View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Edinburgh Research Archive

[chapter] [chapter] [chapter] [chapter] [chapter] [chapter] [chapter]

Decidability and complexity of equivalences for simple process algebras

Jitka Stříbrná

Doctor of Philosophy
University of Edinburgh
1998

Abstract

In this thesis I study decidability, complexity and structural properties of strong and weak bisimilarity with respect to two process algebras, Basic Process Algebras and Basic Parallel Process Algebras.

The decidability of strong bisimilarity for both algebras is an established result. For the subclasses of normed BPA-processes and BPP there even exist polynomial decision procedures. The complexity of deciding strong bisimilarity for the whole class of BPP is unsatisfactory since it is not bounded by any primitive recursive function. Here we present a new approach that encodes BPP as special polynomials and expresses strong bisimulation in terms of polynomial ideals and then uses a theorem about polynomial ideals (Hilbert's Basis Theorem) and an algorithm from computer algebra (Gröbner bases) to construct a new decision procedure.

For weak bisimilarity, Hirshfeld found a decision procedure for the subclasses of totally normed BPA-processes and BPP, and Esparza demonstrated a semidecision procedure for general BPP. The remaining questions are still unsolved. Here we provide some lower bounds on the computational complexity of a decision procedure that might exist. For BPP we show that the decidability problem is NP-hard (even for the class of totally normed BPP), for BPA-processes we show that the decidability problem is PSPACE-hard.

Finally we study the notion of weak bisimilarity in terms of its inductive definition. We start from the relation containing all pairs of processes and then form a non-increasing chain of relations by eliminating pairs that do not satisfy a certain expansion condition. These relations are labelled by ordinal numbers and are called approximants. We know that this chain eventually converges for some α such that $\approx_\alpha = \approx_\beta = \approx$ for all $\alpha < \beta$. We study the upper and lower bounds on such ordinals α. We prove that for BPA, $\alpha \geq \omega^\omega$, and for BPPA, $\alpha \geq \omega \cdot 2$. For some restricted classes of BPA and BPPA we show that $\approx = \approx_{\omega \cdot 2}$.

Acknowledgements

I would like to express my gratitude to Mark Jerrum for his advice and patience, and for all the time that he devoted to me. I have learnt a great deal under his supervision.

Special thank-you goes to John Power for numerous discussions and chats, and for his friendly encouragement throughout my doctorate.

My examiners, Julian Bradfield and Javier Esparza, read the thesis very carefully. I am grateful for their comments and suggestions that helped to improve the final version.

Finally, I would like to thank my family and all my friends.

My stay in Edinburgh was sponsored by a scholarship from the Wolfson Foundation.

Declaration

I declare that this thesis was composed by myself, and the work contained in it is my own, unless stated otherwise.

The results of Chapter 5 were published in [59], and the results of Chapter 3 are contained in [60].

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The area of concurrency has been widely studied in recent years. The need for a theory of concurrent systems comes from the desire to find appropriate theoretical descriptions of real systems so that we can use some mathematical tools in proving properties about systems that are of practical interest. In system design, one wants to specify the properties that the system should possess, and then verify whether the outcome satisfies the given requirements.

Process algebras or process calculi have become a popular tool for describing both systems and their formal specifications. Verifying that a system satisfies a given specification can be done by checking whether the two corresponding descriptions (processes) are equivalent. There are many different process calculi, which vary in the constructions that are considered primitive. There are also various notions of equivalence. One of the most popular calculi is the Calculus of Communicating Systems, and among the favoured equivalences are strong and weak bisimulations.

1.1 Calculus of Communicating Systems

One of the most influential works in the area of process calculi has been Milner's *Calculus of Communicating Systems (CCS)*, which originally appeared in [48] and later as a revised version in [49]. The calculus is built around a few simple operators: we start with a distinguished set of *actions* $Act = \{a, b, c, \ldots, \bar{a}, \bar{b}, \bar{c}, \ldots\} \cup \{\tau\}$, where each action a has a *complement* $\bar{a}$ and $\bar{\bar{a}} = a$, with the exception of the *silent* action τ. The intended meaning is that processes can *synchronise* on complementary actions giving the action τ. The basic operators are: *action prefix*, *summation* over an arbitrary set, *(parallel) composition*, *restriction*, and *relabelling*. The semantics of the process expressions is given in terms of *labelled transition systems*, i.e. graphs whose edges are labelled by actions from Act.

A process P prefixed with an action a can perform a and evolve into P. A sum of processes $\sum_{i \in I} P_i$ can nondeterministically choose to start behaving as any summand P_j thus discarding all other processes P_i for $i \neq j$. A process $P \mid Q$ obtained as composition of two processes P and Q can evolve into either $P' \mid Q$ by performing some transition of P, or $P \mid Q'$ by performing some transition of Q, or it can synchronise on complementary actions a of P and $\bar{a}$ of Q and perform τ to become $P' \mid Q'$. Restriction is specified by a set of actions $L \subseteq Act$ and it forbids a process from performing an action a if $a \in L$ or $\bar{a} \in L$. And finally, relabelling is given by a function f on actions with the convention that $\overline{f(a)} = f(\bar{a})$ and $f(\tau) = \tau$, which results in renamed behaviour of processes.

The combination of composition, restriction and relabelling makes the calculus very powerful since it enables us to define processes that are encodings of Turing machines. That means the calculus is very expressive. On the other hand there are some disadvantages of the expressiveness since in general we will not be able to test whether two processes are equivalent for interesting notions of equivalence.

1.2 Equivalences

Now we will concentrate on the question of under which circumstances two processes will be considered equivalent. We want to distinguish between two processes P and Q if there is a difference that can be detected by another process that might interact with either of them. As an example, let us first consider an equivalence much favoured in automata theory that is *language equivalence*. We call two automata *language equivalent* if they accept identical strings of symbols. However, we can see that this equivalence is not suitable for concurrent processes, as illustrated by the following example.

Example 1.1 We consider processes P and Q whose derivation trees are shown in Fig. 1.1. The languages of P and Q are identical and are given as $L(P) = L(Q) = \{ab, ac\}$. Therefore the processes P and Q are language-equivalent. However, Q, after having performed an action a loses the choice between b and c since it evolves into a state which can only perform one of b or c. That contrasts with the process P which after the a action can choose between b and c. If there was another process R that could perform the action $\bar{b}$ then it would always be able to synchronise with P after P performs an a but it may not be able to synchronise with Q. □

This example demonstrates that language equivalence is too weak for concurrent systems as it does not take into account the branching of processes. There are

Figure 1.1: Language-equivalent processes P and Q

many ways of overcoming this problem and many different notions of process equivalences. The notion of *bisimulation equivalence* is among the most important ones. The basic idea is that two processes will be considered equivalent if they can match each other's transitions and by doing so evolve into processes which are again equivalent. This idea was suggested by Park in [55]. Following Park, Milner used this concept in the definition of bisimilarity in [49] which was originally defined as a limit of a decreasing chain of equivalences [48]. We spell out the definition in **Chapter 2**, meanwhile we will illustrate the concept by means of an example.

Example 1.2 We consider processes P, Q, R and S as depicted in Fig. 1.2.

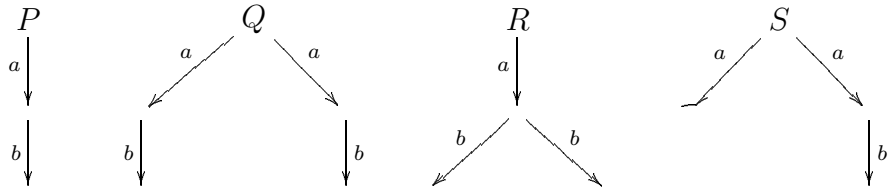


Figure 1.2: Various processes

The processes P, Q and R are all bisimilar since they possess the property that after an a action they have always a b action at their disposal. That is not true of the process S which can choose one branch so that no further action is available. We will also explain why the two processes P and Q from Fig. 1.1 are not bisimilar. The reason is that P can always choose between the actions b and c after having done an a whereas Q determines which of b and c will be available by the choice of the a transition. □

Bisimulation became one of the pivotal notions in concurrency theory. It is a simple and elegant notion with appealing mathematical properties. Bisimulation

is an equivalence relation and its definition in [49] lends itself to an elegant technique of proving bisimilarity simply by providing a binary relation on processes that satisfies a certain closure property. However, in some situations it may prove too discriminative since it requires a matching response to each transition. That may be rather inconvenient in the treatment of the silent action τ since in some circumstances we may want to abstract away from silent behaviour. The importance of the silent action is to hide the inner behaviour of a process from its environment, only conveying a possible change of state to an outside observer. We will define a less strict equivalence that still requires a matching response to any *visible* (non-τ) action which can however include any number of τ transitions. To this end we introduce a *weak derivative* of a process which is obtained by performing an action preceeded and followed by arbitrary finite sequences of τ transitions. Two processes are then equivalent if they have matching weak derivatives that are again equivalent. This equivalence is called *weak bisimulation equivalence* (see Milner in [49]).

Weak bisimulation identifies more processes than strong bisimulation. It is consistent with strong bisimulation, i.e. any pair of processes that are strongly bisimilar are also equivalent under weak bisimulation. However, the opposite does not hold which may be illustrated by the following example: we take a process P which can only do a single action a, and define another process Q as P prefixed with τ. These two processes are not equivalent under strong bisimulation but they are weakly bisimilar. That captures the idea that a weakly bisimilar process may be involved in some inner behaviour, which we do not want to consider, however all the observable actions match those of the other process.

1.3 Decidability of bisimulation equivalences

We would like to be able to test, given a pair of processes and a specific equivalence, whether the processes are related by that equivalence. We may not always be able to do that satisfactorily. The key problem is whether a given equivalence is *decidable* for a particular class of processes that we want to consider. This is an important practical consideration if we want to use an equivalence for design and implementation of systems. So the question we want to ask ourselves is: Can we decide for a given pair of processes whether they are equivalent?

The answer is of course relative to the class of processes and the equivalence we have in mind. For the class of CCS processes and strong/weak bisimulation, the answer is unfortunately negative. As Christensen demonstrated in his the-

sis [6], already the subclass of CCS obtained by application of finite summation, parallel composition and restriction has full Turing power. He proceeds by encoding *two-counter machines* [51] by such processes. Then he reduces the Halting problem of two-counter machines to bisimilarity. Since the Halting problem for two-counter machines is undecidable one can conclude that in general, strong and weak bisimilarity for CCS processes is undecidable.

Another indication of the power that the combination of parallel composition and restriction possesses is showed in [8]. The result presented there states that if we disallow either of parallel composition or restriction and relabelling, we arrive at a subclass of processes on which bisimilarity is decidable. A summary of decidability results for various classes of processes also appears in [53].

1.3.1 Strong bisimilarity

We will investigate the classes of processes for which bisimilarity is decidable, and their expressiveness. It is a folklore result that if we restrict our attention to finite state processes we can decide any reasonable equivalence simply by checking the equivalence condition for finitely many possible combinations of states. But finite state processes are not expressive enough as the description of many systems gives rise to infinite state derivation trees.

In this thesis we will concentrate on two simple classes of possibly infinite state processes. We will avoid the operators of restriction and relabelling altogether. The processes in our process calculi will be obtained from a set of variables by application of the operators of action prefix and finite summation combined with sequential and parallel composition.

The first class of algebras we will consider are *Basic Process Algebras (BPA)*. The processes of a Basic Process Algebra are called *BPA-processes*. The main operators of Basic Process Algebras are finite summation and *sequential composition*. Sequential composition is not among the basic operators of CCS. Its semantics is defined in this way: sequential composition of processes P and Q is a process $P \cdot Q$ which behaves as the process P until termination upon which it behaves as the process Q.

Basic process algebras arose from language theory as the process equivalent of *context-free grammars*. This connection was described by Baeten, Bergstra and Klop in [2] which also contains the first decidability result for BPA, an algorithm deciding strong bisimilarity for the restricted subclass of *normed* BPA-processes. Other results followed with [5], [37], [20], and [38]. Eventually, a polynomial algorithm for normed BPA-processes was demonstrated by Hirshfeld, Jerrum and

Moller in [31]. Decidability for the class of general BPA-processes was established by Christensen, Hüttel and Stirling in [10], and an elementary (estimated as doubly exponential) decision algorithm was presented by Burkart, Caucal and Steffen in [4].

The second class of algebras that we will investigate are *Basic Parallel Process Algebras (BPPA)*. The processes of a BPPA are called *Basic Parallel Processes (BPP)*. The class of BPPA was conceived by Christensen in his doctoral thesis [6]. It is a simple subclass of CCS which only contains the operators of finite summation and *merge*. Merge is a restricted form of parallel composition where no synchronisation occurs, that is the τ action is prescribed explicitly and cannot occur as a result of synchronisation on two complementary actions. The class of Basic Parallel Processes also arises as a simple subclass of Petri nets, so called *communication-free* Petri nets.

The first decidability result for strong bisimilarity of BPP was a decision procedure for *normed* and *live* BPP by Christensen, Hirshfeld and Moller [7] which was later extended to the whole class of BPP by the same authors in [8]. Another decision technique for the class of general BPP was presented by Hirshfeld [27]. For normed BPP, Hirshfeld, Jerrum and Moller constructed a polynomial algorithm deciding bisimilarity [29], [30].

The decidability results for strong bisimilarity on basic process algebras and Basic Parallel Process Algebras are summarised in the table below:

	BPA	normed BPA	BPPA	normed BPPA
strong bisimilarity $\sim$	decidable *doubly exponential upper bound*	decidable *polynomial algorithm*	decidable *no primitive recursive upper bound*	decidable *polynomial algorithm*

Table 1.1: The summary of decidability results for strong bisimilarity

1.3.2 Weak bisimilarity

The problem of deciding weak bisimilarity seems to be harder than deciding strong bisimilarity. Weak bisimulation has an additional complex aspect which is the possibility of infinite branching of processes. When we restrict ourselves to finite

summation then all processes have only finitely many possible derivatives with respect to strong bisimulation. With weak bisimulation we allow the possibility to evolve into potentially infinitely many derivatives with a single (weak) transition.

We have already mentioned that weak bisimilarity for CCS is undecidable due to the fact that we can encode the Halting problem for two-counter machines to strong (weak) bisimilarity of CCS. We have also mentioned that for the classes of BPA and BPPA, strong bisimilarity is decidable. However, we do not yet know exactly what is the situation like for weak bisimilarity and BPA/BPPA.

So far, there are only partial results available. We can single out the subset of *totally normed* BPA-processes and *totally normed* BPP (they can terminate by performing at least one visible action) for which Hirshfeld showed weak bisimilarity to be decidable [28]. For totally normed BPA-processes, decidability of weak bisimilarity can be also derived from Stirling's proof of decidability of strong bisimilarity for *normed pushdown automata* [58]. When we consider the general processes, nothing is known about decidability for BPA. For the class of general BPP, Esparza demonstrated a semidecision procedure in [16], [15].

The decidability results for weak bisimilarity on basic process algebras and Basic Parallel Process Algebras are summarised in the table below:

	BPA	totally normed BPA	BPPA	totally normed BPPA
weak bisimilarity $\approx$	not known	decidable	not known	decidable

Table 1.2: The summary of decidability results for weak bisimilarity

1.4 Computational complexity of equivalences

Several problems arise in the current situation. We may want to investigate weak bisimulation equivalence for BPA and BPPA to complete the picture and establish whether weak bisimilarity is decidable or undecidable for the whole classes. As for strong bisimilarity, the results for normed BPA-processes and normed BPP are satisfactory because of the existence of polynomial decision procedures. For general BPA-processes and BPP the situation is not completely clear. We assume that the computational complexity of an algorithm deciding bisimilarity might be

greater for general processes compared with the special subclass of normed processes. However, so far there is no indication that a polynomial algorithm cannot exist for BPA-processes and BPP in general, that is we do not have any lower bounds on the complexity. When we examine the upper bounds on the complexity we discover a wide gap between the two algebras. The best algorithm for deciding strong bisimilarity of BPA-processes so far runs in estimated doubly exponential time [4] whereas we do not even know of a primitive recursive function which would serve as an upper bound on the complexity of deciding strong bisimilarity of BPP.

In this thesis we try to investigate the complexity of deciding strong bisimilarity for Basic Parallel Processes. We describe a new link between Basic Parallel Process Algebras and classical algebra of polynomials. We express BPP and strong bisimulations in terms of polynomials and polynomial ideals, and devise a condition that enables us to test bisimilarity by testing membership in polynomial ideals. We use some tools of computer algebra designed to test polynomial ideal membership.

We have mentioned above that the problem of deciding weak bisimilarity has not been satisfactorily resolved. In this thesis we study weak bisimulation from two different points of view. We investigate the *hardness* of the weak bisimilarity decision problem, and we study the *structural complexity* of weak bisimulation equivalence in a sense that will be explained shortly.

Hirshfeld in [28] demonstrates decidability of weak bisimilarity for totally normed BPA-processes and totally normed BPP but does not place any estimates on the complexity of the decision procedures. For BPP we face the same problem as in the case of the strong equivalence and with the current techniques no complexity bound can be given. That leaves open a whole range of possibilities. Although it is quite unlikely, so far there are no negative results which would prove that weak bisimilarity cannot be tested in polynomial time.

1.4.1 Hardness of weak bisimilarity

We can by means of a *reduction* convince ourselves about the hardness of a problem, that is a lower bound on the complexity of all algorithms that decide that problem. The reduction technique consists in transforming one problem $\mathcal{P}$ effectively to another problem $\mathcal{Q}$, where the time or space complexity of the problem $\mathcal{P}$ is known. If we make sure that the transformation is *efficient* and transforms instances of $\mathcal{P}$ into equivalent instances of $\mathcal{Q}$ of roughly the same size, then we can deduce that deciding $\mathcal{Q}$ must be as hard as deciding $\mathcal{P}$. Using this technique we

manage to show that deciding weak bisimilarity for totally normed BPA-processes and totally normed BPP is NP-hard. Under the widely accepted conjecture that P $\neq$ NP (that is the class of problems decidable in polynomial time by deterministic Turing machines is strictly smaller than the class of problems decidable in polynomial time by nondeterministic Turing machines) we can conclude that the decision problem for weak bisimilarity is not polynomial.

For general BPA-processes, it still may be the case that weak bisimilarity is undecidable. However, we can construct a reduction which will show that weak bisimilarity is PSPACE-hard for BPA-processes. The class PSPACE is defined as the class of all problems decidable by polynomial-space bounded Turing machines. It is assumed that the class NP is strictly contained in PSPACE and hence this constitutes a stronger result. To our knowledge, these results are the first attempt in the direction of estimating lower bounds for weak bisimilarity.

1.5 Bisimulation approximants

One can look at strong and weak bisimulation equivalences from a rather different perspective and study the properties of these notions following Milner's original definition in [48]. There he defines strong bisimulation in terms of a decreasing binary sequence of *approximants*. This construction was later replaced with a more elegant definition in the spirit of Park [55]. However, the earlier definition is more helpful when we want to argue about non-bisimilarity.

We will develop an analogous approach by defining *weak bisimulation approximants*. We define a sequence of binary relations on processes as follows: we start with the universal relation (containing all pairs of processes) and then we inductively construct smaller relations by removing "unsuitable" pairs (that will be specified later). We will show that for both strong and weak bisimilarity these sequences converge to a limit that is the largest strong or weak bisimulation. For strong bisimilarity we can easily convince ourselves that the maximal strong bisimulation will be obtained as the limit of all finite sequences. However, we will see that in order to reach the maximal weak bisimulation one needs to go on further. We will investigate the length and structural properties of the sequences of approximants, mainly by providing some lower bounds on the length by means of examples.

1.6 Organisation of the thesis

The thesis is organised as follows: in **Chapter 2** we explain all background definitions, namely of Basic Process Algebras and Basic Parallel Process Algebras, strong and weak bisimulation equivalences, and some further notions, and describe in detail a few techniques for deciding strong and weak bisimilarity.

In **Chapter 3** we define weak bisimulation approximants, binary relations on processes labelled by ordinal numbers. We establish their properties and show that they eventually converge at the maximal weak bisimulation. We will search for the value of an ordinal α with the following property: for every $\beta \geq \alpha$, $\approx = \approx_\beta$. Such an ordinal represents the approximant which has already converged to weak bisimulation. The value will differ for BPA and BPPA. We will demonstrate some lower bounds on this α which will be ω^ω for BPA and $\omega \cdot 2$ for BPPA. We will also study approximants on some restricted subclasses of BPA-processes and BPP. Finally, we will use the method of semilinear sets to show decidability of $\approx_n$ for Basic Parallel Processes.

In **Chapter 4** we study hardness of the weak bisimilarity decision problem for BPA-processes and BPP. The question whether weak bisimilarity is decidable for general BPA-processes and BPP is still open. The problem we are looking at is what would be the complexity of a decision procedure that might exist. We use the concept of reduction to provide some lower bounds on the complexity. We present two reductions from two problems that are complete for two complexity classes. That demonstrates various hardness results for weak bisimilarity. The first is a reduction from the problem KNAPSACK, which is NP-complete, to weak bisimilarity of (totally normed) BPA-processes and BPP. The second is a reduction from the problem TOT, which is PSPACE-complete, to weak bisimilarity of BPA-processes. That demonstrates NP-hardness of deciding weak bisimilarity for (totally normed) BPA-processes and BPP, and PSPACE-hardness of deciding weak bisimilarity of BPA-processes.

Chapter 5 describes a new connection between Basic Parallel Process algebras and polynomial rings. We explain how Basic Parallel Processes can be viewed as special one-term polynomials (*power products*), where parallel composition of BPP corresponds to multiplication of power products. A bisimulation relation gives rise to a polynomial ideal and the bisimulation condition can be expressed in terms of polynomial ideal membership. We use an important theorem from polynomial algebra, the Hilbert's Basis Theorem, to show that our ideals have finite bases. Then we use a method from computer algebra, the method of Gröbner bases, to decide membership in polynomial ideals. By combining

all these methods we construct semidecision and decision procedures for strong bisimilarity of BPP.

Finally, **Chapter 6** contains a summary of the results achieved in this thesis and directions for further work.

Chapter 2

Background

In this chapter we are going to define basic concepts which will be used throughout the thesis. We will define two kinds of process algebras that we will study, the *Basic Process Algebras* and the *Basic Parallel Process Algebras*. Two notions of equivalence will be presented, *strong bisimulation equivalence* and *weak bisimulation equivalence*. We will study the relationship between the process algebras and the equivalences. Eventually we will present a review of known decidability and complexity results for these equivalences and various techniques that were used to obtain them.

2.1 Bisimulation equivalences

We will find it convenient to define some basic notions in terms of *labelled transition graphs*. These are accepted as an appropriate semantic model of concurrent computation when interleaving semantics is considered, as it is indeed in our case.

Definition 2.1 *A* labelled transition graph *is a triple* $(\mathcal{S}, A, \longrightarrow)$ *consisting of a set of* states *or* processes $\mathcal{S}$*, a set of* labels *or* actions A *and a* transition relation $\longrightarrow \subseteq \mathcal{S} \times A \times \mathcal{S}$.

For a labelled transition graph $(\mathcal{S}, A, \longrightarrow)$, we will write $P \stackrel{a}{\longrightarrow} P'$ to denote that $(P, a, P') \in \longrightarrow$, and if there is no $P' \in \mathcal{S}$ and no $a \in A$ with $P \stackrel{a}{\longrightarrow} P'$, we will denote that by $P \not\longrightarrow$. We will generalise the transition relation to include sequences of transitions in a straightforward way. For every P, $P \stackrel{\epsilon}{\longrightarrow} P$, where ϵ is the empty word. If $w = a_1 a_2 \ldots a_k$ is a non-empty sequence of labels from A then we write $P \stackrel{w}{\longrightarrow} P'$ if there exists a sequence of transitions $P \stackrel{a_1}{\longrightarrow} P_1 \stackrel{a_2}{\longrightarrow} P_2 \stackrel{a_3}{\longrightarrow} \ldots \stackrel{a_k}{\longrightarrow} P_k = P'$.

2.1.1 Strong bisimulation equivalence

We want to identify processes which exhibit the same observable behaviour that is represented by labelled transitions. Following Milner in [49] we define the notion of a *strong bisimulation* as a binary relation on a transition graph.

Definition 2.2 *Let $(\mathcal{S}, A, \longrightarrow)$ be a transition graph. A binary relation $\mathcal{R}$ over $\mathcal{S}$ is a* strong bisimulation *if whenever $(P,Q) \in \mathcal{R}$ then for every $a \in A$,*

- *if $P \stackrel{a}{\longrightarrow} P'$ then there exists $Q \stackrel{a}{\longrightarrow} Q'$ such that $(P',Q') \in \mathcal{R}$, and*
- *if $Q \stackrel{a}{\longrightarrow} Q'$ then there exists $P \stackrel{a}{\longrightarrow} P'$ such that $(P',Q') \in \mathcal{R}$.*

Processes P and Q are bisimilar*, written $P \sim Q$, if they are related by some strong bisimulation.*

Note that if we want to relate two states P and Q from two different transition graphs $\mathcal{S}$ and $\mathcal{S}'$ we can construct a strong bisimulation on the disjoint union of $\mathcal{S}$ and $\mathcal{S}'$. That ensures the universality of our definition.

It was shown in [49] that the union of all strong bisimulations is also a strong bisimulation. It is the largest strong bisimulation, denoted by $\sim$, and it is an equivalence relation. We will also call it *(strong) bisimulation equivalence.*

There is an alternative approach towards strong bisimilarity which was the original definition stated by Milner in [48]. We start from the relation containing all pairs of processes and then form a non-increasing chain of binary relations which are called *approximants.* Approximants are labelled by ordinal numbers and denoted by $\sim_\alpha$. Originally, Milner only considered the chain of approximants labelled by natural numbers and $\sim$ was taken to be the intersection $\bigcap_{n\in\mathbb{N}} \sim_n$. That is entirely sufficient in the context of finitely branching transition graphs but fails to work for a more general, infinitely branching systems that one might want to consider. Hence the definition stated here is phrased in terms of binary relations labelled by ordinal numbers. We will use Greek letters α, β to denote ordinals and the class of ordinal numbers will be denoted by On. For an introduction to ordinal numbers we refer the reader to standard textbooks (cf. [21], [22], [45]).

Definition 2.3 *Let $(\mathcal{S}, A, \longrightarrow)$ be a transition graph.* Strong bisimulation approximants *labelled by ordinal numbers are binary relations over $\mathcal{S}$ denoted by $\sim_\alpha$ and defined in the following way:*

- *$P \sim_0 Q$ for all P and Q*

- $P \sim_{\alpha+1} Q$ *if for all actions* $a \in A$,
  - *whenever* $P \xrightarrow{a} P'$ *then there exists* $Q \xrightarrow{a} Q'$ *so that* $P' \sim_\alpha Q'$ *and*
  - *whenever* $Q \xrightarrow{a} Q'$ *then there exists* $P \xrightarrow{a} P'$ *so that* $P' \sim_\alpha Q'$
- $P \sim_\lambda Q$ *if* $P \sim_\alpha Q$ *for every* $\alpha < \lambda$, *for a limit ordinal* λ.

We can state another definition of approximants which considers sequences of actions instead of single actions in the clauses. That is the original definition which appears in [48]:

Definition 2.4 *Let* $(\mathcal{S}, A, \longrightarrow)$ *be a transition graph. We define strong bisimulation approximants* $\sim^M_\alpha$ *as binary relations over* $\mathcal{S}$ *in this way:*

- $P \sim^M_0 Q$ *for all* P *and* Q
- $P \sim^M_{\alpha+1} Q$ *if for every sequence of actions* $t \in A^*$,
  - *whenever* $P \xrightarrow{t} P'$ *then there exists* $Q \xrightarrow{t} Q'$ *so that* $P' \sim^M_\alpha Q'$ *and*
  - *whenever* $Q \xrightarrow{t} Q'$ *then there exists* $P \xrightarrow{t} P'$ *so that* $P' \sim^M_\alpha Q'$
- $P \sim^M_\lambda Q$ *if* $P \sim^M_\alpha Q$ *for every* $\alpha < \lambda$, *for a limit ordinal* λ.

The two notions of approximants are equivalent in the sense that both define non-increasing chains of relations that always converge, with the limit being the maximal bisimulation. We will spell that out formally for the approximants defined by 2.3, however analogous statements hold for **Definition 2.4** as well. The following statement says that the relations $\sim_\alpha$ decrease non-strictly as α increases. It is rather straightforward to verify this proposition and we will not concern ourselves with the proof here.

Proposition 2.5 *For all* $\alpha, \beta \in On$, $\alpha > \beta$ *implies* $\sim_\alpha \subseteq \sim_\beta$.

The second claim confirms that the chain of approximants always converges and its limit is the largest strong bisimulation. The proof, which involves arguments from fixed-point theory, can be found in [49].

Proposition 2.6 $\sim = \bigcap_{\alpha \in On} \sim_\alpha$.

It is apparent that the approximants defined by 2.4 distinguish finer aspects of branching behaviour than those of 2.3, and clearly $\sim_\alpha^M \subseteq \sim_\alpha$ for every α. The latter approximants $\sim_\alpha^M$ also converge faster towards the maximal bisimulation. However, checking whether two processes are related by $\sim_\alpha^M$ may not be feasible since it involves checking a property for all sequences of moves that these processes can perform. That can make these approximants undecidable even for some simple classes of processes [36]. Therefore it is more convenient to work with the simpler notion as defined by 2.3.

2.1.2 Weak bisimulation equivalence

The notion of strong bisimilarity requires a process to be capable of matching each transition that an equivalent process may perform. However, sometimes we want to distinguish between *observable (external)* and *internal* behaviour of processes and we wish to regard two processes equivalent if they exhibit the same observable behaviour, irrespective of any intermediate internal behaviour that may occur. To this end we introduce a special *silent* action τ which represents internal behaviour and we define a new composite transition $\stackrel{a}{\Longrightarrow}$ as $(\stackrel{\tau}{\longrightarrow})^* \stackrel{a}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^*$ if $a \neq \tau$ and $(\stackrel{\tau}{\longrightarrow})^*$ for $a = \tau$. Then we can define a more general process equivalence which is called *weak bisimulation.*

Definition 2.7 *Let $(\mathcal{S}, A, \longrightarrow)$ be a transition graph, where the set of labels A also includes the silent action τ. A binary relation $\mathcal{R}$ over $\mathcal{S}$ is a* weak bisimulation *if whenever $(P,Q) \in \mathcal{R}$ then for every $a \in A$,*

- *if $P \stackrel{a}{\longrightarrow} P'$ then there exists $Q \stackrel{a}{\Longrightarrow} Q'$ such that $(P', Q') \in \mathcal{R}$, and*
- *if $Q \stackrel{a}{\longrightarrow} Q'$ then there exists $P \stackrel{a}{\Longrightarrow} P'$ such that $(P', Q') \in \mathcal{R}$.*

Processes P and Q are weakly bisimilar, *written $P \approx Q$, if they are related by some weak bisimulation.*

The requirement on matching transitions is relaxed in comparison with strong bisimulation and so this definition yields a weaker notion of bisimulation. It is easy to convince oneself that for all pairs of processes P and Q, if $P \sim Q$ then also $P \approx Q$. However, the converse does not always hold which is illustrated in the example that follows.

