, namely for CCS and TCSP. The crucial idea is that it is possible to simulate an arbitrary Minsky machine by an effectively constructible process of CCS and TCSP. Taubner also showed that mentioned algebras can simulate counters (and consequently an arbitrary Minsky machine) even without the use of renaming. Obtained sub-algebras correspond to BPP_τ enhanced with the restriction operator, and BPP where the merge operator '$\|$' is replaced with the parallel operator '$\|_A$' of CSP, respectively.

In Section 3.3.2 we have presented another process algebra with full Turing power—PAPDA. We have also extended Taubner's undecidability results to all equivalences of van Glabbeek's hierarchy using a simple reduction of the halting problem.

The first positive decidability result on regularity is due to Mauw and Mulder. They proved in [MM94] that "regularity" is decidable for BPA processes. The quotes are important here because Mauw and Mulder used the word regularity in a different sense—a BPA process Δ is "regular" if for *each* variable $Y \in Var(\Delta)$ there is a finite-state process Δ_Y such that $Y \sim \Delta_Y$. The notion of "regularity" is thus strongly dependent on BPA syntax. It is not clear how to define "regularity" for e.g., Petri nets. Nevertheless, this result is valuable because in case of *normed* BPA processes the notions of regularity and "regularity" coincide (as observed in [Kuč95]). Moreover, our proof of decidability of regularity for nPA processes (see Section 3.1) was inspired by the technique used in [MM94].

Bosscher and Griffionen later proved that regularity is actually decid-

able in a larger subclass of BPA processes (see [BG96]) which includes als some BPA processes which are not normed.

A definitive answer was given by Burkart, Caucal and Steffen [BCS96 They proved that regularity is decidable for *all* BPA processes. The tech nique is rather different from the previous ones—it is shown that for an BPA process Δ which generates a transition system T it is possible to con struct a deterministic graph grammar $\mathcal{G}$ which generates the transition sys tem $T/_\sim$. Hence Δ is non-regular iff $\mathcal{G}$ generates an infinite graph, and is easily decidable.

Jančar and Esparza presented in [JE96] another positive result statin that regularity is decidable for labelled Petri nets. This implies decidabilit of regularity for BPP and PPDA processes because any BPP or PPDA pro cess can also be seen as a (rather special) Petri net. The proof is obtaine by a combination of two semi-decidability results. Semi-decidability o the positive subcase follows from the fact that bisimilarity is decidabl for pairs of labelled Petri nets such that one net of this pair is bounded A labelled Petri net N is regular iff there is a bounded net R such tha $N \sim R$. But this condition is clearly semi-decidable because we can enu merate all bounded nets and check whether we already found R. Sem decidability of the negative subcase is obtained by showing that if a give Petri net is *not* regular, then there is a special marking which fulfills certai semi-decidable conditions. This marking plays the role of finite "witness of non-regularity, whose existence is again semi-decidable by exhaustiv search.

Decidability of regularity w.r.t. other equivalences of van Glabbeek hierarchy is discussed in [JM95]. Jančar and Moller proved that regula ity w.r.t. trace equivalence and simulation equivalence is undecidable fo

[3]A Petri net N is bounded if the total number of tokens which are stored within plac of N cannot exceed certain limit during the execution of N. Bounded Petri nets thu correspond to finite-state processes.

labelled Petri nets. At the same time they proved that mentioned equivalences are decidable for pairs of labelled Petri nets such that one net of this pair is bounded. From this we can conclude that the negative subcase of the regularity problem is even not semi-decidable for these equivalences.

An important open problem in the area of "regularity testing" is decidability of regularity w.r.t. bisimilarity for PDA and PA processes. A recent result [Jan97] due to Jančar says that bisimilarity and regularity w.r.t. bisimilarity are decidable for one-counter processes (i.e., PDA processes where the stack alphabet has just one symbol besides a special bottom symbol). Regularity is also easily decidable for normed PDA processes (if we adopt Stirling's definition of normedness as presented in [Sti96]). Regularity of general PDA processes is at least semi-decidable, because it is possible to check bisimilarity between a PDA process and a finite-state process. The same result holds for PA processes.[4] Our conjecture is that regularity is in fact decidable in both process classes.

[4]Those facts can be presented due to a private communication with Petr Jančar.

Chapter 4

Expressibility of nBPA$_\tau$ and nBPP$_\tau$ Processes

In this chapter we study the relationship between the classes of transition systems which are generated by normed BPA$_\tau$ and normed BPP$_\tau$ processes. We also examine such a relationship between their respective subclasses, namely normed BPA and normed BPP processes (see Section 2.3.1).

BPA processes can be seen as simple sequential programs (they are equipped with a binary sequential operator). This class of processes has been intensively studied by many researchers. Baeten, Bergstra and Klop proved in [BBK87] that bisimilarity is decidable for normed BPA processes. Much simpler proofs of this were later given in [Cau88, HS91, Gro91]. In [HS91] Hüttel and Stirling used a tableau decision method and gave also sound and complete equational theory. Hirshfeld, Jerrum and Moller demonstrated in [HJM94a] that the problem is decidable in polynomial time. The decidability result was later extended to the whole class of BPA processes by Christensen, Hüttel and Stirling in [CHS92].

If we replace the binary sequential operator with the parallel (merge) operator, we obtain BPP processes. They can thus be seen as simple paral-

lel programs. Christensen, Hirshfeld and Moller proved in [CHM93a] tha bisimilarity is decidable for BPP processes. A polynomial decision algo rithm for normed BPP processes was presented in [HJM94b] by Hirshfeld Jerrum and Moller.

If we allow a parallel operator not to specify just merge but also a internal communication between two BPP processes resulting in a specia action τ, we obtain the class of BPP$_\tau$ processes [Chr93]. In order to com pare this class with its sequential counterpart we employ the class of BPA processes [BK88]. Decidability and complexity results just mentioned hol for these classes as well.

This chapter is organized as follows. In Section 4.1 we give an exac characterization of those transition systems which can be equivalently (u to bisimilarity) described by the syntax of nBPA$_\tau$ and nBPP$_\tau$ processes Next we show that if we restrict ourselves to nBPA and nBPP processe we obtain a simpler (and hopefully nicer) characterization of those be haviours which are common to these subclasses. In Section 4.2 we demon strate that it is decidable whether for a given nBPA, nBPA$_\tau$, nBPP, or nBPP process Δ there is some nBPP, nBPP$_\tau$, nBPA, or nBPA$_\tau$ process Δ' such tha $\Delta \sim \Delta'$, respectively. These algorithms are polynomial. We also show tha if the answer to the previous question is positive, then the process Δ' ca be effectively constructed. Unfortunately, this construction is no longe polynomial. As an important consequence we also obtain decidabilit of bisimulation equivalence in the union of nBPA$_\tau$ and nBPP$_\tau$ processes We conclude with remarks on related work and future research. The re sults which are presented is this chapter have been previously publishe as [ČKK96].

Remark 4.1. *In this chapter we use previously established results on regularit of nBPA$_\tau$ nBPP$_\tau$, nBPA and nBPP processes (see Section 3.1.3). Here the wor "regularity" always means regularity w.r.t. bisimilarity.*

4.1 The Characterization of $\mathbf{nBPA}_\tau \cap \mathbf{nBPP}_\tau$

In this section we give an exact characterization of those normed processes which can be equivalently defined by BPA_τ and BPP_τ syntax.

Definition 4.2 ($\mathbf{nBPA}_\tau \cap \mathbf{nBPP}_\tau$). *The* semantical intersection *of $nBPA_\tau$ and $nBPP_\tau$ processes is defined as follows:*

$$\begin{aligned} nBPA_\tau \cap nBPP_\tau &= \{\Delta \in nBPA_\tau, \mid \exists \Delta' \in nBPP_\tau \text{ such that } \Delta \sim \Delta'\} \cup \\ & \quad \{\Delta \in nBPP_\tau, \mid \exists \Delta' \in nBPA_\tau \text{ such that } \Delta \sim \Delta'\} \end{aligned}$$

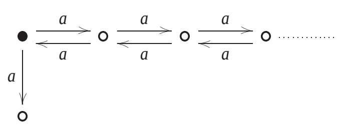
The class $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ is clearly nonempty because each normed finite-state process belongs to $\text{nBPA}_\tau \cap \text{nBPP}_\tau$. But $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ contains also processes with infinitely many states—consider the following process:

$$X \stackrel{def}{=} a(X|X) + a \tag{4.1}$$

X is a nBPP_τ process with infinitely many states. If we replace the '|' operator with the '.' operator, we obtain a bisimilar nBPA_τ process:

$$\overline{X} \stackrel{def}{=} a(\overline{X}.\overline{X}) + a \tag{4.2}$$

Clearly $X \sim \overline{X}$ because transition systems generated by those processes are even isomorphic:



Now we modify the process X slightly:

$$X \stackrel{def}{=} a(X|X) + a + \overline{a} \tag{4.3}$$

Although the process (4.3) does not differ from the process (4.1) too much, it is not hard to prove that there is *no* nBPA_τ process bisimilar to (4.3).

Now we prove that each nBPP_τ processes from $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ can b represented in a special normal form, denoted INF_{BPP} (**I**ntersection **N**orma **F**orm for nBPP_τ processes). Before the definition of INF_{BPP} we first intro duce the notion of *reduced* process:

Definition 4.3 (reduced process). *Let Δ be a $nBPA_\tau$ or $nBPP_\tau$ process. W say that Δ is* reduced *if its variables are pairwise non-bisimilar.*

As bisimilarity is decidable for nBPA_τ and nBPP_τ processes in polynomia time (see [HJM94a], [HJM94b]), each nBPA_τ or nBPP_τ process can be effec tively transformed into a bisimilar reduced process in polynomial time.

In the rest of this chapter we often use the notation α^i where α is a stat of a nBPA_τ or nBPP_τ process. It has the following meaning:

$$\begin{aligned} \alpha^i &= \underbrace{\alpha.\alpha.\cdots.\alpha}_{i} && \text{if } \alpha \text{ is a state of some nBPA}_\tau \text{ or nBPA process} \\ \alpha^i &= \underbrace{\alpha|\alpha|\cdots|\alpha}_{i} && \text{if } \alpha \text{ is a state of some nBPP}_\tau \text{ process} \\ \alpha^i &= \underbrace{\alpha\|\alpha\|\cdots\|\alpha}_{i} && \text{if } \alpha \text{ is a state of some nBPP process} \end{aligned}$$

Definition 4.4 ($\mathbf{INF}_{\mathbf{BPP}}$). *Let Δ be a reduced $nBPP_\tau$ process.*

1. *A variable $Z \in Var(\Delta)$ is* simple *if all summands in the def. equation fo Z are of the form aZ^i, where $a \in Act$ and $i \in N \cup \{0\}$. Moreover, at leas one of those summands must be of the form aZ^k where $a \in Act$ and $k \geq 2$ Finally, the def. equation for Z must not contain two summands of the for $b, \overline{b}$, where $b \in \Lambda$.*

2. *The process Δ is said to be in INF_{BPP} if whenever $a\alpha$ is a summand in a de equation from Δ such that $Length(\alpha) \geq 2$, then $\alpha = Z^i$ for some simp variable Z and $i \geq 2$.*

Note that if Z is a simple variable, then $|Z| = 1$ because Z could not b normed otherwise.

Example 4.5. *The following process as well as process (4.1) are in INF_{BPP}, while the processes (4.3) is not:*

$$\begin{aligned} X &\stackrel{def}{=} aY + b(Z|Z) + b + \overline{b} \\ Y &\stackrel{def}{=} cY + bX + a(Z|Z|Z) \\ Z &\stackrel{def}{=} a(Z|Z) + \overline{a}(Z|Z|Z) + b + \overline{a} \end{aligned}$$

Remark 4.6. *The set of all reachable states of a process Δ in INF_{BPP} looks as follows:*

$$Var(\Delta) \cup \{Z^i \mid Z \in Var(\Delta) \text{ is a simple variable and } i \in N \cup \{0\}\}$$

Proposition 4.7. *Each process Δ in INF_{BPP} belongs to $nBPA_\tau \cap nBPP_\tau$.*

Proof: We show that a bisimilar nBPA$_\tau$ process $\overline{\Delta}$ is even effectively constructible. First we need to define the notion of *closed* simple variable—a simple variable $Z \in Var(\Delta)$ is closed if the following condition holds: If the def. equation for Z contains two summands of the form $bZ^i, \overline{b}Z^j$, then it also contains a summand τZ^{i+j-1} (the case $i = j = 0$ is impossible by Definition 4.4).

The set $Var(\overline{\Delta})$ looks as follows: for each $V \in Var(\Delta)$ we fix a fresh variable $\overline{V}$. Moreover, for each simple non-closed variable $Z \in Var(\Delta)$ we also fix a fresh variable $\overline{Z}_C$. Now we can start to transform Δ to $\overline{\Delta}$. For each equation $Y \stackrel{def}{=} \sum_{i=1}^n a_i\alpha_i$ of Δ we add the equation $\overline{Y} \stackrel{def}{=} \sum_{i=1}^n \mathcal{T}(a_i\alpha_i)$ to $\overline{\Delta}$, where $\mathcal{T}$ is defined as follows:

1. $\mathcal{T}(a_i) = a_i$
2. $\mathcal{T}(a_iV) = a_i\overline{V}$ for each $V \in Var(\Delta)$.
3. If $\alpha_i = Z^j$ where $Z \in Var(\Delta)$ is a closed simple variable and $j \geq 2$, then $\mathcal{T}(a_iZ^j) = a_i\overline{Z}^j$.
4. If $\alpha_i = Z^j$ where $Z \in Var(\Delta)$ is a non-closed simple variable and $j \geq 2$, then $\mathcal{T}(a_iZ^j) = a_i\overline{Z}_C^{\,j-1}.\overline{Z}$.

The defining equation for $\overline{Z}_C$, where $Z \in Var(\Delta)$ is a non-closed simpl variable, is constructed using following rules:

1. if aZ^i is a summand in the def. equation for Z, then $a\overline{Z}_C^{\,i}$ is a summan in the def. equation for $\overline{Z}_C$ in $\overline{\Delta}$.
2. if bZ^i, $\overline{b}Z^j$ are summands in the def. equation for Z, then $\tau\overline{Z}_C^{\,i+j-1}$ is summand in the def. equation for $\overline{Z}_C$ in $\overline{\Delta}$.

The fact $\Delta \sim \overline{\Delta}$ is easy to check. □

Example 4.8. *If we apply the transformation algorithm to the process of Exam ple 4.5, we obtain the following bisimilar $nBPA_\tau$ process:*

$$\begin{aligned} \overline{X} &\stackrel{def}{=} a\overline{Y} + b(\overline{Z}_C.\overline{Z}) + b + \overline{b} \\ \overline{Y} &\stackrel{def}{=} c\overline{Y} + b\overline{X} + a(\overline{Z}_C.\overline{Z}_C.\overline{Z}) \\ \overline{Z} &\stackrel{def}{=} a(\overline{Z}_C.\overline{Z}) + \overline{a}(\overline{Z}_C.\overline{Z}_C.\overline{Z}) + b + \overline{a} \\ \overline{Z}_C &\stackrel{def}{=} a(\overline{Z}_C.\overline{Z}_C) + \overline{a}(\overline{Z}_C.\overline{Z}_C.\overline{Z}_C) + b + \overline{a} + \tau(\overline{Z}_C.\overline{Z}_C.\overline{Z}_C.\overline{Z}_C) + \tau\overline{Z}_C \end{aligned}$$

Now we prove that each nBPP$_\tau$ process from nBPA$_\tau \cap$ nBPP$_\tau$ is bisimila to a process in INF$_{BPP}$. Several auxiliary definitions and lemmas are needed

Definition 4.9 (Assoc set). *Let Δ be a $nBPP_\tau$ process. For each growing var able $Y \in Var(\Delta)$ we define the set $Assoc(Y) \subseteq Var(\Delta)$ in the following way:*

$$\begin{aligned} Assoc(Y) \;=\; & \{P \in Var(\Delta),\ Y \rightarrow^* P\} \cup \\ & \{P \in Var(\Delta),\ P|Y \text{ is a reachable state of } \Delta\} \end{aligned}$$

A variable $L \in Var(\Delta)$ is lonely *if $L \notin Assoc(Y)$ for any growing variab $Y \in Var(\Delta)$.*

Lemma 4.10. *Let $\Delta \in nBPA_\tau \cap nBPP_\tau$ be a reduced $nBPP_\tau$ process. Let Y $Var(\Delta)$ be a growing variable. Then there is exactly one variable $Z_Y \in Var(\Delta$ such that:*

- *Z_Y is non-regular and $|Z_Y| = 1$*
- *If $P \in Assoc(Y)$, then Z_Y is reachable from P and $P \sim Z_Y^{|P|}$.*
- *If $a\alpha$ is a summand in the defining equation for Z_Y in Δ, then $\alpha \sim Z_Y^{|\alpha|}$*

Proof: As Y is growing, $Y \to^* Y|\beta$ where $\beta \in Var(\Delta)^\otimes$, $\beta \neq \emptyset$. As Δ is normed and in GNF, there is $Z_Y \in Var(\Delta)$, $|Z_Y| = 1$ such that $\beta \to^* Z_Y$. Hence $Y \to^* Y|\beta^i \to^* Y|Z_Y^i$ for any $i \in N$ (note that Z_Y is reachable from Y). From this and the definition of *Assoc* set we can easily conclude that if $P \in Assoc(Y)$ then the state $P|Z_Y^i$ is reachable for any $i \in N$.