Example 2.8 We consider the processes P and Q pictured in Fig. 2.1. Clearly P and Q are not strongly bisimilar because to the move $P \stackrel{a}{\longrightarrow} P'$ the process

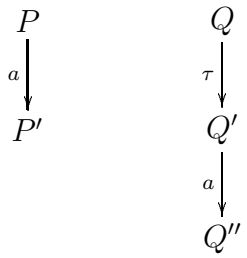


Figure 2.1: Weakly bisimilar processes

Q has no response, and vice versa, the move $Q \xrightarrow{\tau} Q'$ cannot be matched by P. When we consider weak bisimilarity then these processes become equivalent. To the transition $P \xrightarrow{a} P'$ the process Q will respond with $Q \xrightarrow{\tau} Q' \xrightarrow{a} Q''$, written as $Q \stackrel{a}{\Longrightarrow} Q''$ in terms of weak derivations. The resulting derivatives P' and Q'' represent deadlock, i.e. cannot perform any action at all, and hence are equivalent. Conversely, to the transition $Q \xrightarrow{\tau} Q'$ the process P chooses to perform the empty transition $P \stackrel{\epsilon}{\Longrightarrow} P$. The resulting processes Q' and P are even strongly bisimilar. Hence we can conclude that $P \approx Q$. $\square$

The above example illustrates the valid option of a process to refrain from performing any action in response to a τ transition of an equivalent process. This conveys the essence of τ transition as an internal change of state that is not observable by outsiders and hence has to be abstracted from.

Analogously to strong bisimulation, we can define weak bisimulation by a non-increasing chain of binary approximants. That will be carried out in detail in **Chapter 3**.

2.2 Simple process algebras

Simple process algebras are a category of process algebras that are obtained by application of a few basic operators. They are *Basic Process Algebras (BPA)*, whose main operators are sequential composition and summation, and *Basic Parallel Process Algebras (BPPA)*, whose main operators are parallel composition and summation. Now we are going to define the syntax and semantics of the two classes of process algebras. We will define processes by *process expressions* which we will provide with *structured operational semantics*.

We presuppose an infinite set of *actions* $Act = \{a, b, c, \ldots\} \cup \{\tau\}$ with τ being a distinguished element of Act denoting the *silent* action. We let the variables $\mu, \nu, \ldots$ range over Act. We also assume a countably infinite set of *process variables*

$Var = \{X, Y, Z, \ldots\}$ with capital letters P, Q, R, S ranging over strings from Var^* or multisets from $Var^{\otimes}$. The distinguished character ϵ shall denote the empty sequence or the empty (multi)set.

The process expressions are obtained using the following abstract syntax:

$$\begin{array}{rcll} E & ::= & 0 & \text{(inaction)} \\ & | & \mu E & \text{(action prefix } \mu \in Act\text{)} \\ & | & X & \text{(process variable } X \in Var\text{)} \\ & | & E + F & \text{(summation)} \\ & | & E \cdot F & \text{(sequential composition)} \\ & | & E \| F & \text{(parallel composition)} \end{array}$$

The variables E, F will denote process expressions. The intended meaning of our operators is:

- 0 represents the inactive process with no available transitions.
- μE is a process which can evolve into E by performing μ.
- $E + F$ can either behave like E or F.
- $E \cdot F$ behaves like E until termination whereupon it behaves like F.
- $E \| F$ can perform the actions of E and F in an arbitrary interleaved fashion. In fact, this type of parallel composition is called *merge* but for our convenience we will continue to call it parallel composition.

There is no synchronisation in our calculus and therefore the silent action τ only occurs when explicitly stated, it cannot arise as a result of communication between two processes. Finally, to simplify the process expressions, we will omit all superfluous occurrences of 0, that is we will write μ instead of $\mu 0$. We will also omit the symbol for sequential composition, thus writing EF for $E \cdot F$.

A *process* is a process expression whose variables are defined by a family Δ of finitely many recursive process equations

$$\Delta \;=\; \Big\{ X_i \stackrel{\text{def}}{=} E_i \mid i = 1, 2, \ldots, n \Big\},$$

where X_i are distinct variables and E_i are process expressions containing at most the variables $\{X_1, \ldots, X_n\}$. The variable X_1 is singled out as the *leading variable*. We allow recursive definitions and so we need to make sure that all finite families of process equations have a unique solution up to bisimilarity. The following condition takes care of that [49]:

Definition 2.9 *A process expression E is* guarded *if every variable occurrence in E is within the scope of an action prefix. The family of process equations $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, \ldots, n\}$ is guarded if each E_i is guarded for $i = 1, \ldots, n$.*

Hence we will always assume guarded families of process equations. Now we can present the semantics of our language in the style of operational semantics. Any finite family of guarded process equations Δ determines a labelled transition graph; states are process expressions, the set of labels is given by Act and the transition relation is given as the least relation derived from the rules of Table 2.1.

We have already defined strong bisimulation in terms of labelled transition graphs. Every family of guarded process equations determines a labelled transition graph. So we can say that two such families Δ and Δ' are bisimilar, $\Delta \sim \Delta'$, if there exists a strong bisimulation $\mathcal{R}$ relating the corresponding transition graphs of Δ and Δ' such that $(X_1, X_1') \in \mathcal{R}$, where X_1, resp. X_1', are the leading variables of Δ, resp. Δ'.

$$\frac{E \stackrel{\mu}{\longrightarrow} E'}{E + F \stackrel{\mu}{\longrightarrow} E'} \qquad \frac{F \stackrel{\mu}{\longrightarrow} F'}{E + F \stackrel{\mu}{\longrightarrow} F'} \qquad \frac{E \stackrel{\mu}{\longrightarrow} E'}{EF \stackrel{\mu}{\longrightarrow} E'F}$$

$$\frac{E \stackrel{\mu}{\longrightarrow} E'}{E \| F \stackrel{\mu}{\longrightarrow} E' \| F} \qquad \frac{F \stackrel{\mu}{\longrightarrow} F'}{E \| F \stackrel{\mu}{\longrightarrow} E \| F'} \qquad \frac{isnil(E), F \stackrel{\mu}{\longrightarrow} F'}{EF \stackrel{\mu}{\longrightarrow} F'}$$

$$\frac{E \stackrel{\mu}{\longrightarrow} E'}{X \stackrel{\mu}{\longrightarrow} E'} \ (X \stackrel{\text{def}}{=} E \in \Delta) \qquad \mu E \stackrel{\mu}{\longrightarrow} E$$

Table 2.1: Transition rules

For the representation of the rules for sequential composition we introduce a predicate $isnil(E)$ (following Christensen [6]) indicating whether or not E is capable of performing an action, in other words whether or not E is bisimilar to the inactive process 0. We will compute the predicate by induction on the structure of guarded process expressions.

Definition 2.10 *([6]) The predicate $isnil(E)$ is defined by cases on the structure of E as follows:*

- $isnil(0) = true$

- $isnil(\mu E) = false$
- $isnil(E + F) = isnil(EF) = isnil(E \| F) = isnil(E) \wedge isnil(F)$
- $isnil(X) = isnil(E)$ *where* $X \stackrel{\text{def}}{=} E \in \Delta$

We can easily verify that $isnil(E)$ if and only if $E \sim 0$ which justifies the inference rule that defines sequential composition.

2.2.1 Basic Process Algebras

A *Basic Process Algebra (BPA)* consists of *BPA-processes* that are obtained from a finite set of variables using the operators of action prefix, sequential composition and summation. The semantics is defined by transition rules for each operator.

A Basic Process Algebra can be also presented as a free monoid over a finite set of generators (atoms) together with a finite set of rules that define the behaviour of each atom. We will present both definitions and show the relationship between them. First we present the definition of a BPA in the spirit of process calculi.

Definition 2.11 *The* BPA-process expressions *are given by the following abstract syntax:*

$$E ::= 0 \mid \mu E \mid X \mid E + F \mid E \cdot F$$

A family of guarded process equations $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, \ldots, n\}$ *defines a BPA-process if each* E_i *is a guarded BPA expression for* $i = 1, \ldots, n$.

We can present process expressions in a special form which is an analogue of *Greibach normal form* for *context-free grammars*. Indeed, there is a close link between Basic Process Algebras and context-free grammars which was demonstrated by Baeten, Bergstra and Klop in [2]. We will later follow that link and present an alternative definition of BPA.

Definition 2.12 *A finite family* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, \ldots, n\}$ *of guarded BPA equations is in* Greibach normal form (GNF) *if every expression* E_i *is of the form*

$$E_i \equiv \sum_{j=1}^{n_i} \mu_{ij} P_{ij},$$

where $P_{ij} \in Var^*$.

It was shown in [2] and [34] that every system of guarded BPA equations can be effectively presented in Greibach normal form. Therefore we do not need to consider the full generality provided by the process expression definition and we can restrict our attention to processes formed as strings of variables from Var^*.

Theorem 2.13 ([2], [34]) *If Δ is a finite family of guarded BPA equations, we can effectively find a family Δ' in GNF such that $\Delta \sim \Delta'$.*

That leads us to consider a Basic Process Algebra as a pair (Σ^*, Δ), where Σ is a finite set of variables $\{X_1, \ldots, X_n\}$ and $\Delta = \{X \xrightarrow{\mu} P \mid X \in \Sigma, P \in \Sigma^*\}$ is a finite set of *transitions*. Obviously BPA-processes are strings over Σ and we can generalise the rules of Δ to determine a more general transition relation by

$$\text{for every } Q \in \Sigma^*, \quad XQ \xrightarrow{\mu} PQ \quad \text{whenever} \quad X \xrightarrow{\mu} P \in \Delta.$$

Clearly each equation in GNF $X \stackrel{\text{def}}{=} \sum_{i=1}^{n} \mu_i P_i$ can be equivalently viewed as a sequence of transitions $X \xrightarrow{\mu_i} P_i$. Thus every finite family of guarded BPA equations in GNF gives rise to an equivalent Basic Process Algebra of the form (Σ^*, Δ), and vice versa. Therefore we will adopt the notation (Σ^*, Δ) as the definition of a Basic Process Algebra throughout the thesis.

Remark: In fact, the definition expressed in spirit of process calculi slightly differs from the standard approach towards BPA in two aspects. It includes the inactive process 0, and views each μ from Act as a unary operator. In the original calculus BPA [2], each μ would represent a process, namely the process $\mu 0$. We have chosen this uniform presentation along the lines of Christensen [6]. Throughout the thesis we will prefer to view BPA-processes as elements from a free monoid over a finite set of atomic variables, as spelt out above. We shall also assume, without loss of generality, that for each variable $X \in \Sigma$ there is at least one rule $X \xrightarrow{\mu} P$ in Δ. The same condition will be required from Basic Parallel Process Algebras for reasons which will become clear later.

2.2.2 Basic Parallel Process Algebras

Basic Parallel Processes are a natural concept that arises in different contexts. Originally they were defined by Christensen in his thesis [6]. A *Basic Parallel Process Algebra (BPPA)* consists of *Basic Parallel Processes (BPP)* that are obtained from a finite set of variables using the operators of action prefix, parallel composition (merge) and summation. The semantics is defined by transition rules for each operator.

A Basic Parallel Process Algebra can be also presented as a free commutative monoid over a finite set of generators (atoms) together with a finite set of rules that define the behaviour of each atom. We will present both definitions and show the relationship between them. The standard definition of a BPPA in the spirit of process calculi is as follows:

Definition 2.14 *The* Basic Parallel Process (BPP) expressions *are defined by the following abstract syntax:*

$$E ::= 0 \mid \mu E \mid X \mid E + F \mid E \| F$$

A family of guarded process equations $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, \ldots, n\}$ *defines a Basic Parallel Process (BPP) if each* E_i *is a guarded BPP expression for every* $1 \leq i \leq n$.

Analogously to BPA, for every BPP expression there exists a normal form which is called *full standard form* in this context. The definition and the theorem proving existence of an equivalent normal form is due to Christensen [6].

Definition 2.15 *A finite family* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i \mid i = 1, \ldots, n\}$ *of guarded BPP equations is in* full standard form *if every expression* E_i *is of the form*

$$E_i \equiv \sum_{j=1}^{n_i} \mu_{ij} P_{ij},$$

where $P_{ij} \in Var^{\otimes}$.

The set $Var^{\otimes}$ is the commutative monoid consisting of multisets of variables from Var. We can write elements of $Var^{\otimes}$ as multisets or as parallel composition, e.g. $\{X, X, Y, Z\}$ (or equivalently $\{X^2, Y, Z\}$) is the multiset representation, whereas $X \| X \| Y \| Z$ (or equivalently $X^2 \| Y \| Z$) is the parallel composition notation.

Proposition 2.16 *[6] Given any finite family of guarded BPP equations* Δ *we can effectively construct another finite family of BPP equations* Δ' *in full standard form such that* $\Delta \sim \Delta'$.

That eventually leads us to abandon the general abstract syntax and consider a simpler and more straightforward notation for BPPA.

Definition 2.17 *A* Basic Parallel Process Algebra *is a pair* $(\Sigma^{\otimes}, \Delta)$ *where*

- $\Sigma^{\otimes}$ *is the set of multisets over the finite set of variables* $\Sigma = \{X_1, \ldots, X_n\}$, *and*

- Δ *is a finite set of rules of the form* $X \xrightarrow{\mu} P$ *where* $X \in \Sigma$ *and* $P \in \Sigma^{\otimes}$.

The elements of $\Sigma^{\otimes}$ *are then called* Basic Parallel Processes (BPP).

In the commutative algebra $\Sigma^{\otimes}$ there is a natural operation of union which corresponds to parallel composition and hence we will denote it by $\|$. For $P, Q \in \Sigma^{\otimes}$,

$$P\|Q = X_1^{i_1+j_1} \ldots X_n^{i_n+j_n}, \text{ where } P \equiv X_1^{i_1} \ldots X_n^{i_n} \text{ and } Q \equiv X_1^{j_1} \ldots X_n^{j_n}.$$

We can extend the rules of Δ to all BPP in the obvious way:

$$P\|X\|Q \xrightarrow{\mu} P\|R\|Q \text{ if there is a rule } X \xrightarrow{\mu} R \in \Delta.$$

As we showed previously with BPA, the latter notation $(\Sigma^{\otimes}, \Delta)$ seems better suited to the purpose of the thesis and therefore we will accept it as our notion of Basic Parallel Process Algebra.

2.2.2.1 Vector representation of BPP

When we consider BPP as multisets of atomic processes we can represent them as vectors of natural numbers in this way: assuming that the cardinality of Σ is n, then a multiset $\{X_1^{i_1}, \ldots, X_n^{i_n}\}$ can be expressed as a vector $(i_1, \ldots, i_n) \in \mathbb{N}^n$. Each multiset determines a unique vector and union of multisets corresponds to vector summation. Pairs of BPP can be represented as vectors from $\mathbb{N}^n \times \mathbb{N}^n = \mathbb{N}^{2n}$.

Later we shall exploit this representation together with some mathematical structures on vectors. We shall be using two orderings on BPP: the *product order* $\subseteq$ and the *lexicographic order* $<_{\text{Lex}}$. We remind ourselves of the definitions. We assume some processes P and Q given as $P \equiv X_1^{i_1} \ldots X_n^{i_n}$, equivalently $P \equiv (i_1, \ldots, i_n)$, and $Q \equiv X_1^{j_1} \ldots X_n^{j_n}$, equivalently $Q \equiv (j_1, \ldots, j_n)$. Then we say that P is less than Q in the product order, written $P \subseteq Q$, if $i_l \leq j_l$ for every $1 \leq l \leq n$. We say that P is less than Q under lexicographic order, $P <_{\text{Lex}} Q$, if there exists l_0 such that $i_{l_0} < j_{l_0}$, and for all $l < l_0$, $i_l = j_l$. The orderings generalise in the obvious sense to pairs of processes.

The two orderings are consistent with parallel composition, that is $P \subseteq Q$ implies that $P\|R \subseteq Q\|R$, and $P <_{\text{Lex}} Q$ implies that $P\|R <_{\text{Lex}} Q\|R$, for every R. We also have that lexicographic order is a well-order, i.e. it is linear and every strictly decreasing sequence is finite. The following property of lexicographic order will be referred to later.

Lemma 2.18 *For any BPP* P, Q, R *and* S *such that* $P \neq Q$, *it holds that either* $(P\|R, P\|S) <_{\text{Lex}} (P\|R, Q\|S)$ *or* $(Q\|R, Q\|S) <_{\text{Lex}} (P\|R, Q\|S)$.

Proof: As $P \neq Q$ and $<_{\text{Lex}}$ is linear then either $P <_{\text{Lex}} Q$ or $Q <_{\text{Lex}} P$. Assuming that $P <_{\text{Lex}} Q$, we have that $P\|S <_{\text{Lex}} Q\|S$ and hence $(P\|R, P\|S) <_{\text{Lex}} (P\|R, Q\|S)$. From the latter assumption $Q <_{\text{Lex}} P$ we would analogously derive that $(Q\|R, Q\|S) <_{\text{Lex}} (P\|R, Q\|S)$. □

2.3 Further notions

In general, the processes that we shall consider determine infinite state transition graphs. No decision procedure for testing bisimilarity can be based on a simple exhaustive search since for infinitely many states that is not feasible. For that reason we are trying to identify some properties of the processes which would provide us with additional information that would lead to a decision procedure. An obvious guess is to try and identify some structural properties of processes. We want to divide the goal of testing bisimilarity of a given pair of processes into testing several cases of *smaller* processes which would eventually lead to one of a few trivial base cases. We will identify the conditions under which a process can be *decomposed* into smaller processes.

The concept of *size* of a process seems to be a natural criterion to consider. Intuitively, size should satisfy the following requirements. Two equivalent processes should be of the same size, finite state processes should have finite size and infinite state processes would have infinite size. Size should be a function of branching and of the lengths of sequences that can be performed by a process. And finally, size should be additive under composition.

One potential candidate for size is the *height (rank)* of the transition tree determined by a process. The height of a tree is an ordinal number defined inductively in this way: the height of an empty tree is taken to be $\emptyset$, and the height of a node is taken to be the supremum (limit) of the heights of its sons that are increased by 1. Intuitively this concept is rather appealing, however the way it is usually defined does not accommodate trees with infinite branches. That is a serious restriction and there does not seem to be an easy way of overcoming this obstacle.

We will present here a measure on processes which is very simple and still possesses many of the desired properties. We notice that if two processes are equivalent then all their finite behaviour has to coincide. In particular, if there is a terminating sequence of length n available to one of the equivalent processes then the other process has to be able to produce matching behaviour, i.e. the same sequence of the same length n. That applies to minimal (in length) sequences as

well and that is what we will take for our notion of measure on processes. We will define an auxiliary notion of *norm* to be the length of a minimal terminating sequence. Norm does not have any computational value in itself but it is surprisingly useful in classifying processes. It combines additively with respect to composition, sequential and parallel, and it is consistent with bisimilarity.

2.3.1 Strong norm

In the following text we assume that our processes are taken either from a Basic Process Algebra or a Basic Parallel Process Algebra. If P is a process then the *norm* of P, denoted by $|P|$, is the length of a minimal sequence leading from P to the empty process ϵ, that is $|P| = \min\{\text{length}(w) \mid P \xrightarrow{w} \epsilon, w \in A^*\}$. We say that a process is *normed* if it has a finite norm, otherwise it is *unnormed*. We also call this notion *strong norm* to distinguish it from *weak norm* which will be defined later. Norm is consistent with strong bisimilarity and is additive with respect to composition which is expressed in the lemma below.

Lemma 2.19 *Let P and Q be processes. Then the following conditions hold:*

1. *If $P \sim Q$ then $|P| = |Q|$.*
2. *$|P \circ Q| = |P| + |Q|$, where $\circ$ is either sequential or parallel composition.*
3. *If $|P| = 0$ then $P = \epsilon$.*

Proof: Since the statements of the lemma are quite easy to see we will prove them in an informal way. Case 3. is a straightforward consequence of the definition of norm since we do not admit inactive atoms.

To verify 1. and 2. we will note that if $|P| = n > 0$ then there exists a transition $P \xrightarrow{\mu} P'$ with $|P'| = n - 1$. We call this a *norm-reducing* transition. If we have P and Q such that $|P| = m < n = |Q|$ then there is a norm-reducing sequence $P \xrightarrow{\mu_1} P_1 \xrightarrow{\mu_2} P_2 \ldots \xrightarrow{\mu_m} P_m = \epsilon$ of length m. Since the norm of Q is strictly larger than m, for every sequence $Q \xrightarrow{\mu_1} Q_1 \xrightarrow{\mu_2} Q_2 \ldots \xrightarrow{\mu_m} Q_m$ the process Q_m can perform some transition $\xrightarrow{\mu}$. Hence $P_m \not\sim Q_m$ for every such Q_m and from that follows that $P \not\sim Q$.

If we have a composition of two processes P and Q, then we can reach the inactive state by first performing a minimal sequence of P followed by a minimal sequence of Q. Hence clearly $|P \circ Q| \leq |P| + |Q|$. To verify the other direction, we can observe that if we had a sequence starting from $P \circ Q$ leading to ϵ of length less than $|P| + |Q|$, we could reconstruct from it a sequence $P \xrightarrow{s} P' = \epsilon$ with

length$(s) < |P|$ or a sequence $Q \xrightarrow{t} Q' = \epsilon$ with length$(t) < |Q|$, which would contradict our assumptions.

We can observe that the verity of these statements relies strongly on the fact that we do not admit inactive variables in our algebra. If we allow an inactive atom X with no transition rule $X \xrightarrow{\mu} P$ associated with it, then any composite process XQ can perform no actions whatsoever and hence is bisimilar to ϵ. Then it follows from 1. that the norms of ϵ and XQ are equal, i.e. 0. Also the norm of X is 0. By 2., the norm of XQ is equal to $|X| + |Q| = |Q|$. We have that $|XQ| = 0$ yet the norm of Q does not necessarily have to be 0. $\square$

There is a singular process of norm zero (ϵ) and the norm is additive under composition. Hence we can state a simple yet important claim, a corollary of case 2 which applies both to BPA and BPPA.

Corollary 2.20 *For a fixed algebra and a fixed $n \in \mathbb{N}$, there are only finitely many processes of norm n.*

The norm of a process from some algebra (Σ, Δ) (BPA, resp. BPPA) can be easily computed. First we will iteratively construct the set Σ_N of normed atoms from Σ. We initialise N_1 to consist of all atoms X that can reach ϵ within a single step, i.e. there is a transition $X \xrightarrow{\mu} \epsilon$ in Δ. Then we construct N_{i+1} by taking N_i and adding atoms that can with a single transition reach a process from N_i^*, resp. $N_i^{\otimes}$. This algorithm will stop when no more variables can be added. That will certainly occur within k iterations, where k is the size of Σ. All the normed atoms will be contained in $\Sigma_N = N_k$, and all the atoms from $\Sigma \setminus \Sigma_N$ are unnormed. For all normed processes we can easily derive their norms by the following rules:

- $|\epsilon| = 0$
- $|P \circ Q| = |P| + |Q|$, where $\circ$ is either sequential or parallel composition
- $|X| = \min\{|P| \mid \exists X \xrightarrow{\mu} P \in \Delta\} + 1$

The properties of norm which we have just verified are essential in decidability techniques. We will see later that algebras where all processes are of finite norm are simple to deal with since we can decide bisimilarity in polynomial time.

2.3.2 Weak norm

The principle of norm is based on a minimal terminating sequence of a process. In the context of strong bisimilarity all actions contribute equally towards the final length. When we consider weak bisimilarity, the action τ acquires a special status which has to be reflected in this notion. We consider the notion of *weak norm*, originally defined by Hüttel in [35]. The weak norm $\|P\|$ of a process P is taken to be the length of a minimal derivation sequence from P to ϵ not counting τ-moves, i.e. $\|P\| = \min\{\text{length}(w) \mid P \stackrel{w}{\Longrightarrow} \epsilon\}$.

A process P of a finite norm is called *weakly normed*. If the norm of P is finite and positive, i.e. $0 < \|P\| < \infty$, then P is *totally normed*. A process of infinite norm is again called *unnormed*. Weak norm on BPA and BPPA satisfies the following properties:

Lemma 2.21 *Let (Σ, Δ) be an algebra such that for every atom $X \in \Sigma$, if $X \approx \epsilon$ then there is a weak transition $X \stackrel{\tau}{\Longrightarrow} \epsilon$. For all processes P and Q of this algebra holds the following:*

1. *If $P \approx Q$ then $\|P\| = \|Q\|$.*
2. *$\|P \circ Q\| = \|P\| + \|Q\|$, where $\circ$ is either sequential or parallel composition.*

The proof of the lemma above is analogous to the proof of **Lemma 2.19**. We shall now explain the extra condition placed on P and Q. From the definition of weak norm, a process that behaves as a τ loop, for instance a process T defined by a sole transition rule $T \stackrel{\tau}{\longrightarrow} T$, has infinite norm. On the other hand T cannot ever produce any observable behaviour and so is weakly bisimilar to ϵ. Clearly, these two facts combined would violate statement 1, therefore all processes like T have to be excluded.

The weak norm of a process can be computed in a similar way to strong norm with the only difference that silent moves are not taken into account. Again we can design an iterative algorithm that after a finite number of steps halts with the set of weakly normed atoms. Then for weakly normed processes we calculate their respective norms following these rules:

- $\|\epsilon\| = 0$
- $\|P \circ Q\| = \|P\| + \|Q\|$, where $\circ$ is either sequential or parallel composition
- $\|X\| = \min\{\min\{\|P\| \mid \exists X \stackrel{\mu}{\longrightarrow} P \in \Delta, \mu \neq \tau\} + 1, \min\{\|P\| \mid \exists X \stackrel{\tau}{\longrightarrow} P \in \Delta\}\}$

We have showed that analogously to strong norm and strong bisimilarity, weak norm is consistent with weak bisimilarity and combines additively under composition. However, in contrast to strong norm, the class of processes of norm zero is not necessarily so simple. It is no longer true that if $\|P\| = 0$ then $P = \epsilon$ as we can change the norm of a process P to be 0 simply by adding a new transition $P \xrightarrow{\tau} \epsilon$. Yet P can still have many other actions at its disposal and so it need not even be bisimilar to ϵ. If there is a variable X of norm zero in an algebra then automatically we obtain an infinite set of processes of norm zero by taking $X, X^2, X^3, \ldots$. That also means that there may be an infinite set of processes of identical norm. Hence for weak bisimulation we cannot use the standard decidability techniques that work for strong bisimulation. We can achieve decidability by restricting to the subset of totally normed processes which will be showed in **Subsection 2.6.1**.

2.3.3 Image-finiteness

Although the general BPA-processes and BPP define infinite state transition graphs they possess an important property with respect to strong bisimilarity which is *image-finiteness*. In fact because each process algebra is defined by a finite set of rules all processes are finitely branching.

Definition 2.22 *A process P is* image-finite *if the set $\{P' \mid P \xrightarrow{\mu} P'\}$ is finite for every action μ.*

It is a standard result that for every class of image-finite processes the maximal strong bisimulation $\sim$ can be obtained as the intersection of approximants labelled by natural numbers.

Lemma 2.23 $\sim = \bigcap_{n \in \omega} \sim_n$.

Proof: We presuppose a class of image-finite processes and the maximal strong bisimulation $\sim$ on it. It is a matter of fact that the inclusion $\sim \subseteq \sim_\alpha$ holds for every α, and therefore $\sim \subseteq \bigcap_{n \in \omega} \sim_n$. We need to show that for this class of image-finite processes the other direction is satisfied, i.e. $\sim_\omega = \bigcap_{n \in \omega} \sim_n \subseteq \sim$.

The way to do that is to show that $\sim_\omega$ is a strong bisimulation, hence it is included in $\sim$. We need to verify that $\sim_\omega$ is closed under expansion. We assume two processes P and Q from the class such that $P \sim_\omega Q$. That means that for every n, $P \sim_n Q$. We fix a transition $P \xrightarrow{\mu} P'$ and from our assumptions, for every n there is a matching transition $Q \xrightarrow{\mu} Q'_n$ such that $P' \sim_n Q'_n$. Since Q has only finitely many μ derivatives, there must be a Q' that occurs infinitely often

in the sequence $Q'_0, Q'_1, \ldots, Q'_n, \ldots$. Then $P' \sim_n Q'$ for infinitely many indices n, and hence $P' \sim_\omega Q'$. The same argument can be used for any transition of Q. Therefore we have verified that $\sim_\omega$ is a strong bisimulation and thus we come to the conclusion that $\sim_\omega \subseteq \sim$, and finally, $\sim_\omega = \sim$. $\square$

We include this proof here for completeness. The original proof can be found in [23].

2.3.4 Semidecidability of bisimulation

Most of the original decidability results for strong bisimilarity on simple process algebras consist of two semidecision procedures. Semidecidability of non-bisimilarity is a simple consequence of **Lemma 2.23** of the previous subsection. It is straightforward to see that both BPP and BPA-processes give rise to classes of image-finite processes with respect to strong bisimilarity and hence $\sim = \bigcap_{n \in \omega} \sim_n$. Therefore if two BPA-processes or BPP P and Q are not strongly bisimilar there must be an n such that $P \not\sim_n Q$.

We can easily verify that $\sim_n$ is decidable for every n: the base case $\sim_0$ relates all processes, and in order to check whether $P \sim_{n+1} Q$ we need to check only finitely many situations $P' \sim_n Q'$ with $P \stackrel{\mu}{\longrightarrow} P'$ and $Q \stackrel{\mu}{\longrightarrow} Q'$, as P and Q are finitely branching. These two facts combined give us a direct semidecision procedure for non-bisimilarity.

Lemma 2.24 *Strong bisimilarity is semidecidable on Basic Process Algebras and Basic Parallel Process Algebras.*

2.3.5 Unique prime decompositions

We will now focus our attention on normed algebras and show that each process from a normed algebra can be decomposed into a composition of *prime processes*. Prime processes are those that are not further decomposable. We will show that this decomposition is unique with respect to bisimilarity.

Let A be an algebra (BPA or BPPA) with atoms $\Sigma = \{X_1, \ldots, X_n\}$. We say that A is *normed* if all its process variables are normed. A variable $X \in \Sigma$ is called *prime* (with respect to $\sim$) if $X \sim PQ$ entails that $P \sim \epsilon$ or $Q \sim \epsilon$. Let $\Pi = \{Y_1, \ldots, Y_i\}$ be a subset of Σ which contains one selected atom from every equivalence class of primes. A *prime decomposition* of a process P is an expression $Z_1 Z_2 \ldots Z_k \sim P$, where each Z_i is a prime from Π, for $i = 1, \ldots, k$. We say that an algebra has *unique prime decomposition* up to bisimilarity if every process is bisimilar to a unique product of primes from Π.

Historically, the first result on unique prime decompositions was obtained for finite processes by Milner and Moller in [50], [52]. Here we will state a couple of theorems about unique prime decompositions for BPA and BPPA. The proofs can be found in [6], [32].

Theorem 2.25 *Any normed Basic Process Algebra has unique prime decomposition up to bisimilarity.*

Theorem 2.26 *Any normed Basic Parallel Process Algebra has unique prime decomposition up to bisimilarity.*

We will show on an example why normedness is a necessary condition for unique prime decomposition. We assume two processes $X \stackrel{a}{\longrightarrow} \epsilon$ and $Y \stackrel{a}{\longrightarrow} Y$. The atom X is a prime, however Y is not because $Y \sim XY$ (taken as either sequential or parallel composition). We cannot express Y as any composition of a finite number of primes and so clearly prime decomposition fails.

2.4 Decidability of strong bisimilarity on BPA

The first decidability result appeared in [2] where Baeten, Bergstra and Klop demonstrated decidability for normed BPA-processes. They noticed that the concept of context-free grammars can be transposed to the setting of process algebras where it gives rise to context-free processes, or BPA-processes. Their result was a consequence of certain periodic structure displayed by the transition graphs determined by BPA-processes.

Other proofs for the normed subclass of BPA-processes followed with Hüttel and Stirling [37], and Caucal [5]. Then came papers by Groote [20], and Huynh and Lu Tian [38] which improved the computational complexity of the decision problem. Eventually, Hirshfeld, Jerrum and Moller produced a polynomial time decision procedure in [31].