As $\Delta \in$ nBPA$_\tau$ ∩ nBPP$_\tau$, there is a bisimilar nBPA$_\tau$ process Δ'. Let $n = |P|$, $m = \max\{|A|,\ A \in Var(\Delta')\}$. The state $P|Z_Y^{n.m}$ is a reachable state of Δ and therefore there is $\gamma \in Var(\Delta')^*$ such that $P|Z_Y^{n.m} \sim \gamma$. Bisimilar states must have the same norm, hence γ is a sequence of at least $n+1$ variables — $\gamma = A_1.A_2 \dots A_{n+1}.\delta$ where $\delta \in Var(\Delta')^*$. As $|P| = n$, $P \xrightarrow{s} \epsilon$ for some $s \in Act^*$ with $Length(s) = |P|$ — hence $P|Z_Y^{n.m} \xrightarrow{s} Z_Y^{n.m}$. The state $A_1.A_2 \dots A_{n+1}.\delta$ must be able to match the norm reducing sequence of actions s. As $Length(s) = n$, at most the first n variables of $A_1.A_2 \dots A_{n+1}.\delta$ can contribute to the sequence s, i.e., $A_1.A_2 \dots A_{n+1}.\delta \xrightarrow{s} \eta.A_{n+1}.\delta$ where $\eta \in Var(\Delta')^*$. As Δ' is normed, $\eta.A_{n+1}.\delta \xrightarrow{t} A_{n+1}.\delta$ for some $t \in Act^*$ with $Length(t) = |\eta|$. The state $Z_Y^{n.m}$ can match the sequence t only by removing $Length(t)$ copies of Z_Y:

$$\begin{array}{ccc}
P|Z_Y^{n.m} & \sim & A_1 \dots A_{n+1}.\delta \\
\downarrow s & & \downarrow s \\
Z_Y^{n.m} & \sim & \eta.A_{n+1}.\delta \\
\downarrow t & & \downarrow t \\
Z_Y^{n.m-|\eta|} & \sim & A_{n+1}.\delta
\end{array}$$

Now let $k = Length(s) + Length(t)$ (i.e., $k = |A_1 \dots A_n|$). Clearly $k \leq n.m$ and as $|Z_Y| = 1$, $P|Z_Y^{n.m} \xrightarrow{p} P|Z_Y^{n.m-k}$ where $Length(p) = k$. The state

$A_1.A_2 \dots A_{n+1}.\delta$ can match the sequence p only by $A_1.A_2 \dots A_{n+1}.\delta \xrightarrow{p} A_{n+1}$
By transitivity of $\sim$ we now obtain $P|Z_Y^{n.m-k} \sim Z_Y^{n.m-|\eta|}$, hence $P \sim Z_Y^{|P|}$.

As the variable Y is non-regular and $Y \sim Z_Y^{|Y|}$, the variable Z_Y is als non-regular. Moreover, Z_Y is a unique variable with the property $P \sim Z_Y^{|}$ for each $P \in Assoc(Y)$ because Δ is reduced.

A similar argument can be used to prove that Z_Y is reachable from eac $P \in Assoc(Y)$. As P is normed, $P \to^* P'$ where $|P'| = 1$. As $P \sim Z_Y^{|P|}$, $P' \sim Z$ and hence $P' = Z_Y$.

It remains to check that if $a\alpha$ is a summand of the defining equatio for Z_Y in Δ then $\alpha \sim Z_Y^{|\alpha|}$. But each variable $V \in \alpha$ belongs to $Assoc(Y$ (because $Y \to^* Z_Y \to^* V$) and thus $V \sim Z_Y^{|V|}$. Hence $\alpha \sim Z_Y^{|\alpha|}$.

Remark 4.11. *The symbol Z_Y always denotes the unique variable of Lemma 4.1 in the rest of this chapter.*

Lemma 4.12. *Let $\Delta \in$ nBPA$_\tau$ ∩ nBPP$_\tau$ be a reduced nBPP$_\tau$ process. Let $A|$ be a reachable state of Δ such that $A \in Assoc(Y)$ and $B \in Assoc(Q)$. The $Z_Y = Z_Q$.*

Proof: As Δ is reduced, it suffices to prove that $Z_Y \sim Z_Q$. As $A \in Assoc(Y$ $A \to^* Z_Y$ (see Lemma 4.10). Similarly, $B \to^* Z_Q$ and hence $Z_Y|Z_Q$ is reachable state of Δ. As Z_Q is non-regular, it can reach a state of an arb trary norm—for each $i \in N$ there is $\alpha_i \in Var(\Delta)^\otimes$ such that $Z_Q \to^* \alpha_i$ an $|\alpha_i| = i$. Clearly $\alpha_i \sim Z_Q^i$ because each variable of α_i belongs to $Assoc(Q$ Hence $Z_Y|\alpha_i \sim Z_Y|Z_Q^i$.

As $\Delta \in$ nBPA$_\tau$ ∩ nBPP$_\tau$, there is a bisimilar nBPA$_\tau$ process Δ'. Le $m = \max\{|V|,\ V \in Var(\Delta')\}$. The state $Z_Y|\alpha_m$ is a reachable state of and therefore there is $\gamma \in Var(\Delta')^*$ such that $Z_Y|\alpha_m \sim \gamma$ and hence als $Z_Y|Z_Q^m \sim \gamma$. Moreover, γ is a sequence of at least two variables.

Now we can use a similar construction as in the proof of Lemma 4.1 and conclude that $Z_Y|Z_Q^j \sim Z_Q^{j+1}$ for some $j \in N$. This implies $Z_Y \sim Z_Q$.

Lemma 4.13. *Let $\Delta \in nBPA_\tau \cap nBPP_\tau$ be a reduced $nBPP_\tau$ process. Let $L|A$ be a reachable state of Δ such that L is a lonely variable. Then A is a regular process (see Remark 2.6).*

Proof: Let us assume that A is not regular. Then $A \rightarrow^* Y$, where $Y \in Var(\Delta)$ is a growing variable (see Proposition 3.16). But then $L|A \rightarrow^* L|Y$, thus $L \in Assoc(Y)$ and we have a contradiction. □

Proposition 4.14. *Let $\Delta \in nBPA_\tau \cap nBPP_\tau$ be a $nBPP_\tau$ process. Then there is a process Δ' in INF_{BPP} such that $\Delta \sim \Delta'$.*

Proof: We can assume (w.l.o.g.) that Δ is reduced and in 3-GNF. The process Δ' can be obtained by the following transformation of defining equations of Δ (which can also add completely new variables and corresponding defining equations): if $X \stackrel{def}{=} \sum_{j=1}^{m} a_j\alpha_j$ is a defining equation from Δ, then $X \stackrel{def}{=} \sum_{j=1}^{m} \mathcal{T}(a_j\alpha_j)$ is added to Δ', where $\mathcal{T}$ is defined as follows:

- if $card(\alpha_j) \leq 1$, then $\mathcal{T}(a_j\alpha_j) = a_j\alpha_j$
- if $card(\alpha_j) = 2$ (i.e., $\alpha_j = A|B$) then there are three possibilities:
  1. $A \in Assoc(Y)$ and $B \in Assoc(Q)$. Then $A \sim Z_Y^{|A|}$ and $B \sim Z_Q^{|B|}$ (see Lemma 4.10). As $A|B$ is a reachable state, we can conclude (with a help of Lemma 4.12) that $Z_Y = Z_Q$, hence $A|B \sim Z_Y^{|A|+|B|}$. Thus $\mathcal{T}(a(A|B)) = a(Z_Y^{|A|+|B|})$.
  2. $A \in Assoc(Y)$ and B is lonely. But then $A \sim Z_Y^{|A|}$ and as Z_Y is not regular, A is not regular either. As the state $A|B$ is reachable and B is lonely, it contradicts Lemma 4.13. Hence this case is in fact impossible (as well as the case when A is lonely and $B \in Assoc(Q)$).
  3. A and B are lonely. Then A and B are regular (due to Lemma 4.13) and therefore the state $A|B$ is also regular. Each regular process can be represented in normal form (see Definition 2.13).

Let $\Delta_{A|B}$ be a regular process in normal form which is bisimila to $A|B$. We can assume (w.l.o.g.) that $Var(\Delta_{A|B}) \cap Var(\Delta') = \emptyset$. adds all equations from $\Delta_{A|B}$ to Δ' and $\mathcal{T}(a(A|B)) = a.N$ wher N is the leading variable of $\Delta_{A|B}$.

The transformation $\mathcal{T}$ preserves bisimilarity—hence $\Delta \sim \Delta'$. It remains t check that Δ' is in INF_{BPP}. Clearly each summand of each defining equatio from Δ' is of the form which is admitted by INF_{BPP}. If aZ^j is a summand c a defining equation in Δ' such that $j \geq 2$, then $Z = Z_Y$ for some grow ing variable $Y \in Var(\Delta)$. Let $a\alpha$ be a summand in the original definin equation for Z_Y in Δ. We need to show that each such summand mus have been transformed into $aZ_Y^{|\alpha|}$ by $\mathcal{T}$. But it is obvious as each variabl from α belongs to $Assoc(Y)$. If α is composed of a single variable V, the $V = Z_Y$ because $V \sim Z_Y$ (due to Lemma 4.10) and Δ is reduced. More over, at least one summand in the defining equation for Z_Y in Δ' is of th form aZ_Y^l where $l \geq 2$, because Z_Y would be regular otherwise. To com plete the proof we need to show that the defining equation for Z_Y in Δ cannot contain two summands of the form $b, \overline{b}$. Assume the converse. A $\Delta' \in nBPA_\tau \cap nBPP_\tau$, there is a $nBPA_\tau$ process Δ_2 such that $\Delta' \sim \Delta_2$ As Z_Y^i is a reachable state of Δ' for each $i \in N \cup \{0\}$ (see Remark 4.6 there is $\alpha_i \in Var(\Delta_2)^*$ such that $Z_Y^i \sim \alpha_i$ for each i. Moreover, we can as sume (w.l.o.g.) that each α_i is of maximal *Length*, i.e., if $\alpha_i \sim \beta$ for som $\beta \in Var(\Delta_2)^*$, then $Length(\alpha_i) \geq Length(\beta)$. Let k be the minimal num ber with the property $Length(\alpha_k) \geq 2$. Clearly $Length(\alpha_k) = 2$, becaus otherwise we could easily obtain a contradiction with the minimality c k. Hence $\alpha_k = P.Q$ for some $P, Q \in Var(\Delta_2)$. As $Z_Y^k \xrightarrow{b} Z_Y^{k-1}$, we also hav $P.Q \xrightarrow{b} \gamma$ for some $\gamma \sim \alpha_{k-1}$. By definitions of α_i and k, γ must be compose of a single variable. The only such state which can be reached from $P.Q$ i one step is Q, hence $\alpha_{k-1} \sim Q$. As the defining equation for Z_Y contain two summands $b, \overline{b}$, we also have a transition $Z_Y^k \xrightarrow{\tau} Z_Y^{k-2}$. But $P.Q$ canno reach a state which is bisimilar to α_{k-2} in one step, because α_{k-2} is (agai by definitions of α_i and k) composed of at most one variable which must be different from Q because $\alpha_{k-1} \not\sim \alpha_{k-2}$. Hence $\alpha_k \not\sim Z_Y^k$ and we have a contradiction. □

Propositions 4.7 and 4.14 give us the classification of nBPA$_\tau$ ∩ nBPP$_\tau$ in terms of nBPP$_\tau$ syntax.

Theorem 4.15. *The class nBPA$_\tau$ ∩ nBPP$_\tau$ contains exactly (up to bisimilarity) nBPP$_\tau$ processes in INF$_{BPP}$.*

The class nBPA$_\tau$ ∩ nBPP$_\tau$ can also be characterized using nBPA$_\tau$ syntax. To do this, we introduce a special normal form for nBPA$_\tau$ processes:

Definition 4.16 (INF$_{BPA}$). *Let Δ be a reduced nBPA$_\tau$ process.*

1. *Let $X, Y \in Var(\Delta)$ be non-regular variables. We say that Y is a* communication closure *(C-closure) of X if the following conditions hold:*
   - *All summands in the def. equation for X are either of the form a where $a \in Act$, or $a(Y^i.X)$ where $a \in Act$ and $i \in N \cup \{0\}$. Moreover, at least one summand is of the form $a(Y^k.X)$ where $k \geq 1$.*
   - *All summands in the def. equation for Y are of the form aY^i, where $a \in Act$ and $i \in N \cup \{0\}$.*
   - *aY^i is a summand in the def. equation for Y iff one of the following conditions holds:*
     (a) *$i = 0$ and a is a summand in the def. equation for X.*
     (b) *$i \geq 1$ and $a(Y^{i-1}.X)$ is a summand in the def. equation for X.*
     (c) *$a = \tau$ and there are two summands of the form $b\alpha_1, \overline{b}\alpha_2$ in the def. equation for X such that $i = Length(\alpha_1) + Length(\alpha_2) - 1$ (note that this condition ensures that def. equations for X, Y do not contain two summands of the form $b, \overline{b}$).*

2. *The process Δ is said to be in INF$_{BPA}$ if whenever $a\alpha$ is a summand in a de equation from Δ such that $Length(\alpha) \geq 2$, then $\alpha = Y^i.X$ for some $i \in$ 1 and $X, Y \in Var(\Delta)$ such that Y is a C-closure of X (note that X, Y need no be different—variables which are C-closures of themselves may exist).*

Note that if Y is a C-closure of X, then $|Y| = |X| = 1$. Another interestin property of X and Y is presented in the remark below.

Remark 4.17. *It is easy to check that if Y is a C-closure of X, then $Y^i.X \sim \overline{X}^{i+}$ where $\overline{X}$ is a nBPP$_\tau$ process composed of a single variable whose def. equation i obtained from the def. equation for X by substituting '.' with '|' and replacin each occurrence of X and Y with $\overline{X}$.*

Theorem 4.18. *The class nBPA$_\tau$ ∩ nBPP$_\tau$ contains exactly (up to bisimilarity nBPA$_\tau$ processes in INF$_{BPA}$.*

Proof: Each nBPA$_\tau$ process in INF$_{BPA}$ belongs to nBPA$_\tau$ ∩ nBPP$_\tau$, as a bisir ilar nBPP$_\tau$ process can be easily constructed by an algorithm which is ir verse to the algorithm presented in the proof of Proposition 4.7 (see Re mark 4.17). The fact that for each nBPA$_\tau$ process of nBPA$_\tau$ ∩ nBPP$_\tau$ ther is a bisimilar nBPA$_\tau$ process in INF$_{BPA}$ follows directly from Proposition 4. and Proposition 4.14 (note that the algorithm presented in the proof o Proposition 4.7 returns a nBPA$_\tau$ process which is almost in INF$_{BPA}$—th only "problem" is that it can contain different bisimilar variables and henc it need not be reduced.).

Our results can be applied to nBPA and nBPP processes as well. So far w have investigated the intersection of nBPA$_\tau$ and nBPP$_\tau$. It was desirabl to work with this unrestricted syntax, because we could also examine th problem when the "real" communications of a nBPP$_\tau$ process can be sim ulated by a sequential nBPA$_\tau$ process. However, the characterization o nBPA ∩ nBPP is much simpler and therefore we present it explicitly.

Definition 4.19 (INF). *Let Δ be a reduced nBPA (or nBPP) process in GNF.*

1. *A variable $Z \in Var(\Delta)$ is* simple *if all summands in the def. equation for Z are of the form aZ^i, where $a \in Act$ and $i \in N \cup \{0\}$. Moreover, at least one of those summands must be of the form aZ^k where $a \in Act$ and $k \geq 2$.*

2. *The process Δ is said to be in INF if whenever $a\alpha$ is a summand in a def. equation from Δ such that $Length(\alpha) \geq 2$ (or $card(\alpha) \geq 2$), then $\alpha = Z^i$ for some simple variable Z and $i \geq 2$.*

Note that nBPA (or nBPP) processes in INF have a nice property—a bisimilar nBPP (or nBPA) process can be obtained just by replacing the '.' operator with the '‖' operator (or by replacing the '‖' operator with the '.' operator).