For the unrestricted class of general BPA-processes, the first decidability result was demonstrated by Christensen, Hüttel and Stirling in [10], [11]. Their approach was similar to Caucal's proof for normed BPA-processes. An improvement on this was an elementary decision procedure reported by Burkart, Caucal and Steffen [4]. We will now concentrate on the technique developed by Caucal in [5] since it enables us, for some classes of processes, to test bisimilarity by constructing a finite *base* for bisimulation.

2.4.1 Caucal bases and normed BPA

Caucal in his paper [5] introduced the notion of *self-bisimulation* which then became widely used by other authors, also under the name of *Caucal base*. Originally conceived in order to construct a decision procedure for normed BPA-processes, it was used later in combination with other techniques to decide strong bisimilarity for other classes of processes. Here the definition of Caucal base will be given and Caucal's original decidability proof will be explained. All the definitions and theorems that follow are due to Caucal and taken from [5] unless stated otherwise.

Caucal phrases his results in terms of *graphs of right derivations of context-free grammars*, however we will present the work in the framework of Basic Process Algebras. In the following we will assume a normed BPA (Σ, Δ), although the notion of a Caucal base and its properties carry over to general BPA.

For a binary relation $\mathcal{R}$ on BPA-processes, the least congruence generated by $\mathcal{R}$ is denoted by $\overset{\mathcal{R}}{\equiv}$, and it is the least equivalence relation containing $\mathcal{R}$ which is also closed under sequential composition. To spell this out, $PP' \overset{\mathcal{R}}{\equiv} QQ'$ whenever $P \overset{\mathcal{R}}{\equiv} Q$ and $P' \overset{\mathcal{R}}{\equiv} Q'$.

Definition 2.27 *A binary relation $\mathcal{R}$ on (general) BPA-processes is a* Caucal base *if whenever $(P, Q) \in \mathcal{R}$ we have that*

- *if $P \xrightarrow{\mu} P'$ then $Q \xrightarrow{\mu} Q'$ for some Q' with $P' \overset{\mathcal{R}}{\equiv} Q'$, and*
- *if $Q \xrightarrow{\mu} Q'$ then $P \xrightarrow{\mu} P'$ for some P' with $P' \overset{\mathcal{R}}{\equiv} Q'$.*

The following result relates Caucal bases and bisimulation equivalence.

Proposition 2.28 *If $\mathcal{R}$ is a Caucal base then $\overset{\mathcal{R}}{\equiv} \subseteq \sim$.*

In fact a stronger property holds, that if $\mathcal{R}$ is a Caucal base then the least congruence generated by $\mathcal{R}$ is a bisimulation. In order to prove that we would show by induction on the depth of inference that if $P \overset{\mathcal{R}}{\equiv} Q$ then for every $P \xrightarrow{\mu} P'$ there exists $Q \xrightarrow{\mu} Q'$ such that $P' \overset{\mathcal{R}}{\equiv} Q'$, and vice versa. The base case is that $(P, Q) \in \mathcal{R}$, then the sought condition follows from $\mathcal{R}$ being a Caucal base. If $P \overset{\mathcal{R}}{\equiv} Q$ follows from the closure properties of the congruence then the result follows by induction, using the fact that bisimulation equivalence is a congruence. We can easily see that $\sim$ satisfies the conditions of a Caucal base and so as a corollary we obtain the following:

Corollary 2.29 *$P \sim Q$ if and only if $(P, Q) \in \mathcal{R}$ for some Caucal base $\mathcal{R}$.*

We can immediately see the importance of a Caucal base. Clearly one cannot test bisimilarity by going through the whole possibly infinite bisimulation equivalence. A Caucal base is intended as a finite representation of $\sim$ that would enable effective testing. However, not every Caucal base would suffice as the feasibility of testing with a base depends on the check whether processes are related by the least congruence generated by the base.

For normed BPA-processes Caucal solves this problem by considering self-bisimulations that are *fundamental*. As we work within a slightly different framework we present a slight modification of the notion. It will be convenient to assume that the variables from Σ are ordered by nondecreasing norm, so that $X < Y$ implies $|X| \leq |Y|$. Then we say that a binary relation $\mathcal{R}$ on processes is *fundamental* if it satisfies the following two conditions:

1. $\mathcal{R}$ consists of pairs (X, YP) where $Y < X$ and $|X| = |YP|$
2. there is at most one pair (X, YP) in $\mathcal{R}$ for every variable X.

Every fundamental relation will be finite as it contains at most one pair for each process variable. For a fundamental relation $\mathcal{R}$ it is decidable whether $P \stackrel{\mathcal{R}}{\equiv} Q$ hence we can test whether a given fundamental relation is a Caucal base. It also follows from the definition that there are only finitely many fundamental relations and they can be effectively enumerated.

We can intuitively see that a correct fundamental self-bisimulation will consist of pairs *(atom, its decomposition into primes)*, whose existence is guaranteed for normed BPA. Before we express the exact nature of the relationship between a fundamental self-bisimulation and bisimulation equivalence we will formulate the property of normed BPA-processes that captures the essence of decomposition.

Lemma 2.30 *If $XP \sim YQ$ and $|X| \leq |Y|$ then there exists a process R such that $XR \sim Y$ and $P \sim RQ$.*

The final item of the decision procedure is the criterion that states when a fundamental Caucal base generates the maximal bisimulation.

Theorem 2.31 *Every fundamental Caucal base that is maximal with respect to inclusion generates the maximal strong bisimulation.*

Proof (sketch): Clearly every fundamental Caucal base generates a subset of the maximal bisimulation. That every bisimilar pair belongs to the congruence generated by a fundamental self-bisimulation $\mathcal{R}$ that is maximal with respect to

inclusion is obtained as a consequence of **Lemma 2.30**. If that there exists a bisimilar pair $XP \sim YQ$ that does not appear in $\overset{\mathcal{R}}{\equiv}$, we choose a minimal such pair. Then there exists a process R such that, without loss of generality, $XR \sim Y$ and $P \sim RQ$. As both pairs (XR, Y) and (P, RQ) have smaller norm, we can conclude from the assumptions that $XR \overset{\mathcal{R}}{\equiv} Y$ and $P \overset{\mathcal{R}}{\equiv} RQ$. The relation $\overset{\mathcal{R}}{\equiv}$ is a congruence and hence $XP \overset{\mathcal{R}}{\equiv} YQ$ which contradicts the choice of (XP, YQ). Thus we have that $\overset{\mathcal{R}}{\equiv} \subseteq \sim$. $\square$

Given a normed BPA and a pair of processes P, Q, the algorithm for deciding bisimilarity will enumerate all fundamental relations on the algebra. For each such relation it will test whether it is a Caucal base maximal with respect to inclusion. As the set of all fundamental relations is finite and the test on self-bisimulation is decidable, the algorithm will eventually stop with some maximal Caucal base $\mathcal{R}$. Then it will check whether the input pair P, Q is related by the least congruence generated by $\mathcal{R}$. The algorithm will output that $P \sim Q$ if $P \overset{\mathcal{R}}{\equiv} Q$, otherwise it will output that $P \not\sim Q$. This is clearly a correct algorithm that decides strong bisimilarity on any normed BPA hence we can conclude:

Theorem 2.32 *Strong bisimilarity is decidable on any normed Basic Process Algebra.*

Now we can briefly explain the polynomial algorithm for deciding $\sim$ of normed BPA-processes as it was presented in [31]. The idea is analogous to the decision procedure above, however we actually construct a finite base for the largest bisimulation and we do that in time polynomial in the size of the input algebra.

We start from a base of size quadratic in the number of atomic variables and then refine it in a series of steps into smaller bases which eventually converge to a base for $\sim$. All the bases satisfy this condition: they consist of pairs (Y, XP) such that $|Y| = |XP|$. We include at most one pair for each choice of variables X, Y and moreover, for every bisimilar pair $Y \sim XQ$ there is a pair (Y, XP) with $P \sim Q$ in the base. These conditions altogether ensure that we do not omit any bisimilar pair and also that the sizes are polynomial.

The refinement step mirrors the condition in the definition of a Caucal base. If there is a pair (X, P) in the current base such that some derivative (Q, R) is not in the congruence generated by the base then $X \not\sim P$ and hence we remove that pair from the base. We use the decomposition theorem to test in polynomial time whether two processes belong to the congruence $\overset{B}{\equiv}$ for the current base B.

2.4.2 Decidability for unnormed BPA

Having explained the decidability procedure for normed BPA-processes we will move onto general BPA. Christensen, Hüttel and Stirling were the first to apply Caucal bases for the purpose of proving bisimilarity decidable for all BPA-processes [10]. We will present their approach so all the definitions and theorems that follow are taken from [10].

We assume a Basic Process Algebra (Σ, Δ) whose variables may be unnormed. The notion of Caucal base carries over together with **Proposition 2.28** and **Corollary 2.29**. However, decomposition based on norm no longer works as there may be infinitely many unnormed processes and hence infinitely many possibilities of matching them up. Still, there exists a finite Caucal base for bisimulation and we will explain how it can be obtained.

Due to the sequential structure of BPA-processes we do not need to consider processes XP for unnormed X since then clearly $X \sim XP$. We will divide Σ into the set of normed variables Σ_N, and the set $\Sigma_U = \Sigma \setminus \Sigma_N$ consisting of unnormed variables. Then we will only consider processes from $\Sigma_N^* \cup \Sigma_N^* \Sigma_U$, i.e. processes in the form of sequences of normed variables possibly followed with a single unnormed variable.

The next definition stipulates what we mean by decomposition.

Definition 2.33 *A pair (XP, YQ) satisfying $XP \sim YQ$ is decomposable if X and Y are normed and for some R,*

- *$X \sim YR$ and $RP \sim Q$, or*
- *$Y \sim XR$ and $RQ \sim P$.*

We have noted earlier that in the context of normed algebras all bisimilar pairs (XP, YQ) are decomposable (see **Lemma 2.30**). However, in the presence of unnormed variables there can be bisimilar pairs that are not decomposable. Fortunately we will manage to show that there are only finitely many of them. The following finiteness lemma is crucial:

Lemma 2.34 *If $PR \sim QR$ for infinitely many non-bisimilar processes R, then $P \sim Q$.*

To verify this lemma we would show that the relation $\mathcal{R} = \{(P, Q) \mid PR \sim QR, \text{ for infinitely many non} - \text{bisimilar } R\}$ is a bisimulation. The argument rests on the fact that the processes are image-finite. If there is a move $P \xrightarrow{\mu} P'$ then for

infinitely many non-bisimilar R we must have $Q \xrightarrow{\mu} Q_R$ such that $P'R \sim Q_R R$. There must be some Q' occurring infinitely many times among these Q_R, hence (P', Q') is again in $\mathcal{R}$.

We will call two pairs (XP, YQ), (XP', YQ') *distinct* if $P \not\sim P'$ or $Q \not\sim Q'$. Then we have the following lemma about non-decomposable pairs:

Lemma 2.35 *For any X,Y, any set $\mathcal{R}$ of the form*

$$\mathcal{R} = \{(XP, YQ) \mid XP \sim YQ \text{ and } (XP, YQ) \text{ are not decomposable } \}$$

which contains only distinct pairs is finite.

This lemma is a consequence of image-finiteness and the previous **Lemma 2.34**.

For the final part we need to introduce a well-founded ordering $\sqsubseteq$ on pairs of processes. To that end we define a measure s on $\Sigma_N^* \cup \Sigma_N^* \Sigma_U$ as follows. For $P \in \Sigma_N^*$ and $X \in \Sigma_U$, $s(P) = s(PX) = |P|$. Now we put $(P, P') \sqsubseteq (Q, Q')$ if $\max\{s(P), s(P')\} \leq \max\{s(Q), s(Q')\}$. We will make use of the ordering in the construction of a Caucal base for $\sim$.

Theorem 2.36 *There exists a finite Caucal base for $\sim$.*

Proof (sketch): The base $\mathcal{R}$ is obtained as a union of two (finite) relations $\mathcal{R}_1$ and $\mathcal{R}_2$. The relation $\mathcal{R}_1$ is the largest set $\{(X, P) \mid X \in \Sigma_N \text{ and } X \sim P\}$. The relation $\mathcal{R}_2$ is the largest set $\{(XP, YQ) \mid XP, YQ \in \Sigma_N^* \cup \Sigma_N^* \Sigma_U, XP \sim YQ \text{ and } (XP, YQ) \text{ not decomposable}\}$, such that each pair (XP, YQ), (XP', YQ') is distinct, and we only assume minimal elements with respect to $\sqsubseteq$.

The two sets are clearly finite; $\mathcal{R}_1$ is finite as there are only finitely many processes of a given finite norm and the finiteness of $\mathcal{R}_2$ follows from **Lemma 2.35**. Thus $\mathcal{R}$ is finite.

As $\mathcal{R}$ contains only bisimilar pairs, $\mathcal{R} \subseteq \sim$ and thus $\overset{\mathcal{R}}{\equiv} \subseteq \sim$ because $\sim$ is a congruence with respect to sequential composition. The other direction, $\sim \subseteq \overset{\mathcal{R}}{\equiv}$, is then checked separately for decomposable, resp. not decomposable, bisimilar pairs (XP, YQ) by induction on $\sqsubseteq$. $\square$

To conclude the semidecision procedure we note that it is semidecidable for a finite relation $\mathcal{R}$ whether it is a Caucal base as the membership test in $\overset{\mathcal{R}}{\equiv}$ is also semidecidable. The algorithm for semideciding bisimilarity then proceeds by generating finite relations and testing for a Caucal base. This procedure combined with the algorithm for semideciding non-bisimilarity (**Subsection 2.3.4**) yields a decision procedure, although no complexity bound can be obtained in this way.

For more detailed treatment consult [10], [11], [29]. An improvement on this approach is presented in [4] where an actual bisimulation base is constructed thus yielding an elementary decision procedure, with the complexity estimated as doubly exponential.

2.5 Decidability of strong bisimilarity on BPP

The first decidability result for strong bisimilarity of BPP was a decision procedure for the subclasses of normed and live BPP by Christensen, Hirshfeld and Moller in [7]. That was later extended to the whole class of BPP by the same authors in [8].

For the subclass of normed BPP, Hirshfeld, Jerrum and Moller constructed a polynomial algorithm in [29], [30]. For the class of general BPP, there appeared various other decidability results following different techniques. There is a decision procedure that makes use of tableaux by Hirshfeld and Moller [32], or a method based on semilinear sets suggested by Jančar in [40], [41]. However, without further assumptions it is not possible to place any primitive recursive upper bound on the computational complexity of deciding bisimilarity of BPP.

Hirshfeld in [27] presented an elegant method of testing bisimilarity by constructing trees that yield a finite representation of bisimulation if used for bisimilar pairs. This technique is quite general and can be used for various process algebras and different notions of bisimulation. We will give a brief overview of his technique.

2.5.1 Bisimulation trees

The commutative nature of BPP in general requires the use of different techniques from those that we could apply to BPA-processes. Here the method of *bisimulation trees* devised by Hirshfeld is introduced. All the definitions and results of this section are taken from [27].

When we consider parallel composition it is no longer true that if X is unnormed then for any process P, $X\|P \sim X$. Therefore we need to take into account processes that may contain more unnormed atoms. Adding to that, there is no finiteness theorem for non-decomposable pairs (although we can prove the existence of a finite base for bisimilarity) and thus we have to follow a different path. We will construct a tree - a witness for bisimilarity, which will be always finite and will contain a successful branch if and only if the input is a bisimilar pair. The nodes of the tree will be formed by finite sets, slightly refined versions

of *expansions*:

Definition 2.37 *Let A be a finite set of pairs of Basic Parallel Processes. A set A' of pairs of BPP is an* expansion *of A if*

- *for every pair (P,Q) in A and for every derivation $P \xrightarrow{\mu} P'$ there is a derivation $Q \xrightarrow{\mu} Q'$ with $(P',Q') \in A'$;*
- *for every pair (P,Q) in A and for every derivation $Q \xrightarrow{\mu} Q'$ there is a derivation $P \xrightarrow{\mu} P'$ with $(P',Q') \in A'$;*
- *A' is minimal, that is no proper subset of A' satisfies these two conditions.*

We say that A has an expansion in R if some expansion of A is a subset of R. If we start off from a bisimilar pair then there must be a (possibly infinite) sequence of expansions whose union yields a bisimulation relating the original pair. Those expansions correspond to correct derivations of the input pair. We will put that idea into practise and construct a tree whose nodes correspond almost exactly to expansion sets.

Definition 2.38 *A* bisimulation tree *for a pair of processes (P,Q) is a tree whose nodes are labelled by finite sets of pairs such that the root is labelled by the singleton $\{(P,Q)\}$ and the sons of each node are labelled by different expansions of the set of pairs at this node, with this modification: we do not include any pairs that have occurred in some ancestor node and every pair (P,P) is omitted. A leaf of a bisimulation tree is* successful *if it is labelled by the empty set. A leaf of a bisimulation tree is* unsuccessful *if it is not empty and yet has no expansion. A branch is* successful *if it ends with a successful node or if it is infinite.*

Every pair of the form (P,P) constitutes bisimilar processes and hence it is not necessary to include such pairs in the construction of a bisimulation witness. If we have already considered a pair (P,Q) at some point then by the definition of expansions we have already examined and included all possible matching derivatives and so we do not need to check this pair again if it appears later on.

Since every BPP has only finitely many derivatives in one step, every finite set of pairs of processes has finitely many possible expansions and every expansion is finite. Therefore every bisimulation tree is finitely branching. It is not difficult to see that the definition yields the following characterisation:

Proposition 2.39 *P and Q are bisimilar if and only if their bisimulation tree has a successful branch. In fact, the union of the sets labelling the nodes of a successful branch together with all the pairs (P,P) forms a bisimulation.*

The bisimulation tree as it is defined can still contain infinite branches. It is the unbounded growth of the processes which may make the branches in a tree infinite. We shall use an idea similar to decomposition which will ensure that the sizes of the processes that may occur in the tree will be bounded.

We will use the natural correspondence between BPP and vectors over natural numbers that was explained in 2.2.2.1. Assuming that the size of the set of variables Σ is k, we will represent a BPP as a vector in $\mathbb{N}^k$. Then we say that P *dominates* Q if there is a process R such that $P = Q\|R$. Equivalently, P dominates Q if P is greater than Q in the product order. For pairs of vectors, we say that (P, Q) dominates (R, S) if P dominates R and Q dominates S. Finally, a sequence of BPP is *proper* if it does not contain a process dominating a process previously occurring in the sequence. Proper sequences cannot go on forever which was originally formulated and proved by Dickson in [13]. We will also refer to the following lemma as Dickson's lemma.

Lemma 2.40 *Every proper sequence of BPP is finite.*

In order to remove infinite branches, we will modify the construction of the bisimulation tree. The idea is that "big" processes will be replaced with smaller ones so that pairs of processes from nodes along a branch will determine proper sequences. That will ensure finite lengths of branches of the tree.

We start from the root and assuming we have modified up to some node, we carry out the modification of its sons. If a son includes a pair $(P\|R, Q\|S)$ which dominates a previously occurring pair (P, Q) then we consider the pairs $(P\|R, P\|S)$ and $(Q\|R, Q\|S)$. If either of them does not dominate (P, Q) then we replace $(P\|R, Q\|S)$ with the respective pair.

However, it may still happen that both pairs dominate (P, Q). Then we make use of the fact that either $(P\|R, P\|S)$ or $(Q\|R, Q\|S)$ is less than $(P\|R, Q\|S)$ in the lexicographic order which was stated in **Lemma 2.18**. Assuming it is $(P\|R, P\|S) <_{\text{Lex}} (P\|R, Q\|S)$ we continue the modification with $(P\|R, P\|S)$ (still with respect to (P, Q)). Thus we are forming a decreasing sequence of pairs which will be finite as $<_{\text{Lex}}$ is a well-order. Therefore we will eventually come across a pair (P', Q') that will not dominate (P, Q).

However, it may happen that the new pair still dominates another previously occurring pair. As there are only finitely many of such pairs to consider this algorithm will stop after a finite number of modification steps. The outcome will be a pair that does not dominate any pair occurring at a previous node along the branch and it will be the replacement for the original pair $(P\|R, Q\|S)$.

In order to show that the modification preserves the soundness of the tree we will need the following lemma:

Lemma 2.41 *Assuming that $P \sim Q$, then for every R and S, $P\|R \sim Q\|S$ iff $P\|R \sim P\|S$ iff $Q\|R \sim Q\|S$.*

To verify the lemma we would employ the property of bisimulation being a congruence with respect to parallel composition. If $P \sim Q$ then for any R and S also $P\|R \sim Q\|R$ and $P\|S \sim Q\|S$. That combined with the assumption of $P\|R \sim Q\|S$ and $\sim$ being symmetric and transitive implies that also $P\|R \sim P\|S$ and $Q\|R \sim Q\|S$. The other implications would be deduced in a similar way.

Analogously we would show that if the dominated pair does not consist of bisimilar processes then modification does not add processes that would produce a bisimulation witness. All successors of a node containing a non-bisimilar pair will eventually fail, i.e. reach a (non-empty) set that has no expansion. Therefore the soundness of the tree is preserved.

The modification of the bisimulation tree that we have just described ensures that all nodes alongside every branch determine proper sequences of processes. Then the application of Dickson's lemma ensures that all branches have to be finite. **Lemma 2.41** ensures that the modification preserves the correctness and so the modified bisimulation tree has a successful branch if and only if the root is labelled by a bisimilar pair of processes. In fact, the union of the nodes alongside a successful branch forms a finite Caucal base for a bisimulation.

Finally we will remark on the complexity of this decision procedure. There are two factors to consider. The first is the size of the branching at each node which is bounded by a function exponential in the size of the input algebra. The second is the upper bound on the lengths of the branches. The only fact concerning the lengths of branches available to us is Dickson's lemma. McAloon in [47] found an effective upper bound on the length of a maximal proper sequence which unfortunately is not even primitive recursive.

For normed BPP there exists a polynomial algorithm deciding bisimilarity. It makes use of Caucal bases and decomposition into primes and although the technique is rather different than polynomial decision procedure for BPA-processes, the idea also involves constructing a series of bases which starts from a large base and is eventually pruned down to a bisimulation base.

2.6 Decidability problem for weak bisimilarity

When we move into the realm of weak bisimilarity we encounter more complex behaviour even in the case of processes defined by a few simple operators. By allowing any number of τ transitions within a single step we lose the image-finiteness since now processes have the capability of evolving into one of potentially infinitely many options. That brings in the problem that we may not be able to test a property for all derivatives of a process unless they can be somehow represented by a finite number of *base processes*. We shall see that that is the case for BPP where the set of derivatives of a BPP forms a semilinear set which has a finite characterisation. This fact will form a basis for a semidecision procedure for weak bisimilarity.

Another problem that we encounter is semideciding weak non-bisimilarity. There does not seem to exist any straightforward semidecision procedure for $\not\approx$ that would work along the lines of the semidecision procedure for strong non-bisimilarity. Just to remind ourselves, the algorithm for semideciding $\not\sim$ consecutively enumerated and tested approximants $\sim_n$ which are decidable for both BPA and BPPA, and converge with the limit being $\sim$. In **Chapter 3** we will define an analogue of strong bisimulation approximants for weak bisimulation, *weak bisimulation approximants* $\approx_\alpha$. However, we shall see that the equality $\approx = \bigcap_{n\in\mathbb{N}} \approx_n$ does not hold for general BPA-processes and BPP which is just another consequence of the capability to perform infinite branching. We will provide more explanation and examples in **Chapter 3**.

We have discussed the concept of norm around which the current decidability techniques for $\sim$ centre. For weak norm and the class of totally normed processes we will show that we can actually recover enough of the results that work for strong bisimilarity to construct a decision procedure.

2.6.1 Decidability of $\approx$ for totally normed BPA and BPP

Decidability of weak bisimilarity for totally normed process algebras was demonstrated by Hirshfeld in [28]. Totally normed process algebras consist of process variables whose weak norm is finite and positive. We will therefore assume that we are given some algebra (BPA or BPPA) with a set of atoms Σ and set of transitions Δ, where for every $X \in \Sigma$, $0 < \|X\| < \infty$. Hirshfeld noticed that for weak bisimulation on such algebras one can use similar techniques as for strong bisimulation.

When we disallow atomic processes to have norm zero then there remains

only one process of norm zero which is the empty process ϵ. Then all other processes have a positive norm and, as a consequence of additivity of norm, the set of processes of a fixed finite norm is finite. As weak bisimilarity is consistent with weak norm we can request the condition that matching derivatives should be of equal norm. Thus we will be able to adjust the technique of bisimulation trees to work for weak bisimilarity. We shall define weak expansion and weak bisimulation trees with appropriate modifications that reflect the character of weak bisimilarity.

Definition 2.42 *Let A be a finite set of pairs of Basic Parallel Processes. A set A' of pairs of BPP is a* weak expansion *of A if*

- *for every pair (P,Q) in A and for every derivation $P \xrightarrow{\mu} P'$ there is a derivation $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $(P',Q') \in A'$;*
- *for every pair (P,Q) in A and for every derivation $Q \xrightarrow{\mu} Q'$ there is a derivation $P \stackrel{\mu}{\Longrightarrow} P'$ with $(P',Q') \in A'$;*
- *A' is minimal, that is no proper subset of A' satisfies these two conditions.*

We define a weak bisimulation tree exactly as for strong bisimilarity with the following alterations: expansions are replaced with weak expansions, and we only consider matching derivatives of equal weak norm. It is not difficult to see that if we start off with a bisimilar pair then the constructed tree contains a successful branch and the union of weak expansions along that branch together with all pairs (P,P) forms a weak bisimulation.

For this particular case of totally normed BPP, the bisimulation tree is finitely branching. The reason for that is the requirement that the weak norm be preserved, and for totally normed processes, there are only finitely many processes of a given norm. Hence if we are at a node and we consider a pair (P,Q), then for each derivation $P \xrightarrow{\mu} P'$ (and there are only finitely many of those) there exist only finitely many matching derivations $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $\|P'\| = \|Q'\|$. Therefore the tree we are constructing is finitely branching with nodes being labelled with finite weak expansions.

As for strong bisimilarity, the original tree may contain infinite branches. The way around that is to replace dominating pairs with smaller pairs so that the branches correspond to proper sequences. Dickson's lemma then takes care of finiteness. The modification for weak bisimulation works precisely as for the strong case. To justify the soundness of the modification we make use of the

following lemma, a weak bisimulation analogue of **Lemma 2.41**. More details can be found in [28].

Lemma 2.43 *Assuming that $P \approx Q$, then for every R and S, $P\|R \approx Q\|S$ iff $P\|R \approx P\|S$ iff $Q\|R \approx Q\|S$.*

For totally normed BPA-processes the situation is rather more complicated but still the technique of bisimulation trees works. On the other hand, we can make use of a result proved by Stirling in [58] which shows decidability of strong bisimilarity of normed pushdown automata (PDA).

Another relevant result regarding weak bisimulation of BPA-processes is that of Sénizergues [57] which settles decidability of bisimilarity for a larger class of processes that includes all pushdown processes. When pushdown processes are enriched with arbitrary ϵ-transitions then weak bisimilarity of BPA-processes corresponds to strong bisimilarity of PDA. However, in the class of [57] ϵ-moves are restricted in such a way that it covers only finitely branching BPA-processes, i.e. processes which by a $\stackrel{\mu}{\Longrightarrow}$ transition can create only a finite class of derivatives. That naturally implies that the question of decidability of $\approx$ cannot be settled by this approach.

2.6.2 Semidecidability of $\approx$ for general BPP

The last technique we wish to discuss applies solely to commutative algebras. It was suggested as a method to decide strong bisimilarity for BPP by Jančar in [40] and developed to deal with weak bisimilarity by Esparza in [16], [15]. The underlying idea is to combine the facts that every congruence in a commutative monoid is finitely generated and that the maximal strong (and weak) bisimulation on BPP is a congruence. That exploits the multiset character of BPP which will be again represented as vectors of natural numbers.

We assume that our BPP-algebra consists of a set of atoms $\Sigma = \{X_1, \ldots, X_k\}$, and a set of transitions Δ. Again we will identify processes with vectors from $\mathbb{N}$ and a pair of processes with a vector from $\mathbb{N}^{2k}$. The set $\mathbb{N}^{2k}$ together with the operation of vector addition is a commutative monoid. We know that the largest weak bisimulation is a congruence, that means $\approx$ is an equivalence relation such that $P \approx Q$ and $R \approx S$ implies that $P\|R \approx Q\|S$. Parallel composition can be directly translated as vector addition and hence it is straightforward that $\approx$ is a congruence when expressed as a subset of $\mathbb{N}^{2k}$. Then we will show that $\approx$ is finitely generated as a *semilinear set* and we will show that membership in a semilinear set can be encoded by a formula from *Presburger arithmetic*, a decidable first

order theory of addition. Now we are going to explain all necessary notions and state all relevant theorems.

Definition 2.44 *A subset A of $\mathbb{N}^k$ is* linear *if there exist vectors $v_0, v_1, \ldots, v_n$ in $\mathbb{N}^k$ such that $A = \{v_0 + a_1 v_1 + \ldots + a_n v_n \mid a_1, \ldots, a_n \in \mathbb{N}\}$. A set is* semilinear *if it is a finite union of linear sets.*

Theorem 2.45 ([14]) *Every congruence in a finitely generated commutative monoid M is a semilinear subset of $M \times M$.*

A corollary of this theorem is the following:

Corollary 2.46 *The largest weak bisimulation $\approx$ on $(\Sigma^{\otimes}, \Delta)$ is a semilinear subset of $\mathbb{N}^k \times \mathbb{N}^k$.*

If we translate this result into the process language it says that there exists a finite set of sequences of pairs of BPP $\Big\{\langle (P_{10}, Q_{10}), (P_{11}, Q_{11}) \ldots, (P_{1i_1}, Q_{1i_1})\rangle, \ldots, \langle (P_{n0}, Q_{n0}), (P_{n1}, Q_{n1}), \ldots, (P_{ni_n}, Q_{ni_n})\rangle\Big\}$ such that for each bisimilar pair (P, Q) there exists a $j \leq n$ so that

$$(P, Q) = (P_{j0}, Q_{j0}) + \sum_{i=1}^{n_j} a_i (P_{ji}, Q_{ji}), \text{ for some } a_1, \ldots, a_{n_j} \in \mathbb{N}.$$

Semilinear sets can be encoded as formulas of *Presburger arithmetic*. Presburger arithmetic is the first order theory of addition with formulas that are built out of variables, logical connectives, quantifiers and the symbols $0, \leq$ and $+$. Formulas are interpreted on the natural numbers and the symbols are interpreted as the number 0, the natural total order on $\mathbb{N}$ and addition. The decidability of Presburger arithmetic was originally proved by Presburger in [56], for other references consult [24], [19].

Theorem 2.47 *It is decidable whether an arbitrary Presburger sentence is true.*

What it means to encode a set of vectors as a Presburger formula is defined below:

Definition 2.48 *A subset $A \subseteq \mathbb{N}^k$ is* expressible *in Presburger arithmetic if there exists a Presburger formula $A(x_1, \ldots, x_k)$ with free variables $x_1, \ldots, x_k$ such that for every $(n_1, \ldots, n_k) \in \mathbb{N}^k$, the closed formula $A(n_1, \ldots, n_k)$ is true if and only if $(n_1, \ldots, n_k) \in A$.*

And the final connection between formulas and semilinear sets is expressed in a theorem by Ginsburg and Spanier:

Theorem 2.49 ([19]) *A subset $A \subseteq \mathbb{N}^k$ is semilinear iff it is expressible in Presburger arithmetic. Moreover, the transformations between semilinear sets and Presburger formulas are effective.*

We will use this fact to show that we can encode the property of being a bisimulation as a Presburger formula. That will enable us to check whether a binary relation, given by its finite representation as a semilinear set, is a bisimulation. To do that we will make use of the following proposition (consult [16] for details):

Proposition 2.50 *For a fixed BPP-algebra and a fixed action μ, the set $\{(P, P') \mid P \stackrel{\mu}{\Longrightarrow} P'\}$ is semilinear.*

We will not specify the details of the proof here but for a fixed action μ we can actually construct a formula $\phi_\mu(\vec{x}, \vec{y})$, where $\vec{x} = (x_1, \ldots, x_k), \vec{y} = (y_1, \ldots, y_k)$ are vectors of variables, so that

$$\phi_\mu(P, P') \equiv P \stackrel{\mu}{\Longrightarrow} P'$$

Assuming a given binary semilinear set $\mathcal{R}$, we know that it is expressible as a formula of Presburger arithmetic (**Theorem 2.49**) and hence we can construct a formula ψ of $2k$ free variables so that $\psi(\vec{x}, \vec{y}) \equiv (\vec{x}, \vec{y}) \in \mathcal{R}$. Then we can put the two formulas together and define a closed formula $\Phi(\mathcal{R})$ which will be true if and only if the relation $\mathcal{R}$ is a weak bisimulation:

$$\begin{array}{lll} \Phi(\mathcal{R}) \equiv \forall \vec{x}, \vec{y} \in \mathbb{N}^k. & \psi(\vec{x}, \vec{y}) & \Rightarrow \\ & \bigwedge_{\mu \in L} \Big(& \forall \vec{x'}.\ [\phi_\mu(\vec{x}, \vec{x'}) \Rightarrow \exists \vec{y'}.\ \phi_\mu(\vec{y}, \vec{y'}) \wedge \psi(\vec{x'}, \vec{y'})] \wedge \\ & & \forall \vec{y'}.\ [\phi_\mu(\vec{y}, \vec{y'}) \Rightarrow \exists \vec{x'}.\ \phi_\mu(\vec{x}, \vec{x'}) \wedge \psi(\vec{x'}, \vec{y'})] \ \Big), \end{array}$$

where L is a finite set of actions that occur in the definition of the BPP-algebra in question.

Therefore we can decide whether a given binary relation is a weak bisimulation. We can now present the sketch of the semidecision procedure where we make use of the standard fact that we can recursively enumerate all finite sets, hence also all (generators of) semilinear sets.

1. Input a pair of BPP P and Q.
2. Compute an effective enumeration of all semilinear sets $\mathcal{S}_1$, $\mathcal{S}_2$, ..., such that each $\mathcal{S}_i$ contains the pair (P, Q).
3. Check $\Phi(\mathcal{S}_i)$ until you find i such that the formula holds.

Clearly if $P \approx Q$ then there exists a weak bisimulation relating the two processes and eventually we will generate the corresponding semilinear set and successfully terminate. On the other hand, we have no means of encoding maximality into Presburger arithmetic which means that we are not able to test for the largest weak bisimulation. For that reason this method cannot verify that two processes are not weakly bisimilar.

We already know that the straightforward semidecision procedure for non-bisimilarity does not work for infinitely branching processes and hence for weak bisimilarity. Thus we are confronted with the curious case where we can semidecide bisimilarity but we cannot say anything more specific about verifying non-bisimilarity.

2.6.2.1 Bisimulation trees for general BPP

Another technique that can be applied to general BPP to obtain semidecidability is Hirshfeld's method of bisimulation trees. The construction is essentially the same as for totally normed BPP. The weak bisimulation tree consists of nodes labelled by weak expansions. These are always finite, as in the case of totally normed BPP, simply because for a pair (P, Q) from some node we consider only derivatives (P', Q') where either $P \xrightarrow{\mu} P'$ and $Q \stackrel{\mu}{\Longrightarrow} Q'$ or $Q \xrightarrow{\mu} Q'$ and $P \stackrel{\mu}{\Longrightarrow} P'$. However, since there may be an infinite number of possibilities of a matching derivative to some move even when we require that they preserve weak norm, there may be an infinite number of possible weak expansions for some set (node) and the bisimulation tree may be infinitely branching.

We can ensure finite lengths of branches by never including any dominating pair, as in the case of (totally normed) BPP. Therefore the only obstacle that we are facing is the potential infinite branching which cannot be prevented. Still the finite depth of the tree enables us to search the tree by *dove-tailing* for a potential bisimulation witness, in the form of an empty leaf.

The dove-tailing technique can be briefly summed up as follows. The nodes of the tree that is being constructed are labelled by sequences of natural numbers in this fashion: the root is labelled by 1; if a node has label $n_1 n_2 \ldots n_k$ then its sons have labels $n_1 n_2 \ldots n_k 1$, $n_1 n_2 \ldots n_k 2$, ..., $n_1 n_2 \ldots n_k i$, and so on. The algorithm for every $i \in \mathbb{N}$ constructs nodes whose labels add up to i, i.e. in the first step the root is computed, then the first son labelled by 11, next the first son of 11, labelled by 111, and the second son of the root, labelled by 12, etc. Since for every i there are only finitely many possible sequences adding up to it, and the nodes are finite, the iteration for every i is finite and each node of the tree will

be eventually computed. The algorithm will terminate if it generates an empty node (leaf).

Obviously, the question that immediately arises in this context is whether one can somehow curb infinite branching. One way of conquering the obstacle of infinite branching would finding some *upper bound* N such that for any bisimilar pair (P, Q) and any move $P \xrightarrow{\mu} P'$, there would exist a matching derivative $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx Q'$ such that the size of Q' would be bounded by N. The upper bound would be some function of the size of a given algebra, i.e. would depend on the number of atoms and sizes of defining transition rules, although it is not entirely clear what measure would be best suited for this purpose. If we had such a bound we would be able to preserve finiteness of the branching as there would be only finitely many weak expansions that would need to be checked. Analogously, this upper bound could be expressed in Presburger arithmetic so that we would only need to check bounded quantification instead of general quantification. Then we would immediately achieve decidability.

Chapter 3

Approximants

In this chapter we will develop an idea introduced by Milner in [48]. He defines an equivalence relation on processes which he calls *strong equivalence* $\sim$ and defines in terms of a decreasing sequence of equivalence relations (*approximants*) $\sim_0, \sim_1, \ldots, \sim_k, \ldots$. The definition of the approximants $\sim_k$ is stated in **Chapter 2**. Milner then puts $\sim$ to be $\bigcap_{k\in\omega} \sim_k$. That coincides with the alternative definition of strong bisimulation as presented in [49] on the classes of both Basic Process Algebras and Basic Parallel Process Algebras. The reason for that is the image-finiteness of these processes.

As we have already mentioned, the fact that the maximal strong bisimulation can be obtained as the limit of the chain of approximants labelled by natural numbers can be employed to yield a semidecision procedure for non-bisimilarity. Each approximant can be tested in a simple straightforward way, and if two processes are not bisimilar then they are not related by an approximant $\sim_k$ for some k.

We can adopt a similar approach towards weak bisimilarity. Analogously to the strong bisimulation approximants we can define a sequence of binary relations labelled by ordinal numbers that will approximate the maximal weak bisimulation relation from above. We will call these relations *weak bisimulation approximants*. The approximant labelled by 0 is the largest relation in the sequence and contains all pairs of processes. Approximants labelled by larger ordinals are defined in terms of smaller approximants so that the resulting sequence is non-increasing. Note that we need to go beyond natural numbers in this construction. The reason for that will become clear later.

We will show that in general this sequence converges to the maximal weak bisimulation. For the specific cases of BPA and BPPA we will search for the minimum ordinal that labels an approximant equal to $\approx$. Finally we will demonstrate that in the case of Basic Parallel Processes all approximants labelled by

natural numbers are decidable. In this chapter we will use notions and results from the theory of ordinal numbers and so we will commence with an informal revision of ordinal numbers.

3.1 Ordinal arithmetic

In this section we will present an informal overview of ordinal arithmetic (for more details consult [1], [21], [22], [45]). We assume that the reader is familiar with the notion of *ordinal numbers* as representatives of classes of well-ordered sets. Ordinal numbers form a class denoted by On with the least element being the empty set $\emptyset$. Ordinal numbers are constructed starting from $\emptyset$ by two operations: by taking an ordinal α and adding $\{\alpha\}$ to it to form a set $\alpha \cup \{\alpha\}$ which gives rise to an ordinal denoted by $\alpha + 1$, or by taking a union of a possibly infinite set of ordinals. The ordinal of the form $\alpha \cup \{\alpha\}$ is called a *successor* ordinal and α is its *predecessor*. An ordinal $\lambda \neq \emptyset$ which does not have a predecessor is called a *limit* ordinal. Ordinals themselves are well-ordered by the element-of relation which will be denoted by $<$. We will denote successor ordinals by Greek letters α, β, γ, and limit ordinals by λ, κ.

The ordinals we will be using form an initial segment of On and they start with the natural numbers $0, 1, 2, \ldots, n, \ldots$ together with the linear order. The limit of this chain is the first *limit* ordinal ω after which we continue in a similar fashion $\omega, \omega + 1, \omega + 2, \ldots, \omega + n, \ldots, \omega + \omega$, and so on.

Before we define some arithmetical operations on the class of ordinal numbers we will recall the notion of *lexicographic order* on On^2. We say that the pair (α_1, β_1) is less than (α_2, β_2) with respect to lexicographic order if and only if $\alpha_1 < \alpha_2$ or $(\alpha_1 = \alpha_2 \wedge \beta_1 < \beta_2)$. Now we are ready to spell out the definitions of ordinal *summation* and *multiplication*.

Definition 3.1 *The sum of ordinals α and β is an ordinal number denoted by $\alpha + \beta$ and defined as the representative of the set $(\{0\} \times \alpha) \cup (\{1\} \times \beta)$ under lexicographic order.*

Definition 3.2 *The multiplication of ordinals α and β is an ordinal number denoted by $\alpha \cdot \beta$ and defined as the representative of the set $\beta \times \alpha$ under lexicographic order.*

The asymmetry in the definition of ordinal multiplication is due to historical reasons. On natural numbers, the operations of ordinal summation and multiplication correspond to (natural) summation and multiplication. However, in

general these operations are not symmetric. Hence for example $1+\omega = \omega \neq \omega+1$, and $2 \cdot \omega = \omega \neq \omega \cdot 2$. We will pay particular attention to sums and multiples of ω so to clarify our notation let us note that we will abbreviate $\omega+\omega$ to $\omega \cdot 2$, etc., and $\omega \cdot \omega$ with ω^2, etc. It should be clear now what we mean by the expression $\omega^m e_m + \omega^{m-1} e_{m-1} + \ldots + \omega e_1 + e_0$ which will be abbreviated as $\sum_{i=0}^{m} \omega^i e_i$, where $e_0, \ldots, e_m$ are natural numbers and we assume that $e_m \neq 0$. In this thesis we will not get very far in the ordinal hierarchy, however we will encounter the ordinal ω^ω in some claims. It is not necessary to understand the structure of this ordinal and it suffices to consider ω^ω as a supremum (limit) of the sequence of ordinals $\omega, \omega^2, \omega^3, \ldots, \omega^n, \ldots$.

Before we move on to explain some proof methods that deal with ordinals we will state this important yet simple property concerning sums.

Proposition 3.3 *For two arbitrary sums,* $\sum_{i=0}^{m} \omega^i e_i = \sum_{i=0}^{n} \omega^i f_i$ *if and only if* $m = n$ *and* $e_i = f_i$ *for every* $i = 0, 1, \ldots, m$.

We will not concern ourselves with a proof here. Intuitively, the validity of this observation follows from the fact that ω^{n+1} is the limit of $\omega^n, \omega^n \cdot 2, \ldots, \omega^n \cdot k, \ldots$, and hence we cannot reach ω^{n+1} with any finite multiple of a smaller power of ω.

When we deal with the whole class of ordinals the common induction principle for natural numbers becomes too weak for proving theorems. We need a more powerful proof method than that and, fortunately, the well-ordered structure of ordinal numbers enables us to formulate a statement which is a generalisation of the induction principle. It is called *transfinite* or *ordinal induction.*

The Principle of Transfinite Induction: *Let* $P(\alpha)$ *be a statement for each ordinal* α. *Assume that*

1. $P(0)$
2. $P(\alpha) \Rightarrow P(\alpha+1)$ *for every* α
3. *if* λ *is a limit ordinal then* $(\forall \alpha < \lambda.\ P(\alpha)) \Rightarrow P(\lambda)$.

Then for every $\alpha \in On$, $P(\alpha)$.

Now if we want to verify that some property P holds for the class On we only have to test three cases: the base case $P(0)$, the successor case $P(\alpha) \Rightarrow P(\alpha+1)$ and the limit case $(\forall \alpha < \lambda.\ P(\alpha)) \Rightarrow P(\lambda)$. If we manage to prove all three cases we can be confident that all ordinals possess the desired property P.

The difference from the induction on $\mathbb{N}$ is obviously the case of a limit ordinal. In order to test that case we need to understand the structure of a limit ordinal and there is one special property that will prove to be useful. Clearly if we have a limit ordinal λ and any $\alpha < \lambda$, then also $\alpha + 1 < \lambda$, and so on. Thus there always exists an ordinal β such that $\alpha < \beta < \lambda$ and, in fact, there exists an infinite increasing sequence $\beta_0 < \beta_1 < \ldots < \beta_n < \ldots$ with $\beta_i < \lambda$ for every $i \in \mathbb{N}$.

We will also touch briefly on the subject of *cardinality* of ordinal numbers. A set X is called *countable* if there exists a one-to-one mapping from X to ω. All the ordinals $0, 1, \ldots, \omega, \omega \cdot 2, \ldots, \omega^2, \ldots, \omega^n, \ldots, \omega^\omega, \ldots$ are countable. The first *uncountable* ordinal is denoted by ω_1. Later we will make use of the following fact: assume that we have a sequence of sets $\{X_\alpha \mid \alpha < \omega_1\}$ such that the set X_0 is countable and $X_\alpha \subseteq X_\beta$ for every $\beta < \alpha$. Then there must be a $\beta < \omega_1$ so that $X_\beta = X_\gamma$ for every $\beta \leq \gamma < \omega_1$. Expressed in other terms, there is no uncountable strictly decreasing sequence of countable sets.

3.2 Weak bisimulation approximants

In this section we will state several possible ways of defining weak bisimulation approximants. The individual definitions differ in the type of moves we require the processes to perform (a single transition $\stackrel{a}{\longrightarrow}$, an action augmented with τ actions $\stackrel{a}{\Longrightarrow}$ or a sequence of such actions $\stackrel{t}{\Longrightarrow}$). Accordingly to the type of transition the different definitions also possess different attributes.

In the definitions below we assume a fixed labelled transition graph $(\mathcal{S}, A, \longrightarrow)$ where the set of labels A contains the silent action τ. The processes range over $\mathcal{S}$ and the actions range over A.

Definition 3.4 Weak bisimulation approximants $\approx^s_\alpha$

- $P \approx^s_0 Q$ *for all* P *and* Q
- $P \approx^s_{\alpha+1} Q$ *if for all actions* a,
  - *whenever* $P \stackrel{a}{\longrightarrow} P'$ *then there exists* $Q \stackrel{a}{\Longrightarrow} Q'$ *so that* $P' \approx^s_\alpha Q'$ *and*
  - *whenever* $Q \stackrel{a}{\longrightarrow} Q'$ *then there exists* $P \stackrel{a}{\Longrightarrow} P'$ *so that* $P' \approx^s_\alpha Q'$
- $P \approx^s_\lambda Q$ *if* $P \approx^s_\alpha Q$ *for every* $\alpha < \lambda$, *for a limit ordinal* λ.

Definition 3.5 Weak bisimulation approximants $\approx_\alpha$

- $P \approx_0 Q$ *for all* P *and* Q

- $P \approx_{\alpha+1} Q$ *if for all actions* a,
  - *whenever* $P \stackrel{a}{\Longrightarrow} P'$ *then there exists* $Q \stackrel{a}{\Longrightarrow} Q'$ *so that* $P' \approx_\alpha Q'$ *and*
  - *whenever* $Q \stackrel{a}{\Longrightarrow} Q'$ *then there exists* $P \stackrel{a}{\Longrightarrow} P'$ *so that* $P' \approx_\alpha Q'$
- $P \approx_\lambda Q$ *if* $P \approx_\alpha Q$ *for every* $\alpha < \lambda$, *for a limit ordinal* λ.

Definition 3.6 Weak bisimulation approximants $\approx^M_\alpha$

- $P \approx^M_0 Q$ *for all* P *and* Q
- $P \approx^M_{\alpha+1} Q$ *if for every sequence of actions* t,
  - *whenever* $P \stackrel{t}{\Longrightarrow} P'$ *then there exists* $Q \stackrel{t}{\Longrightarrow} Q'$ *so that* $P' \approx^M_\alpha Q'$ *and*
  - *whenever* $Q \stackrel{t}{\Longrightarrow} Q'$ *then there exists* $P \stackrel{t}{\Longrightarrow} P'$ *so that* $P' \approx^M_\alpha Q'$
- $P \approx^M_\lambda Q$ *if* $P \approx^M_\alpha Q$ *for every* $\alpha < \lambda$, *for a limit ordinal* λ.

The three definitions determine different types of approximation. The approximants $\approx^s_\alpha$ are asymmetric in that only single transitions are included in the premise. The relations $\approx_\alpha$ are symmetric and composite transitions are inspected as both initial and matching moves. The last relations $\approx^M_\alpha$ are consistent with the original Milner's definition of strong bisimulation approximants where sequences of moves are considered rather than single moves. We can observe some simple relations between the various approximants and also between the individual approximants and the maximal weak bisimulation $\approx$ as defined in **Chapter 2**. To make things simpler we will state the lemma below in terms of **Definition 3.5**, however it holds for **Definitions 3.4** and **3.6** as well.

Lemma 3.7

1. *for every* $\alpha \in On$, $\approx^M_\alpha \subseteq \approx_\alpha \subseteq \approx^s_\alpha$
2. *for every* $\alpha, \beta \in On$, $\alpha < \beta \Rightarrow \approx_\beta \subseteq \approx_\alpha$
3. *for every* $\alpha \in On$, $\approx \subseteq \approx_\alpha$
4. *if there is an* α *such that* $\approx_\alpha = \approx_{\alpha+1}$ *then for all* $\beta \geq \alpha$, $\approx_\alpha = \approx_\beta = \approx$
5. $\approx = \bigcap_{\alpha \in On} \approx_\alpha$
6. *for BPA and BPPA,* $\approx = \approx_{\omega_1}$

We will not directly call upon any of the claims above hence the proofs are placed in **Appendix A**. We can interpret the statements as follows: claim 1. says that $\approx_\alpha^M$ determine the strictest relations, $\approx_\alpha^s$ the weakest and $\approx_\alpha$ are between the two. Claims 2. and 3. assert that all three definitions determine non-increasing chains of approximants, and each approximant contains the maximal weak bisimulation. Claims 4., 5. and 6. resolve convergence towards weak bisimulation, with 6. asserting that for BPA and BPP, convergence occurs at most at the level $\approx_{\omega_1}$.

Now we will compare the above definitions of weak bisimulation approximants. **Definition 3.4** is an analogue of the original definition of a weak bisimulation relation as stated in [49]. Algorithmically it is the simplest definition of the three although the double arrow in the second part may still cause infinite behaviour. Still, it would make proofs easier if it was not for the fact that the resulting relations fail to be equivalences. The mismatch between the single and double arrows means that the approximants $\approx_\alpha^s$ fail to be transitive which is shown in the following example:

Example 3.8 We will define the following BPA:

$$A \xrightarrow{a} \epsilon \qquad A \xrightarrow{\tau} \epsilon \qquad C \xrightarrow{\tau} CA \qquad C \xrightarrow{\tau} \epsilon \qquad B \xrightarrow{a} B \qquad B \xrightarrow{\tau} \epsilon$$

We will illustrate that the approximants $\approx_k^s$ fail to be transitive for all $k \geq 3$. We will demonstrate that by showing that $B \approx_k^s A^k$ and $A^k \approx_k^s CA$ for every k but $B \not\approx_3^s CA$. The two equivalences can be easily verified by induction hence we will only concentrate on the fact that B fails to be $\approx_3^s$-equivalent with CA.

We make the process CA do τ and become A. The variable B can either respond by doing $B \xrightarrow{\tau} \epsilon$ but then $A \not\approx_1^s \epsilon$ or $B \stackrel{\epsilon}{\Longrightarrow} B$. Now B can perform a sequence of a actions of an arbitrary length, however A is capable of only one a transition after which it evolves into ϵ. Therefore we come to the conclusion that $A \not\approx_2^s B$ hence $B \not\approx_3^s CA$ and $B \not\approx_k^s CA$ for all $k \geq 3$. With **Definition 3.5** this could never occur because of this simple fact: $A^k \not\approx_3 B$ for any $k \geq 2$. The reason for that is the possibility of doing τ actions in the preamble so A^k can perform $\stackrel{a}{\Longrightarrow}$ and become a single copy A. The response of B is either $B \stackrel{a}{\Longrightarrow} \epsilon$ with $A \not\approx_1 \epsilon$ or $B \xrightarrow{a} B$ with $A \not\approx_2 B$ hence, finally, $A^k \not\approx_3 B$. □

The approximants defined by **3.5** and **3.6** are equivalence relations which is not difficult to verify. The last **Definition 3.6** possesses similar disadvantage as the analogous **Definition 2.4** of strong bisimulation approximants - it may not be feasible to check a property for all sequences of actions. This can be very well demonstrated on the example of BPP. We can check whether a process is

reachable from another process via a sequence of actions, more specifically we can decide the membership in the set $\{(P, P') \mid P \stackrel{w}{\Longrightarrow} P'\}$ for a given word w ([16]). However, the test over all possible sequences w causes a problem. Hirshfeld showed in [26] that trace equivalence ($\sim_1^M$) for BPP is undecidable. Later Hüttel generalised this result by proving that all strong bisimulation approximants $\sim_n^M$ are undecidable (see [36]). It is a straightforward generalisation that weak bisimulation approximants possess the same property, i.e. $\approx_n^M$ are undecidable for every $n > 0$. On the other hand, we will establish later that for BPP, the approximants $\approx_n$ are decidable. Hence we have chosen to work with weak bisimulation approximants as defined in **3.5** and we will assume that throughout the following text unless stated otherwise.

3.2.1 Congruence

Another property we shall discuss is *congruence*. The approximants $\approx_\alpha$ are congruence relations with respect to Basic Parallel Processes because of the commutativity of parallel composition. For BPA-processes $\approx_\alpha$ are not congruences in general. Clearly, if $P \approx_\alpha Q$ then also $RP \approx_\alpha RQ$ but the opposite direction does not always hold.

Lemma 3.9 (Congruence) *Assume that a Basic Process Algebra (Σ^*, Δ) is such that every process variable $X \in \Sigma$ satisfies the following condition:*

- *if X is weakly bisimilar to the empty process ϵ then X can evolve into ϵ with $a \stackrel{\tau}{\Longrightarrow}$ move.*

Then for every ordinal number α and processes P, Q and R from Σ^*, if $P \approx_\alpha Q$ then also $PR \approx_\alpha QR$.

Proof: We will prove this statement in full generality using transfinite induction on α. For $\alpha = 0$ the claim holds trivially as all processes are equivalent at level 0. Now assuming that the claim holds for some α we will prove that it also holds for its successor $\alpha + 1$. We presuppose a BPA satisfying the required condition and three BPA-processes P, Q and R with $P \approx_{\alpha+1} Q$. In order to demonstrate that $PR \approx_{\alpha+1} QR$ we have to check all possible moves of PR and QR. When PR performs a move $PR \stackrel{\mu}{\Longrightarrow} P'$ there may occur two different situations:

1. The process P is not exhausted and there is a remainder $\bar{P}$ such that $P' = \bar{P}R$. In this case we use the assumption that $P \approx_{\alpha+1} Q$ to conclude that

there must be a response $Q \stackrel{\mu}{\Longrightarrow} \bar{Q}$ with $\bar{P} \approx_\alpha \bar{Q}$. Since $\bar{Q}$ is an $\stackrel{\mu}{\Longrightarrow}$ derivative of Q then $\bar{Q}R$ is an $\stackrel{\mu}{\Longrightarrow}$ derivative of QR and it follows from the induction hypothesis that these composites $\bar{P}R$ and $\bar{Q}R$ are equivalent at level α.

2. The process P is exhausted and P' is actually equal to some R'. Either the derivation sequence is $PR \stackrel{\mu}{\Longrightarrow} R \stackrel{\tau}{\Longrightarrow} R'$ or $PR \stackrel{\tau}{\Longrightarrow} R \stackrel{\mu}{\Longrightarrow} R'$. The former is analogous to the previous case so we assume that $P \stackrel{\tau}{\Longrightarrow} \epsilon$ and $R \stackrel{\mu}{\Longrightarrow} R'$. From the assumption that $P \approx_{\alpha+1} Q$ we have that there must be a matching response of Q in the form $Q \stackrel{\tau}{\Longrightarrow} Q'$ such that $\epsilon \approx_\alpha Q'$. The empty process ϵ cannot perform any action at all and so there cannot be any visible transition available for Q'. Since we assume that every process weakly bisimilar to ϵ must have the possibility to become ϵ by performing $\stackrel{\tau}{\Longrightarrow}$ then there must be a finite sequence of τ moves leading from Q' to ϵ. Therefore the composite QR has the ability to get rid of Q and become R' with the sequence of moves $QR \stackrel{\tau}{\Longrightarrow} Q'R \stackrel{\tau}{\Longrightarrow} R \stackrel{\mu}{\Longrightarrow} R'$. Since the relations $\approx_\alpha$ are equivalences we can conclude the proof with the fact that $R' \approx_\alpha R'$.

The analysis of the moves of the process QR is symmetrical and hence we can draw the conclusion that indeed $PR \approx_{\alpha+1} QR$.

The analysis of the limit case is straightforward. If $P \approx_\lambda Q$ for a limit ordinal λ then for all $\alpha < \lambda$, $P \approx_\alpha Q$. Then also $PR \approx_\alpha QR$ for any process R and hence $PR \approx_\lambda QR$. The conclusion is that $\approx_\alpha$ are congruence relations for every $\alpha \in On$. $\square$

Example 3.10 We are now going to demonstrate that the condition we place on the processes in the statement above is essential for congruence to hold. We can define a process P to be a τ loop, that is $P \stackrel{\tau}{\longrightarrow} P$ is the only transition available to P, then we take Q to be a process capable of a single τ action, that is $Q \stackrel{\tau}{\longrightarrow} \epsilon$ is the transition that defines Q, and we take a process R to be $R \stackrel{a}{\longrightarrow} \epsilon$. Clearly $P \approx Q$ which implies that $P \approx_\alpha Q$ for every α. The process PR is weakly bisimilar to P because we can never get past P but on the other hand the process QR can do $\stackrel{\tau}{\longrightarrow}$ and become R and then perform $\stackrel{a}{\longrightarrow}$. Since the action a is not available to P we come to the conclusion that $PR \not\approx_1 QR$ despite the fact that $P \approx Q$. $\square$

3.2.2 Infinite branching

As we mentioned in the previous chapter, both BPA-processes and BPP are image-finite with respect to strong bisimilarity (for any process P and any action

μ, there are only finitely many process reachable from P via μ). A corollary of this fact is that the chain of strong bisimulation approximants stops at the level ω, i.e. $\sim = \sim_\omega$. When we move from strong bisimilarity to weak bisimilarity we lose the important property of image-finiteness for both BPA-processes and BPP. Now we can define processes that may have an infinite number of derivatives obtained by performing $\stackrel{\mu}{\Longrightarrow}$ for some action μ, hence they are capable of infinite branching.

Example 3.11 Here we demonstrate two infinitely branching BPA-processes:

$$X \stackrel{a}{\longrightarrow} \epsilon \qquad Y \stackrel{\tau}{\longrightarrow} YX \qquad Y \stackrel{b}{\longrightarrow} \epsilon$$

The process Y can perform the sequence $\stackrel{\tau^n}{\longrightarrow}\stackrel{b}{\longrightarrow} = \stackrel{b}{\Longrightarrow}$ and become X^n for an arbitrary n. That clearly determines an infinitely branching transition tree. We present the tree in Fig. 3.1 in two versions: in terms of single transitions and then using the appropriate $\Longrightarrow$ notation.