Theorem 4.20. *The class $nBPA \cap nBPP$ contains exactly (up to bisimilarity) nBPA (or nBPP) processes in INF.*

4.2 Deciding whether $\Delta \in$ nBPA$_\tau \cap$ nBPP$_\tau$

In this section we prove that the problem whether a given nBPA$_\tau$ or nBPP$_\tau$ process Δ belongs to nBPA$_\tau \cap$ nBPP$_\tau$ is decidable in polynomial time. The technique is essentially similar in both cases—we check whether each summand of each defining equation of Δ whose form is not admitted by INF$_{\text{BPA}}$ (or INF$_{\text{BPP}}$) can be in principal transformed so that requirements of INF$_{\text{BPA}}$ (or INF$_{\text{BPP}}$) are satisfied. We also show that if a nBPA$_\tau$ (or nBPP$_\tau$) process belongs to nBPA$_\tau \cap$ nBPP$_\tau$, then a bisimilar process Δ' in INF$_{\text{BPA}}$ (or INF$_{\text{BPP}}$) is effectively constructible. Simplified versions of our algorithms which work for nBPA and nBPP processes are presented as well.

Definition 4.21 ($\mathrm{S}(\Delta)$, $\mathrm{R}(\Delta)$ and $\mathrm{G}(\Delta)$ sets). *Let Δ be a nBPA$_\tau$ or nBPP$_\tau$ process in GNF.*

- *The set $S(\Delta) \subseteq Var(\Delta)$ is composed of all variables V such that $|V| = 1$, V is non-regular and if $a\alpha$ is a summand in the defining equation for V in Δ, then $\alpha \sim V^{|\alpha|}$.*

- *The set $R(\Delta) \subseteq Var(\Delta)$ contains all regular variables of Δ.*
- *The set $G(\Delta) \subseteq Var(\Delta)$ contains all growing variables of Δ.*

The sets $S(\Delta)$, $R(\Delta)$ and $G(\Delta)$ can be constructed in polynomial time be cause bisimilarity and regularity are decidable for nBPA$_\tau$ and nBPP$_\tau$ pro cesses in polynomial time (see [HJM94a], [HJM94b] and Section 3.1.3).

If Δ is a nBPA$_\tau$ (or nBPP$_\tau$) process from nBPA$_\tau \cap$ nBPP$_\tau$, then there i Δ' in INF$_{\text{BPA}}$ (or INF$_{\text{BPP}}$) such that $\Delta \sim \Delta'$. In case of nBPP$_\tau$ processes th set $S(\Delta)$ contains in fact variables which can be (potentially) bisimilar t simple variables of Δ'. In case of nBPA$_\tau$ processes the set $S(\Delta)$ contain variables which can be bisimilar to C-closures of variables from $Var(\Delta')$.

The three lemmas below together prove correctness of our algorith which decides the membership to nBPA$_\tau \cap$ nBPP$_\tau$ for nBPP$_\tau$ processes.

Lemma 4.22. *Let Δ be a reduced nBPP$_\tau$ process in 3-GNF and let $a(A|B)$ b a summand of a defining equation from Δ such that A is regular and B is non regular. Then $\Delta \notin nBPA_\tau \cap nBPP_\tau$.*

Proof: Assume there is a nBPP$_\tau$ process Δ' in INF$_{\text{BPP}}$ such that $\Delta \sim \Delta'$. Le $n = \max\{|Y|,\ Y \in Var(\Delta')\}$. As B is non-regular, it can reach a state of a arbitrary norm—let $B \rightarrow^* \beta$ where $|\beta| > n$. Then $A|\beta$ is a reachable state c Δ and thus $A|\beta \sim \beta'$ for some reachable state β' of Δ'. As $|A|\beta| > n$, w can conclude that $\beta' = Z^{|A|\beta|}$ where $Z \in Var(\Delta')$ is a simple variable (se Remark 4.6). Hence $A \sim Z^{|A|}$ and as each simple variable is growing (se Definition 4.4), it contradicts regularity of A. □

Lemma 4.23. *Let Δ be a reduced nBPP$_\tau$ process in 3-GNF which belongs t nBPA$_\tau \cap$ nBPP$_\tau$. Let $a(A|B)$ be a summand of a defining equation from Δ suc that A and B are non-regular. Then there is exactly one variable $Z \in S(\Delta)$ suc that $A|B \sim Z^{|A|B|}$.*

Proof: Let Δ' be a nBPP$_\tau$ process in INF$_{\text{BPP}}$ such that $\Delta \sim \Delta'$. Let n = $\max\{|Y|,\ Y \in Var(\Delta')\}$. Using the same argument as in the proof c

Lemma 4.22 we obtain $A \sim P^{|A|}$, $B \sim Q^{|B|}$ where $P, Q \in Var(\Delta')$ are simple variables. We show that $P = Q$. Let $A \to^* \alpha$ where $|\alpha| > n$. Then clearly $\alpha \sim P^{|\alpha|}$ and as $\alpha|B$ is a reachable state of Δ, $\alpha|B \sim R^{|\alpha|B|}$ where $R \in Var(\Delta')$ is a simple variable. To sum up, we have $\alpha|B \sim P^{|\alpha|}|Q^{|B|} \sim R^{|\alpha|B|}$. Hence $P \sim R \sim Q$ and thus $P = R = Q$ because Δ' is reduced. As e.g. P is a reachable state of Δ', there is a reachable state γ of Δ such that $P \sim \gamma$. As $|P| = 1$, we can conclude $\gamma = Z$ for some $Z \in Var(\Delta)$ which clearly belongs to $S(\Delta)$. Moreover, Z is unique because Δ is reduced. □

Lemma 4.24. *Let Δ be a nBPP$_\tau$ process in GNF and let $X \in S(\Delta)$. If the defining equation for X contains two summands of the form $b, \bar{b}$, then $\Delta \notin$ nBPA$_\tau \cap$ nBPP$_\tau$.*

Proof: Assume there is a nBPP$_\tau$ process Δ' in INF$_{BPP}$ such that $\Delta \sim \Delta'$. Using the same kind of argument as in the proof of Lemma 4.22 we obtain $X \sim Z$ for some simple variable $Z \in Var(\Delta')$. As the def. equation for X contains two summands of the form $b, \bar{b}$ and $X \sim Z$, the def. equation for Z must contain those summands too—hence Z is not simple and we have a contradiction. □

The promised (constructive) algorithm which decides the membership to nBPA$_\tau \cap$ nBPP$_\tau$ for nBPP$_\tau$ processes is presented in Figure 4.1. Steps which are executed only by the constructive algorithm are shaded—if we omit everything on a grey background, we obtain a non-constructive polynomial algorithm. The abbreviation “NFR(Δ)” stands for the **N**ormal **F**orm of the **R**egular process Δ, which can be effectively constructed (see Section 3.1.2). We always assume that NFR(Δ) contains fresh variables which are not contained in any other process we are working with. When the command <u>return</u> is executed, the algorithm *halts* and returns the value which follows immediately after the keyword <u>return</u>.

The constructive algorithm is not polynomial because the construction of NFR is not polynomial—a regular nBPP$_\tau$ process in 3-GNF with n

Algorithm: A constructive test of the membership to nBPA$_\tau \cap$ nBPP$_\tau$ for nBPP$_\tau$ processes.

Input: A reduced nBPP$_\tau$ process Δ in 3-GNF.

Output: **YES** and a nBPP$_\tau$ process Δ' in INF$_{BPP}$ such that $\Delta \sim \Delta'$ if $\Delta \in$ nBPA$_\tau \cap$ nBPP$_\tau$.
NO otherwise.

1. Construct the sets $S(\Delta)$, $R(\Delta)$ and $G(\Delta)$.
2. <u>If</u> there is $X \in S(\Delta)$ whose def. equation contains two summands of the form $b, \bar{b}$ <u>then</u>
   <u>return</u> **NO**;
3. <u>If</u> $G(\Delta) = \emptyset$ <u>then</u>
   Δ' :=NFR(Δ) ;
   <u>return</u> **YES** and Δ' ;
4. $\Delta' := \Delta$;
5. <u>for</u> <u>each</u> summand of the form $a(A|B)$ in defining equations of Δ <u>do</u>
   <u>if</u> $A, B \in R(\Delta)$ <u>then</u>
   Construct NFR($A|B$) ;
   Replace the summand $a(A|B)$ with aN in Δ', where N is the leading variable of NFR($A|B$) ;
   $\Delta' := \Delta' \cup$NFR($A|B$) ;
   <u>if</u> ($A \in R(\Delta)$ and $B \notin R(\Delta)$) or ($A \notin R(\Delta)$ and $B \in R(\Delta)$) <u>then</u>
   <u>return</u> **NO**;
   <u>if</u> $A, B \notin R(\Delta)$ <u>then</u>
   <u>if</u> there exists $Z \in S(\Delta)$ such that $A|B \sim Z^{|A|B|}$
   <u>then</u> Replace the summand $a(A|B)$ with $a(Z^{|A|B|})$ in Δ' ;
   <u>else</u> <u>return</u> **NO**;
6. <u>return</u> **YES** and Δ' ;

Figure 4.1: An algorithm which (constructively) decides the membershi to nBPA$_\tau \cap$ nBPP$_\tau$ for nBPP$_\tau$ processes.

variables can generally reach exponentially many pairwise non-bisimilar states and each of these states requires a special variable.

Our algorithm for nBPP_τ processes works for pure nBPP processes as well. It suffices to replace the '|' operator with the '‖' operator in our description. As there are no communications in nBPP, the notion of dual action is no longer sensible—hence the second step of our algorithm can be removed in case of nBPP processes.

Now we provide an analogous algorithm for nBPA_τ processes. We start with some auxiliary definitions and lemmas.

Definition 4.25 (CL sets). *Let Δ be a $nBPA_\tau$. For each $Y \in S(\Delta)$ we define the set $CL(Y)$, composed of all $X \in Var(\Delta)$ which satisfy the following conditions:*

- *If $a\alpha$ is a summand in the def. equation for X such that $Length(\alpha) \geq 1$, then $\alpha \sim Y^{|\alpha|-1}.X$.*
- *The def. equation for Y contains a summand bisimilar to aY^k, $k \in N \cup \{0\}$, iff one of the following conditions holds:*
  1. *$k = 0$ and the def. equation for X contains a summand 'a'.*
  2. *$k > 0$ and the def. equation for X contains a summand which is bisimilar to $a(Y^{k-1}.X)$.*
  3. *$a = \tau$ and the def. equation for X contains two summands of the form $b\alpha_1, \overline{b}\alpha_2$ such that $k = Length(\alpha_1) + Length(\alpha_2) - 1$.*

It is easy to see that the set $CL(Y)$ can be constructed in polynomial time for each $Y \in S(\Delta)$. The following lemma is due to D. Caucal (see [Cau88]):

Lemma 4.26. *Let Δ, Δ' be $nBPA_\tau$ processes in GNF and let $\alpha, \beta \in Var(\Delta)^*$, $\alpha', \beta' \in Var(\Delta')^*$ such that $\beta \sim \beta'$ and $\alpha.\beta \sim \alpha'.\beta'$. Then $\alpha \sim \alpha'$*

Lemma 4.27. *Let Δ, Δ' be $nBPA_\tau$ processes. Let $A_1, \ldots, A_k \in Var(\Delta)$, $X, Y \in Var(\Delta')$ such that $|X| = |Y| = 1$ and $A_1.\cdots.A_k \sim Y^l.X$ where $l = |A_1.\cdots.A_k| - 1$. Then $A_k \sim Y^{|A_k|-1}.X$ and $A_i \sim Y^{|A_i|}$ for $1 \leq i < k$.*

Proof: Clearly $A_k \sim Y^{|A_k|-1}.X$. Hence $A_1.\cdots.A_{k-1} \sim Y^{|A_1.\cdots.A_{k-1}|}$ (due t Lemma 4.26). The fact $A_i \sim Y^{|A_i|}$ for $1 \leq i < k$ can be proved by inductio on k. If $k = 2$ then $A_1 \sim Y^{|A_1|}$ and our lemma holds. If $k > 2$, then clearl $A_{k-1} \sim Y^{|A_{k-1}|}$ and due to Lemma 4.26 we have $A_1.\cdots.A_{k-2} \sim Y^{|A_1.\cdots.A_{k-2}}$ Now we can use the inductive hypothesis and conclude that $A_i \sim Y^{|A_i|}$ fo $1 \leq i < (k-2)$. □

Lemma 4.28. *Let Δ be a reduced $nBPA_\tau$ process in 3-GNF which belongs t $nBPA_\tau \cap nBPP_\tau$. Let $Q.\alpha$ be a reachable state of Δ such that $Q \in G(\Delta)$, α ≠ ϵ. Then there are unique variables $Y \in S(\Delta)$, $X \in CL(Y)$ such that $Q.\alpha$ ∼ $Y^{|Q.\alpha|-1}.X$.*

Proof: As $\Delta \in \text{nBPA}_\tau \cap \text{nBPP}_\tau$, there is a nBPA_τ process Δ' in INF_{BPA} suc that $\Delta \sim \Delta'$. Let $n = \max\{|A|,\ A \in Var(\Delta')\}$. As Q is growing, $Q \to^* Q.$ where $\gamma \neq \epsilon$. Hence the state $Q.\gamma^n.\alpha$ is a reachable state of Δ and therefor there is a reachable state δ of Δ' such that $Q.\gamma^n.\alpha \sim \delta$. As $|Q.\gamma^n.\alpha| >$ r we can conclude $\delta = R^{|Q.\gamma^n.\alpha|-1}.S$, where R is a C-closure of S (see Defin tion 4.16). Hence $Q.\gamma^n.\alpha \sim R^{|Q.\gamma^n.\alpha|-1}.S$ and due to Lemma 4.27 we hav $\alpha \sim R^{|\alpha|-1}.S$ and $Q \sim R^{|Q|}$, thus $Q.\alpha \sim R^{|Q.\alpha|-1}.S$. Now it suffices to sho that there are $Y \in S(\Delta)$, $X \in CL(Y)$ such that $Y \sim R$ and $X \sim S$. As Δ is normed, $Q \xrightarrow{s} Y$ where $|Y| = 1$ and s is a norm-decreasing sequenc of actions. Then $Q.\alpha \xrightarrow{s} Y.\alpha$ and as $Q.\alpha \sim R^{|Q.\alpha|-1}.S$, the state $R^{|Q.\alpha|-1}.$ must be able to match the sequence s and enter a state bisimilar to $Y.\alpha$. A s is norm-decreasing and $|R| = 1$, the only such state is $R^{|Y.\alpha|-1}.S$. Henc $Y.\alpha \sim R^{|Y.\alpha|-1}.S$ and due to Lemma 4.27 we have $Y \sim R$. The fact $Y \in S(\Delta$ follows directly from Definition 4.16. As S is a reachable state of Δ', ther is a variable $X \in S(\Delta)$ such that $X \sim S$. Clearly $X \in CL(Y)$ (see Defin tion 4.16). Variables X, Y are unique because Δ is reduced. □

It is worth noting that the variables X, Y of the previous lemma need no be different—if a nBPA_τ process Δ belongs to $\text{nBPA}_\tau \cap \text{nBPP}_\tau$, then eac $Y \in S(\Delta)$ belongs to $CL(Y)$.