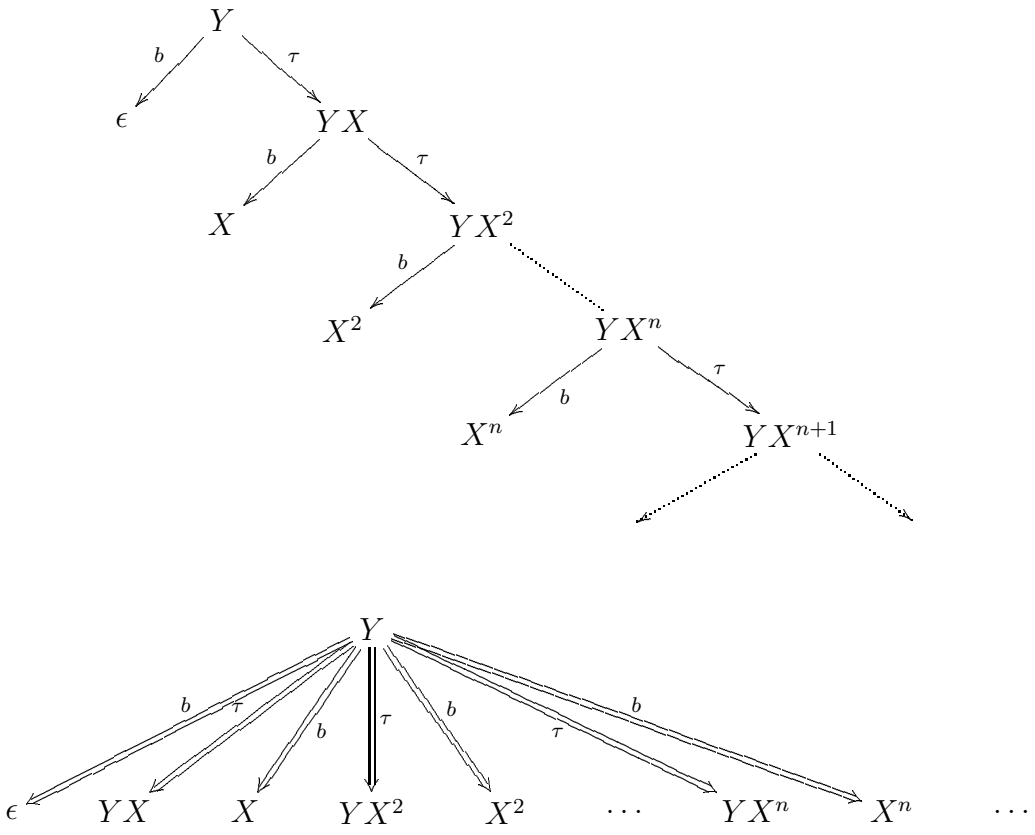


Figure 3.1: Infinitely branching BPA-process

We can interpret the expression YX as parallel composition $Y\|X$ and that gives rise to an infinitely branching BPP as presented in Fig. 3.2. Notice that since in the parallel composition each composite variable can perform an action, the processes $Y\|X^n$ have the $\stackrel{a}{\longrightarrow}$ transition at their disposal, as opposed to the sequential processes YX^n. $\square$

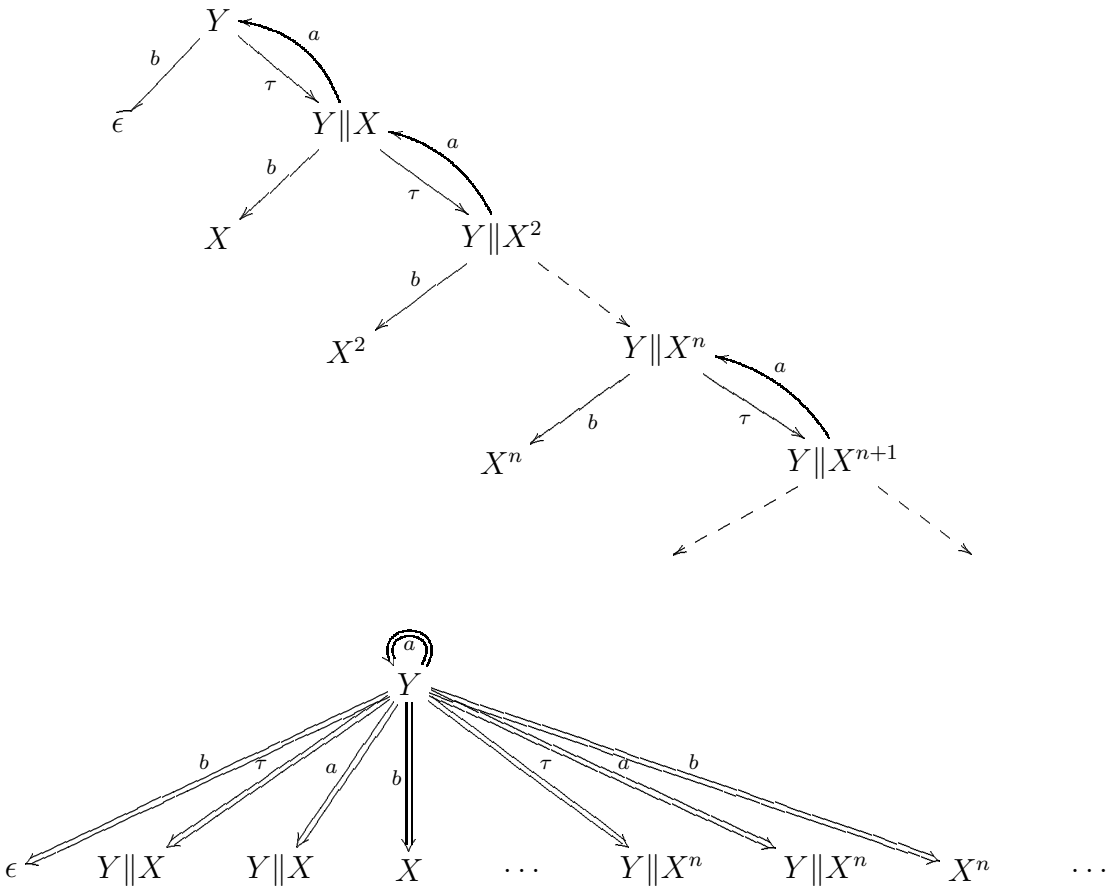


Figure 3.2: Infinitely branching BPP

Most importantly, along with image-finiteness we also lose the property that the maximal bisimulation relation can be obtained as the intersection of all approximants over the set of natural numbers. In the following part we will show the existence of processes that distinguish the level $\approx_\omega$ from $\approx$. We will construct a sequence of pairs of BPP P_i and Q_i such that $P_i \approx_{\omega+i} Q_i$ and $P_i \not\approx Q_i$, for all $i \in \omega$. For BPA-processes we will show an even stronger result - we will construct pairs of processes P_i, Q_i with the property that for every i, $P_i \approx_{\omega^i} Q_i$ but $P_i \not\approx Q_i$. Thus we will obtain two lower bounds on the convergence towards $\approx$. For BPPA, $\approx \subset \approx_{\omega+i}$ for every $i \in \omega$. For BPA, $\approx \subset \approx_\alpha$ for every $\alpha < \omega^\omega$.

Remark: Both weak bisimilarity and weak bisimulation approximants are binary relations defined on a fixed process algebra (Σ, Δ). Most of the time we will not specify that algebra explicitly but we will assume that it can be determined from the context.

3.3 BPA-processes and $\approx_\alpha$

In this section we will focus our attention onto the relationship between BPA-processes and weak bisimulation approximants. The aim of this section is to provide a lower bound on the ordinal number α which is the label of an approximant that coincides with the maximal weak bisimulation. We will construct an infinite sequence of pairs of processes (P_n, Q_n) that will distinguish the approximant $\approx_{\omega^n}$ from $\approx$ and then deduce that the lower bound on such an α is ω^ω.

We will start with a pair of BPA-processes that distinguishes $\approx_\omega$ from $\approx$. For that purpose we will use process variables A and C as defined in **Example 3.8**:

$$A \stackrel{a}{\longrightarrow} \epsilon \qquad A \stackrel{\tau}{\longrightarrow} \epsilon \qquad C \stackrel{\tau}{\longrightarrow} CA \qquad C \stackrel{\tau}{\longrightarrow} \epsilon$$

Now we can show the basic result that the processes C and AC are equivalent at level ω but not weakly bisimilar, in fact not equivalent at level $\omega + 1$ which implies that $\approx \neq \approx_\omega$ (more precisely, there is a proper inclusion $\approx \subsetneq \approx_\omega$). The two processes are pictured in Fig. 3.3.

Proposition 3.12 *$C \approx_\omega AC \wedge C \not\approx_{\omega+1} AC$.*

Proof: First we will make two important observations: $A^k \approx_k A^l$ for any $k \leq l$ and $A^k \approx_k CA^l$ for any k, l. The validity of the two observations can be seen by induction on k: the processes A^k, A^l and CA^k can only perform sequences of a actions and there is no branching involved so equivalence at level k is determined solely by the ability to generate a sequence $(\stackrel{a}{\Longrightarrow})^k$. Also, if any process decides to dispose of a number of copies of A in one go, the other process is always able to simulate that and become an identical process.

From the definition, two processes are equivalent at level ω if they are equivalent at level n for all n. Hence we will continue by induction on n. The base case is $C \approx_0 AC$ which holds trivially. Assuming $C \approx_n AC$ we will show that $C \approx_{n+1} AC$ by analysing all possible moves of C and AC. First we will note that any move of C can be copied by $AC \stackrel{\tau}{\longrightarrow} C$ hence the only 'interesting' moves are those of AC. They are displayed in the table below:

$$AC \stackrel{a}{\Longrightarrow} \begin{cases} CA^l, & \text{for any } l \\ A^l, & \text{for any } l \end{cases} \qquad AC \stackrel{\tau}{\Longrightarrow} \begin{cases} CA^l, & \text{for any } l \\ A^l, & \text{for any } l \end{cases}$$

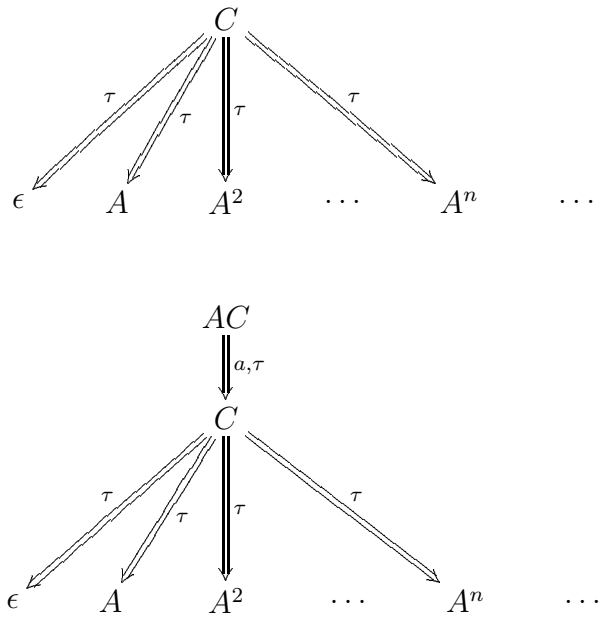


Figure 3.3: The processes C and AC

The response of C to the move $AC \stackrel{a}{\Longrightarrow} CA^l$ is $C \stackrel{\tau}{\Longrightarrow} A^{n+1} \stackrel{a}{\longrightarrow} A^n$ and we use the second observation to conclude that $A^n \approx_n CA^l$. The response to $AC \stackrel{a}{\Longrightarrow} A^l$ is $C \stackrel{\tau}{\Longrightarrow} A^{l+1} \stackrel{a}{\longrightarrow} A^l$. For the $\stackrel{\tau}{\Longrightarrow}$ moves, if $AC \stackrel{\tau}{\Longrightarrow} CA^l$ or A^l then C is actually capable of generating the same processes CA^l or A^l by a sequence of τ moves. Therefore we can conclude that $C \approx_{n+1} AC$ from which follows that $C \approx_\omega AC$.

In order to see that $C \not\approx_{\omega+1} AC$ we observe that if AC does $\stackrel{a}{\longrightarrow}$ and becomes C then C has to match the action and the only way of doing that is by generating CA^{k+1} with a sequence of τ moves, disposing of the C in front and doing $\stackrel{a}{\longrightarrow}$ to become A^k. Then we have the process C on the one hand and A^k on the other. These two processes cannot be equivalent at level ω because we can choose any $N > k$ and generate A^N from C. A^N can enforce a sequence of N actions a which A^k cannot match. Hence $A^k \not\approx_N A^N$, $A^k \not\approx_\omega C$ and $C \not\approx_{\omega+1} AC$. $\square$

Now we can try to explain the construction of processes which will be equivalent at levels $\approx_{\omega^n}$ without being actually weakly bisimilar. We will present a discussion to convey the idea, the precise statements and proofs will follow later. The construction builds on the variables A and C defined earlier. If we take C and AC and compose them both from the right with the same number of copies of C then we obtain processes C^n and AC^n, for some n, which are related by $\approx_{\omega n}$ but are not related by $\approx_{\omega n+1}$ and hence are not weakly bisimilar. These pairs of processes therefore distinguish between $\approx$ and $\approx_{\omega n}$. Intuitively, the reason why this is so

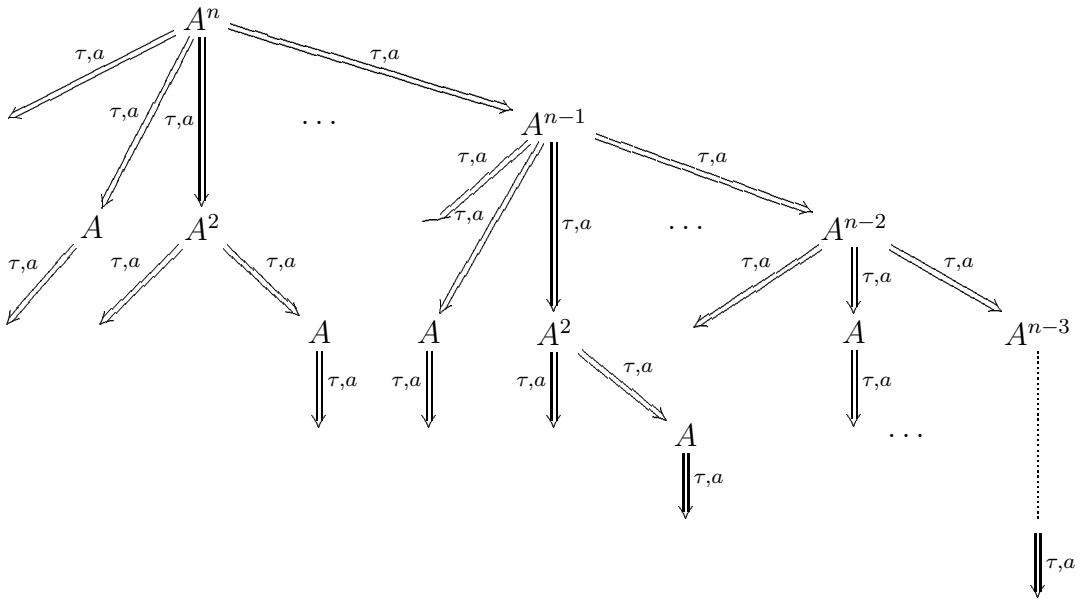


Figure 3.4: The process A^n

is the following: each C has got the power to generate any number of copies of A by a sequence of τ moves hence it defines an infinitely branching transition tree with branches of unbounded length and therefore of height ω (height is also called rank). BPA-processes compose in a way that uses the full power of each component so a process C^n will give rise to a tree of height $\omega \cdot n$. Each ω tier of the tree provides for one ω level in the $\approx_{\omega \cdot n}$ equivalence hence height $\omega \cdot n$ implies equivalence at $\approx_{\omega \cdot n}$. That is the reason for the processes being $\approx_{\omega \cdot n}$ equivalent. On the other hand, that the processes cannot be weakly bisimilar can be seen as follows: the single copy of A in front of one of the processes gives the possibility of an extra a action. So while AC^n can do $\stackrel{a}{\longrightarrow}$ and become C^n the other process C^n already has to commit itself to becoming $A^k C^{n-1}$ for some k. However big this k, the former process can always choose to become $A^N C^{n-1}$ for $N > k$ so it has the advantage of having some extra a actions. Then it can perform such a sequence of choices which will demonstrate that AC^n and C^n are not equivalent at $\omega \cdot n + 1$.

This construction clearly does not have to stop at $\approx_{\omega \cdot n}$ and indeed we can generalise the idea and define process variables which will give rise to trees of larger height. We will define a new process variable which will be capable of generating any power of C which will give rise to a tree of height ω^2 and we will continue in this manner. So we are going to introduce an infinite hierarchy of new variables D_i such that $D_0 = A$, $D_1 = C$ and each new variable D_{i+1} will be

able to generate any number of copies of D_i. Formally, the construction is done inductively as follows:

1. $D_0 \xrightarrow{a} \epsilon$, $D_0 \xrightarrow{\tau} \epsilon$
2. assuming we have defined the variables $D_0, \ldots, D_i$, the variable D_{i+1} is defined by $D_{i+1} \xrightarrow{\tau} D_{i+1}D_i$, $D_{i+1} \xrightarrow{\tau} \epsilon$.

Notice that the only variable capable of performing a visible action is D_0. The purpose of the other variables is to create bigger and bigger branching. We will show later that each variable D_i determines a tree of height ω^i. Now we are ready to state the main result of the section:

Theorem 3.13 *$D_n \approx_{\omega^n} D_0D_n \wedge D_n \not\approx D_0D_n$ for every $n \in \mathbb{N}$.*

We will formulate and prove a couple of more general claims from which the **Theorem 3.13** will easily follow. First we will analyse all possible moves and derivatives of the processes in question. In order to gain intuition about the behaviour of the processes we will examine all the possibilities that can arise from a single copy of D_i with $i > 0$. We can observe that except for the process variable D_0 none of the other process variables can do a visible action. The only available behaviour is to generate some variables of one level lower and then disappear.

Starting from a variable D_i we can only perform a $\stackrel{\tau}{\Longrightarrow}$ sequence with which we will obtain the process $D_iD_{i-1}^{e_{i-1}}$ for some e_{i-1}. We cannot get a more complex shape without removing the D_i in front. After having disposed of the D_i we can continue and from $D_{i-1}^{e_{i-1}}$ generate (with another $\stackrel{\tau}{\Longrightarrow}$ sequence) the process $D_{i-1}D_{i-2}^{e_{i-2}}D_{i-1}^{e_{i-1}-1}$. We can repeat the procedure several times and finally derive a process either of the form $D_{k+1}D_k^{e_k}D_{k+1}^{e_{k+1}}\ldots D_m^{e_m}$ or $D_k^{e_k}D_{k+1}^{e_{k+1}}\ldots D_m^{e_m}$, where $k \geq 0$, $m < i$ and $e_k, \ldots, e_m \geq 0$. The latter process is an ascending product of variables which will be denoted by $\prod_{i=0}^{m} D_i^{e_i}$ and called simply a *product*.

So we can easily deduce that a single variable evolves into a product, resp. a variable followed by a product, which can then only evolve into another product or a process of the form a variable followed by a product. In both cases the resulting process is smaller in a certain sense which we will specify later. Before we make this observation precise in the **Proposition 3.14** below we will introduce the following auxiliary notation. To a product $\prod_{i=0}^{m} D_i^{e_i}$ we will assign an ordinal number $\sum_{i=0}^{m} \omega^i e_i = \omega^m e_m + \omega^{m-1}e_{m-1} + \ldots + \omega e_1 + e_0$ which will provide the measure for product processes. This measure actually corresponds to the height of product processes but we will not follow that link here.

Proposition 3.14 *For any product* $\prod_{i=0}^{m} D_i^{e_i}$ *and any process* P*, there is a derivation* $\prod_{i=0}^{m} D_i^{e_i} \stackrel{\mu}{\Longrightarrow} P$ *if and only if* P *is of the form* $\prod_{i=0}^{m} D_i^{f_i}$ *or* $D_l \prod_{i=l-1}^{m} D_i^{f_i}$*, with* $\sum_{i=0}^{m} \omega^i f_i < \sum_{i=0}^{m} \omega^i e_i$ *in case* $\mu = a$*, and* $\sum_{i=0}^{m} \omega^i f_i \leq \sum_{i=0}^{m} \omega^i e_i$ *in case* $\mu = \tau$*.*

Proof: Since we deal with sequential processes we know that all moves available to a product process are induced by the transitions of its foremost variable. Hence in order to analyse all possible sequences of moves a product can perform it suffices to investigate the behaviour of each potential front process variable before it eventually disappears and gives way to another variable.

To carry out the proof we fix a product $\prod_{i=0}^{m} D_i^{e_i}$ and the corresponding $\sum_{i=0}^{m} \omega^i e_i$ and assume that e_j is the first non-zero exponent. If $j = 0$ then we know that all available transitions are $D_0 \stackrel{\mu}{\longrightarrow} \epsilon$ where μ is either τ or a. Each such transition diminishes the respective ordinal number by 1. Hence we can conclude that as long as D_0 remains the front variable all possible sequences of transitions lead to a product of the form $D_0^e D_1^{e_1} \ldots D_m^{e_m}$ with $e < e_0$ and therefore also $\omega^m e_m + \ldots + \omega e_1 + e < \omega^m e_m + \ldots + \omega e_1 + e_0$.

Next we assume that $j > 0$. For such j the process variable D_j can decide to disappear using the rule $D_j \stackrel{\tau}{\longrightarrow} \epsilon$ which gives rise to the product $D_j^{e_j - 1} \ldots D_m^{e_m}$. The respective ordinal is then $\omega^m e_m + \ldots + \omega^j (e_j - 1)$ which is smaller than the original $\omega^m e_m + \ldots + \omega^j e_j$. Before D_j disappears it can perform a sequence of transitions $D_j \stackrel{\tau}{\longrightarrow} D_j D_{j-1}$ which results in the process $D_j D_{j-1}^{e_{j-1}} D_j^{e_j - 1} \ldots D_m^{e_m}$. However, the respective ordinal $\omega^m e_m + \ldots + \omega^j (e_j - 1) + \omega^{j-1} e_{j-1}$ is again smaller than $\omega^m e_m + \ldots + \omega^j e_j$. Since we can break any (non-empty) sequence of transitions into subsequences of the form we have just analysed we can conclude that any process derived from $\prod_{i=0}^{m} D_i^{e_i}$ has to be of the form $\prod_{i=0}^{m} D_i^{f_i}$ or $D_{l+1} \prod_{i=l}^{m} D_i^{f_i}$, with $\sum_{i=0}^{m} \omega^i f_i < \sum_{i=0}^{m} \omega^i e_i$. In case the product performs an empty sequence $\stackrel{\epsilon}{\Longrightarrow}$ the assigned ordinal remains unaltered.

Finally we want to show that for any sequence $f_0, f_1, \ldots, f_m$ with $\sum_{i=0}^{m} \omega^i f_i \leq \sum_{i=0}^{m} \omega^i e_i$ we can perform a derivation resulting in the product $\prod_{i=0}^{m} D_i^{f_i}$. Clearly in case of $\sum_{i=0}^{m} \omega^i f_i = \sum_{i=0}^{m} \omega^i e_i$, $f_i = e_i$ for every i, and hence it is the empty sequence $\stackrel{\epsilon}{\Longrightarrow}$. If case $\sum_{i=0}^{m} \omega^i f_i < \sum_{i=0}^{m} \omega^i e_i$ we define $j = \max\{i \mid f_i \neq e_i\}$. Then $f_j < e_j$ and the product $\prod_{i=0}^{m} D_i^{f_i}$ is equal to $D_0^{f_0} \ldots D_j^{f_j} D_{j+1}^{e_{j+1}} \ldots D_m^{e_m}$. In the explanation preceding this proposition we showed that from a single variable D_j we can derive any product $D_0^{f_0} \ldots D_{j-1}^{f_{j-1}}$ hence we proceed in this way: with a $\stackrel{\tau}{\Longrightarrow}$ sequence we reduce the product $\prod_{i=0}^{m} D_i^{e_i}$ into $D_j^{f_j+1} D_{j+1}^{e_{j+1}} \ldots D_m^{e_m}$ and then we generate from the front copy of D_j the required prefix $D_0^{f_0} \ldots D_{j-1}^{f_{j-1}}$ obtaining the result $D_0^{f_0} \ldots D_{j-1}^{f_{j-1}} D_j^{f_j} D_{j+1}^{e_{j+1}} \ldots D_m^{e_m}$. This is all achieved with a $\stackrel{\tau}{\Longrightarrow}$ sequence,

however if we require the sequence $\stackrel{a}{\Longrightarrow}$ we can also generate one more copy of D_0 that will finally perform $\stackrel{a}{\longrightarrow}$. Of course, in this way we can only obtain products of strictly smaller heights. $\Box$

We have shown in **Proposition 3.14** that there are only two types of derivatives of a product process. We will simplify the forthcoming proofs by showing that a process of the form $D_l \prod_{i=l-1}^{m} D_i^{f_i}$ is in fact weakly bisimilar to the product $D_l^{f_l+1} \prod_{i=l+1}^{m} D_i^{f_i}$. We will make use of this property in the equivalence check for $\approx_\alpha$. Suppose we want to establish that $P \approx_{\alpha+1} Q$ for some products P, Q. Then if there is a derivation $P \stackrel{\mu}{\Longrightarrow} D_l \prod_{i=l-1}^{m} D_i^{f_i}$ we look for a matching response $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx_\alpha Q'$. If there exists some $P \stackrel{\mu}{\Longrightarrow} P'$ with $D_l \prod_{i=l-1}^{m} D_i^{f_i} \approx P'$ for which $P' \approx_\alpha Q'$ then, as the relations $\approx_\alpha$ are equivalences and contain $\approx$, also $D_l \prod_{i=l-1}^{m} D_i^{f_i} \approx_\alpha Q'$. We can conclude that it suffices to carry out the check solely for P'. All that has been said combined will permit us to consider solely product processes in the analysis of derivatives of some product.

In order to show that $D_l \prod_{i=l-1}^{m} D_i^{f_i}$ is weakly bisimilar to $D_l^{f_l+1} \prod_{i=l+1}^{m} D_i^{f_i}$ we will show that the processes D_l and $D_l D_{l-1}^n$ are weakly bisimilar for every $l > 0$ and every n, that means that it does not matter how many copies of D_{l-1} we have already generated as long as we still keep the front copy of D_l. Then we will use the fact that weak bisimulation is a congruence in this special case hence $D_l \approx D_l D_{l-1}^n$ implies that $D_l D_{l-1}^{f_{l-1}} D_l^{f_l} \ldots D_m^{f_m} \approx D_l^{f_l+1} \ldots D_m^{f_m}$. It is convenient to state and prove a stronger result.

Proposition 3.15 *For every k, m and $l > 0$, $D_l^{k+1} \approx D_l D_{l-1}^m D_l^k$.*

Proof: In order to demonstrate that two processes are weakly bisimilar it suffices to construct a binary relation containing the pair of the processes in question and show that the relation is a weak bisimulation. For that purpose we will define a binary relation $\mathcal{R} = \{(D_l D_{l-1}^m D_l^k, D_l D_{l-1}^n D_l^k) | k, l > 0, m, n \in \mathbb{N}\} \cup \{(\hat{D}, \hat{D}) | D_l D_{l-1}^m D_l^k \stackrel{a^*}{\Longrightarrow} \hat{D}, k, l > 0, m \in \mathbb{N}\}$. Clearly the relation $\mathcal{R}$ contains the pairs $(D_l^{k+1}, D_l D_{l-1}^m D_l^k)$ for every k, m and $l > 0$. Now we have to check that it is closed under expansion with $\stackrel{\mu}{\longrightarrow}$, that means for every pair (P, Q) from $\mathcal{R}$, if there is a transition $P \stackrel{\mu}{\longrightarrow} P'$ then there has to be a matching transition $Q \stackrel{\mu}{\Longrightarrow} Q'$ with the resulting pair (P', Q') again in $\mathcal{R}$, and conversely, also starting from Q.

Firstly we will choose a pair $(D_l D_{l-1}^m D_l^k, D_l D_{l-1}^n D_l^k)$ for some fixed $k, l > 0, m$ and n. We remind ourselves that for any $l > 0$, the only possible transitions D_l can do are $D_l \stackrel{\tau}{\longrightarrow} \epsilon$ and $D_l \stackrel{\tau}{\longrightarrow} D_l D_{l-1}$. If either $D_l D_{l-1}^m D_l^k$ or $D_l D_{l-1}^n D_l^k$ chooses to perform the transition $D_l \stackrel{\tau}{\longrightarrow} D_l D_{l-1}$ the other process does exactly the same which results in a pair $(D_l D_{l-1}^{m+1} D_l^k, D_l D_{l-1}^{n+1} D_l^k)$ that belongs to $\mathcal{R}$ by definition.

To analyse the case when one process decides to employ the transition $D_l \xrightarrow{\tau} \epsilon$ we will assume that $m \leq n$. Hence the response to $D_l D_{l-1}^n D_l^k \xrightarrow{\tau} D_{l-1}^n D_l^k$ will be $D_l D_{l-1}^m D_l^k \xrightarrow{\tau^{n-m}} D_l D_{l-1}^n D_l^k \xrightarrow{\tau} D_{l-1}^n D_l^k$ and the pair $(D_{l-1}^n D_l^k, D_{l-1}^n D_l^k)$ will belong to $\mathcal{R}$ since $D_{l-1}^n D_l^k$ is derived from $D_l D_{l-1}^n D_l^k$. If it is $D_l D_{l-1}^m D_l^k$ that disposes of D_l and becomes $D_{l-1}^m D_l^k$ then the other process $D_l D_{l-1}^n D_l^k$ responds by removing D_l in the first place and then all superfluous copies of D_{l-1} to become $D_{l-1}^m D_l^k$. Again the pair $(D_{l-1}^m D_l^k, D_{l-1}^m D_l^k)$ is in $\mathcal{R}$.

Lastly, if we have a pair $(\hat{D}, \hat{D})$ from $\mathcal{R}$ with $\hat{D}$ being an $\stackrel{a^*}{\Longrightarrow}$ derivative of some $D_l D_{l-1}^m D_l^k$ then any $\bar{D}$ obtained from $\hat{D}$ by performing $\xrightarrow{\mu}$ is also an $\stackrel{a^*}{\Longrightarrow}$ derivative of $D_l D_{l-1}^m D_l^k$ and hence the pair $(\bar{D}, \bar{D})$ belongs to $\mathcal{R}$. $\square$

And a simple consequence of the preceding two statements is the following corollary:

Corollary 3.16 *The processes $D_l \prod_{i=l-1}^m D_i^{f_i}$ and $D_l^{f_l+1} \prod_{i=l+1}^m D_i^{f_i}$ are weakly bisimilar and there is a derivation $\prod_{i=0}^m D_i^{e_i} \stackrel{\mu}{\Longrightarrow} D_l \prod_{i=l-1}^m D_i^{f_i}$ if and only if there is a derivation $\prod_{i=0}^m D_i^{e_i} \stackrel{\mu}{\Longrightarrow} D_l^{f_l+1} \prod_{i=l+1}^m D_i^{f_i}$.*

Proof: The first part of the statement follows from the fact that for Basic Process Algebras which do not allow infinite τ sequences, weak bisimulation is a congruence. No process over the variables $D_0, D_1 \ldots, D_n$ has the capability to perform the action τ infinitely many times hence we can deduce that since $D_l \approx D_l D_{l-1}^{f_{l-1}}$ then also $D_l^{f_l+1} D_{l+1}^{f_{l+1}} \ldots D_m^{f_m} \approx D_l D_{l-1}^{f_{l-1}} D_l^{f_l} D_{l+1}^{f_{l+1}} \ldots D_m^{f_m}$.

One implication of the second part is straightforward. If we can have a derivation $\prod_{i=0}^m D_i^{e_i} \stackrel{\mu}{\Longrightarrow} D_l^{f_l+1} \prod_{i=l+1}^m D_i^{f_i}$ then with repeated application of the rule $D_l \xrightarrow{\tau} D_l D_{l-1}$ we can obtain the process $D_l \prod_{i=l-1}^m D_i^{f_i}$. On the other hand we know that this rule is the only means of obtaining copies of D_{l-1} so obviously we cannot generate $D_l \prod_{i=l-1}^m D_i^{f_i}$ without having generated the product $D_l^{f_l+1} D_{l+1}^{f_{l+1}} \ldots D_m^{f_m}$ first. $\square$

Now we are ready to prove the main **Theorem 3.13** in two parts, the positive part by demonstrating equivalence at the specified level, and the negative by showing that the two processes are not related at the level above. We will make use of the ordinals assigned to each product since they determine the highest level that relates two processes. Intuitively this corresponds to the heights of respective transition trees which we discussed earlier. The height of the smaller tree is the maximal level that can relate two trees which is formalised in the following lemma.

Lemma 3.17 *For all products,* $\prod_{i=0}^{m} D_i^{e_i} \approx_\alpha \prod_{i=0}^{m} D_i^{f_i}$, *where* $\alpha \leq \min\{\beta, \gamma\}$ *with* $\beta = \omega^m e_m + \omega^{m-1} e_{m-1} + \ldots + \omega e_1 + e_0$ *and* $\gamma = \omega^m f_m + \omega^{m-1} f_{m-1} + \ldots + \omega f_1 + f_0$.

Proof: We will prove this statement by transfinite induction on α which consists of proving the claim for the cases of α being 0, then a successor ordinal number and finally a limit ordinal number. The claim obviously holds for $\alpha = 0$ since all processes are related at zero level.

In order to prove the successor case we assume that the claim holds for some α and we will want to prove it for $\alpha+1$. We presuppose two processes $P = \prod_{i=0}^{m} D_i^{e_i}$ and $Q = \prod_{i=0}^{m} D_i^{f_i}$ such that $\alpha + 1 \leq \beta = \sum_{i=0}^{m} \omega^i e_i \leq \gamma = \sum_{i=0}^{m} \omega^i f_i$ and we will show that $P \approx_{\alpha+1} Q$. We know that $\beta = \gamma$ if and only if $e_i = f_i$ for every $i = 0, \ldots, m$. In that case P and Q are two identical processes which are trivially equivalent at every level. Hence we can without loss of generality assume that $\beta < \gamma$.

We remind ourselves that $P \approx_{\alpha+1} Q$ if for every move $P \stackrel{\mu}{\Longrightarrow} P'$ there is a matching transition $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx_\alpha Q'$, and conversely, starting from Q. Since $\sum_{i=0}^{m} \omega^i e_i < \sum_{i=0}^{m} \omega^i f_i$ the process Q can evolve into P by a $\stackrel{\tau}{\Longrightarrow}$ sequence so in case P takes the initiative and performs a transition $P \stackrel{\mu}{\Longrightarrow} P'$ the process Q will copy P and become P' as well. Then we can conclude by $P' \approx_\alpha P'$.