To prove correctness of our algorithm which decides the membership to nBPA$_\tau$ $\cap$ nBPP$_\tau$ for nBPA$_\tau$ processes we need some lemmas about summands:

Lemma 4.29. *Let Δ be a reduced nBPA$_\tau$ process in 3-GNF and let $a(A.B)$ be a summand of a defining equation from Δ such that A is non-regular and B is regular. Then $\Delta \notin nBPA_\tau \cap nBPP_\tau$.*

Proof: As $a(A.B)$ is a summand of a defining equation from Δ and Δ is normed and in GNF, there is a reachable state of the form $A.B.\beta$. As A is non-regular, $A \to^* Q.\alpha$ where $Q \in G(\Delta)$. Hence $Q.\alpha.B.\beta$ is a reachable state of Δ and due to Lemma 4.28 we have $Q.\alpha.B.\beta \sim Y^{|Q.\alpha.B.\beta|-1}.X$ for some $Y \in S(\Delta)$, $X \in CL(Y)$. With a help of Lemma 4.27 we obtain that $B \sim Y^{|B|}$ or $B \sim Y^{|B|-1}.X$ (the latter possibility holds if $\beta = \epsilon$). As X, Y are growing, it contradicts regularity of B. $\square$

Lemma 4.30. *Let Δ be a reduced nBPA$_\tau$ process in 3-GNF. Let $a(A.B)$ be a summand of a defining equation from Δ such that A is regular and B is non-regular. Then it is possible to replace the summand $a(A.B)$ with aN where $N \notin Var(\Delta)$ and to add a finite number of new equations in INF_{BPA} to Δ such that the resulting process Δ_1 is bisimilar to Δ.*

Proof: As A is regular, it is possible to construct $\Delta_A := NFR(A)$ such that $Var(\Delta) \cap Var(\Delta_A) = \emptyset$. Now we modify defining equations of Δ_A slightly—each summand of the form a where $a \in Act$ is replaced with aB. The resulting system of equations is in INF$_{BPA}$. If we add the modified system Δ_A to Δ and replace the summand $a(A.B)$ with aN where N is the leading variable of Δ_A, we obtain a process Δ_1 which is clearly bisimilar to Δ. $\square$

Lemma 4.31. *Let Δ be a reduced nBPA$_\tau$ process in 3-GNF and let $a(A.B)$ be a summand of a defining equation from Δ such that A and B are non-regular. Then*

1. *If $\Delta \in nBPA_\tau \cap nBPP_\tau$ then there are unique variables $Y \in S(\Delta)$, $X \in CL(Y)$ such that $B \sim Y^{|B|-1}.X$*

2. *Let $B \sim Y^{|B|-1}.X$ for some $Y \in S(\Delta)$ and $X \in CL(Y)$. If there is a sequenc of transitions $A = A_0 \xrightarrow{a_0} A_1.\alpha_1 \xrightarrow{a_1} A_2.\alpha_2 \xrightarrow{a_2} \cdots \xrightarrow{a_k} A_k.\alpha_k$ such that $k \geq$ ($A_k \in G(\Delta)$ and $A_k.\alpha_k \not\sim Y^{|A_k.\alpha_k|}$, then $\Delta \notin nBPA_\tau \cap nBPP_\tau$.*

3. *Let $B \sim Y^{|B|-1}.X$ for some $Y \in S(\Delta)$ and $X \in CL(Y)$. If for each sequenc of transitions $A = A_0 \xrightarrow{a_0} A_1.\alpha_1 \xrightarrow{a_1} A_2.\alpha_2 \xrightarrow{a_2} \cdots \xrightarrow{a_k} A_k.\alpha_k$ such tha $A_k \in G(\Delta)$ the state $A_k.\alpha_k$ is bisimilar to $Y^{|A_k.\alpha_k|}$, then the summand $a(A.E$ can be replaced with aN where $N \notin Var(\Delta)$ and a finite number of ne equations in INF_{BPA} can be added to Δ such that the resulting process Δ_2 i bisimilar to Δ.*

Proof:

1. As A is non-regular, $A \to^* Q.\alpha$ where $Q \in G(\Delta)$. The proof can b easily completed with a help of Lemma 4.27 and Lemma 4.28.

2. This is a consequence of Lemma 4.27 and Lemma 4.28.

3. It suffices to realize that if $A = A_0 \xrightarrow{a_0} A_1.\alpha_1 \xrightarrow{a_1} A_2.\alpha_2 \xrightarrow{a_2} \cdots \xrightarrow{a_k} A_k.\alpha$ is a sequence of transitions such that $A_0, \ldots, A_{k-1} \notin G(\Delta)$ and A_k ($G(\Delta)$, then $Length(A_i.\alpha_i) \leq card(Var(\Delta))$ for $0 \leq i \leq k-1$ (her we use the assumption that Δ is in 3-GNF. Naturally, $Length(A_i.\alpha$ is bounded also in case of general GNF). As there are only finitel many sequences of variables of this bounded length, we can intro duce a fresh variable for each of them. To construct the process Δ we use a similar procedure as in the proof of Lemma 4.30.

An existence of a sequence $A = A_0 \xrightarrow{a_0} A_1.\alpha_1 \xrightarrow{a_1} A_2.\alpha_2 \xrightarrow{a_2} \cdots \xrightarrow{a_k} A_k.\alpha_k$ suc that $A_k \in G(\Delta)$ and $A_k.\alpha_k \not\sim Y^{|A_k.\alpha_k|}$ is decidable in polynomial time:

Lemma 4.32. *Let Δ be a reduced nBPA$_\tau$ process in 3-GNF. Let $A \in Var(\Delta)$ be non-regular variable and let $Y \in S(\Delta)$. The problem whether A can reach a sta of the form $Q.\alpha$ where $Q \in G(\Delta)$ and $Q.\alpha \not\sim Y^{|Q.\alpha|}$ is decidable in polynomi time.*

Proof: We divide the set $Var(\Delta)$ into two disjoint subsets of *successful* and *unsuccessful* variables. $P \in Var(\Delta)$ is unsuccessful if one of the following conditions holds:

- P is growing and $P \not\sim Y^{|P|}$.
- The defining equation for P in Δ contains a summand of the form $a(R.S)$ where R is non-regular and $S \not\sim Y^{|S|}$.

A variable is successful if it is not unsuccessful. Furthermore, we define the binary relation '$\Rightarrow$' on $Var(\Delta)$: $U \Rightarrow V$ iff U is successful and the defining equation for U in Δ contains a summand which is of one of the following forms:

- aV
- $a(V.W)$ where $W \in Var(\Delta)$
- $a(W.V)$ where $W \in Var(\Delta)$ is regular

Let '$\Rightarrow^*$' be the reflexive and transitive closure of '$\Rightarrow$'. It is not hard to prove that A can reach a state of the form $Q.\alpha$ where Q is growing and $Q.\alpha \not\sim Y^{|Q.\alpha|}$ iff $A \Rightarrow^* T$ for some unsuccessful variable T. As the relation '$\Rightarrow^*$' can be constructed in polynomial time, the proof is finished. $\square$

An algorithm which decides the membership to $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ for nBPA_τ processes is presented in Figure 4.2. We use the same notation as in the case of nBPP_τ.

In case of nBPA processes our algorithm must be slightly modified (and simplified). This is a consequence of the fact that a nBPA process Δ belongs to nBPA $\cap$ nBPP iff it can be represented in INF—and INF is a little different from INF_{BPA} (see Definitions 4.19 and 4.16). Lemma 4.29 and Lemma 4.30 are valid also for nBPA processes. Instead of Lemma 4.31 we can prove the following (in a similar way):

Algorithm: A constructive test of the membership to $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ for nBPA_τ processes.

Input: A reduced nBPA_τ process Δ in 3-GNF.

Output: **YES** and a nBPA_τ process Δ' in INF_{BPA} such that $\Delta \sim \Delta'$ if $\Delta \in \text{nBPA}_\tau \cap \text{nBPP}_\tau$.
NO otherwise.

1. Construct the sets $S(\Delta)$, $R(\Delta)$, $G(\Delta)$ and for each $Y \in S(\Delta)$ construct the set $CL(Y)$.
2. <u>If</u> $(G(\Delta) = \emptyset)$ <u>then</u>
   $\Delta' :=$NFR(Δ);
   <u>return</u> **YES** and Δ';
3. $\Delta' := \Delta$;
4. <u>for</u> each summand of the form $a(A.B)$ in defining equations of Δ <u>do</u>
   - <u>if</u> $A, B \in R(\Delta)$ <u>then</u>
     Construct NFR$(A.B)$;
     Replace the summand $a(A.B)$ with aN in Δ', where N is the leading variable of NFR$(A.B)$;
     $\Delta' := \Delta' \cup$NFR$(A.B)$;
   - <u>if</u> $A \notin R(\Delta)$ and $B \in R(\Delta)$ <u>then</u>
     <u>return</u> **NO**;
   - <u>if</u> $A \in R(\Delta)$ and $B \notin R(\Delta)$ <u>then</u>
     Construct the process Δ_1 of Lemma 4.30;
     $\Delta' := \Delta_1$;
   - <u>if</u> $A, B \notin R(\Delta)$ <u>then</u>
     <u>if</u> there exist $Y \in S(\Delta), X \in CL(Y)$ such that $B \sim Y^{|B|-1}.X$
     <u>then</u> <u>if</u> A can reach a state $Q.\alpha$ where $Q \in G(\Delta)$ and $Q.\alpha \not\sim Y^{|Q.\alpha|}$
     <u>then</u> <u>return</u> **NO**;
     <u>else</u> Construct the process Δ_2 of Lemma 4.31;
     $\Delta' := \Delta_2$;
     <u>else</u> <u>return</u> **NO**;
5. <u>return</u> **YES** and Δ';

Figure 4.2: An algorithm which (constructively) decides the membershi to $\text{nBPA}_\tau \cap \text{nBPP}_\tau$ for nBPA_τ processes.

Lemma 4.33. *Let Δ be a reduced nBPA process in 3-GNF and let $a(A.B)$ be a summand of a defining equation from Δ such that A and B are non-regular. Then*

1. *If $\Delta \in nBPA \cap nBPP$ then there is a unique variable $Z \in S(\Delta)$ such that $B \sim Z^{|B|}$*

2. *Let $B \sim Z^{|B|}$ for some $Z \in S(\Delta)$. If there is a sequence of transitions $A = A_0 \stackrel{a_0}{\to} A_1.\alpha_1 \stackrel{a_1}{\to} A_2.\alpha_2 \stackrel{a_2}{\to} \cdots \stackrel{a_k}{\to} A_k.\alpha_k$ such that $k \geq 0$, $A_k \in G(\Delta)$ and $A_k.\alpha_k \not\sim Z^{|A_k.\alpha_k|}$, then $\Delta \notin nBPA \cap nBPP$.*

3. *Let $B \sim Z^{|B|}$ for some $Z \in S(\Delta)$. If for each sequence of transitions $A = A_0 \stackrel{a_0}{\to} A_1.\alpha_1 \stackrel{a_1}{\to} A_2.\alpha_2 \stackrel{a_2}{\to} \cdots \stackrel{a_k}{\to} A_k.\alpha_k$ such that $A_k \in G(\Delta)$ the state $A_k.\alpha_k$ is bisimilar to $Z^{|A_k.\alpha_k|}$, then the summand $a(A.B)$ can be replaced with aN where $N \notin Var(\Delta)$ and a finite number of new equations in INF can be added to Δ such that the resulting process Δ_2 is bisimilar to Δ.*

Our algorithm for nBPA processes differs from the algorithm of Figure 4.2 in two things—the sets $CL(Y)$ for $Y \in S(\Delta)$ are not computed at all and the last if statement in the loop of step 4 is replaced with the following code:

if $A, B \notin R(\Delta)$ then
  if there exist $Z \in S(\Delta)$ such that $B \sim Z^{|B|}$
    then if A can reach a state $Q.\alpha$ where $Q \in G(\Delta)$ and $Q.\alpha \not\sim Z^{|Q.\alpha|}$
      then return **NO**;
      else Construct the process Δ_2 of Lemma 4.33 ;
        $\Delta' := \Delta_2$;
    else return **NO**;

The existence of constructive variants of presented algorithms allow us to prove the following theorem:

Theorem 4.34. *Bisimilarity is decidable in the union of nBPA$_\tau$ and nBPP$_\tau$ processes.*

Proof: Given two nBPA$_\tau$ or nBPP$_\tau$ processes, it is possible to check bisimilarity using algorithms which were published in [HJM94a] and [HJM94b].

If we get a nBPP$_\tau$ process Δ_1 and a nBPA$_\tau$ process Δ_2, then we run one o the constructive algorithms presented earlier. We can choose e.g., the firs algorithm with Δ_1 on input. If it answers **NO**, then $\Delta_1 \not\sim \Delta_2$. Otherwis we obtain a nBPP$_\tau$ process Δ_1' in INF$_{BPP}$ which is bisimilar to Δ_1. Nov it suffices to check bisimilarity between two nBPA$_\tau$ processes $\overline{\Delta_1'}$ and Δ_2 where $\overline{\Delta_1'}$ is obtained by running the algorithm presented in the proof o Proposition 4.7 with Δ_1' on input. □

Note that the corresponding statement holds for nBPA and nBPP processe by specialization.

4.3 Related Work and Future Research

The problem whether a given nBPP process belongs to nBPA ∩ nBPP ha been independently examined by Blanco in [Bla95] where it is shown tha given a nBPP process, one can decide whether there is a bisimilar nBPA process. Blanco's approach is based on special properties of BPA transi tion graphs (see [CM90]). A test whether a given nBPP graph has thes properties is given in the work. Consequently, this result does not allov for testing whether a given *nBPA* process belongs to the intersection. Th generalization to nBPA$_\tau$ and nBPP$_\tau$ classes is not considered at all.

Our result on the classification of nBPA ∩ nBPP might be of some in terest from the point of view of formal languages/automata theory a well. INF (for nBPA processes) can be taken as a special type of CF gram mars which generate languages of the form $R.(L_1 \cup \ldots \cup L_n)$, where R i regular and each L_i can be generated by a CF grammar having just on nonterminal and rules of the form $Z \to aZ^k$, $k \geq 0$. Considering lan guage equivalence only, it is obvious that languages of the mentioned typ $R.(L_1 \cup \ldots \cup L_n)$ can be recognized by nondeterministic one-counter au tomata. Hence our result on the classification of nBPA ∩ nBPP can be con sidered as a refinement of the result achieved in [Sch92] on the contex freeness of languages generated by Petri nets, as BPP processes form a proper subclass of Petri nets.

An obvious question is whether our results can be extended to classes of all (not only normed) BPA and BPP processes. The class BPA ∩ BPP contains also processes which cannot be presented in INF. Consider the following BPP process (this example is due to I. Černá):

$$\begin{aligned} X &\stackrel{def}{=} a(Y \| X) \\ Y &\stackrel{def}{=} b \end{aligned}$$

The process X cannot be presented in INF. But it obviously belongs to BPA ∩ BPP; a bisimilar BPA process looks as follows:

$$\begin{aligned} A &\stackrel{def}{=} a(B.A) \\ B &\stackrel{def}{=} a(B.B) + b \end{aligned}$$

Transition systems generated by X and A are isomorphic:

$$\begin{tikzcd} \bullet \arrow[r, shift left, "a"] & \circ \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] & \circ \arrow[l, shift left, "b"] \arrow[r, shift left, "a"] & \circ \arrow[l, shift left, "b"] & \cdots \end{tikzcd}$$

This indicates that the problem is actually more complicated. Techniques which were used for normed processes cannot be applied—it seems however, that a deeper study of the structure of BPA and BPP transition graphs could help.

Chapter 5

Parallelization of nBPA Processes

A general problem considered by many researchers is how to improve performance of sequential programs by parallelization. In this chapter we study this problem within the framework of process algebras. They provide us with a pleasant formalism which allows to specify sequential as well as parallel programs. Here we adopt nBPA processes as a simple model of sequential behaviours.

The problem of possible decomposition of processes into a parallel product of primes[1] was first addressed by Milner and Moller in [MM93]. A more general result was later proved by Christensen, Hirshfeld and Moller (see [CHM93b])—it says that *each* normed process has a unique decomposition into primes up to bisimilarity. However, the proof is non-constructive.

This chapter is organized as follows. In Section 5.1 we characterize all decomposable nBPA processes together with their decompositions via special normal forms. As a consequence we also obtain a refinement of the result achieved in [BS94].

In Section 5.2 we show that any nBPA process can be decomposed into

[1] A process is prime if it cannot be equivalently expressed as a parallel product of two nontrivial processes.

a parallel product of primes effectively. We also prove several related de cidability results. Finally, we prove that bisimilarity is decidable in a larg subclass of nPA processes (see [BW90]), which consists of processes of th form $\Delta_1\|\cdots\|\Delta_n$, where each Δ_i is a nBPA or nBPP process. As bisimilarit coincides with language equivalence in the class of normed deterministi processes, obtained results can also be applied to determistic context-fre grammars (which are in fact deterministic nBPA processes). For exampl it is decidable whether a given deterministic CF grammar generates a lan guage which can be defined as a *shuffle* of two nonempty deterministic C languages L_1 and L_2. If the answer is positive, then deterministic CF gram mars generating L_1 and L_2 can be effectively constructed. See Section 5. for details. Presented results have been previously published as [Kuč97].

Remark 5.1. *In this chapter we rely on previously proved results on regularit of nBPA and nBPP processes (see Section 3.1.3) and decidability of bisimilarit in the union of nBPA and nBPP processes (see Theorem 4.34).*

Remark 5.2 (special notation). *In the rest of this chapter we also use some spe cial notation (due to the lack of general standard). To improve readability, we pu all specialties to one place:*

- *if α is a regular state of a nBPA or nBPP process (see Remark 2.6), the $NFR(\alpha)$ denotes a bisimilar regular process in normal form, which can b effectively constructed (see Section 3.1.2). Furthermore, we always assum that $NFR(\alpha)$ contains completely fresh variables which are not containe in any other process we deal with.*

- *the class of all processes for which there is a bisimilar nBPA (or nBPF process is denoted $\mathcal{S}$(nBPA) (or $\mathcal{S}$(nBPP)).*

- *if $\Delta_1,\ldots,\Delta_n$ are processes from nBPA$\cup$nBPP and X_i is the leading variab of Δ_i for $1 \leq i \leq n$, then $\Delta_1\|\cdots\|\Delta_n$ denotes the process $X_1\|\cdots\|X_n$ i the sense of Remark 2.6.*

- *square brackets '[' and ']' indicate optional occurrence—if we say that some expression is of the form* $a[A][B]$, *we mean that this expression is either* a, aA, aB *or* aAB.