It remains to check the moves of $Q = \prod_{i=0}^{m} D_i^{f_i}$. Either $Q \stackrel{\mu}{\Longrightarrow} \prod_{i=0}^{m} D_i^{g_i}$, where $\sum_{i=0}^{m} \omega^i g_i \leq \gamma$, or $Q \stackrel{\mu}{\Longrightarrow} D_j \prod_{i=j-1}^{m} D_i^{g_i}$. The latter is by **Proposition 3.15** weakly bisimilar to the product $D_j^{g_j+1} \prod_{i=j+1}^{m} D_i^{g_i}$. We can replace $D_j \prod_{i=j-1}^{m} D_i^{g_i}$ with the bisimilar product because of **Corollary 3.16** and the facts that for every ordinal δ, $\approx \subseteq \approx_\delta$ and $\approx_\delta$ is transitive. Therefore if $P' \approx_\alpha D_j^{g_j+1} \prod_{i=j+1}^{m} D_i^{g_i}$ for some P-derivative P' then also $P' \approx_\alpha D_j \prod_{i=j-1}^{m} D_i^{g_i}$. Hence we will assume that Q evolves into a product $\prod_{i=0}^{m} D_i^{g_i}$, for some g_i such that $\sum_{i=0}^{m} \omega^i g_i \leq \gamma$. We have to distinguish two cases according to the height of the Q-derivative:

1. $\alpha < \sum_{i=0}^{m} \omega^i g_i$

   We will show that P can do a matching action and evolve into a product $\prod_{i=0}^{m} D_i^{h_i}$ with $\sum_{i=0}^{m} \omega^i h_i \geq \alpha$. Then we will use the induction hypothesis and conclude that $\prod_{i=0}^{m} D_i^{h_i} \approx_\alpha \prod_{i=0}^{m} D_i^{g_i}$. There are two ways in which P will respond depending on e_0 (the exponent of D_0 in P).

   - If $e_0 > 0$ then P contains at least one copy of D_0 which will perform the appropriate action using the transition $D_0 \stackrel{a/\tau}{\longrightarrow} \epsilon$. P will therefore evolve into $D_0^{e_0-1} \ldots D_m^{e_m}$ with $\sum_{i=0}^{m} \omega^i e_i - 1 \geq \alpha$. Hence $\alpha \leq \min\{\sum_{i=0}^{m} \omega^i e_i - 1, \sum_{i=0}^{m} \omega^i g_i\}$ and from the induction hypothesis we can conclude that $D_0^{e_0-1} \ldots D_m^{e_m} \approx_\alpha \prod_{i=0}^{m} D_i^{g_i}$.

- If $e_0 = 0$ then $\beta = \sum_{i=0}^{m} \omega^i e_i$ is a limit ordinal. Since $\alpha{+}1$ is a successor ordinal and $\alpha{+}1 \leq \beta$ then from the nature of ordinal numbers $\alpha{+}1 < \beta$ and, moreover, there exists an ordinal δ with $\alpha < \delta < \beta$. Now we can use the statement of **Proposition 3.14** and deduce that there has to be a matching move of P resulting in a product $\prod_{i=0}^{m} D_i^{h_i}$ with $\alpha < \delta = \sum_{i=0}^{m} \omega^i h_i$. Then again $\alpha \leq \min\{\sum_{i=0}^{m} \omega^i h_i, \sum_{i=0}^{m} \omega^i g_i\}$ and the conclusion is that $\prod_{i=0}^{m} D_i^{h_i} \approx_\alpha \prod_{i=0}^{m} D_i^{g_i}$.

2. $\alpha \geq \sum_{i=0}^{m} \omega^i g_i$

   In this case also $\sum_{i=0}^{m} \omega^i g_i < \sum_{i=0}^{m} \omega^i e_i$ which means that by **Proposition 3.14** the process P can simulate the move of Q and become exactly the product $\prod_{i=0}^{m} D_i^{g_i}$. Again we conclude with the argument that the relation $\approx_\alpha$ is an equivalence and so $\prod_{i=0}^{m} D_i^{g_i} \approx_\alpha \prod_{i=0}^{m} D_i^{g_i}$.

Lastly we have to check the case of a limit ordinal λ. The argument is the following: we assume that the two processes $\prod_{i=0}^{m} D_i^{e_i}$ and $\prod_{i=0}^{m} D_i^{f_i}$ are such that $\lambda \leq \min\{\sum_{i=0}^{m} \omega^i e_i, \sum_{i=0}^{m} \omega^i f_i\}$. Hence the same holds for every $\alpha < \lambda$. From the induction hypothesis we conclude that $\prod_{i=0}^{m} D_i^{e_i} \approx_\alpha \prod_{i=0}^{m} D_i^{f_i}$ for every $\alpha < \lambda$ and from the definition of a limit approximant we know that $\prod_{i=0}^{m} D_i^{e_i} \approx_\lambda \prod_{i=0}^{m} D_i^{f_i}$. □

And the second technical lemma which deals with the negative part of the main **Theorem 3.13** goes as follows:

Lemma 3.18 *If* $\prod_{i=0}^{m} D_i^{e_i} \neq \prod_{i=0}^{m} D_i^{f_i}$ *then* $\prod_{i=0}^{m} D_i^{e_i} \not\approx_\alpha \prod_{i=0}^{m} D_i^{f_i}$ *where* $\alpha > \min\{\beta, \gamma\}$ *with* $\beta = \omega^m e_m + \omega^{m-1} e_{m-1} + \ldots + \omega e_1 + e_0$ *and* $\gamma = \omega^m f_m + \omega^{m-1} f_{m-1} + \ldots + \omega f_1 + f_0$.

Proof: We will prove this statement by transfinite induction on α. For $\alpha = 0$ the statement holds vacuously. Next we check the case of a successor ordinal. We assume that the claim holds for an ordinal α and we will argue that it also holds for $\alpha{+}1$. We know that $\prod_{i=0}^{m} D_i^{e_i} = \prod_{i=0}^{m} D_i^{f_i}$ if and only if $\sum_{i=0}^{m} \omega^i e_i = \sum_{i=0}^{m} \omega^i f_i$ hence without loss of generality we can presuppose two products $\prod_{i=0}^{m} D_i^{e_i}$ and $\prod_{i=0}^{m} D_i^{f_i}$ such that $\sum_{i=0}^{m} \omega^i e_i > \sum_{i=0}^{m} \omega^i f_i$ and $\alpha + 1 > \sum_{i=0}^{m} \omega^i f_i$.

Now let the larger product $\prod_{i=0}^{m} D_i^{e_i}$ take the initiative and perform $\stackrel{a}{\Longrightarrow}$ to become $\prod_{i=0}^{m} D_i^{e'_i}$ with $\sum_{i=0}^{m} \omega^i e'_i \geq \sum_{i=0}^{m} \omega^i f_i$. The possibility of such a move follows from our earlier assumption and **Proposition 3.14**. Again using the **Propositions 3.14** and **3.16** we conclude that any matching move $\stackrel{a}{\Longrightarrow}$ of $\prod_{i=0}^{m} D_i^{f_i}$ will

necessarily be sum decreasing, that is if $\prod_{i=0}^{m} D_i^{f_i} \stackrel{a}{\Longrightarrow} \prod_{i=0}^{m} D_i^{f'_i}$ then $\sum_{i=0}^{m} \omega^i f'_i < \sum_{i=0}^{m} \omega^i f_i$. Hence the two derivatives are distinct with $\sum_{i=0}^{m} \omega^i e'_i > \sum_{i=0}^{m} \omega^i f'_i$ and moreover, also $\alpha > \sum_{i=0}^{m} \omega^i f'_i$ and we can use the induction hypothesis to conclude that $\prod_{i=0}^{m} D_i^{e'_i} \not\approx_\alpha \prod_{i=0}^{m} D_i^{f'_i}$. Since this is true for all matching responses of $\prod_{i=0}^{m} D_i^{f_i}$, the products $\prod_{i=0}^{m} D_i^{e_i}$ and $\prod_{i=0}^{m} D_i^{f_i}$ cannot be equivalent at $\alpha + 1$.

Finally we assume that λ is a limit ordinal and $P = \prod_{i=0}^{m} D_i^{e_i}$, $Q = \prod_{i=0}^{m} D_i^{f_i}$ are distinct products such that without loss of generality $\sum_{i=0}^{m} \omega^i e_i > \sum_{i=0}^{m} \omega^i f_i$ and $\lambda > \sum_{i=0}^{m} \omega^i f_i$. From the definition, $P \approx_\lambda Q$ if for every $\alpha < \lambda$, $P \approx_\alpha Q$, so if there exists an $\alpha < \lambda$ with $P \not\approx_\alpha Q$ then also $P \not\approx_\lambda Q$. We have to distinguish two cases:

1. $\lambda > \sum_{i=0}^{m} \omega^i e_i$

   Then we know that there exists an ordinal α such that $\lambda > \alpha > \sum_{i=0}^{m} \omega^i e_i$. From the induction hypothesis it follows that $\prod_{i=0}^{m} D_i^{e_i} \not\approx_\alpha \prod_{i=0}^{m} D_i^{f_i}$ and we can deduce that $\prod_{i=0}^{m} D_i^{e_i} \not\approx_\lambda \prod_{i=0}^{m} D_i^{f_i}$.

2. $\sum_{i=0}^{m} \omega^i e_i \geq \lambda > \sum_{i=0}^{m} \omega^i f_i$

   In this case we can find an ordinal number α such that $\lambda > \alpha > \sum_{i=0}^{m} \omega^i f_i$. By **Proposition 3.14** there exists a transition $\prod_{i=0}^{m} D_i^{e_i} \stackrel{a}{\Longrightarrow} \prod_{i=0}^{m} D_i^{e'_i}$ with $\alpha = \sum_{i=0}^{m} \omega^i e'_i$. For any matching move $Q \stackrel{a}{\Longrightarrow} \prod_{i=0}^{m} D_i^{f'_i}$ the sum $\sum_{i=0}^{m} \omega^i f'_i$ is smaller than α. Hence we can deduce that $\prod_{i=0}^{m} D_i^{e'_i} \neq \prod_{i=0}^{m} D_i^{f'_i}$ and $\alpha > \min\{\sum_{i=0}^{m} \omega^i e'_i, \sum_{i=0}^{m} \omega^i f'_i\}$ which means that $\prod_{i=0}^{m} D_i^{e'_i} \not\approx_\alpha \prod_{i=0}^{m} D_i^{f'_i}$ and finally, $\prod_{i=0}^{m} D_i^{e_i} \not\approx_{\alpha+1} \prod_{i=0}^{m} D_i^{f_i}$ and $\prod_{i=0}^{m} D_i^{e_i} \not\approx_\lambda \prod_{i=0}^{m} D_i^{f_i}$. □

To conclude the proof of the **Theorem 3.13** we notice that the positive part follows from **Lemma 3.17** because for each i, ω^i is the minimum of ω^i and $\omega^i + 1$ and hence the two processes D_i and $D_0 D_i$ are equivalent at level ω^i. The negative part is a straightforward consequence of **Lemma 3.18** since clearly D_i and $D_0 D_i$ will not be $(\omega^i + 1)$-equivalent and hence they cannot be weakly bisimilar.

To summarise the above constructions, if we define a Basic Process Algebra (Σ_n^*, Δ_n) to be $\Sigma_n = \{D_0, D_1, \ldots, D_n\}$ and $\Delta_n = \{D_0 \stackrel{\tau}{\longrightarrow} \epsilon, D_0 \stackrel{a}{\longrightarrow} \epsilon, D_{i+1} \stackrel{\tau}{\longrightarrow} D_{i+1} D_i, D_{i+1} \stackrel{\tau}{\longrightarrow} \epsilon \mid 0 \leq i < n\}$ then we know that the weak bisimulation over this BPA cannot be equal to the approximant $\approx_{\omega^n}$ which is demonstrated by the two processes D_n and $D_0 D_n$ that distinguish $\approx$ from $\approx_{\omega^n}$. Hence we obtain a lower bound on weak bisimulation approximants over all Basic Process Algebras which can be expressed as follows:

Proposition 3.19 *For every $\alpha < \omega^\omega$ there exists a Basic Process Algebra (Σ^*, Δ) such that $\approx \subset \approx_\alpha$ on (Σ^*, Δ).*

On the other hand, to reach every higher level we need to introduce a new variable. Since we are only allowed to use a finite number of variables in the definition of a BPA this leads to the following conjecture:

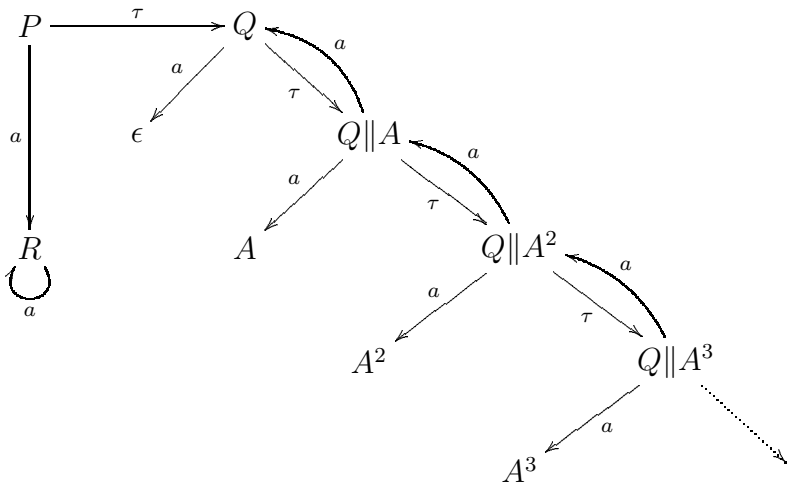
Conjecture 3.20 *For Basic Process Algebras,* $\approx = \approx_{\omega^\omega}$.

3.4 Basic Parallel Processes and $\approx_\alpha$

In this section we are going to study the properties of weak bisimulation approximants with respect to Basic Parallel Processes. We will start by demonstrating a pair of BPP related at the level ω but not weakly bisimilar. Then we will present pairs of processes that distinguish the approximants $\approx_{\omega+n}$ from $\approx$ and we will spell out a conjecture (suggested independently by Hirshfeld and Jančar) that for Basic Parallel Process Algebras, $\approx = \approx_{\omega\cdot 2}$. We will demonstrate the validity of this conjecture for a special subclass of processes. To conclude with, we will present the decidability of $\approx_n$ for all $n \in \mathbb{N}$. The following example introduces BPP that distinguish $\approx_\omega$ from $\approx$.

Example 3.21 We define Basic Parallel Processes P and Q in this way:

$$\begin{array}{lll} P \xrightarrow{\tau} Q & Q \xrightarrow{\tau} Q\|A & R \xrightarrow{a} R \\ P \xrightarrow{a} R & Q \xrightarrow{a} \epsilon & A \xrightarrow{a} \epsilon \end{array}$$



Now we will verify that $P \approx_\omega Q$ and $P \not\approx Q$. To test that $P \approx_\omega Q$ we have to show that $P \approx_n Q$ for every n. Of course $P \approx_0 Q$ holds trivially hence it remains to show that $P \approx_{n+1} Q$ for every $n \geq 0$. We will fix an n and analyse the

possible moves of P and Q. To any move $Q \stackrel{a}{\Longrightarrow} Q'$ the variable P can respond with the transitions $P \stackrel{\tau}{\longrightarrow} Q \stackrel{a}{\Longrightarrow} Q'$, thus becoming an identical process. The transition $P \stackrel{\tau}{\longrightarrow} Q$ is also easy since Q can choose to perform the empty sequence $Q \stackrel{\epsilon}{\Longrightarrow} Q$. Therefore the only interesting move is $P \stackrel{a}{\longrightarrow} R$ to which Q will respond by performing $Q(\stackrel{\tau}{\longrightarrow})^n Q \| A^n \stackrel{a}{\longrightarrow} A^n$. Since the only transition R and A can do is $\stackrel{a}{\longrightarrow}$ we come to the conclusion that $R \approx_n A^n$ and hence $P \approx_{n+1} Q$.

To see that P is not weakly bisimilar to Q we will actually show that $P \not\approx_{\omega+1} Q$. We let the process P perform the transition $\stackrel{a}{\longrightarrow}$ to become R. The only available transition of R is $R \stackrel{a}{\longrightarrow} R$ which defines an infinite a sequence. The process Q has two options: it can generate a number of copies of A and then either disappear or stay. In both cases it cannot maintain equivalence at level ω:

1. If the response of Q is $Q \stackrel{\tau}{\Longrightarrow} Q \| A^n \stackrel{a}{\longrightarrow} A^n$ then we take an $N > n$ and clearly $A^n \not\approx_N R$ which implies that $A^n \not\approx_\omega R$.

2. If Q decides to perform the sequence $Q \stackrel{\tau}{\Longrightarrow} Q \| A^{n+1} \stackrel{a}{\longrightarrow} Q \| A^n$ then we take an $N > n$ and make $Q \| A^n$ perform the transition $Q \| A^n \stackrel{a}{\longrightarrow} A^n$. We know that $A^n \not\approx_N R$ and hence $Q \| A^n \not\approx_\omega R$. $\square$

Following up on this idea, for every n we can construct a pair of processes P_n, Q_n such that $P_n \approx_{\omega+n} Q_n$ and $P_n \not\approx Q_n$. The construction is as follows:

1. $P_0 = P$ and $Q_0 = Q$

2. assuming we have defined P_n and Q_n,

   $P_{n+1} \stackrel{a}{\longrightarrow} P_n$ and $Q_{n+1} \stackrel{a}{\longrightarrow} Q_n$.

$$
\begin{array}{ccc}
P_n & \approx_{\omega+n} & Q_n \\
\downarrow a & & \downarrow a \\
\vdots & \cdots & \vdots \\
\downarrow a & & \downarrow a \\
P_1 & \approx_{\omega+1} & Q_1 \\
\downarrow a & & \downarrow a \\
P & \approx_{\omega} & Q
\end{array}
$$

We can easily verify the following proposition:

Proposition 3.22 *For every n, $P_n \approx_{\omega+n} Q_n$ and $P_n \not\approx Q_n$.*

Proof: It is quite straightforward to show the validity of this proposition. We assume that $P_0 \approx_\omega Q_0$ and $P_0 \not\approx Q_0$ which was shown earlier. For the positive part, since the only transition both sides can do is $\xrightarrow{a}$, more precisely $P_{n+1} \xrightarrow{a} P_n$ and $Q_{n+1} \xrightarrow{a} Q_n$, the equivalence $P_{n+1} \approx_{\omega+n} Q_{n+1}$ follows from the assumption that $P_n \approx_{\omega+n} Q_n$. For the negative part, again because of the form of the available transitions we can argue that $P_n \approx Q_n$ if and only if $P_0 \approx Q_0$. Earlier we have demonstrated that $P_0 \not\approx Q_0$ which implies that also $P_n \not\approx Q_n$. $\square$

As a consequence of the former proposition we can observe that $\approx \subset \approx_{\omega+n}$ for every n. We may interpret it as a lower bound in this way: if α is an ordinal such that for every Basic Parallel Process Algebra, if $\approx = \approx_\beta$ then $\beta \leq \alpha$, then $\alpha \geq \omega \cdot 2$.

On the other hand, there is no known example of a pair of BPP which would be equivalent at level $\omega \cdot 2$ and yet not weakly bisimilar which leads to the following conjecture:

Conjecture 3.23 (Hirshfeld, Jančar) *For Basic Parallel Process Algebras, $\approx_{\omega \cdot 2} = \approx$.*

However simple this conjecture may seem, the question of its validity still remains open. Let us reason about a possible proof technique. We need to demonstrate the inclusion $\approx_{\omega \cdot 2} \subseteq \approx$. One way of doing that is to use the fact that $\approx$ is the largest weak bisimulation equivalence and contains all other weak bisimulation relations. Hence it suffices to show that $\approx_{\omega \cdot 2}$ is closed under expansion which then would imply that it is indeed a weak bisimulation. So we assume two arbitrary BPP P and Q such that $P \approx_{\omega \cdot 2} Q$ and we assume a transition $P \stackrel{\mu}{\Longrightarrow} P'$. Now we need to find a matching move $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx_{\omega \cdot 2} Q'$. However, there may be an infinite sequence of matching moves resulting in processes $Q'_0, Q'_1, \ldots, Q'_n, \ldots$ and we can only argue that $P' \approx_{\omega+n} Q'_n$ for every n. We can place some further restrictions on the sequence, for instance there is Dickson's lemma (cf. [27]) which allows us to assume that the processes form a non-decreasing sequence with respect to product order. Even this fact does not seem to be enough to overcome the hurdle of infinity and therefore we have to resort to smaller subclasses of BPP.

We will restrict ourselves to process algebras which only use one visible, i.e. non-τ, action. Although this restriction seems rather strict, we can argue that a single action was enough to construct BPA-processes that could distinguish

approximants $\approx_{\omega^n}$ from $\approx$, and in the case of BPP we could distinguish the level $\approx_{\omega+n}$ from $\approx$.

We need one further assumption which restricts the norm of the process variables. We recall that the weak norm $\|P\|$ of a process P is the length of the shortest derivation sequence from P to ϵ not counting τ-moves. Variables of weak norm zero are not allowed in our process algebra. Again, we can recall the BPA-processes D_i that were all of norm zero. Then we can demonstrate the following claim:

Proposition 3.24 *For all Basic Parallel Process Algebras with one visible action and no variables of norm zero, $\approx = \approx_{\omega \cdot 2}$.*

Proof: First we will make an important observation: if we have an algebra where all the rules use only one visible action then all processes of infinite norm are actually weakly bisimilar. The reason for that is that if a process has infinite norm then it can only perform infinite sequences of moves. When we consider a single non-τ action a then such a process can only perform infinite sequences of $\stackrel{a}{\Longrightarrow}$ moves and two such processes are indistinguishable. We also remind ourselves that for any two processes of different norms there exists an n such that these processes are not equivalent at level n. Hence if two processes P and Q are equivalent at level ω (or higher) then necessarily $\|P\| = \|Q\|$.

To verify the statement of the theorem we presuppose two processes P and Q such that $P \approx_{\omega \cdot 2} Q$. Hence for any transition $P \stackrel{a}{\Longrightarrow} P'$ and every i there exists $Q \stackrel{a}{\Longrightarrow} Q'_i$ such that $P' \approx_{\omega+i} Q'_i$ and we need to distinguish two cases according to the norm of P':

1. If $\|P'\| = \infty$ then also for every i the norm of Q'_i is infinite. We have observed earlier that all processes of infinite norm are weakly bisimilar and hence they must be related by $\approx_{\omega \cdot 2}$, that is $P' \approx_{\omega \cdot 2} Q'_i$ for every i.

2. If $\|P'\| = n$ for some natural number n, then also $\|Q'_i\| = n$ for all i. Each Q'_i is of the form $X_1^{q_{1i}} \parallel \ldots \parallel X_k^{q_{ki}}$ and we assume that all variables X_j have a non-zero finite norm, let us say n_j. We know that the norm is additive and so each process Q'_i has norm $\sum_{j=1}^{k} q_{ji} n_j$ which we assume equals n. Since all n_j are greater than 0, there can be only finitely many different combinations of exponents q_{ji} such that the sum adds up to n and hence only finitely many distinct Q'_i. Therefore there must be at least one Q' that occurs infinitely many times in the sequence $Q'_1, \ldots, Q'_i, \ldots$, and hence for infinitely many i, $P' \approx_{\omega+i} Q'$. Therefore $P' \approx_{\omega \cdot 2} Q'$ and we will choose Q' to be the required response from Q. $\Box$

It is well worth noting why we cannot replace $\approx_{\omega \cdot 2}$ with $\approx_\omega$ in the proof above. The point is rather subtle. It is the presence of unnormed processes which would create problems. Assume that $P \approx_\omega Q$ and we allow P to evolve with some move into an unnormed process P'. Then the response of Q is a sequence of processes $Q'_0, Q'_1, \ldots, Q'_i, \ldots$ but these may be normed because we are only guaranteed that $P' \approx_i Q'_i$ and so we cannot deduce that $\|P'\| = \|Q'_i\|$. Hence they cannot be forced to be weakly bisimilar and equivalent at $\approx_\omega$. We will see later that when we restrict ourselves to totally normed variables (that is variables of a positive finite norm) we will be able to prove that $\approx = \approx_\omega$.

If we study the proof of the above theorem thoroughly we find out that we do not use any special properties that only apply to BPP. The proof goes through without any problems for sequential algebras and hence we can conclude with the following proposition:

Proposition 3.25 *For all Basic Process Algebras with one visible action and no variables of norm zero,* $\approx = \approx_{\omega \cdot 2}$.

3.4.1 Decidability of $\approx_n$ for BPP

Now we will demonstrate the decidability of finite approximants $\approx_n$ for Basic Parallel Processes. We will follow the approach of Esparza in [16] that uses semilinear sets and their encoding as formulae of Presburger arithmetic which is decidable. The definition of semilinear sets, Presburger arithmetic and all related theorems are introduced in **Chapter 2**. Now we only recall the definition of $\approx_n$:

- $P \approx_0 Q$ for all P and Q
- $P \approx_{n+1} Q$ if for every action μ,
  - whenever $P \stackrel{\mu}{\Longrightarrow} P'$ then there exists $Q \stackrel{\mu}{\Longrightarrow} Q'$ so that $P' \approx_n Q'$ and
  - whenever $Q \stackrel{\mu}{\Longrightarrow} Q'$ then there exists $P \stackrel{\mu}{\Longrightarrow} P'$ so that $P' \approx_n Q'$.

It suffices to show that we can encode the individual approximants as formulae of Presburger arithmetic. It was proved in [16] that for every BPP-algebra and every action μ, the set of pairs $\{(P, P') \mid P \stackrel{\mu}{\Longrightarrow} P'\}$ is semilinear. A theorem by Ginsburg and Spanier [19] ensures that every semilinear set can be effectively expressed as a formula of Presburger arithmetic. Hence we presuppose a fixed BPPA $(\Sigma^\otimes, \Delta)$ and we can assume that we can construct a formula $\phi_\mu(P, P')$

of Presburger arithmetic such that $\phi_\mu(P,P')$ iff $P \stackrel{\mu}{\Longrightarrow} P'$. Assuming that we have such a formula we can now proceed to define formulae $\Phi_n(P,Q)$ such that $\Phi_n(P,Q)$ iff $P \approx_n Q$. The construction is done inductively as follows:

$$\begin{aligned}
\Phi_0(P,Q) &\equiv tt \\
\Phi_{n+1}(P,Q) &\equiv \bigwedge_{\mu\in L} \Big\{ \quad \forall P'.[\phi_\mu(P,P') \Rightarrow \exists Q'.(\phi_\mu(Q,Q') \wedge \Phi_n(P',Q'))] \\
&\qquad\qquad \wedge \qquad \forall Q'.[\phi_\mu(Q,Q') \Rightarrow \exists P'.(\phi_\mu(P,P') \wedge \Phi_n(P',Q'))] \quad \Big\}
\end{aligned}$$

The set L in the definition of formulas $\Phi_n(P,Q)$ consists of all actions that appear in the transitions rules of Δ. It is important that L can be easily enumerated and is always finite.

It is easy to verify that the construction is correct. The formula $\Phi_0(P,Q)$ is always true which corresponds to the approximant $\approx_0$ relating all processes P and Q. All the other formulas are a straightforward translation of the definition of approximants so we can conclude that

Proposition 3.26 *For any two BPP P and Q and for every n, $\Phi_n(P,Q)$ if and only if $P \approx_n Q$.*

For every P, Q the formulae $\Phi_n(P,Q)$ are closed formulae of Presburger arithmetic and thus decidable and so we can conclude with the final theorem:

Theorem 3.27 *On every Basic Parallel Process algebra, the approximants $\approx_n$ are decidable for every n.*

3.5 General properties of $\approx_\alpha$

Now we can demonstrate what we have claimed in the previous section, that the restriction to totally normed algebras is enough to ensure that $\approx = \approx_\omega$. That means we could construct a semidecision procedure for weak non-bisimilarity in an analogous way as in the case of strong non-bisimilarity, provided we could test each approximant $\approx_n$.

We recall that a process P is totally normed if the weak norm of P is positive and finite, i.e. $0 < \|P\| < \infty$. An algebra is totally normed if all its process variables are totally normed. First we will show a little lemma which relates weak norm and approximants.

Lemma 3.28 *For any pair of processes P and Q, if $\|P\| = n$, $P \approx_m Q$ and $n < m$, then $\|Q\| = n$.*

This lemma is intuitively true; if the process P has a weak norm n then P can perform a sequence of moves $\stackrel{w}{\Longrightarrow}$ that leads to ϵ and is of length n. As $m > n$, there must be a matching response $Q \stackrel{w}{\Longrightarrow} \epsilon$ of exactly the same length, so $\|Q\| \leq n$. On the other hand, if the weak norm of Q was less than n then Q would be able to perform a strictly shorter terminating sequence to which P would not have an adequate response and Q would not be equivalent with P at $\approx_m$.

Now we are ready to verify the claim about totally normed algebras. We will state it in a general way so that it can be applied to both BPA and BPPA.

Lemma 3.29 *For totally normed algebras, $\approx = \bigcap_{i \in \omega} \approx_i = \approx_\omega$.*

Proof: The proof follows the line of the proof for image-finite processes. We only need to verify one inclusion which is $\approx_\omega \subseteq \approx$.

We assume a pair of processes P and Q from a totally normed (BPA or BPP) algebra with the property $P \approx_\omega Q$. From the definition this is equivalent to the fact that $P \approx_n Q$ for all n. We consider a transition $P \stackrel{\mu}{\Longrightarrow} P'$. That will be matched by a sequence $Q'_0, Q'_1, \ldots,$ of $\stackrel{\mu}{\Longrightarrow}$ derivatives of Q such that $P' \approx_n Q'_n$ for every n. We know that the weak norm of P' is some positive number N. Now we make use of the lemma above that if we have a process P' such that $\|P'\| = N$, $P' \approx_m Q'$ and $N < m$, then $\|Q'\| = N$.

Therefore for every $m > N$ the Q-derivatives Q'_m must agree with P' on the norm, that is $\|Q'_m\| = \|P'\| = N$. Finally we wheel in the fact that there are only finitely many processes of a given norm in totally normed algebras, from which we can conclude that there must be a Q' occurring infinitely often in the sequence $Q'_m, Q'_{m+1}, \ldots$, thus being equivalent with P' for infinitely many indices, and so $P' \approx_\omega Q'$. We would follow a symmetric argument for any initial transition of the process Q. Hence we have shown that $\approx_\omega$ is a weak bisimulation and thus $\approx_\omega \subseteq \approx$. Since the opposite inclusion is trivially true we have indeed verified that $\approx_\omega = \approx$. $\Box$

We can now combine several facts to obtain another proof of decidability of weak bisimilarity for totally normed BPP. In **Section 3.4.1** we demonstrated that weak bisimulation approximants $\approx_n$ are decidable. The **Lemma 3.29** above then ensures that we can semidecide non-bisimilarity. That combined with Esparza's

semidecision procedure for bisimilarity presents another proof of the fact that $\approx$ is decidable for totally normed BPP.

We have been trying to estimate the ordinal number α such that for every algebra (BPA or BPPA), the chain of weak bisimulation approximants will have converged to maximal weak bisimulation at the level α. We need to consider the maximum ordinal taken over the whole class of BPA, resp. BPPA. So far, we have produced some lower bounds on such an ordinal which is ω^ω for basic process algebras and $\omega \cdot 2$ for Basic Parallel Process Algebras.