- *upper indexes are used heavily; they appear in two forms:*

$$\alpha^i = \underbrace{\alpha\|\cdots\|\alpha}_{i}$$
$$\alpha^{\bullet i} = \underbrace{\alpha.\cdots.\alpha}_{i}$$

5.1 The Characterization of Decomposable nBPA Processes

In this section we design special normal forms for nBPA processes which allow us to characterize all decomposable nBPA processes together with their decompositions.

Definition 5.3 (prime processes). *Let* nil *be a special name for the process which cannot emit any action (i.e.,* nil $\sim \epsilon$*). A nBPA or nBPP process* Δ *is* prime *if* $\Delta \not\sim$ nil *and whenever* $\Delta \sim \Delta_1\|\Delta_2$ *we have that either* $\Delta_1 \sim$ nil *or* $\Delta_2 \sim$ nil.

Natural questions are, what processes have a decomposition into a finite parallel product of primes and whether this decomposition is unique. This problem was first examined by Milner and Moller in [MM93]. They proved that each normed finite-state process has a unique decomposition up to bisimilarity. A more general result is due to Christensen, Hirshfeld and Moller—they proved the following proposition (see [CHM93b]):

Proposition 5.4. *Let* Δ *be a nBPP process. Then* Δ *has a unique decomposition (up to bisimilarity) into a parallel product of primes.*

Proof: An existence of a finite decomposition of Δ into a parallel produc of primes is obvious—it suffices to realize that the norm is additive ove the '$\|$' operator. For uniqueness, suppose that Δ has two distinct prim decompositions given by

$$\alpha \equiv \varphi_1^{k_1}\|\cdots\|\varphi_n^{k_n}$$
$$\beta \equiv \varphi_1^{l_1}\|\cdots\|\varphi_n^{l_n}$$

where $\varphi_i \not\sim \varphi_j$ for $i \neq j$ and $|\varphi_i| \leq |\varphi_j|$ for $i \leq j$. Furthermore, assume tha Δ is a counterexample of the smallest norm, i.e., each process Δ' such tha $|\Delta'| < |\Delta|$ has a unique decomposition. Let i be the maximal number wit the property $k_i \neq l_i$. We can assume (w.l.o.g.) that $k_i > l_i$. Now we distin guish three cases, and in each case we show that process α may perform norm-reducing transition $\alpha \xrightarrow{a} \alpha'$ that cannot be matched by any transitio $\beta \xrightarrow{a} \beta'$ with $\alpha' \sim \beta'$, which will supply the desired contradiction. Observ that by minimality of the counterexample if α' and β' are to be bisimila then their prime decompositions must be identical.

- If $k_j > 0$ for some $j < i$, then α can perform some norm-reducin action via process φ_j. Process β cannot match this transition, as cannot increase the exponent l_i without decreasing the exponent o some prime with norm greater than that of φ_j.

- If $k_j > 0$ for some $j > i$, then α can perform a norm-reducing trans tion via process φ_j that maximizes (after reduction into primes) th increase in the exponent k_i. Again the process β is unable to matc this transition.

- If the process $\alpha \equiv \varphi_i^{k_i}$ is a prime power, then note that $l_j = 0$ for a $j > i$ by choice of i, and that $k_i \geq 2$ by the definition of prime. If $l_i > 0$ then β can perform a norm-reducing transition via φ_i. This transitio cannot be matched by α, because it would require the exponent k_i t decrease by at least two. On the other hand, if $l_i = 0$ then α ca perform a norm-reducing transition via φ_i and this transition cannot be matched by β, because β is unable to increase the exponent l_i.

These cases are inclusive, so the proof is finished. □

Remark 5.5. *Proposition 5.4 in fact holds for* any *normed process (namely for nBPA). The proof does not depend on a concrete syntax—it could be easily formulated in terms of normed transition systems.*

Proposition 5.4 actually says that each normed process Δ can be parallelized in the "best" way and that this way is in some sense unique. However, this nice theoretical result is non-constructive. It is not clear how to *construct* the decomposition and how to test whether some process is prime. This is the subject of next sections.

An immediate consequence of Proposition 5.4 is the following "cancelation" lemma (see [Chr93]):

Lemma 5.6. *Let $\Delta, \Gamma, \Psi, \Phi$ be normed processes such that $\Delta \| \Psi \sim \Gamma \| \Phi$ and $\Psi \sim \Phi$. Then $\Delta \sim \Gamma$.*

5.1.1 Decomposability of nBPP Processes

Each nBPP processes Δ can be easily decomposed into a parallel product of primes—all what has to be done is a construction of a bisimilar *canonical* process (see [Chr93]).

Theorem 5.7. *Let Δ be a nBPP process. It is decidable whether Δ is prime and if not, its decomposition into primes can be effectively constructed.*

Proof: By induction on $n = |\Delta|$:

- **n=1:** each nBPP process whose norm is 1 is prime.
- **Induction step:** Suppose $\Delta \sim \Delta_1 \| \Delta_2$. As Δ_1, Δ_2 are reachable states of $\Delta_1 \| \Delta_2$, there are $\alpha_1, \alpha_2 \in Var(\Delta)^{\otimes}$ such that $\Delta_1 \sim \alpha_1$ and $\Delta_2 \sim \alpha_2$, thus $\Delta \sim \alpha_1 \| \alpha_2$. Furthermore, $|\Delta| = |\alpha_1| + |\alpha_2|$. We show tha there are only finitely many candidates for α_1, α_2. First, there ar only finitely many pairs $[k_1, k_2] \in N \times N$ such that $k_1 + k_2 = |\Delta|$. Fo each such pair $[k_1, k_2]$ there are only finitely many pairs $[\beta_1, \beta_2]$ suc that $\beta_1, \beta_2 \in Var(\Delta)^{\otimes}$, $|\beta_1| = k_1$ and $|\beta_2| = k_2$. It is obvious that the se $\mathcal{M}$ of all such pairs can be effectively constructed. For each elemen $[\beta_1, \beta_2]$ of $\mathcal{M}$ we check whether $\Delta \sim \beta_1 \| \beta_2$ (it can be done becaus bisimilarity is decidable for nBPP processes). If there is no such pai then Δ is prime. Otherwise, we check whether β_1, β_2 are prime (it i possible by ind. hypothesis) and construct their decompositions. we combine obtained decompositions in parallel, we get a decompo sition of Δ. □

As each normed regular process in normal form can be seen as a nBP process in GNF, Theorem 5.7 (and especially its constructive proof) ca be used also for regular nBPA processes. In the next section we can thu concentrate on non-regular nBPA processes.

5.1.2 Decomposability of nBPA Processes

It this section we give an exact characterization of non-prime nBPA pro cesses. As we already know from the previous section, the problem i actually interesting only for non-regular nBPA processes, hence the mai characterization theorem does not concern regular nBPA processes. Ou results bring also interesting consequences—we obtain a refinement of th result achieved in [BS94] (see Remark 5.19).

The layout of this subsection is as follows: first we prove two technica lemmas (Lemma 5.8 and 5.9). Then we consider the following problem: Δ is a non-regular nBPA process such that $\Delta \sim \Delta_1 \| \Delta_2$, where Δ_1, Δ_2 ar some (unspecified) processes, how do the processes $\Delta, \Delta_1, \Delta_2$ look like? is clear that $\Delta_1, \Delta_2 \in \mathcal{S}(nBPA)$, hence the assumption that Δ_1, Δ_2 are nBP processes can be used w.l.o.g. This problem is solved by Proposition 5.12 and 5.17, with a help of several definitions. Having this, the proof of Theorem 5.23 is easy to complete.

Lemma 5.8. *Let Δ be a nBPA process. Let $\alpha, \gamma \in Var(\Delta)^+$, $Q, C \in Var(\Delta)$ such that $|Q| = |C| = 1$ and $\alpha \| Q \sim C.\gamma$. Then $\alpha \sim Q^{|\alpha|}$.*

Proof: It suffices to prove that if $\beta \| Q^i \sim C.\gamma$ where $\beta \in Var(\Delta)^+$ and $i \in N$, then $\beta \| Q^i \sim \beta' \| Q^{i+1}$ for some $\beta' \in Var(\Delta)^*$. As $|C| = 1$, all states which are reachable from $\beta \| Q^i$ in one norm-decreasing step are bisimilar. As Δ is normed, $\beta \xrightarrow{a} \beta'$ where $|\beta| = |\beta'| + 1$ and $a \in Act$. Hence $\beta \| Q^{i-1} \sim \beta' \| Q^i$ and by substitution we obtain $\beta \| Q^i \sim \beta' \| Q^{i+1}$. □

The proof of the following lemma is probably the most technical part of this chapter. Diagrams of Figure 5.1 could ease the reading.

Lemma 5.9. *Let Δ be a nBPA process, $\alpha, \beta, \gamma \in Var(\Delta)^*$ such that α is non-regular and $\alpha \| \beta \sim \gamma$. Let $\beta \rightarrow^* Q$ where $|Q| = 1$. Then $\beta \sim Q^{|\beta|}$.*

Proof: As α is non-regular, it can reach a state of an arbitrary length, i.e., for each $i \in N$ there is α' such that $\alpha \rightarrow^* \alpha'$ and $Length(\alpha') = i$. Let $m = \max\{|X|,\ X \in Var(\Delta)\}$ and let $\alpha \rightarrow^* \alpha_1$ where $Length(\alpha_1) \geq m.(|\beta|+1)$. Then $\alpha_1 \| \beta \sim \gamma_1$ for some $\gamma_1 \in Var(\Delta)^*$. As $\beta \rightarrow^* Q$, $\alpha_1 \| Q \sim \gamma_2$ where $\gamma_2 \in Var(\Delta)^*$ and $Length(\gamma_2) > 1$ — hence γ_2 is of the form $P.\omega$ where $\omega \in Var(\Delta)^+$. Let $\alpha_1 \xrightarrow{s} \alpha_2$ where s is a norm-decreasing sequence of actions such that $Length(s) = |P| - 1$. As $\alpha_1 \| Q \xrightarrow{s} \alpha_2 \| Q$ and $\alpha_1 \| Q \sim P.\omega$, $P.\omega \xrightarrow{s} C.\omega$ where $|C| = 1$ and $\alpha_2 \| Q \sim C.\omega$. Now we can apply Lemma 5.8 and conclude $\alpha_2 \sim Q^{|\alpha_2|}$. As $\alpha_1 \xrightarrow{s} \alpha_2$ where $Length(s) = |P| - 1 < m$, only the first $m - 1$ variables of α_1 could contribute to the sequence s — hence α_1, α_2 must have a common suffix whose length is at least $m.|\beta|$, i.e., $\alpha_1 = \nu.\eta$, $\alpha_2 = \delta.\eta$ where $Length(\eta) \geq m.|\beta|$. As $\alpha_1 \| \beta \sim \gamma_1$ and $\alpha_1 = \nu.\eta$, we can conclude $\eta \| \beta \sim \gamma_3$ for some $\gamma_3 \in Var(\Delta)^*$. Clearly $Length(\gamma_3) > |\beta|$, because $Length(\eta) \geq m.|\beta|$ (and thus also $|\eta| \geq m.|\beta|$) and therefore

$$
\begin{tikzcd}
\alpha \| \beta \arrow[d] & \sim & \gamma \arrow[d] \\
\alpha_1 \| \beta \arrow[d] & \sim & \gamma_1 \arrow[d] \\
\alpha_1 \| Q \arrow[d, "s"] & \sim & \gamma_2 = P.\omega \arrow[d, "s"] \\
\alpha_2 \| Q & \sim & C.\omega
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & Q^{|\eta|} \| \beta \arrow[d, "t"] \arrow[dddll, "v"'] & \sim & A_1. \cdots .A_{|\beta|+1}.\rho \arrow[d, "t"] \arrow[dd, bend left, "v"] \\
 & & Q^{|\eta|} \arrow[d, "u"] & \sim & \varphi.A_{|\beta|+1}.\rho \arrow[d, "u"] \\
Q^{|\eta|-|A_1.\cdots.A_{|\beta|}|} \| \beta & \sim & Q^{|\eta|-|\varphi|} & \sim & A_{|\beta|+1}.\rho
\end{tikzcd}
$$

Figure 5.1: Diagrams for the proof of Lemma 5.9

$|\eta \| \beta| > m.|\beta|$. Thus γ_3 is of the form $A_1. \cdots .A_{|\beta|+1}.\rho$ where $\rho \in Var(\Delta)$
Furthermore, $\eta \sim Q^{|\eta|}$ because $\alpha_2 \sim Q^{|\alpha_2|}$ and $\alpha_2 = \delta.\eta$. To sum up, w
have $Q^{|\eta|} \| \beta \sim A_1. \cdots .A_{|\beta|+1}.\rho$. Now we prove that $\beta \sim Q^{|\beta|}$. Let $\beta \xrightarrow{t}$
where $Length(t) = |\beta|$. Then $Q^{|\eta|} \| \beta \xrightarrow{t} Q^{|\eta|}$ and the state $A_1. \cdots .A_{|\beta|+1}.$
must be able to match the sequence t and enter a state bisimilar to $Q^{|\eta|}$. A
$Length(t) = |\beta|$, only the first $|\beta|$ variables of $A_1. \cdots .A_{|\beta|+1}.\rho$ can contribut
to the sequence t, i.e., $A_1. \cdots .A_{|\beta|+1}.\rho \xrightarrow{t} \varphi.A_{|\beta|+1}.\rho$ where $\varphi \in Var(\Delta)$
Now let $\varphi.A_{|\beta|+1}.\rho \xrightarrow{u} A_{|\beta|+1}.\rho$ where $Length(u) = |\varphi|$. The state $Q^{|\eta|}$ ca
match the sequence u only by removing $|\varphi|$ copies of Q — hence $Q^{|\eta|-|\varphi|}$
$A_{|\beta|+1}.\rho$. As $|\eta| \geq m.|\beta|$, it is clear that $|\eta| \geq |A_1. \cdots .A_{|\beta|}|$. Therefore the
is $v \in Act^*$, $Length(v) = |A_1. \cdots .A_{|\beta|}|$ such that $Q^{|\eta|} \xrightarrow{v} Q^{|\eta|-|A_1.\cdots.A_{|\beta|}|}$ an
thus $Q^{|\eta|} \| \beta \xrightarrow{v} Q^{|\eta|-|A_1.\cdots.A_{|\beta|}|} \| \beta$. The state $A_1. \cdots .A_{|\beta|+1}.\rho$ can match the se
quence v only by removing $A_1. \cdots .A_{|\beta|}$ — hence $Q^{|\eta|-|A_1.\cdots.A_{|\beta|}|} \| \beta \sim A_{|\beta|+1}.$
and by transitivity of bisimilarity we have $Q^{|\eta|-|\varphi|} \sim Q^{|\eta|-|A_1.\cdots.A_{|\beta|}|} \| \beta$. Fro
this we obtain $\beta \sim Q^{|\beta|}$.

Definition 5.10 (simple processes). *A nBPA process Δ is* simple *if $Var(\Delta$
contains just one variable, i.e., $card(Var(\Delta)) = 1$.

We will often identify simple processes with their leading (and only) var
ables in the rest of this chapter. Moreover, it is easy to see that a simpl

process Q is non-regular iff the def. equation for Q contains a summand of the form $aQ^{\bullet k}$ where $a \in Act$ and $k \geq 2$. The norm of Q is one, because Q could not be normed otherwise. Another important property of simple processes is presented in the remark below:

Remark 5.11. *Each simple nBPA process Q belongs to $\mathcal{S}$(nBPP)—a bisimilar nBPP process can be obtained just by replacing the '.' operator with '$\|$' operator in the def. equation for Q. Consequently, any process expressions built over k copies of Q using '.' and '$\|$' operators are bisimilar (e.g., $(Q.(Q\|Q))\|Q \sim (Q\|Q).(Q\|Q)$).*

Proposition 5.12. *Let Δ_1, Δ_2 be non-regular nBPA processes. Then $\Delta_1\|\Delta_2 \in \mathcal{S}$(nBPA) iff $\Delta_1 \sim Q^{|\Delta_1|}$ and $\Delta_2 \sim Q^{|\Delta_2|}$ for some non-regular simple process Q.*

Proof:

"$\Leftarrow$" Easy—see Remark 5.11.

"$\Rightarrow$" Assume there is some nBPA process Δ such that $\Delta_1\|\Delta_2 \sim \Delta$. Then there are $\alpha_1, \alpha_2 \in Var(\Delta)^*$ such that $\Delta_1 \sim \alpha_1$ and $\Delta_2 \sim \alpha_2$. Thus $\alpha_1\|\alpha_2 \sim \Delta$ and as α_1, α_2 are non-regular, we can use Lemma 5.9 and conclude that there are $Q_1, Q_2 \in Var(\Delta)$ such that $|Q_1| = |Q_2| = 1$, $\alpha_1 \rightarrow^* Q_1$, $\alpha_2 \rightarrow^* Q_2$ and $\alpha_1 \sim Q_1^{|\alpha_1|}$, $\alpha_2 \sim Q_2^{|\alpha_2|}$. First we prove that $Q_1 \sim Q$ for some simple process Q. To do this, it suffices to prove that if $a\gamma$ is a summand in the def. equation for Q_1, then $\gamma \sim Q_1^{\bullet|\gamma|}$. As $\alpha_1\|\alpha_2 \rightarrow^* Q_1\|\alpha_2 \xrightarrow{a} \gamma\|\alpha_2$, the process $\gamma\|\alpha_2$ belongs to $\mathcal{S}$(nBPA). Let $\gamma \rightarrow^* R$ where $|R| = 1$. Then $\gamma \sim R^{|\gamma|}$ (due to Lemma 5.9) and as $\alpha_1 \rightarrow^* \gamma \rightarrow^* R$, we also have $\alpha_1 \sim R^{|\alpha_1|}$. Hence $R \sim Q_1$ and $\gamma \sim Q_1^{|\gamma|} \sim Q_1^{\bullet|\gamma|}$.

To finish the proof, we need to show that $Q_1 \sim Q_2$. Let $m = \max\{|X|, X \in Var(\Delta)\}$. As α_1 is non-regular, it can reach a state of an arbitrary norm—let $\alpha_1 \rightarrow^* \alpha_1'$ where $|\alpha_1'| = m$. Then $\alpha_1'\|Q_2 \sim \delta$ for some $\delta \in Var(\Delta)^*$ whose length is at least two—$\delta = A.B.\delta'$. Clearly $\alpha_1' \sim Q_1^{|\alpha_1'|}$ (we can use the same argument as in the first part of this proof—Q_2 is non-regular and α' plays the role of γ), hence $Q_1^{|\alpha_1'|}\|Q_2 \sim A.B.\delta'$. As $Q_1^{|\alpha_1'|-|A|}\|Q_2 \sim B.\delta'$ and $Q_1^{|\alpha_1'|-|A|+1} \sim B.\delta'$, we have $Q_1^{|\alpha_1'|-|A|}\|Q_2 \sim Q_1^{|\alpha_1'|-|A|+1}$ by transitivity an
thus $Q_1 \sim Q_2$. □

Proposition 5.12 in fact says that if Δ is a non-regular nBPA process suc
that $\Delta \sim \Delta_1\|\Delta_2$, where Δ_1, Δ_2 are non-regular processes, then each o
those three processes can be equivalently represented as a power of som
non-regular simple process. This representation is very special and can b
seen as normal form.

If Δ is a non-regular nBPA process such that $\Delta \sim \Delta_1\|\Delta_2$, it is als
possible that Δ_1 is non-regular and Δ_2 regular. Before we start to examin
this possibility, we introduce a special normal form for nBPA processes (a
we shall see, Δ and Δ_1 can be represented in this normal form):

Definition 5.13 ($DNF(Q)$). *Let Δ be a non-regular nBPA process in GNF, Q
$Var(\Delta)$. We say that Δ is in $DNF(Q)$ if all summands in all defining equatio
from Δ are of the form $a([Y].[Q^{\bullet i}])$, where $Y \in Var(\Delta)$, $i \in N$ and $a \in Ac$
Furthermore, all summands in the def. equation for Q must be of the form $a[Q$
where $a \in Act$.*

Example 5.14. *The following process is in $DNF(Q)$:*

$$\begin{aligned} X &\stackrel{def}{=} a(Y.Q.Q) + bX + a(Q.Q.Q) + c \\ Y &\stackrel{def}{=} bQ + cX + c(Y.Q) + b \\ Q &\stackrel{def}{=} aQ + bQ + a + c \end{aligned}$$

Remark 5.15. *Reachable states of a nBPA process Δ in $DNF(Q)$ are of the for
$[Y].[Q^{\bullet i}]$ where $Y \in Var(\Delta)$ and $i \in N \cup \{0\}$. As Δ is non-regular, the state $Q^{\bullet}$
is reachable for each $k \in N$.*

Note that the variable Q itself is a regular simple process. The next lemm
says that if Δ is a process in $DNF(Q)$, then the variable Q is in some sens
unique:

Lemma 5.16. *Let Δ and Δ' be processes in $DNF(Q)$ and $DNF(R)$, respectively. If $\Delta \sim \Delta'$, then $Q \sim R$.*

Proof: Let $m = \max\{|X|,\ X \in Var(\Delta')\}$. As the state $Q^{\bullet m+1}$ is a reachable state of Δ, $Q^{\bullet m+1} \sim [Y].R^{\bullet i}$ for some $Y \in Var(\Delta')$, $i \in N$ (see Remark 5.15). Hence $Q \sim R$. □

Proposition 5.17. *Let Δ_1, Δ_2 be nBPA processes such that Δ_1 is non-regular and Δ_2 is regular. Then $\Delta_1 \| \Delta_2 \in \mathcal{S}$(nBPA) iff there is a process Δ'_1 in $DNF(Q)$ such that $\Delta_1 \sim \Delta'_1$ and $\Delta_2 \sim Q^{|\Delta_2|}$.*

Proof:
"$\Rightarrow$" Let $\Delta_2 \rightarrow^* Q'$ where $Q' \in Var(\Delta_2)$, $|Q'| = 1$. Using the same kind of argument as in the proof of Proposition 5.12 we obtain that $Q' \sim Q$ for some regular simple process Q such that $\Delta_2 \sim Q^{|\Delta_2|}$. It remains to prove that there is a process Δ'_1 in $DNF(Q)$ such that $\Delta_1 \sim \Delta'_1$. We show that each summand of each defining equation from Δ_1 can be transformed into a form which is admitted by $DNF(Q)$. First, let us realize two facts about summands—if $a\alpha$ is a summand in a def. equation from Δ_1, then

1. If $\alpha = \beta.Y.\gamma$ where Y is a non-regular variable, then each variable P of γ is bisimilar to $Q^{|P|}$.
2. α contains at most one non-regular variable.

The first fact is a consequence of Lemma 5.8—let Δ be a nBPA process such that $\Delta_1 \| \Delta_2 \sim \Delta$. As Δ_1 is normed, $\Delta_1 \rightarrow^* Y.\gamma.\delta$ for some $\delta \in Var(\Delta_1)^*$. As Y is non-regular, it can reach a state of an arbitrary length—let $m = \max\{|X|,\ X \in Var(\Delta_1)\}$ and let $Y \rightarrow^* \omega$ where $Length(\omega) = m$. As $\Delta_1 \| \Delta_2 \rightarrow^* \omega.\gamma.\delta \| Q'$, there is $\varphi \in Var(\Delta)^*$ such that $\omega.\gamma.\delta \| Q' \sim \varphi$. Let $\varphi = C.\varphi'$ and let s be a norm-decreasing sequence of actions such that $Length(s) = |C| - 1$ and $\omega \xrightarrow{s} \omega'$. Then $\omega'.\gamma.\delta \| Q' \sim C'.\varphi'$ where $|C'| = 1$ and due to Lemma 5.8 (and the fact that $Q' \sim Q$) we have $\omega'.\gamma.\delta \sim Q^{|\omega'.\gamma.\delta|}$, hence $\gamma \sim Q^{|\gamma|}$ and $P \sim Q^{|P|}$ for each variable P which appears in γ.

The second fact is a consequence of the first one—assume that $\alpha =$ $\beta.Y.\gamma.Z.\delta$ where Y, Z are non-regular. Then $Z \sim Q^{|Z|}$ and as Q is regula $Q^{|Z|}$ is regular too. Hence Z is regular and we have a contradiction.

Now we can describe the promised transformation of Δ_1 into Δ'_1: $X \stackrel{def}{=} \sum_{i=1}^{n} a_i\alpha_i$ is a def. equation in Δ_1, then $X \stackrel{def}{=} \sum_{i=1}^{n} a_i\mathcal{T}(\alpha_i)$ is a de equation in Δ'_1, where $\mathcal{T}$ is defined as follows:

- If α_i does not contain any non-regular variable, then $\mathcal{T}(\alpha_i) = A$ where A is the leading variable of $NFR(\alpha_i)$. Moreover, defining equa tions of $NFR(\alpha_i)$ are added to Δ'_1.
- If $\alpha_i = \beta.Y.\gamma$ where Y is a non-regular variable, then $\mathcal{T}(\alpha_i) = A$ where A is the leading variable of the process Δ' which is obtaine by the following modification of the process $NFR(\beta)$: each summan in each def. equation of $NFR(\beta)$ which is of the form b, where $b \in Ac$ is replaced with $b(Y.Q^{\bullet|\gamma|})$ — remember $B \sim Q^{|\gamma|} \sim Q^{\bullet|\gamma|}$. Moreove def. equations of Δ' are added to Δ'_1.

The defining equation for Q is also added to Δ'_1. The resulting process i in $DNF(Q)$ and as $\mathcal{T}$ preserves bisimilarity, $\Delta_1 \sim \Delta'_1$.

"$\Leftarrow$" We show how to construct a nBPA process Δ which is bisimilar t $\Delta'_1 \| Q^{|\Delta_2|}$. Let $k = |\Delta_2|$. The set of variables of Δ looks as follows:

$$Var(\Delta) = \{Q\} \cup \{Y_i,\ Y \in Var(\Delta'_1), Y \neq Q \text{ and } i \in \{0, \ldots, k\}\}$$

Defining equations of Δ are constructed using following rules:

- the def. equation for Q is the same as in Δ'_1
- if $a(Y.Q^j)$, where $j \in N \cup \{0\}$, $Y \neq Q$, is a summand in the def. equatio for $Z \in Var(\Delta'_1)$, then $a(Y_i.Q^j)$ is a summand in the def. equation fo Z_i for each $i \in \{0, \ldots, k\}$

- if $a(Q^j)$ where $j \in N \cup \{0\}$ is a summand in the def. equation for $Z \in Var(\Delta_1')$, then $a(Q^{j+i})$ is a summand in the def. equation for Z_i for each $i \in \{0, \ldots, k\}$
- if aQ is a summand in the def. equation for Q and $Z \in Var(\Delta_1')$, $Z \neq Q$, then aZ_i is a summand in the def. equation for Z_i for each $i \in \{1, \ldots, k\}$
- if a is a summand in the def. equation for Q and $Z \in Var(\Delta_1')$, $Z \neq Q$, then aZ_{i-1} is a summand in the def. equation for Z_i for each $i \in \{1, \ldots, k\}$

The intuition which stands behind this construction is that lower indexes of variables indicate how many copies of Q in $Q^{|\Delta_2|}$ have not disappeared yet. The fact $\Delta_1' \| Q^{|\Delta_2|} \sim \Delta$ is easy to check. □

Example 5.18. *If we apply the algorithm presented in the "⇐" part of the proof of Proposition 5.17 to the process $X \| Q^2$, where X, Q are variables of the process presented in Example 5.14, we obtain the following output:*

$$\begin{aligned}
X_2 &\stackrel{def}{=} a(Y_2.Q.Q) + bX_2 + a(Q.Q.Q.Q.Q) + c(Q.Q) + aX_2 + bX_2 + aX_1 + cX_1 \\
X_1 &\stackrel{def}{=} a(Y_1.Q.Q) + bX_1 + a(Q.Q.Q.Q) + cQ + aX_1 + bX_1 + aX_0 + cX_0 \\
X_0 &\stackrel{def}{=} a(Y_0.Q.Q) + bX_0 + a(Q.Q.Q) + c \\
Y_2 &\stackrel{def}{=} b(Q.Q.Q) + cX_2 + c(Y_2.Q) + b(Q.Q) + aY_2 + bY_2 + aY_1 + cY_1 \\
Y_1 &\stackrel{def}{=} b(Q.Q) + cX_1 + c(Y_1.Q) + bQ + aY_1 + bY_1 + aY_0 + cY_0 \\
Y_0 &\stackrel{def}{=} bQ + cX_0 + c(Y_0.Q) + b \\
Q &\stackrel{def}{=} aQ + bQ + a + c
\end{aligned}$$

Remark 5.19. *Proposition 5.17 can also be seen as a refinement of the result achieved in [BS94]—Burkart and Steffen proved that PDA processes are closed under parallel composition with finite-state processes, while BPA processes lack this property. Proposition 5.17 says precisely, what nBPA processes can remain nBPA if they are combined in parallel with a regular process. Moreover, it also characterizes all such regular processes.*

It is easy to see that the algorithm from the proof of Proposition 5.17 a ways outputs a process in $DNF(Q)$ (see Example 5.18). Moreover, th structure of this process is very specific; we can observe that each var able belongs to a special "level". This intuition is formally expressed b the following definition (it is a little complicated—but it pays because w will be able to characterize all non-prime nBPA processes):

Definition 5.20. *Let Δ be a nBPA process in $DNF(Q)$. The* level *of Δ, denote $Level(\Delta)$, is the maximal $l \in N$ such that the set $Var(\Delta) - \{Q\}$ can be divided int l disjoint linearly ordered subsets $L_1, \ldots, L_l$ of the same cardinality k. Moreove the following conditions must be true (the j^{th} element of L_i is denoted $A_{i,j}$):*

- $A_{l,1}$ is the leading variable of Δ.
- Defining equations for variables of L_1 contain only variables fror $L_1 \cup \{Q\}$
- The defining equation for $A_{i,j}$, where $i \geq 2$, $1 \leq j \leq k$, contains exactl those summands which can be derived by one of the following rule
  1. If aQ is a summand in the defining equation for Q, then $aA_{i,j}$ i a summand in the defining equation for $A_{i,j}$ for each $2 \leq i \leq$ $1 \leq j \leq k$.
  2. If a is a summand in the defining equation for Q, then $aA_{i-1,j}$ i a summand in the defining equation for $A_{i,j}$ for each $2 \leq i \leq$ $1 \leq j \leq k$.
  3. If $a(A_{1,m}.Q^{\bullet n})$ is a summand in the defining equation for A_1 such that $A_{1,m} \neq Q$, then $a(A_{i,m}.Q^{\bullet n})$ is a summand in the defir ing equation for $A_{i,j}$ for each $2 \leq i \leq l$.
  4. If $aQ^{\bullet n}$ is a summand in the defining equation for $A_{1,j}$, the $aQ^{\bullet (n+i-1)}$ is a summand in the defining equation for $A_{i,j}$, wher $2 \leq i \leq l$.

Example 5.21. *The process of Example 5.18 has the level 3; $L_1 = \{X_0, Y_0\}$, $L_2 = \{X_1, Y_1\}$ and $L_3 = \{X_2, Y_2\}$.*

Lemma 5.22. *Let Q be a non-regular simple process and let Δ be a nBPA process such that $\Delta \| Q \in \mathcal{S}(nBPA)$. Then $\Delta \sim Q^{|\Delta|}$.*

Proof: Let $\Delta \to^* R$ where $|R| = 1$. As Q is non-regular, we can use Lemma 5.9 and conclude that $\Delta \sim R^{|\Delta|}$. Now it suffices to prove that $R \sim Q$. Let Δ' be a nBPA process such that $\Delta \| Q \sim \Delta'$ and let $m = \max\{|X|,\ X \in Var(\Delta')\}$. As Q is simple and non-regular, $Q \to^* Q^{\bullet m}$ (see Remark 5.15). Hence $R \| Q^{\bullet m} \sim \alpha$ for some $\alpha \in Var(\Delta')^*$ whose length is at least 2 — thus $\alpha = A.\beta$ for some $\beta \in Var(\Delta')^+$. Let $k = |A|$. Then each two states, which are reachable from $R \| Q^{\bullet m}$ in k norm-decreasing steps are bisimilar—hence $R \| Q^{\bullet m-k} \sim Q^{\bullet m-k+1}$ and from this we have $R \sim Q$. $\square$

Now we can prove the first main theorem of this chapter:

Theorem 5.23. *Let Δ be a non-regular nBPA process and let $\Delta \sim \Delta_1 \| \cdots \| \Delta_n$, where $n \geq 2$, Δ_i is a prime process for each $1 \leq i \leq n$ and Δ_1 is non-regular. Then one of the following possibilities holds:*

- *There is a non-regular simple process Q such that $\Delta \sim Q^{\bullet|\Delta|}$ and $\Delta_i \sim Q$ for each $1 \leq i \leq n$.*
- *There are nBPA processes Δ', Δ'_1 in $DNF(Q)$ such that $\Delta \sim \Delta'$, $\Delta_1 \sim \Delta'_1$, $Level(\Delta') = n$, $Level(\Delta'_1) = 1$ and $\Delta_i \sim Q$ for each $2 \leq i \leq n$.*

Proof: We proceed by induction on n:

- **n=2:** this is an immediate consequence of Proposition 5.12 and Proposition 5.17.
- **Induction step:** let $\Delta \sim \Delta_1 \| \cdots \| \Delta_n$. As $\Delta_1 \| \cdots \| \Delta_n \to^* \Delta_1 \| \cdots \| \Delta_{n-1}$, there is a reachable state α of Δ such that $\alpha \sim \Delta_1 \| \cdots \| \Delta_{n-1}$ — hence we can use ind. hypothesis (note that α must be non-regular) and conclude that there are two possibilities:

1. There is a non-regular simple process Q such that $\Delta_i \sim Q$ fo each $1 \leq i \leq n-1$. We prove that $\Delta_n \sim Q$. As $\Delta \sim Q^{n-1} \| \Delta$ and $Q^{n-1} \| \Delta_n \to^* Q \| \Delta_n$, we can use Lemma 5.22 and conclud $\Delta_n \sim Q^{|\Delta_n|}$. Hence $\Delta_n \sim Q$ because Δ_n would not be prim otherwise.