Now we can try to establish some upper bounds on the level of convergence. That does not seem to be so easy as we do not have appropriate tools that could establish the maximal level of convergence, even for a specific algebra. It seems that the only claim we can make stems from the fact that the process algebras we deal with are countable. We have already showed that

$$\approx_\alpha = \approx_{\alpha+1} \Longrightarrow \approx_\alpha = \approx,$$

that is if two subsequent levels α and $\alpha + 1$ define the same equivalence then all levels β for $\alpha \leq \beta$ are equal and hence equal the maximal weak bisimulation. We can define only countably many processes and hence countably many pairs of processes which means we can never distinguish more than countably many approximants. That can be expressed as follows:

Lemma 3.30 $\approx = \approx_{\omega_1}$.

Obviously, this is a rather crude upper bound (ω_1 is the first uncountable ordinal). Bradfield observed that there exists a stronger upper bound that can be obtained as follows. Non-bisimulation is an inductively defined property, and the monotone (and indeed positive) operator over which induction occurs is arithmetical, since the $\stackrel{\mu}{\Longrightarrow}$ relation for BPA is clearly arithmetical. There is a theorem due to Spector (consult Theorem IV.2.15 in [25]) that any inductive definition over a monotone arithmetical (or even Π^1_1) operator has closure ordinal $\leq \omega_1^{CK}$, the least non-recursive ordinal.

However, it seems plausible that there must be yet smaller (countable) ordinals that will provide an upper bound, such as ω^ω for BPA and $\omega \cdot 2$ for BPP. It seems that in order to prove any stronger claim we need to develop some technique that would enable us to estimate the convergence for any given process algebra. Such a technique might require further involvement of ordinal numbers that would in some way capture the ability of processes to maintain equivalence with other processes at high levels of $\approx_\alpha$ whilst being actually weakly non-bisimilar.

3.6 Conclusions

In this chapter we were concerned with the relation between Basic Process Algebras and Basic Parallel Process Algebras, and weak bisimulation, more precisely weak bisimulation approximants. We established lower bounds on ordinal numbers labelling the approximants that are equal to $\approx$ which was $\approx_{\omega \cdot 2}$ in case of BPP and $\approx_{\omega^\omega}$ in case of BPA. We showed decidability of $\approx_n$ for BPP and we discussed the importance of processes of zero and infinite norm.

However, there are a few questions to which we did not manage to find satisfactory answers. What are the least ordinals, both for BPA and BPPA, that label the approximant equal to the maximal weak bisimulation. Hirshfeld and Jančar independently conjectured that for Basic Parallel Process Algebras, $\approx = \approx_{\omega \cdot 2}$. We stated a hypothesis that in the case of Basic Process Algebras, $\approx = \approx_{\omega^\omega}$. We did not manage to verify those hypotheses since the only upper bound we could establish was the first uncountable ordinal ω_1 (in fact, Bradfield observed that it is ω_1^{CK}, the least non-recursive ordinal). Related to the question of decidability of BPP (or rather semidecidability of $\not\approx$) is the following: is $\approx_\omega$ decidable, and are $\approx_{\omega+n}$ decidable for every n?

With BPP, we can also investigate Milner's approximants $\approx_\alpha^M$ which may contribute to the solution to the semidecidability of $\not\approx$ problem. Although we know that $\approx_n^M$ are undecidable for every $n > 0$, we are not certain about the status of $\not\approx_n^M$ and we can spell out another conjecture: $\approx = \approx_\omega^M$. In case that $\not\approx_n^M$ were semidecidable and the conjecture was proved that would establish decidability of $\approx$ for BPP.

Chapter 4

Lower bound results

The decidability of weak bisimilarity for the restricted subclass of *totally normed* BPP and BPA-processes was established by Hirshfeld in [28]. In the case of totally normed BPA-processes it can be also viewed as a consequence of Stirling's result of decidability for strong bisimilarity on normed pushdown automata [58]. A semidecision procedure for weak bisimilarity of BPP was presented by Esparza [16], however decidability for general BPA-processes and BPP remains open. Since the decision problem seems to be rather difficult to solve we may try to study easier aspects of weak bisimilarity, for example its *hardness*. To be precise, we will try to establish some lower bounds on the computational complexity of a decision procedure that might exist.

So far, decidability of (strong) bisimilarity for simple process algebras has been consistent with a polynomial-time decision procedure. Polynomial algorithms deciding strong bisimilarity for normed BPP and BPA-processes were demonstrated by Hirshfeld, Jerrum, and Moller in [29], [30], [31]. Even though there is no known polynomial decision procedure for the classes of general BPP and BPA-processes, there is no evidence (lower bound) that would contradict its existence. Since weak bisimilarity is a much more complex notion we might expect that any existing decision procedure would be of rather high computational complexity. Indeed, a possible result in that direction would be to show that weak bisimilarity cannot be decided in polynomial time. To the best of our knowledge even this result has not been showed yet. Here we provide a strong evidence in favour by demonstrating that for weak bisimilarity and (totally normed) BPP and BPA-processes the decision problem is NP-hard and for general BPA-processes, decidability would imply a PSPACE-hard decision procedure.

In order to obtain these results we will make use of a *reduction*, a concept which is widely used in both recursion theory and computational complexity theory. The principle of a reduction is that we construct a translation of a decision problem $\mathcal{P}$

to a decision problem $\mathcal{Q}$ which is *efficient* and maps instances of $\mathcal{P}$ to *equivalent* instances of $\mathcal{Q}$. If we know the complexity of our chosen problem $\mathcal{P}$ then we will be able to gauge the hardness of $\mathcal{Q}$. First we will recall a few important notions from the field of computational complexity.

4.1 Computational complexity

In order to define *time* or *space complexity* of a problem (language) we employ the concept of Turing machines (see e.g. [33]). We say that a Turing machine M *decides* a language L over some alphabet Σ if M halts on every input string $w \in \Sigma^*$ and M accepts w if and only if $w \in L$. Assuming that $T(n)$ is a function on natural numbers, we say that a machine M has *time complexity* $T(n)$ if for every n and for every input string w of length n the amount of time required for the computation of the machine on w is bounded by $T(n)$. Also the language decided by M is said to be of time complexity $T(n)$. If $S(n)$ is a function on natural numbers, we say that a Turing machine M has *space complexity* $S(n)$ if for every n and for every input string w of length n the amount of space required for the computation of M on the input w is bounded by $S(n)$. The language decided by M is then of space complexity $S(n)$.

Remark: If there is a Turing machine of time complexity $T(n)$ deciding a language L then the function $T(n)$ is an upper bound on the inherent hardness of L. There may exist other Turing machines of lower time complexity deciding L. However, that is inevitable because there are languages for which no "best" Turing machine exists (in terms of time complexity). The same applies to space complexity $S(n)$ of languages.

4.1.1 Complexity classes

We distinguish between *deterministic* and *nondeterministic* computation, that is computation performed by *deterministic* and *nondeterministic* Turing machines. We group languages into complexity classes according to their time or space complexity. DTIME($f(n)$) consists of languages decidable by deterministic Turing machines whose time complexity is bounded by $f(n)$, DSPACE($f(n)$) consists of languages decidable by deterministic Turing machines whose space complexity is bounded by $f(n)$. The classes NTIME($f(n)$) and NSPACE($f(n)$) are defined analogously in terms of nondeterministic Turing machines. We are particularly interested in classes that involve polynomial time and space complexity, as defined below.

$$\begin{aligned} \text{P} &= \cup_{c>0}\, \text{DTIME}(n^c) \\ \text{NP} &= \cup_{c>0}\, \text{NTIME}(n^c) \\ \text{PSPACE} &= \cup_{c>0}\, \text{DSPACE}(n^c) \\ \text{NPSPACE} &= \cup_{c>0}\, \text{NSPACE}(n^c) \end{aligned}$$

The relationship between these classes is as follows:

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE}$$

It is not known whether any of these inclusions is proper, however it is widely assumed that P $\neq$ NP and NP $\neq$ PSPACE. Any problem that belongs to P is considered feasible because there exists an algorithm that solves it in polynomial time. Generally, problems outside P are looked upon as hard to solve. Assuming P $\neq$ NP, already the class NP contains some hard problems. The class PSPACE is widely assumed to contain some problems harder than all problems in NP and so problems complete for PSPACE are considered not feasible. Now we will explain how we can compare complexities of various languages.

4.1.2 Reductions

We define computational complexity of languages in terms of Turing machines whose output is either *accept* or *reject*. Hence we may view a Turing machine to be computing a function from the set of strings over the input alphabet to the two-element set $\{0, 1\}$. In the context of reductions we need to be able to compute functions from one set of strings to another set of strings and so we will consider special kinds of Turing machines that are called *transducers*. A transducer is a Turing machine which consists of a read-only input tape, a work-tape and a write-only output tape on which the head always moves to the right. The function computed by some transducer is defined in the obvious way.

Assume two languages L_1 over some alphabet Σ_1 and L_2 over an alphabet Σ_2. A *reduction* from L_1 to L_2 is a function f from Σ_1^* to Σ_2^* such that

$$\text{for all } w \in \Sigma_1^*, \qquad w \in L_1 \iff f(w) \in L_2.$$

A reduction f is *polynomial-time* if there exists a polynomial-time bound transducer that computes the function f. A reduction f is *log-space (logarithmic space)* if it is computable by a transducer with log-space bounded work-tape. Every log-space reduction is also a polynomial-time reduction. Polynomial-time reduction

will suffice for our purposes as will be shown later. We will denote the fact that L_1 is polynomial-time reducible to L_2 by $L_1 \preccurlyeq L_2$.

We say that a language L is *complete* for a class $\mathcal{C}$ (with respect to polynomial-time reduction) if L is in $\mathcal{C}$ and every language in $\mathcal{C}$ is polynomial-time reducible to L. A language L is *hard* for $\mathcal{C}$ (with respect to polynomial-time reduction) if every language in $\mathcal{C}$ is reducible to L, but L is not necessarily in $\mathcal{C}$. The concept of hardness is not enough to determine the complexity of a language, however it provides us with a lower bound. We will be mainly dealing with the classes NP and PSPACE and the following theorem [33] confirms that polynomial-time reduction is suitable for our purposes.

Lemma 4.1 *If $L_2 \in$ P and $L_1 \preccurlyeq L_2$ then also $L_1 \in$ P.*

If there are languages $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ such that L_1 is polynomial-time reducible to L_2 then there exists a Turing machine M that transforms input strings over Σ_1 into equivalent strings over Σ_2 and whose time complexity is bounded by some polynomial $p(n)$. That means that each word w_1 of length n will be transformed into a word w_2 of length at most $p(n)$. If we can decide the language L_2 in polynomial time, let us say $q(n)$, then clearly via the machine M we can decide the language L_1 in time $p(n) + q(p(n))$ which is clearly also polynomial.

Since the composition of two polynomial-time reductions is also polynomial-time we obtain the following statement:

Lemma 4.2 *If L_1 is $\mathcal{C}$-complete and $L_1 \preccurlyeq L_2$ then L_2 is $\mathcal{C}$-hard.*

4.1.3 Decision problems

We have defined our complexity notions in terms of languages. It is more convenient to deal with a slightly less formal concept of *decision problems*. A *decision problem* is characterised by a set of *instances* of the problem and a *YES/NO question* that one asks about the instances. Each language defines a decision problem in this way: for a language L, a subset of Σ^*, the corresponding decision problem is to decide whether a word w belongs to L, for any word over the alphabet Σ. Obviously, if we have a Turing machine that decides L we can easily modify it into a Turing machine deciding the corresponding decision problem hence languages and decision problems are closely related. From now on, we will only consider decision problems.

Example 4.3 Let us consider the *Hamilton circuit problem (*HAM*)* which is defined as follows: given a graph G, we want to determine if there exists a path in G which visits each vertex exactly once and returns to its starting point. To encode HAM formally we presuppose an alphabet Σ (whose letters will be used to encode graphs), for instance $\Sigma = \{0, 1, (,)\}$. The language L_{Ham} is a subset of Σ^* which corresponds to encodings of graphs with Hamilton circuits. The decision problem HAM is then given as follows:

Instance: A graph G.
Question: Does G contain a Hamilton circuit? $\square$

More details and examples of complete problems can be found in Papadimitriou [54], and Garey and Johnson [17].

4.2 Weak bisimilarity of BPP is NP-hard

In order to show NP-hardness of weak bisimilarity we choose the problem KNAPSACK which is known to be NP-complete (originally proved by Karp in [42]). We will reduce it to weak bisimilarity of Basic Parallel Processes. As we have explained earlier that will make the decidability of $\approx$ for BPP at least NP-hard.

KNAPSACK (also called SUBSET SUM) is a combinatorial problem which compares sums of natural numbers. We are given a sequence of natural numbers $m_1, m_2, \ldots, m_n$ and a total t and we want to find out whether we can choose a subsequence $m_{i_1}, \ldots, m_{i_k}$ that adds up to t. That brings us to the idea of having two processes, one representing the total t by being defined as a trace of length t, and the other representing the choices of subsequences of $m_1, m_2, \ldots, m_n$ and hence giving rise to a tree whose branches correspond to traces of lengths specified by the individual subsequences. Formally, the definition of the problem is as follows:

Definition 4.4 KNAPSACK *is the following problem:*

Instance: $t, m_1, m_2, \ldots, m_n \in \mathbb{N}$
Question: $\exists\ i_1, i_2, \ldots, i_n \in \{0, 1\}.\ \sum_{j=1}^{n} i_j m_j = t$*?*

We will follow the convention that all the input values $t, m_1, m_2, \ldots, m_n$ are encoded in binary [17]. That is an essential requirement because if we consider KNAPSACK with the input encoded in unary then we can actually construct an algorithm that will solve it in polynomial time. We say that KNAPSACK is not

strongly NP-complete [17]. However, with respect to binary encoding it is NP-complete.

The fact that we assume input values encoded in binary means that we can encode large numbers in a succinct way. That means we will have to deal with large values in the definition of processes that occur in the reduction. That will require a trick in the definition so that we remain within the limits of polynomial-time reduction. We will now proceed to demonstrate a polynomial time many-one reduction of KNAPSACK to weak bisimilarity of BPP.

Lemma 4.5 KNAPSACK $\preccurlyeq \approx$.

Proof: Let $t, m_1, m_2, \ldots, m_n \in \mathbb{N}$ be an instance of KNAPSACK. We will demonstrate two Basic Parallel Processes P and Q such that there exist $i_1, i_2, \ldots, i_n \in \{0,1\}$ with $\sum_{j=1}^{n} i_j m_j = t$ if and only if $P \approx Q$. Following the aforementioned idea, the process P will simulate branching defined by individual subsequences, and the process Q will simulate a trace of length t. For the purpose of counting we will use a single visible action a that will help us to test if there is a branch in the tree defined by P of length t, i.e. equivalent with Q.

For each m_j, resp. t, we will introduce a process variable M_j, resp. T, that will be able to perform exactly a sequence of $\stackrel{a}{\Longrightarrow}$ transitions of length m_j, resp. t. The process P then will be capable of generating any subset of $\{M_1, \ldots, M_n\}$ whereas the process Q will be able to evolve into T. Finally we will demonstrate that P is weakly bisimilar to Q if and only if the answer to the corresponding instance is yes. Now we will present the transition rules that define the process variables P and Q:

$$P \stackrel{\tau}{\longrightarrow} P_1 \parallel \ldots \parallel P_n \qquad Q \stackrel{\tau}{\longrightarrow} P \qquad P_j \stackrel{\tau}{\longrightarrow} M_j, \quad j = 1, \ldots, n$$
$$Q \stackrel{\tau}{\longrightarrow} T \qquad P_j \stackrel{\tau}{\longrightarrow} \epsilon, \quad j = 1, \ldots, n$$

In order to complete the definitions of P and Q we have to define process variables M_j and T. Our only concern is that the resulting reduction is polynomial time hence we have to use a little trick in the definition. We define a sequence of variables $S_0, S_1, \ldots, S_k$ in this way: $S_0 \stackrel{a}{\longrightarrow} \epsilon$, $S_{i+1} \stackrel{\tau}{\longrightarrow} S_i \| S_i$ for $i < k$, where k is taken to be $\lfloor \log(\max\{t, m_1, \ldots, m_n\}) \rfloor$. Thus we have obtained variables such that $S_i \approx a^{2^i}$, where we use the expression a^m in the obvious meaning. Now we can define $T \stackrel{\tau}{\longrightarrow} S_k^{e_k} \| \ldots \| S_1^{e_1} \| S_0^{e_0}$ where $e_k \ldots e_1 e_0$ is the binary encoding of t (the expression on the right-hand side of the rule is written as $S_k^{e_k} \| \ldots \| S_1^{e_1} \| S_0^{e_0}$ in order to make the idea clear; in fact the variables S_i with the respective exponent e_i being equal to 0 will not be present). The variables $M_0, \ldots, M_n$ are defined in a similar fashion: $M_j \stackrel{\tau}{\longrightarrow} S_k^{e_{kj}} \| \ldots \| S_1^{e_{1j}} \| S_0^{e_{0j}}$, where $e_{kj} \ldots e_{0j}$ is the binary

encoding of m_j, for $j = 1, \ldots, n$. To summarise, we only need $k+1$ extra variables in order to define the processes T and M_j.

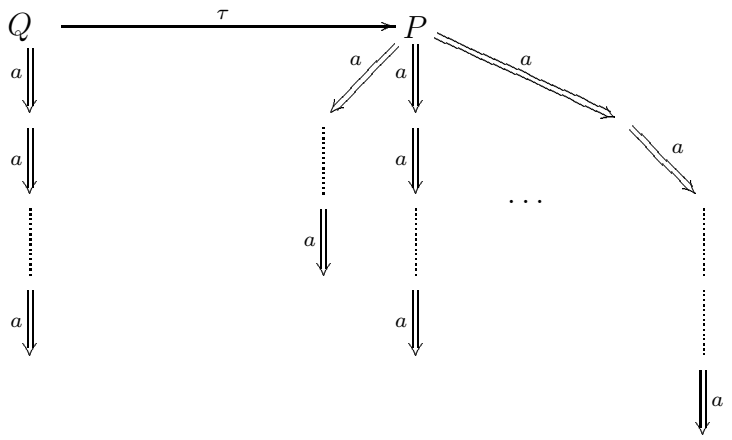


Figure 4.1: The processes P and Q

The transition systems determined by processes P and Q are illustrated in Fig. 4.1. It is easily seen from the construction that P can only perform sequences of $\stackrel{a}{\Longrightarrow}$ transitions of length $\sum_{j=1}^{n} i_j m_j$ for some $i_1, i_2, \ldots, i_n \in \{0, 1\}$. Therefore if this sum never adds up to t the process Q can become T and thus force non-bisimilarity with P. On the other hand, if there exist $i_1, i_2, \ldots, i_n \in \{0, 1\}$ such that $\sum_{j=1}^{n} i_j m_j = t$ the process P will generate the composition $M_1^{i_1} \| \ldots \| M_n^{i_n}$ as an answer to the move $Q \stackrel{\tau}{\longrightarrow} T$ and preserve weak bisimilarity. If it is P that takes the initiative then the process Q simply makes use of the rule $Q \stackrel{\tau}{\longrightarrow} P$ and then copies any move of P. $\square$

It remains to check that the reduction is polynomial-time. We will prove that by examining the reduction in detail and we will actually show that the reduction is log-space. We define a log-space transducer that performs this reduction as follows: it consists of a read-only input tape, a work-tape, and a write-only output tape on which the head only moves to the right. The input is encoded in binary and hence the size of the input is roughly $n \cdot k$, where $k = \lfloor \log \max\{t, m_1, \ldots, m_n\} \rfloor + 1$. In the construction we use $2n+k+4$ variables for which we need $\lfloor \log(2n+k+4) \rfloor + 1$ space, that is about $\log(n+k)$. We will set up a counter on the work-tape which will contain the following information: (the encoding of) the last variable, the variable being currently defined and then some auxiliary information. All this takes up space of about $\log n + \log k + \log(n+k)$.

We proceed to encode the algebra in this fashion: we start with the sequence of rules $S_{i+1} \xrightarrow{\tau} S_i \| S_i$. In order to do that we need to keep track of the previously defined variable, that is to encode the transition of S_{i+1} we need to remember S_i which requires $\log k$ space. Then we continue with the transition rules for individual M_j and T. For that we just need to be able to work out the encoding of some S_i if it appears on the right hand side of some rule. We will also write down the encoding of M_1 so that we know how to start in the definition of P_1. That requires a counter for keeping track of which P_j is being defined. Finally, we finish off with the encoding of P and Q. Clearly all the information stored in the work-tape only takes up space which is logarithmic in the size of the input. Since the head only moves to the right on the output tape we can conclude that the reduction is indeed polynomial-time. Hence we have verified the following theorem:

Theorem 4.6 *The decidability of weak bisimilarity of Basic Parallel Processes is NP-hard.*

4.2.1 Totally normed BPP

The result that we have just proved appears rather weak in the light of the fact that so far there exists only a semidecision procedure. Now we will present a stronger result by modifying the reduction so that the resulting processes belong to the restricted subclass of totally normed BPP, for which weak bisimilarity is actually decidable [28]. In the original reduction we define several variables which are of norm zero. The class of totally normed processes does not admit such variables and so we will replace them with variables of positive norm.

The problematic processes are P_i since they have at their disposal the transition rules $P_i \xrightarrow{\tau} \epsilon$. We can get rid of such processes by considering the problem $\sum_{j=1}^{n} i_j m_j + \sum_{j=1}^{n} m_j = t + \sum_{j=1}^{n} m_j$ instead. Clearly, there exist coefficients $i_j \in \{0,1\}$ such that $\sum_{j=1}^{n} i_j m_j = t$ if and only if there exist i'_j such that $\sum_{j=1}^{n} i'_j m_j = t + \sum_{j=1}^{n} m_j$, where $i'_j \in \{1,2\}$. Following this idea we define new process variables P'_i and P' using the transitions $P'_i \xrightarrow{\tau} M_i \| M_i$, $P'_i \xrightarrow{\tau} M_i$, and $P' \xrightarrow{\tau} P'_1 \| \ldots \| P'_n$. The fact that we are now simulating either the number m_j or its double $2m_j$ means that the moves $P_i \xrightarrow{\tau} \epsilon$ are no longer present. We also need to replace the process Q representing t with a process Q' representing $t + \sum_{j=1}^{n} m_j$. The process Q' is defined by the two transitions $Q' \xrightarrow{\tau} P'$ and $Q' \xrightarrow{\tau} T \| M_1 \| \ldots \| M_n$.

It is obvious that for the processes P' and Q' defined above, $P' \approx Q'$ if and only if they correspond a positive instance of KNAPSACK. This modification does

not have any impact on the size of the reduction which can still be done in log-space. All the newly defined processes are totally normed and we can conclude with the following theorem:

Theorem 4.7 *The decidability of weak bisimilarity of totally normed Basic Parallel Processes is NP-hard.*

4.2.2 Totally normed BPA-processes

For general BPA-processes we will demonstrate PSPACE-hardness of the decision problem for weak bisimilarity but before doing that we will modify the reduction from KNAPSACK to sequential composition in order to show the following theorem:

Theorem 4.8 *To decide weak bisimilarity of totally normed BPA-processes is NP-hard.*

Proof: The proof follows very much the ideas we used in the reduction to weak bisimilarity of BPP. Given an instance $t, m_1, \ldots, m_n$ of KNAPSACK, we will define processes P and Q in this way:

$$\begin{array}{llll} P & \stackrel{\tau}{\longrightarrow} P_1 M_1 & P & \stackrel{\tau}{\longrightarrow} P_1 M_1^2 \qquad Q \stackrel{\tau}{\longrightarrow} P \\ P_1 & \stackrel{\tau}{\longrightarrow} P_2 M_2 & P_1 & \stackrel{\tau}{\longrightarrow} P_2 M_2^2 \qquad Q \stackrel{\tau}{\longrightarrow} T \\ & \vdots & & \vdots \\ P_{n-1} & \stackrel{\tau}{\longrightarrow} M_n & P_{n-1} & \stackrel{\tau}{\longrightarrow} M_n^2 \end{array}$$

Note that it is not enough to simply replace parallel composition with sequential in the definition of P because of the leftmost derivation in case of BPA. If we took P to be $P_1 P_2 \ldots P_n$ then we would not be able to derive any process $M_1^{i_1} M_2^{i_2} \ldots M_n^{i_n}$ in a single $\stackrel{\tau}{\Longrightarrow}$ transition because P_i is only enabled when all P_j have terminated for all $j < i$. Therefore we would have to start producing $\stackrel{a}{\Longrightarrow}$ transitions which would introduce more possibilities for branching and spoil bisimilarity.

The definitions of T and each M_j are again expressed in terms of auxiliary variables S_i:

$$S_0 \stackrel{a}{\longrightarrow} \epsilon \quad S_{i+1} \stackrel{\tau}{\longrightarrow} S_i S_i \quad T \stackrel{\tau}{\longrightarrow} S_k^{e_k} \ldots S_1^{e_1} S_0^{e_0} \quad M_j \stackrel{\tau}{\longrightarrow} S_k^{e_{kj}} \ldots S_1^{e_{1j}} S_0^{e_{0j}},$$

where $1 \leq i < \lfloor \log(\max\{t, m_1, \ldots, m_n\}) \rfloor$, and $e_k \ldots e_1 e_0$, resp. $e_{kj} \ldots e_{1j} e_{0j}$, are the binary encodings of t, resp. m_j. For these processes there is no branching available, they only determine a sequence of $\stackrel{a}{\Longrightarrow}$ moves of the right length.

When we examine the possible behaviour of P we can see that before doing an $\stackrel{a}{\Longrightarrow}$ transition it must have evolved into a composition of variables of the form

$M_n^{i_n} \ldots M_2^{i_2} M_1^{i_1}$ with $i_j \in \{1,2\}$. That represents some subset of $m_1, \ldots, m_n$ and P can perform exactly $\sum_{j=1}^{n} i_j m_j$ actions $\stackrel{a}{\Longrightarrow}$ for the corresponding i_j. Then we apply a similar argument as in the case for BPP and we can conclude that $P \approx Q$ if and only if there are $i_1, \ldots, i_n \in \{1,2\}$ so that $\sum_{j=1}^{n} i_j m_j = t$. All the variables are totally normed which concludes the proof. $\Box$

We can see that the presented reduction constructs roughly the same number of processes as the reduction to (totally normed) BPP. Hence we can rely on the analysis of the former reduction and conclude that also the reduction to totally normed BPA-processes is log-space.

4.3 Weak bisimilarity of BPA is PSPACE-hard

Sequential composition, however, enables us to go even further. With the sequential structure of BPA-processes we are able to encode finite automata and hence achieve a stronger result. We will use the *totality problem for finite automata* TOT which is PSPACE-complete and construct a polynomial time reduction to weak bisimilarity of BPA-processes. Thus we will show that this problem is at least PSPACE-hard. First we will define the totality problem for finite automata:

Definition 4.9 TOT *is the following problem:*

> *Instance: A nondeterministic finite automaton $\mathcal{A}$ over some alphabet Σ.*
> *Question: Is $L(\mathcal{A})$, the language accepted by the automaton $\mathcal{A}$, equal to the total language Σ^*?*

This problem is PSPACE-complete even for a two-letter alphabet (cf. [17]) hence in the following we will assume that $\Sigma = \{a, b\}$. We will in fact demonstrate a linear time reduction of TOT to weak bisimilarity of BPA-processes.

Theorem 4.10 TOT $\preccurlyeq \approx$.

Now we will explain the main idea behind the reduction. We presuppose a nondeterministic finite automaton $\mathcal{A} = (\Sigma, \mathcal{Q}, \delta, q_0, \{q_k\})$, where $\Sigma = \{a, b\}$ is the input alphabet, $\mathcal{Q} = \{q_0, q_1, \ldots, q_k\}$ is the set of states with q_0 being the initial and q_k the final states, and $\delta : \mathcal{Q} \times \Sigma \to 2^{\mathcal{Q}}$ is the transition function. We will write $(q_i, x) \longmapsto q_j$ to express the fact that the state q_j belongs to the set $\delta(q_i, x)$ with x being either a or b. Also note that without loss of generality we can assume a single final state q_k.

We will simulate words over $\{a, b\}$ by introducing two variables A and B such that A can only perform the transition $A \xrightarrow{a} \epsilon$ and B can only perform the transition $B \xrightarrow{b} \epsilon$. Then any process over A and B will determine a single word over the alphabet Σ. To simulate the total language Σ^* we will introduce a process P that will be capable of producing any string of atoms A and B. Next we define a process Q that can generate all strings from $L(\mathcal{A})$. However since we are only allowed the leftmost derivation in the case of BPA-processes, we will define a process that generates exactly all the reverse words from $L(\mathcal{A})$. Still, we will be able to show that $L(\mathcal{A}) = \Sigma^*$ if and only if $P \approx Q$.

For each state $q_0, q_1, \ldots, q_k$ of the automaton $\mathcal{A}$ we define a process variable $Q_0, Q_1, \ldots, Q_k$ using the following rules:

if $(q_i, a) \longmapsto q_j$ then $Q_i \xrightarrow{\tau} Q_j A$, and
if $(q_i, b) \longmapsto q_j$ then $Q_i \xrightarrow{\tau} Q_j B$.

For the variable Q_k that corresponds to the final state q_k we add a special rule $Q_k \xrightarrow{s} \epsilon$, where s is a special *initial* action. The purpose of s is to mark that the automaton $\mathcal{A}$, resp. the corresponding process Q, has reached a final state and halted. Finally, we put $Q = Q_0$. It is quite straightforward to observe that a word w is in $L(\mathcal{A})$ if and only if we can with $\stackrel{\tau}{\Longrightarrow}$ transition from the variable Q generate the process $Q_k R$ where $R \in \{A, B\}^*$ determines precisely the word $\bar{w}$, the reverse of w.

Example 4.11 To illustrate the construction above we will now show an example of a nondeterministic finite automaton and the process that it gives rise to. The automaton $\mathcal{A}$ (Fig. 4.2) consists of a set of states $\{q_0, q_1, q_2, q_3\}$ with q_0 being the initial and q_3 the final states, the alphabet is $\Sigma = \{a, b\}$ and the transition function is given by $\delta(q_0, c) = \{q_1\}$, $\delta(q_1, a) = \{q_2, q_3\}$, and $\delta(q_2, b) = \{q_1, q_3\}$. The language it defines is given by the regular expression $c(ab)^*a + c(ab)^* = \{c(ab)^i a \mid i \geq 0\} \cup \{c(ab)^i \mid i > 0\}$.