2. There is a nBPA process Δ'_1 in $DNF(Q)$ such that $\Delta_1 \sim \Delta'$ $Level(\Delta'_1) = 1$ and $\Delta_i \sim Q$ for each $1 \leq i \leq n-1$. First w prove that $\Delta_n \sim Q$. As $\Delta_1 \| \Delta_n$ is a reachable state of $\Delta_1 \| \cdots \| \Delta$ it belongs to $\mathcal{S}(nBPA)$. Let us realize that Δ_n is regular. Assum the converse—then we can use Proposition 5.12 and conclud that $\Delta_1 \sim R^{|\Delta_1|}$ for some non-regular simple process R. Fro this and Remark 5.15 we can easily prove that $R \sim Q$ and contradicts regularity of Q.

   As Δ_n is regular and $\Delta_1 \| \Delta_2 \in \mathcal{S}(nBPA)$, we can apply Propo sition 5.17; from this (and also from Lemma 5.16) we get tha $\Delta_n \sim Q^{|\Delta_n|}$ and thus $\Delta_n \sim Q$ because Δ_n is prime.

   It remains to prove that there is a process Δ' in $DNF(Q)$ suc that $Level(\Delta') = n$ and $\Delta \sim \Delta'$. But the process Δ' can be easil constructed by the algorithm from the proof of Proposition 5.1 with $\Delta'_1 \| Q^{n-1}$ on input.

5.2 Decidability Results

In this section we present several positive decidability results. We sho that it is decidable whether a given nBPA process is prime and if the an swer is negative, then its decomposition into primes can be effectively con structed. There are also other decidable properties which are summarize in Theorem 5.28.

5.2.1 Effective Decomposability of nBPA Processes

Lemma 5.24. *Let Δ be a nBPA process. It is decidable whether there is a nBPA process Δ' in $DNF(Q)$ such that $\Delta \sim \Delta'$. Moreover, if the answer to the previous question is positive, then the process Δ' can be effectively constructed.*

Proof: We can assume (w.l.o.g.) that Δ is in 3-GNF. If there is a process Δ' in $DNF(Q)$ such that $\Delta \sim \Delta'$, then there is $R \in Var(\Delta)$ such that $R \sim Q$, because Q is a reachable state of Δ'. As Q is a regular simple process, each summand in the def. equation for R must be of the form $a[P]$, where $R \sim P$. As bisimilarity is decidable for nBPA processes, we can construct the set $\mathcal{M}$ of all variables of $Var(\Delta)$ with this property. Each variable from this set is a potential candidate for the variable which is bisimilar to Q (if the set $\mathcal{M}$ is empty, then Δ cannot be bisimilar to any process in $DNF(Q)$).

For each variable $V \in \mathcal{M}$ we now modify the process Δ slightly—we replace each summand of the form aP in the def. equation for V with aV. The resulting process is denoted Δ_V (clearly $\Delta \sim \Delta_V$). For each Δ_V we check whether Δ_V can be transformed into a process in $DNF(V)$. To do this, we first need to realize the following fact: if there is Δ'_V in $DNF(V)$ such that $\Delta_V \sim \Delta'_V$ and $a(A.B)$ is a summand in a def. equation from Δ_V such that A is non-regular, then $B \sim V^{\bullet|B|}$. It is easy to prove by the technique we already used many times in this chapter—as A is non-regular, it can reach a state of an arbitrary norm. Furthermore, there is a reachable state of Δ_V which is of the form $A.B.\gamma$ where $\gamma \in Var(\Delta_V)^*$. We choose sufficiently large α such that $A \rightarrow^* \alpha$ and $\alpha.B.\gamma$ must be bisimilar to a state of Δ'_V which is of the form $[Y].V^{\bullet i}$ where $i \geq |B.\gamma|$. From this we get $B \sim V^{\bullet|B|}$.

Now we can describe the promised transformation $\mathcal{T}$ of Δ_V into a process Δ'_V in $DNF(V)$. If this transformation fails, then there is *no* process in $DNF(V)$ bisimilar to Δ_V. $\mathcal{T}$ is invoked on each summand of each def. equation from Δ_V and works as follows:

- $\mathcal{T}(a) = a$

- $\mathcal{T}(aA) = aA$
- $\mathcal{T}(a(A.B)) = aN$ if A is regular. The variable N is the leading var able of $NFR(A)$, whose def. equations are also added to Δ'_V afte the following modification: each summand in each def. equation o $NFR(A)$ which is of the form b where $b \in Act$ is replaced with bB.
- $\mathcal{T}(a(A.B)) = a(A.V^{\bullet|B|})$ if A is non-regular and $B \sim V^{\bullet|B|}$. If A i non-regular and $B \not\sim V^{\bullet|B|}$, then $\mathcal{T}$ *fails*.

If there is $V \in \mathcal{M}$ such that $\mathcal{T}$ succeeds for Δ_V, then the process $\Delta'_V \sim \Delta$ i the process we are looking for. Otherwise, there is no process in $DNF(Q$ bisimilar to Δ. □

Proposition 5.25. *Let $\Delta_1, \ldots, \Delta_n$, $n \geq 2$ be nBPA processes. It is decidab whether $\Delta_1 \| \cdots \| \Delta_n \in \mathcal{S}$(nBPA). Moreover, if the answer to the previous ques tion is positive, then a nBPA process Δ such that $\Delta_1 \| \cdots \| \Delta_n \sim \Delta$ can be effec tively constructed.*

Proof: By induction on n:

- **n=2:** we distinguish three possibilities (it is decidable which one ac tually holds—see Remark 5.1):
  1. Δ_1 and Δ_2 are regular. Then $\Delta_1 \| \Delta_2 \in \mathcal{S}$*(nBPA)* and a bisimila regular process Δ in normal form can be easily constructed.
  2. Δ_1 and Δ_2 are non-regular. Proposition 5.12 says that there is non-regular simple process Q such that $\Delta_1 \sim Q^{|\Delta_1|} \sim Q^{\bullet|\Delta_1|}$ an $\Delta_2 \sim Q^{|\Delta_2|} \sim Q^{\bullet|\Delta_2|}$. As Q is a reachable state of $Q^{\bullet|\Delta_2|}$, ther is $R \in Var(\Delta_1)$ such that $Q \sim R$. As reachable states of Q are o the form $Q^{\bullet i}$ where $i \in N \cup \{0\}$, each summand $a\alpha$ in the de equation for R has the property $\alpha \sim R^{\bullet|\alpha|}$. As bisimilarity is de cidable for nBPA processes, we can find all variables of $Var(\Delta$

which have this property—we obtain a set of possible candidates for R (if this set is empty, then $\Delta_1\|\Delta_2 \notin \mathcal{S}(nBPA)$). Now we check whether the constructed set of candidates contains a variable R such that $\Delta_1 \sim R^{\bullet|\Delta_1|}$. If not, then $\Delta_1\|\Delta_2 \notin \mathcal{S}(nBPA)$. Otherwise we have R which is bisimilar to Q.

The same procedure is now applied to Δ_2. If it succeeds, it outputs some $S \in Var(\Delta)$. Now we check whether $R \sim S$. If not, then $\Delta_1\|\Delta_2 \notin \mathcal{S}(nBPA)$. Otherwise $\Delta_1\|\Delta_2 \in \mathcal{S}(nBPA)$ and $\Delta_1\|\Delta_2 \sim R^{\bullet|\Delta_1|+|\Delta_2|}$.

3. Δ_1 is non-regular and Δ_2 is regular (or Δ_1 is regular and Δ_2 is non-regular—this is symmetric). Due to Proposition 5.17 we know that there is a regular simple process Q and a nBPA process Δ_1' in $DNF(Q)$ such that $\Delta_1 \sim \Delta_1'$ and $\Delta_2 \sim Q^{|\Delta_2|} \sim Q^{\bullet|\Delta_2|}$. An existence of Δ_1' can be checked effectively (see Lemma 5.24). If it does not exist, then $\Delta_1\|\Delta_2 \notin \mathcal{S}(nBPA)$. If it exists, it can be also constructed and thus the only thing which remains is to test whether $\Delta_2 \sim Q^{\bullet|\Delta_2|}$. If this test succeeds, then $\Delta_1\|\Delta_2 \in \mathcal{S}(nBPA)$ and we invoke the algorithm from the proof of Proposition 5.17 with $\Delta_1'\|Q^{|\Delta_2|}$ on input—it outputs a nBPA process which is bisimilar to $\Delta_1\|\Delta_2$.

- **Induction step:** if $\Delta_1\|\cdots\|\Delta_n \in \mathcal{S}(nBPA)$, then also $\Delta_1\|\cdots\|\Delta_{n-1} \in \mathcal{S}(nBPA)$ and this is decidable by ind. hypothesis—if the answer is negative, then $\Delta_1\|\cdots\|\Delta_n \notin \mathcal{S}(nBPA)$ and if it is positive, then we can construct a nBPA process Δ' such that $\Delta_1\|\cdots\|\Delta_{n-1} \sim \Delta'$. Now we check whether $\Delta'\|\Delta_n \in \mathcal{S}(nBPA)$ and construct a bisimilar nBPA process Δ if needed. □

As an immediate consequence of Proposition 5.25 we get:

Proposition 5.26. *Let $\Delta, \Delta_1, \ldots, \Delta_n$ be nBPA processes. It is decidable whether $\Delta \sim \Delta_1\|\cdots\|\Delta_n$.*

Now it is easy to prove the following theorem:

Theorem 5.27. *Let Δ be a nBPA process. It is decidable whether Δ is prime an if not, its decomposition into primes can be effectively constructed.*

Proof: The technique is the same as in the proof of Theorem 5.7. We ca almost copy the whole proof—the crucial result which allows us to do s is Proposition 5.26. □

Decidability results which were proved in this section are summarized i the following theorem:

Theorem 5.28. *Let $\Delta, \Delta_1, \ldots, \Delta_n$ be nBPA processes. The following problem are decidable:*

- *Is Δ prime? (If not, its decomposition can be effectively constructed)*
- *Is Δ bisimilar to $\Delta_1\|\cdots\|\Delta_n$?*
- *Does the process $\Delta_1\|\cdots\|\Delta_n$ belong to $\mathcal{S}(nBPA)$?*
- *Is there any process Δ' such that $\Delta\|\Delta' \in \mathcal{S}(nBPA)$? (if so, an example o such a process can be effectively constructed).*
- *Is there any process Δ' such that $\Delta \sim \Delta_1\|\cdots\|\Delta_n\|\Delta'$? (if so, Δ' can b effectively constructed).*

5.2.2 Decidability of Bisimilarity for sPA Processes

A "structural" way how to construct new processes from older ones is t combine them together in parallel. If we do this with nBPA and nBP processes, we obtain a natural subclass of normed PA processes denote sPA (**s**imple PA processes):

Definition 5.29 (sPA processes). *The class of sPA processes is defined as follows:*

$$sPA = \{\Delta_1 \| \cdots \| \Delta_n \mid n \in N,\ \Delta_i \in nBPA \cup nBPP \text{ for each } 1 \leq i \leq n\}$$

The class sPA is strictly greater than the union of nBPA and nBPP processes. This is demonstrated by the following example:

Example 5.30. *Let Δ_1, Δ_2 be nBPA processes defined as follows:*

$$\begin{array}{llll} \Delta_1: & X \stackrel{def}{=} zX + i(Y.X) + q & \Delta_2: & A \stackrel{def}{=} aA + b(B.A) + r \\ & Y \stackrel{def}{=} i(Y.Y) + d & & B \stackrel{def}{=} b(B.B) + c \end{array}$$

Then there is no *nBPA or nBPP process bisimilar to the sPA process $\Delta_1 \| \Delta_2$. This can be easily proved with the help of pumping lemmas for context-free languages and for languages generated by nBPP processes—see [Chr93].*

Theorem 5.31. *Let $\Phi = \varphi_1 \| \cdots \| \varphi_n$, $\Psi = \psi_1 \| \cdots \| \psi_m$ be sPA processes. It is decidable whether $\Phi \sim \Psi$.*

Proof: As each φ_i, $1 \leq i \leq n$ and ψ_j, $1 \leq j \leq m$ can be effectively decomposed, we can also construct decompositions of Φ and Ψ. If $\Phi \sim \Psi$, then these decompositions must be the same up to bisimilarity (see Remark 5.5). In other words, there must be a one-to-one correspondence between primes forming the two decompositions which preserves bisimilarity. An existence of such a correspondence can be checked effectively, because bisimilarity is decidable in the union of nBPA and nBPP processes (see Theorem 4.34). □

5.3 Conclusions, Future Research

The main characterization theorem (Theorem 5.23) says that non-regular nBPA processes which are not prime can be divided into two groups:

1. Processes which can be equivalently expressed as a power of som non-regular simple process. It is obvious that each such nBPA pro cess belongs to $\mathcal{S}$*(nBPP)*—see Remark 5.11.

2. Processes which can be equivalently represented in $DNF(Q)$. It ca be proved (with the help of results achieved in Section 5.1) that eac such process does *not* belong to $\mathcal{S}$*(nBPP)*.

From this we can observe that our division based on normal forms corre sponds to the membership to $\mathcal{S}$*(nBPP)*.

It is worth mentioning that our results are also of some interest from th point of view of formal languages / automata theory. Bisimilarity coincide with language equivalence in the class of deterministic normed transitio systems. Deterministic normed BPA processes in GNF are in fact dete ministic context-free grammars. Parallel composition of processes (the '‖ operator) has also its counterpart in the theory of formal languages in th form of *shuffle* operator (see [HU79] for definition). All decidability result of Theorem 5.28 can be easily reformulated for deterministic CF gram mars, language equivalence, and shuffle.

The first possible generalization of our results could be the replacemen of the '‖' operator with the parallel operator of CCS which allows synchro nizations on complementary actions. This should not be hard, but we ca expect more complicated normal forms. Decidability results should be th same.

A natural question is whether our results can be extended to the clas of all (not necessarily normed) BPA processes. The answer is no, becaus there are quite primitive BPA processes which do not have any decompo sition at all—assume e.g., the process $X \stackrel{def}{=} aX$.

Another related open problem is decidability of bisimilarity for norme PA processes. It seems that it should be possible to design at least rich sub classes of normed PA processes where bisimilarity remains decidable.

Chapter 6

Conclusions

In this chapter we give a brief summary of main results achieved in this thesis and we also mention some major open problems.

6.1 Summary of the Main Results

In Chapter 3 we have concentrated on regularity problem. Regularity w.r.t. bisimilarity has been proved to be decidable for normed PA processes in polynomial time (Theorem 3.11). Furthermore, if a normed PA process Δ is regular, then a bisimilar finite-state process in normal form can be effectively constructed (Section 3.1.2). From this we have obtained decidability of bisimilarity for pairs of processes such that one process of this pair is a normed PA process and the other process has finitely many states (Theorem 3.14).

The notion of regularity can also be defined w.r.t. other equivalences from van Glabbeek's hierarchy. We have designed and justified new notions of finite characterization and strong regularity. Strong regularity guarantees an existence of a finite characterization in case of all equivalences from van Glabbeek's hierarchy (Theorem 3.25). Moreover, we have shown that the conditions of regularity and strong regularity express dif-

ferent features w.r.t. all equivalences from van Glabbeek's hierarchy excep
bisimilarity (Theorem 3.30).

In the last section of Chapter 3 we have extended PA processes with
finite-state control unit. As the resulting calculus (denoted PAPDA) ha
full Turing power, regularity and strong regularity are undecidable in th
class of processes (Theorem 3.34).

In Chapter 4 we have studied the relationship between sequential an
parallel compositions. The semantical intersection of nBPA_τ and nBPP
(denoted $\mathrm{nBPA}_\tau \cap \mathrm{nBPP}_\tau$) has been exactly characterized in terms of nor
mal forms $\mathrm{INF}_{\mathrm{BPP}}$ (Theorem 4.15) and $\mathrm{INF}_{\mathrm{BPA}}$ (Theorem 4.18), designed fo
nBPP_τ and nBPA_τ processes, respectively.

We have also demonstrated that the membership to $\mathrm{nBPA}_\tau \cap \mathrm{nBPP}_\tau$ i
decidable for nBPA_τ and nBPP_τ processes in polynomial time (Section 4.2
Moreover, each nBPA_τ or nBPP_τ process from $\mathrm{nBPA}_\tau \cap \mathrm{nBPP}_\tau$ can be effec
tively transformed into $\mathrm{INF}_{\mathrm{BPA}}$ or $\mathrm{INF}_{\mathrm{BPP}}$, respectively. Simplified version
of mentioned algorithms which work for nBPA and nBPP processes ha
been given too. Finally, as an immediate consequence we have obtaine
decidability of bisimilarity in the union of nBPA_τ and nBPP_τ processe
(Theorem 4.34).

The problem of effective decomposability of nBPA processes has bee
examined in Chapter 5. First, we have presented a complete character
ization of decomposable nBPA processes together with their decompos
tions by means of special normal forms (Theorem 5.23). Using this re
sult, we have shown that any nBPA process can be decomposed into
parallel product of primes effectively (Theorem 5.27), i.e. "the most para
lel" version of a given nBPA process is effectively constructible. Relate
decidability results are summarized in Theorem 5.28. Finally, we hav
demonstrated decidability of bisimilarity in a natural subclass of norme
PA processes (Theorem 5.31).

6.2 Open Problems

An interesting problem which remains open is decidability of regularity w.r.t. bisimilarity in other process classes, namely PDA and PA. This problem is at least semi-decidable (see Section 3.4), hence it suffices to establish semi-decidability of the negative subcase. Our conjecture is that regularity w.r.t. bisimilarity is in fact decidable for PDA and PA processes.

Theorem 4.34 says that bisimilarity is decidable in the union of nBPA_τ and nBPP_τ processes. However, our algorithm is exponential because it involves transformations of regular nBPA_τ (or nBPP_τ) processes into normal form. From the practical point of view it would be more interesting to obtain a better (polynomial-time) algorithm. Furthermore, one may wonder if the decidability result can be extended to the union of all (not only normed) BPA_τ and BPP_τ processes. In Section 4.3 we have mentioned that the class BPA ∩ BPP contains also processes which cannot be equivalently represented in INF. Moreover, techniques which have been used in Chapter 4 cannot be applied, hence the characterization of BPA ∩ BPP seems to be a more complicated task.

Naturally, it would be nice to compare other classes of behaviours which are generated by different types of syntax, e.g., Petri nets and BPA. A "complete" result should contain an exact characterization of the "semantical intersection" and two (constructive) algorithms which can decide the membership to the intersection for both types of syntax (and possibly construct an equivalent description in the other syntax).

A prime decomposition of a process Δ expresses all internal concurrency of Δ explicitly—the problem of effective decomposability is thus especially interesting in process classes which contain sequential behaviours. It would be nice to obtain some positive results for e.g., normed PDA processes.

The problem of effective decomposability is also related to decidability of bisimilarity in various process classes. For example, if bisimilarity is

decidable for normed PA processes, then normed PA processes can be effectively decomposed. On the other hand, we have obtained decidability of bisimilarity for sPA processes (Theorem 5.31) as a simple consequence of effective decomposability of nBPA and nBPP processes.

Bibliography

[BBK87] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proceedings of PARLE'87*, volume 259 of *LNCS*, pages 93–114. Springer-Verlag, 1987.

[BBK93] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the Association for Computing Machinery*, 40:653–682, 1993.

[BCS96] O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96* [Con96], pages 247–262.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. Verifying infinite state processes with sequential and parallel composition. In *Proceedings of POPL'95*, pages 95–106. ACM Press, 1995.

[BG96] D.J.B. Bosscher and W.O.D. Griffionen. Regularity for a large class of context-free processes is decidable. In *Proceedings of ICALP'96* [Ica96], pages 182–192.

[BK88] J.A. Bergstra and J.W. Klop. Process theory based on bisimulation semantics. In *Advanced Topics in Artificial Intelligence*, volume 345 of *LNCS*, pages 50–122. Springer-Verlag, 1988.

[Bla95] J. Blanco. Normed BPP and BPA. In *Proceedings of ACP'9* Workshops in Computing, pages 242–251. Springer-Verla 1995.

[BS94] O. Burkart and B. Steffen. Pushdown processes: Parallel com position and model checking. In *Proceedings of CONCUR'9* [Con94], pages 98–113.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 1 in Cambridge Tracts in Theoretical Computer Science. Cam bridge University Press, 1990.

[Cau88] D. Caucal. Graphes canoniques de graphes algebriques. Rap port de Recherche 872, INRIA, 1988.

[CHM93a] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is de cidable for all basic parallel processes. In *Proceedings of CON CUR'93*, volume 715 of *LNCS*, pages 143–157. Springer-Verla 1993.

[CHM93b] S. Christensen, Y. Hirshfeld, and F. Moller. Decomposabilit decidability and axiomatisability for bisimulation equivalenc on basic parallel processes. In *Proceedings of LICS'93*. IEE Computer Society Press, 1993.

[Chr93] S. Christensen. *Decidability and Decomposition in Process Alge bras*. PhD thesis, The University of Edinburgh, 1993.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equiv alence is decidable for all context-free processes. In *Proceeding of CONCUR'92*, volume 630 of *LNCS*, pages 138–147. Springe Verlag, 1992.

[ČKK96] I. Černá, M. Křetínský, and A. Kučera. Bisimilarity is decidab in the union of normed BPA and normed BPP processes. I

Proceedings of INFINITY'96, MIP-9614, pages 32–46. University of Passau, 1996.

[CM90] D. Caucal and R. Monfort. On the transition graphs of automata and grammars. In *Graph-Theoretic Concepts in Computer Science*, volume 484 of *LNCS*, pages 311–337. Springer-Verlag, 1990.

[Con94] *Proceedings of CONCUR'94*, volume 836 of *LNCS*. Springer-Verlag, 1994.

[Con96] *Proceedings of CONCUR'96*, volume 1119 of *LNCS*. Springer-Verlag, 1996.

[Flo67] R.W. Floyd. Assigning meanings to programs. In *Mathematical Aspects of Computer Science. Proc. Symp. Appl. Math., 19*, pages 19–32. American Math. Society, 1967.

[Gro91] J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters*, 42:167–171, 1991.

[HJM94a] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. Technical report ECS-LFCS-94-286, Department of Computer Science, University of Edinburgh, 1994.

[HJM94b] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes. Technical report ECS-LFCS-94-288, Department of Computer Science, University of Edinburgh, 1994.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[HS91] H. Hüttel and C. Stirling. Actions speak louder than word Proving bisimilarity for context-free processes. In *Proceeding of LICS'91*, pages 376–386. IEEE Computer Society Press, 199

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theor Languages, and Computation*. Addison-Wesley, 1979.

[Ica96] *Proceedings of ICALP'96*, volume 1099 of *LNCS*. Springe Verlag, 1996.

[Jan94] P. Jančar. Decidability questions for bisimilarity of Petri ne and some related problems. In *Proceedings of STACS'94*, vo ume 775 of *LNCS*, pages 581–592. Springer-Verlag, 1994.

[Jan97] P. Jančar. Bisimulation equivalence is decidable for on counter processes. To appear in Proc. of ICALP'97. LNC Springer-Verlag, 1997.

[JE96] P. Jančar and J. Esparza. Deciding finiteness of Petri nets up t bisimilarity. In *Proceedings of ICALP'96* [Ica96], pages 478–48

[JM95] P. Jančar and F. Moller. Checking regular properties of Pet nets. In *Proceedings of CONCUR'95*, volume 962 of *LNCS*, page 348–362. Springer-Verlag, 1995.

[Kuč95] A. Kučera. Deciding regularity in process algebras. BRICS R port Series RS-95-52, Department of Computer Science, Un versity of Aarhus, October 1995.

[Kuč96a] A. Kučera. Regularity is decidable for normed BPA an normed BPP processes in polynomial time. In *Proceedings SOFSEM'96*, volume 1175 of *LNCS*, pages 377–384. Springe Verlag, 1996.

[Kuč96b] A. Kučera. Regularity is decidable for normed PA processes in polynomial time. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 111–122. Springer-Verlag, 1996.

[Kuč97] A. Kučera. How to parallelize sequential processes. To appear. In *Proceedings of CONCUR'97*, LNCS. Springer-Verlag, 1997.

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

[Min67] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[MM93] R. Milner and F. Moller. Unique decomposition of processes. *Theoretical Computer Science*, 107(2):357–363, 1993.

[MM94] S. Mauw and H. Mulder. Regularity of BPA-systems is decidable. In *Proceedings of CONCUR'94* [Con94], pages 34–47.

[Mol96] F. Moller. Infinite results. In *Proceedings of CONCUR'96* [Con96], pages 195–216.

[MS85] D.E. Muller and P.E. Schupp. The theory of ends, pushdown automata, and second order logic. *Theoretical Computer Science*, 37(1):51–75, 1985.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[Plo81] G. Plotkin. A structural approach to operational semantics. Technical Report Daimi FN-19, Department of Computer Science, University of Aarhus, 1981.

[Rei85] W. Reisig. *Petri Nets—An Introduction*. Springer-Verlag, 1985.

[Sch86] D.A. Schmidt. *Denotational Semantics*. Allyn and Bacon, Inc 1986.

[Sch92] S.R. Schwer. The context-freeness of the languages associate with vector addition systems is decidable. *Theoretical Compute Science*, 98(2):199–247, 1992.

[Sti92] C. Stirling. Modal and temporal logics. In S. Abramsk D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Com puter Science*, volume I. Oxford University Press, 1992.

[Sti96] C. Stirling. Decidability of bisimulation equivalence fo normed pushdown processes. In *Proceedings of CONCUR'9* [Con96], pages 217–232.

[SW89] C. Stirling and D. Walker. Local model checking in the moda mu-calculus. In *Proceedings of TAPSOFT'89, I*, volume 351 o *LNCS*, pages 369–383. Springer-Verlag, 1989.

[Tau89] D. Taubner. *Finite Representations of CCS and TCSP Program by Automata and Petri Nets*. Number 369 in LNCS. Springe Verlag, 1989.

[vG90] R.J. van Glabbeek. The linear time—branching time spectrum In *Proceedings of CONCUR'90*, volume 458 of *LNCS*, pages 278 297. Springer-Verlag, 1990.

[vGW89] R.J. van Glabbeek and W.P. Weijland. Branching time and ab straction in bisimulation semantics. *Information Processing 8* pages 613–618, 1989.

Appendix A

Behavioural Equivalences

In this appendix we present definitions of behavioural equivalences of van Glabbeek's hierarchy. Here we adopt the definition of transition system from Section 2.1, i.e., $T = (S, Act, \to, r)$. If $s \in S$, then

$$I(s) = \{a \in Act \mid \exists t \in S \text{ such that } s \xrightarrow{a} t\}$$

denotes the set of initial actions of s. Furthermore, $\mathcal{P}(M)$ denotes the power-set of M.

Definition A.1 (Trace equivalence). *Let T be a transition system. We define the set of* traces *of T in the following way:*

$$tr(T) = \{w \in Act^* \mid \exists s \in S \textit{ such that } r \xrightarrow{w} s\}$$

Transition systems T_1, T_2 are trace equivalent, *written $T_1 =_{tr} T_2$, if $tr(T_1) = tr(T_2)$.*

Definition A.2 (Completed trace equivalence). *Let T be a transition system. We define the set of* completed traces *of T in the following way:*

$$ct(T) = \{w \in Act^* \mid \exists s \in S \textit{ such that } r \xrightarrow{w} s \textit{ and } I(s) = \emptyset\}$$

Transition systems T_1, T_2 are completed trace equivalent, *written $T_1 =_{ct} T_2$, if $tr(T_1) = tr(T_2)$ and $ct(T_1) = ct(T_2)$.*

Definition A.3 (Failure equivalence). *Let T be a transition system. A pa $(w, \Phi) \in Act^* \times \mathcal{P}(Act)$ is a* failure pair *of T, if there is a state $s \in S$ such tha $r \xrightarrow{w} s$ and $I(s) \cap \Phi = \emptyset$. Let $F(T)$ denote the set of all failure pairs of T. Transitio systems T_1, T_2 are* failure equivalent, *written $T_1 =_f T_2$, if $F(T_1) = F(T_2)$*

Definition A.4 (Readiness equivalence). *Let T be a transition system. pair $(w, \Phi) \in Act^* \times \mathcal{P}(Act)$ is a* ready pair *of T, if there is a state $s \in S$ such tha $r \xrightarrow{w} A$ and $I(A) = \Phi$. Let $R(T)$ denote the set of all ready pairs of T. Transitio systems T_1, T_2 are* readiness equivalent, *written $T_1 =_r T_2$, if $R(T_1) = R(T_2)$.*

Definition A.5 (Failure trace equivalence). *Let T be a transition system. Th* refusal relations *$\xrightarrow{\Phi}$ for $\Phi \in \mathcal{P}(L)$ are definined by:*

$$A \xrightarrow{\Phi} B \textit{ iff } A = B \textit{ and } I(A) \cap \Phi = \emptyset$$

The failure trace relations *$\xrightarrow{\delta}$ for $\delta \in (L \cup \mathcal{P}(L))^*$ are defined as the reflexiv and transitive closure of both the transition and the refusal relations. $\delta \in (Act \cup$ $\mathcal{P}(Act))^*$ is a* failure trace *of T, if there is a state $s \in S$ such that $r \xrightarrow{\delta} s$. Le $FT(T)$ denote the set of failure traces of T. Transition systems T_1, T_2 are* failur trace equivalent, *written $T_1 =_{ft} T_2$, if $FT(T_1) = FT(T_2)$.*

Definition A.6 (Ready trace equivalence). *Let T be a transition system. Th* ready trace relations *$\overset{\delta}{\Rightarrow}$ for $\delta \in (Act \cup \mathcal{P}(Act))^*$ are defined inductively by:*

1. *$s \overset{\epsilon}{\Rightarrow} s$ for any $s \in S$.*

2. *$s \xrightarrow{a} t$ implies $s \overset{a}{\Rightarrow} t$.*

3. *$s \overset{\Phi}{\Rightarrow} t$ with $\Phi \in \mathcal{P}(Act)$ whenever $s = t$ and $I(s) = \Phi$.*

4. *$s \overset{\delta}{\Rightarrow} t \overset{\rho}{\Rightarrow} u$ implies $s \overset{\delta\rho}{\Rightarrow} u$.*

$\delta \in (Act \cup \mathcal{P}(Act))^*$ is a ready trace *of T if there is a state $s \in S$ such that $r \overset{\delta}{\Rightarrow}$ Let $RT(T)$ denote the set of ready traces of T. Transition systems T_1, T_2 are* read trace equivalent, *written $T_1 =_{rt} T_2$, if $RT(T_1) = RT(T_2)$.*

Definition A.7 (Simulation equivalence). *Let T_1, T_2 be transition systems. A binary relation $R \subseteq S_1 \times S_2$ is a* simulation *if whenever $s_1 R s_2$ then*

$$\forall a \in Act_1 : \; s_1 \xrightarrow{a} s_1' \;\Rightarrow\; \exists s_2' : \; s_2 \xrightarrow{a} s_2' \wedge s_1' R s_2'$$

Transition systems T_1, T_2 are simulation equivalent, *written $T_1 =_s T_2$, if there exists a simulation R with $r_1 R r_2$ and a simulation S with $r_2 S r_1$.*

Definition A.8 (Ready simulation equivalence). *Let T_1, T_2 be transition systems. A binary relation $R \subseteq S_1 \times S_2$ is a* ready simulation *if whenever $s_1 R s_2$ then:*

- $\forall a \in Act_1 : \; s_1 \xrightarrow{a} s_1' \;\Rightarrow\; \exists s_2' : \; s_2 \xrightarrow{a} s_2' \wedge s_1' R s_2'$
- $I(s_1) = I(s_2)$

Transition systems T_1, T_2 are ready simulation equivalent, *written $T_1 =_{rs} T_2$, if there exists a ready simulation R with $r_1 R r_2$ and a ready simulation S with $r_2 S r_1$.*

Definition A.9 (Possible futures equivalence). *Let T be a transition system. A pair $(w, \Phi) \in Act^* \times \mathcal{P}(Act^*)$ is a* possible future *of T if there is a state $s \in N$ such that $r \xrightarrow{w} s$ and $tr(s) = \Phi$. The set of all possible futures of T is denoted $PF(T)$. Transition systems T_1, T_2 are* possible-futures equivalent, *written $T_1 =_{pf} T_2$, if $PF(T_1) = PF(T_2)$.*

Definition A.10 (2-nested simulation equivalence). *Let T_1, T_2 be transition systems. A binary relation $R \subseteq S_1 \times S_2$ is a* 2-nested simulation *if whenever $s_1 R s_2$ then*

- $\forall a \in Act_1 : \; s_1 \xrightarrow{a} s_1' \;\Rightarrow\; \exists s_2' : \; s_2 \xrightarrow{a} s_2' \wedge s_1' R s_2'$
- $s_1 =_s s_2$

Transition systems T_1, T_2 are 2-nested simulation equivalent, *written $T_1 =_2 T_2$, if there exists a 2-nested simulation R with $r_1 R r_2$ and a 2-nested simulation S with $r_2 S r_1$.*