The process Q is defined by the following transition rules and is pictured in Fig. 4.3:

$$\begin{array}{lll} Q_0 \xrightarrow{\tau} Q_1 C & Q_1 \xrightarrow{\tau} Q_2 A & Q_1 \xrightarrow{\tau} Q_3 A \\ Q_2 \xrightarrow{\tau} Q_1 B & Q_2 \xrightarrow{\tau} Q_3 B & Q_3 \xrightarrow{s} \epsilon \end{array}$$

It is not difficult to verify that the only processes derivable from the process Q_0 with $\stackrel{s}{\Longrightarrow}$ move are either of the form $A(BA)^i C$ or $(BA)^i C$. Clearly a process

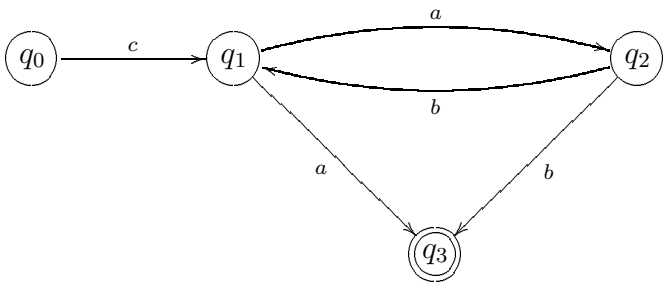


Figure 4.2: The nondeterministic automaton $\mathcal{A}$

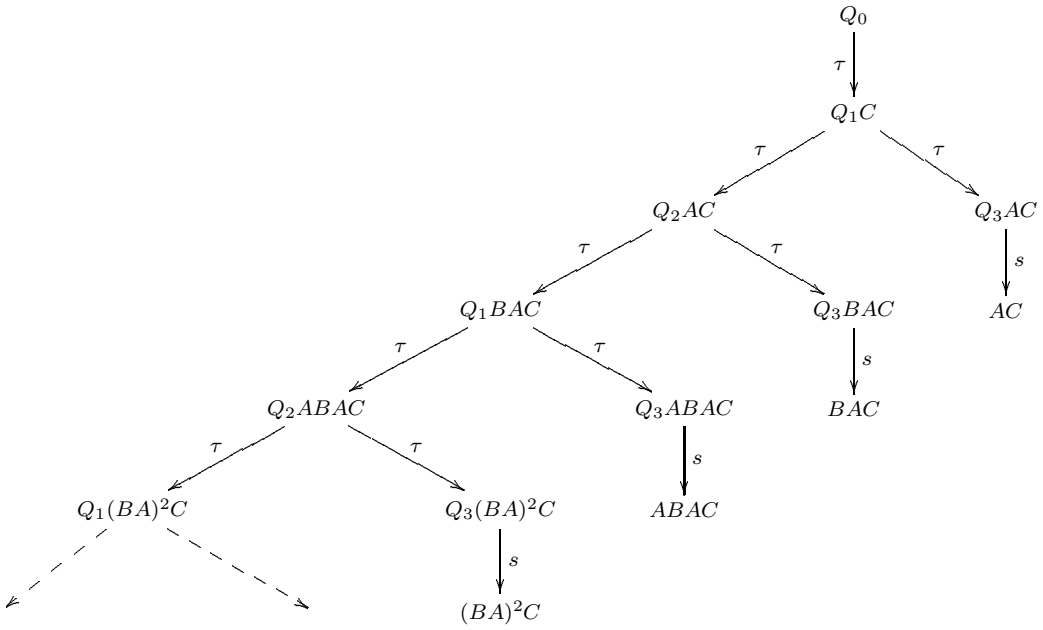


Figure 4.3: The corresponding process Q

$A(BA)^iC$, resp. $(BA)^iC$, determines the word $a(ba)^ic$, resp. $(ba)^ic$, which is the reverse of $c(ab)^ia$, resp. $c(ab)^i$. $\square$

Now we define the process P whose task is to be able to represent all strings from $\{a,b\}^*$ and simulate the process Q, and also the variables A, resp. B, that simulate the letters a, resp. b.

$$P \xrightarrow{\tau} QT \qquad P \xrightarrow{\tau} PA \qquad A \xrightarrow{a} \epsilon \qquad T \xrightarrow{\tau} T$$
$$P \xrightarrow{s} \epsilon \qquad P \xrightarrow{\tau} PB \qquad B \xrightarrow{b} \epsilon$$

For technical reasons we need a process that will block any sequence of variables that the process P may have generated. The process T forms such a block since

it is defined as a τ loop and therefore it is clear that $TR \approx \epsilon$ for any process R. The presence of T in the algebra means that the algebra fails to be totally normed which opens the question about the complexity of weak bisimilarity for totally normed processes. The final step is to show the correctness of the reduction.

Lemma 4.12 *Assume a given automaton $\mathcal{A}$ and P and Q defined as above. Then $L(\mathcal{A}) = \{a, b\}^*$ iff $P \approx Q$.*

Proof: One implication is straightforward. Assume that $L(\mathcal{A}) \neq \{a, b\}^*$ and let $w \in \{a, b\}^* \setminus L(\mathcal{A})$. Since P is constructed to generate all strings of a and b it can produce a sequence of variables capable of performing the word $s\bar{w}$, where $\bar{w}$ is the reverse of w. However, as Q simulates the automaton $\mathcal{A}$ it cannot produce the string $\bar{w}$ and thus $P \not\approx Q$.

In order to show the other direction we need to analyse the moves of P and Q. The idea is that P will wait for Q to make a move and then respond by doing $P \xrightarrow{\tau} QT$ which blocks anything which P may have generated in the meantime. Clearly $Q \approx QTR$ for any process R because we can never get past T. On the other hand, Q has to respond only when P decides to generate a sequence of A's and B's and then disappear. Hence the responses of Q are:

1. $P \stackrel{\tau}{\Longrightarrow} PR, R \in \{A, B\}^*$ - in this case Q does the empty sequence $Q \stackrel{\epsilon}{\Longrightarrow} Q$
2. $P \stackrel{\tau}{\Longrightarrow} QTR, R \in \{A, B\}^*$ - in this case $Q \approx QTR$ and hence again the response is $Q \stackrel{\epsilon}{\Longrightarrow} Q$
3. $P \stackrel{s}{\Longrightarrow} R, R \in \{A, B\}^*$ - since Q can generate all strings over the alphabet $\{A, B\}$ it will be able to generate the process R via $\stackrel{s}{\Longrightarrow}$.

We will make a formal argument out of the informal analysis above by actually constructing a weak bisimulation relation containing the pair (P, Q). We define a binary relation $\mathcal{R}$ as a union of three subrelations, $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$, where

$$\mathcal{R}_1 = \{(Q_i RTR', Q_i R) \mid R, R' \in \{A, B\}^*\}$$

$$\mathcal{R}_2 = \{(PR, Q) \mid R \in \{A, B\}^*\}$$

$$\mathcal{R}_3 = \{(RTR', R) \mid R, R' \in \{A, B\}^*\}$$

To establish that $\mathcal{R}$ is a weak bisimulation relation we have to verify that $\mathcal{R}$ is closed under expansion. This naturally falls into three cases:

1. Given a pair (Q_iRTR', Q_iR) from $\mathcal{R}_1$, the possible transitions are only those of Q_i and each process will always simulate the other's move. There are two transitions to distinguish: either $Q_iR \xrightarrow{\tau} Q_jXR$ with X being from $\{A, B\}$ in which case (Q_jXRTR', Q_jXR) belongs to $\mathcal{R}_1$, or $i = k$ and the transition is $Q_iR \xrightarrow{s} R$, in which case we end up with the pair (RTR', R) which is contained in $\mathcal{R}_3$.

2. Consider a pair (PR, Q) from $\mathcal{R}_2$. If P takes the initiative the possible transitions are as follows:

   - $PR \xrightarrow{\tau} PXR$, where X is either A or B. Then Q will respond with the empty move $Q \stackrel{\epsilon}{\Longrightarrow} Q$ and from the definition, the pair (PXR, Q) belongs to $\mathcal{R}_2$.
   - $PR \xrightarrow{\tau} QTR$, to which move Q will again respond with $Q \stackrel{\epsilon}{\Longrightarrow} Q$, and now the resulting pair (QTR, Q) belongs to $\mathcal{R}_1$.
   - $PR \xrightarrow{s} R$, then Q will respond with a sequence of τ transitions generating precisely the process Q_kR and then removing the final variable Q_k with $\xrightarrow{s}$. The possibility of such a move follows from the assumption that the automaton generates the total language. The result is a pair (R, R) which is contained in $\mathcal{R}_3$.

   That sorts out all moves available to P. When the process Q decides to perform any move $Q \xrightarrow{\mu} Q'$ the other process PR makes use of the transition $PR \xrightarrow{\tau} QTR \xrightarrow{\mu} Q'TR$ and from the definition, the pair $(Q'TR, Q')$ is in $\mathcal{R}_1$.

3. Any pair from $\mathcal{R}_3$ is of the form (RTR', R) where R and R' are sequences over process variables A and B. Since A can only do $\xrightarrow{a}$ and disappear, and B can only do $\xrightarrow{b}$ and disappear, it is clear that any move $R \xrightarrow{\mu} \hat{R}$ results again in a process $\hat{R} \in \{A, B\}^*$, and so the pair $(\hat{R}TR', \hat{R})$ is contained in $\mathcal{R}_3$.

We have exhausted all possible moves of any P', Q' such that $(P', Q') \in \mathcal{R}$ and thus showed that $\mathcal{R}$ is closed under expansion. Since the pair (P, Q) belongs to $\mathcal{R}_2$ and we have demonstrated that $\mathcal{R}$ is a weak bisimulation relation we can conclude that $P \approx Q$. $\square$

Finally we need to verify the size of the reduction. We assume that we start from an automaton with $k + 1$ states. We need to measure the size of the transition

function δ and we will choose the size to be $N = \sum_{i=0}^{k}(|\delta(q_i, a)| + |\delta(q_i, b)|)$, that is the total number of states reachable from each single state by accepting a or b. For each $q_j \in \delta(q_i, x)$ we define a transition rule, moreover there are several more rules that define the processes P, T, A and B. Eventually we end up with $N+8$ transition rules of size linear in the size of encoding of the states $q_0, \ldots, q_k$. To write down the rules of the algebra we need a work-tape that will contain the position of the currently defined process variable which requires roughly $\log k$ space. Hence it is quite straightforward to see that we can make do with a log-space work-tape and the resulting reduction is linear in size of the input. And so we come to the conclusion expressed in the theorem below:

Theorem 4.13 *The decidability problem of weak bisimilarity of BPA-processes is PSPACE-hard.*

4.3.1 EXPSPACE-complete problem versus $\approx$ of BPA

In the light of the current state of knowledge, with the decidability question for BPA-processes completely open, it seems natural to try to reduce even harder problems to weak bisimilarity. A natural choice might be to consider the problem TOT^2 which is a generalisation of TOT stated in terms of *regular expressions with squaring*. Just to remind ourselves, the set of *regular expressions (r.e.)* over some alphabet Σ is constructed recursively from the basic regular expressions which are $\emptyset$, and a for every $a \in \Sigma$. If r and s are r.e. then $r + s$, $r \cdot s$ and r^* are also r.e. The languages they determine are $L(\emptyset) = \emptyset$, $L(a) = \{a\}$ for every $a \in \Sigma$, $L(r + s) = L(r) \cup L(s)$, $L(r \cdot s) = L(r) \cdot L(s)$, and finally, $L(r^*) = (L(r))^*$. The *regular expressions with squaring (r.e.w.s.)* have a special symbol r^2 which stands for $r \cdot r$ and allows a succinct notation for exponentiation. For a more detailed account the reader should consult [33], [54].

Definition 4.14 TOT^2 *is the following problem:*

> *Instance: A regular expression r with squaring over some alphabet Σ.*
> *Question: Is $L(r)$, the language generated by the regular expression r, equal to the language Σ^*?*

This problem is EXPSPACE-complete, even for a two-letter alphabet (cf. [54], [33]). We will not go into detail here but roughly speaking, any computation of a Turing machine that works in space exponential in the size of input can be encoded by a regular expression with squaring of length polynomial in the size of input, thanks to the squaring operator. We have seen in previous reductions that

(BPA-)processes also permit a succinct way of writing down large (exponential) strings and so we hoped for a polynomial-time reduction from regular expressions with squaring to BPA. Unfortunately, it does not appear to work and we will briefly sketch why it is so.

For a given r.e.w.s. r over some alphabet Σ we want to compute, in polynomial time, processes P and Q such that $P \approx Q$ if and only if $L(r) = \Sigma^*$. Following the spirit of the reduction from TOT to $\approx$, testing weak bisimilarity will be simplified to string comparison. Then the goal is to construct the two processes so that they generate some strings with a $\stackrel{\tau}{\Longrightarrow}$ sequence and when they have stopped generating they proceed with string comparison, which is done by non$-\tau$ actions. Thus visible actions appear only at the end and all branching is performed exclusively by τ moves. As in the case of the previous reduction, this simplification seems necessary as otherwise it may be impossible to ensure bisimilarity.

Assuming we have an r.e.w.s. $r' = (...(r\underbrace{^2)^2...)^2}_{n}$ we need to construct a sequence of processes of polynomial size that will in some suitable way represent r'. We may stipulate that a process P represents r if every w is from $L(r)$ iff $P \stackrel{w}{\Longrightarrow} \epsilon$. It seems that any reduction method, computable in polynomial time, produces for r' a slight variation on the following design: assuming that P_0 represents r, we define rules $P_1 \stackrel{\tau}{\longrightarrow} P_0P_0$, ..., $P_n \stackrel{\tau}{\longrightarrow} P_{n-1}P_{n-1}$. Then the process P_n will represent r'. However, the principle of generating complete strings solely with τ actions is marred. We can see that on an example: if $r = a + b$ then $r^2 = (a + b)(a + b)$. We can put $P_0 \stackrel{\tau}{\longrightarrow} A$, $P_0 \stackrel{\tau}{\longrightarrow} B$, where $A \stackrel{a}{\longrightarrow} \epsilon$, $B \stackrel{b}{\longrightarrow} \epsilon$ (that would be done analogously to the reduction from TOT), and $P_1 \stackrel{\tau}{\longrightarrow} P_0P_0$. But then a possible execution of actions of P_1 is $P_1 \stackrel{\tau}{\longrightarrow} P_0P_0 \stackrel{\tau}{\longrightarrow} AP_0 \stackrel{a}{\longrightarrow} P_0 \stackrel{\tau}{\longrightarrow} B$ which clearly shows that we cannot generate the string AB with $\stackrel{\tau}{\Longrightarrow}$. As we have noted before this would be a serious hindrance for bisimilarity.

In the reduction from TOT we employed finite automata which offered a straightforward transformation to BPA-processes. Regular expressions with squaring can be also transformed into nondeterministic finite automata, however, this algorithm is in general exponential and hence not useful for our purposes. As there does not appear to be another way of encoding regular expressions by BPA-processes that would make it possible to argue about bisimilarity, other than via finite automata, we conclude that there does not seem to exist a way of showing EXPSPACE-hardness of weak bisimilarity for BPA-processes that would be based on reduction from regular expressions with squaring.

4.4 Conclusions

We have shown NP-hardness for the weak bisimilarity decision problem of totally normed BPA-processes and BPP. Those are the restricted classes of processes for which weak bisimilarity is decidable [28]. The two decision procedures of [28] do not produce any immediate complexity estimate of the problem. In the case of BPP the explicit upper bound is the Ackerman function (as is for the strong bisimilarity decision procedure for general BPP) which is not even primitive recursive. However, Hirshfeld also showed that unique prime decomposition holds for totally normed BPP with respect to weak bisimilarity which might be used to construct a more efficient decision procedure.

Rather surprisingly, there is no unique prime decomposition for totally normed BPA-processes [35]. Also for general BPA-processes there are no known results concerning decidability of weak bisimilarity. It might be the case that deciding weak bisimilarity will be harder for BPA-processes than for BPP but up till now there exist only weak concrete results that support this conjecture.

If we study in detail the structure of the processes that were constructed by the presented reductions we come to the conclusion that the full power of alternation between two processes, and hence also branching, is not used. In all the reductions we use τ transitions to generate an appropriate choice or response before any visible action is performed. Once we have performed some visible transition we are only allowed to continue with a single sequence which has been already determined. Then testing (weak) bisimilarity boils down to comparing two strings because no branching is available.

To be more concrete, in the KNAPSACK reduction we use the ability of processes to *count*. Each of the two processes generates a string, one of which is of a prescribed length. Then by comparing the strings we only compare the lengths and bisimilarity amounts to both strings being of the same length.

The TOT reduction is more subtle but basically works on the same principle. Here we compare two languages over a two letter alphabet. That means we need to compare strings generated from two letters. The two strings are again determined by a sequence of τ transitions before any visible action is performed. Then testing bisimilarity boils down to string comparison. A similar approach for the harder problem TOT^2, however, does not seem to work.

There is a plethora of hard problems from the area of language theory, model checking, and game theory. It is worth investigating how weak bisimilarity compares with these other problems and it will be a subject of further research.

Chapter 5

Connecting BPP and polynomial rings

Decidability of strong bisimilarity for Basic Parallel Processes was first established by Christensen, Hirshfeld and Moller in [7] for the restricted subclasses of normed and live BPP. Hirshfeld, Jerrum and Moller later constructed a polynomial time algorithm for the subclass of normed BPP [29], [30]. Finally, decidability was demonstrated for the whole class of BPP by Christensen, Hirshfeld and Moller in [8]. There were other approaches towards decidability of BPP, for instance by Hirshfeld [27] and Jančar [40], [41].

Despite a number of various decision procedures for bisimilarity on Basic Parallel Processes, so far there does not exist any algorithm whose computational complexity would be satisfactory. Some algorithms consist of two semidecision procedures with no available upper bounds. Hirshfeld in [27] constructs a *bisimulation tree*, a tool for deciding strong bisimilarity which is always finite. The width of the tree can be easily calculated from the size of the input processes, and is fairly small (exponential). The length of branches of the tree constitutes the real problem. Hirshfeld constructs the branches so that they form *proper sequences* (cf. **Chapter 2**) and then applies a lemma originally stated and proved by Dickson in [13] to conclude that every proper sequence of BPP must be finite. There is an effective upper bound on the length of a maximal proper sequence of BPP found by McAloon [47]. This bound is expressed as a function of the number of atoms in the algebra, the size of the first process in the sequence, and of the bound on the growth of the size in each step. Unfortunately, this upper bound is primitive recursive in the Ackerman function and hence not very useful.

In this chapter we make an attempt at improving the current situation. We provide a new technique for deciding bisimilarity by exploiting a connection between Basic Parallel Processes and polynomials. We construct a decision proce-

dure which is based on membership test for polynomial ideals. The technique of *Gröbner bases* is a powerful technique from the area of computational algebra that enables us to solve many problems involving polynomials, among others polynomial ideal membership. The resulting decision procedure is constructed in the spirit of Hirshfeld's bisimulation trees using an analogue of Caucal base (cf. **Chapter 2**) expressed in terms of polynomials.

5.1 Polynomial algebra

We will give a brief overview of some basic algebraic notions; for a more detailed account consult for instance Jacobson [39], Lang [44]. A *semigroup* $(S, \circ)$ is a set S together with an associative binary operation $\circ$ on S. A *monoid* $(S, \circ, e)$ is a semigroup with an *identity* element e, that is $e \circ x = x \circ e = x$ for every $x \in S$. A *group* $(G, \circ, \diamond, e)$ is a monoid $(G, \circ, e)$ together with a unary *inverse* operation $\diamond$, that is $x \circ \diamond x = \diamond x \circ x = e$. G is *commutative* if the operation $\circ$ is commutative.

A *ring* $(R, +, -, \cdot, 0)$ is a commutative group $(R, +, -, 0)$ and a semigroup $(R, \cdot)$, satisfying the laws of distributivity $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(x + y) \cdot z = x \cdot z + y \cdot z$. A *commutative* ring is one in which the operation $\cdot$ is commutative. A *ring with identity* is a ring R together with an element $1 \neq 0$ such that $(R, \cdot, 1)$ is a monoid. A *field* $(F, +, -, \cdot, ^{-1}, 0, 1)$ is a commutative ring with identity $(F, +, -, \cdot, 0, 1)$ and simultaneously a group $(F \setminus \{0\}, \cdot, ^{-1}, 1)$.

Let R be a commutative ring with identity. A nonempty subset I of R is an *ideal* if $a + b \in I$ for all $a, b \in I$ and $ac \in I$ for all $a \in I$ and $c \in R$. A set $B \subseteq R$ *generates* the ideal I or B is a *basis* for I if

$$I = \Big\{ \sum_{i=1}^{m} r_i b_i \mid m \in \mathbb{N}, r_i, \ldots, r_m \in R, b_1, \ldots, b_m \in B \Big\}$$

In this case we say that I is the *ideal generated by* B and we denote it by $I = \mathrm{Id}(B)$. I is *finitely generated* if it has a finite basis.

We are going to work with the two-element field $\mathbb{F}_2 = (\{0, 1\}, +, -, \cdot, ^{-1}, 0, 1)$ and the polynomial ring (with identity) $\mathbb{F}_2[x_1, \ldots, x_n]$. The two binary operations $+$ and $\cdot$ of the field $\mathbb{F}_2$ are given in Fig. 5.1 and the unary operation $-$ and $^{-1}$ are defined in the obvious way so that they satisfy the appropriate axioms: $-(0) = 0$, $-(1) = 1$, and finally $1^{-1} = 1$.

For a commutative ring with identity R, we denote the polynomial ring over R in indeterminates $x_1, \ldots, x_n$ with $R[x_1, \ldots, x_n]$. Special terms of the form

$$\begin{array}{ll} 0+0=0 & 0\cdot 0=0 \\ 0+1=1 & 0\cdot 1=0 \\ 1+0=1 & 1\cdot 0=0 \\ 1+1=0 & 1\cdot 1=1 \end{array}$$

Figure 5.1: The summation and multiplication operations on $\mathbb{F}_2$

$x_1^{i_1}\dots x_n^{i_n}$, where $i_1,\dots,i_n \in \mathbb{N}$, are called *power products*. The empty power product $x_1^0\dots x_n^0$ is denoted with 1, and the set of power products over $R[x_1,\dots,x_n]$ will be denoted with $\mathcal{P}$. Terms of the form $a.x_1^{i_1}\dots x_n^{i_n}$, where a is a coefficient from the ring R and $x_1^{i_1}\dots x_n^{i_n}$ is a power product, are called *monomials*. A polynomial is then identified with a function $p:\mathcal{P}\to R$ with a finite support, that is p assigns a non-zero coefficient only to finitely many terms from $\mathcal{P}$. It is convenient to express polynomials as sums of monomials where the order is not taken into consideration, that is the two expressions $x_1x_2^2+2x_1x_3$ and $2x_1x_3+x_1x_2^2$ represent the same polynomial. When we deal with polynomials in the method of Gröbner bases it will be convenient to define an ordering on power products. However, unless explicitly stated we will assume that two expressions represent the same polynomial if they are identical up to reordering of terms.

We consider polynomials over the field $\mathbb{F}_2$ which means that the only coefficients are 0 or 1. Hence for the polynomial ring $\mathbb{F}_2[x_1,\dots,x_n]$ monomials and power products coincide. The terms $x_1x_2^2$, x_1^5, and 1 are examples of power products from the polynomial ring $\mathbb{F}_2[x_1,x_2]$. We will make a convention that capital letters P,Q,R,S range over power products, small letters p,q,r range over polynomials.

We will see later how useful ideals of polynomials will be to us. There are special cases when all polynomial ideals can be finitely generated. This is expressed in the following theorem by Hilbert [39], [61]:

Hilbert's Basis Theorem:

If R is a field or the ring of integers, then any ideal in the polynomial ring $R[x_1,\dots,x_n]$ has a finite set of generators.

5.2 Ideal membership and Gröbner bases

Now we can explain the method of *Gröbner bases* which is widely used in computer algebra to deal with problems concerning polynomials, such as polynomial ideal

membership, equivalence of polynomials with respect to a polynomial ideal, and others. We will only touch briefly on this method; for a thorough treatment consult computer algebra textbooks ([12], [18], [62]).

We are going to consider the problem of ideal membership, that is, given an ideal I (most likely in the form of a basis for I) and given a polynomial p, we want to determine whether $p \in I$. To decide this problem may not be straightforward even if we have a finite basis for I since $p \in I$ if and only if p is a linear combination of some polynomials from the basis of I and there might be an infinite number of combinations we might need to check. Therefore we are looking for a special kind of basis that will provide us with a straightforward test for membership. We will see that the test will consist of a series of reductions starting from the original polynomial via smaller polynomials that will terminate with a trivial case. Before we introduce the definition of reduction we need to define an appropriate ordering.

Let us consider polynomials in the variables $x_1, \ldots, x_n$, the coefficients of which belong to a field F. An *admissible ordering* $<$ over power products of $F[x_1, \ldots, x_n]$ satisfies the following two conditions:

1. for every power product P, $1 \leq P$
2. if $P < Q$ then for every power product R, $P \cdot R < Q \cdot R$

The lexicographic order on the vector of exponents $(i_1, \ldots, i_n)$ is an example of an admissible ordering and we assume a fixed admissible ordering $<$ for the rest of the section. Let us suppose that every polynomial is written in decreasing order (according to $<$) of its power products as $\sum_{i=1}^{m} a_i P_i$, where $a_i \neq 0$ and $P_i > P_{i+1}$ for every i. The leftmost power product P_1 is then called the *leading power product* and the term $a_1 P_1$ is called the *leading monomial*.

Let G be a finite set of polynomials, p a polynomial, then we say that p is *reduced with respect to* G if no leading power product of an element of G divides the leading power product of p. If p is not reduced with respect to G then we can subtract from it a multiple of an element of G to obtain a new polynomial p'. This polynomial is smaller than p in the sense that the leading power product of p' is smaller than the leading power product of p, and also $p' \in \mathrm{Id}(G)$ if and only if $p \in \mathrm{Id}(G)$. This process is called a *reduction* of p with respect to G. A fundamental property of reduction is that a polynomial p cannot have an infinite chain of reductions with respect to a fixed set G.

Now we can define the concept of a *Gröbner basis* for an ideal. A basis G of an ideal I is called a *Gröbner basis* (with respect to $<$) if every reduction of a

polynomial p from I to a reduced polynomial (with respect to G) always leads to 0. Since we already know that no polynomial determines an infinite chain of reductions we can determine the ideal membership in this way: given an ideal I and a polynomial p, compute a Gröbner basis G for I and reduce p with respect to G. If the outcome is 0 then $p \in I$ otherwise $p \notin I$. The following theorem proves the universality of our approach [12].

Theorem 5.1 *Every ideal has a Gröbner basis with respect to any admissible order.*

There is an algorithm originally devised by Buchberger in his Ph.D. thesis [3] which transforms a finite basis B of an ideal I into a Gröbner basis G for I. That justifies the algorithm for ideal membership test we outlined above. Hence we can assume in the following text that we can decide polynomial ideal membership. The computational complexity of polynomial membership test is at least exponential space and the best upper bound on the complexity appears to be "double exponential or more" [46]. However, as we shall see later, we will only consider bases that will consist of two-term polynomials. They generate *binomial ideals* and in [43] was presented an optimal exponential space algorithm for constructing Gröbner bases of such ideals.

Example 5.2 Consider the ideal I generated by the polynomials

$$p_1 = xy^2z - xyz \qquad p_2 = x^2y^2 - z$$

Then the polynomial $p_3 = zp_2 - xp_1 = x^2yz - z^2$ is a linear combination of p_1 and p_2 and hence belongs to I. However, p_3 is reduced with respect to $\{p_1, p_2\}$ which means that $\{p_1, p_2\}$ is not a Gröbner basis of I. Next we consider the set $\{p_1, p_2, p_3\}$, however there are still polynomials $p_4 = xp_1 - (y-1)p_3 = yz^2 - z^2$ and $p_5 = zp_3 - x^2p_4 = x^2z^2 - z^3$ which are in the ideal I but do not reduce to 0. Finally, the set $\{p_1, p_2, p_3, p_4, p_5\}$ is a Gröbner basis (for more detail consult [12]). □

5.3 Bisimulation and polynomial ideals

We are going to relate Basic Parallel Processes with power products and express the bisimulation condition as a condition defined over polynomial ideals and their bases. We presuppose a fixed set of process variables $\Sigma = \{X_1, \ldots, X_n\}$ and a

set of rules Δ. We consider the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_n]$ where the indeterminates $x_1, \ldots, x_n$ correspond to process variables from Σ. Then we can identify a BPP $X_1^{e_1} \| X_2^{e_2} \| \ldots \| X_n^{e_n}$ from $\Sigma^{\otimes}$ with a power product $x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$ from $\mathbb{F}_2[x_1, \ldots, x_n]$, and parallel composition of BPP corresponds to multiplication of power products. Hence we see that power products and BPP, and indeterminates and process variables are closely related and therefore they will be freely interchanged in the following text. The process algebra $(\Sigma^{\otimes}, \Delta)$ and the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_n]$ remain fixed for the rest of the chapter and whenever we speak about a bisimulation relation, resp. an ideal I, we mean a bisimulation relation with respect to $(\Sigma^{\otimes}, \Delta)$, resp. an ideal $I \subseteq \mathbb{F}_2[x_1, \ldots, x_n]$.

We can express a bisimulation relation $\mathcal{R}$ in polynomial terms as the set $B_{\mathcal{R}} = \{P + Q \mid (P, Q) \in \mathcal{R}\}$, and the largest bisimulation relation $\sim$ in particular gives rise to the set $B_{\sim} = \{P + Q \mid P \sim Q\}$. Clearly, these sets are possibly infinite and hence it may not be feasible to test membership in them. However we may consider the polynomial ideals generated by these sets and then make use of the method of Gröbner bases which allows us to test membership in polynomial ideals.

Example 5.3 We will consider a Basic Parallel Process Algebra $\Sigma = \{X, Y\}$ together with the transition rules $\Delta = \{X \xrightarrow{a} \epsilon, Y \xrightarrow{a} X\}$. It is straightforward that for instance the pairs of processes X^2 and Y, X^3 and $X \| Y$, and X^4 and Y^2 are strongly bisimilar and then from that observation we can easily arrive at the conclusion that all bisimilar processes are pairs of the form $X^i \| Y^j$ and $X^m \| Y^n$ where $i + 2j = m + 2n$. Therefore the maximal bisimulation relation gives rise to the set $B_{\sim} = \{x^i y^j + x^m y^n \mid i + 2j = m + 2n\}$. Then we consider the ideal $I_{\sim}$ generated by $B_{\sim}$. **Hilbert's Basis Theorem** ensures that this ideal has a finite basis. Indeed, it is not difficult to verify that all polynomials from $I_{\sim}$ (and in particular from $B_{\sim}$) can be obtained as a multiple of the polynomial $x^2 + y$. □

To be able to test whether a set of polynomials represents a bisimulation relation we are going to devise a condition which was inspired by Caucal's condition for bisimulation equivalence. We will denote this Caucal-like condition by CC.

Definition 5.4 *Let B be an arbitrary (finite) set of polynomials. We say that $CC(B)$ if for all $P + Q \in B$ and for every $\mu \in Act$*

- *for all $P \xrightarrow{\mu} P'$ there exists $Q \xrightarrow{\mu} Q'$ with $P' + Q' \in Id(B)$ and*

- *for all* $Q \xrightarrow{\mu} Q'$ *there exists* $P \xrightarrow{\mu} P'$ *with* $P' + Q' \in Id(B)$,

where $Id(B)$ *is the ideal generated by the (finite) set* B.

Note that this condition only applies to two-term polynomials from B. The reason for that will become clear later. Now we will study the ideal $I_\sim$ corresponding to the maximum bisimulation $\sim$ and the properties of its (finite) bases. Firstly, however, in order to simplify the proofs that will follow, we will show that an ideal generated by a set A can be also defined inductively, as follows:

Lemma 5.5 *For every set* $A \subseteq \mathbb{F}_2[x_1, \ldots, x_n]$ *the polynomial ideal* $Id(A) = I(A)$, *where* $I(A)$ *is defined in the following way:*

$$\begin{array}{lcl} I_0 & = & A \\ I_{i+1} & = & I_i \cup \{p + q \mid p, q \in I_i\} \\ & & \cup \{P \cdot p \mid p \in I_i, P \text{ is a power product in } \mathbb{F}_2[x_1, \ldots, x_n]\} \\ I(A) & = & \bigcup_{i \in \omega} I_i \end{array}$$

Proof: We assume a fixed set A and the corresponding sets $\mathrm{Id}(A)$ and $I(A)$. We need to verify that $\mathrm{Id}(A)$ is included in $I(A)$ and also that $I(A)$ is a subset of $\mathrm{Id}(A)$. Firstly we will check that $I(A) \subseteq \mathrm{Id}(A)$, and we will do that by an inductive argument that follows the definition of $I(A)$. The set I_0 which is taken to be A is clearly a subset of $\mathrm{Id}(A)$. Assuming that I_i is contained in $\mathrm{Id}(A)$, we consider the elements of the set I_{i+1}. They either already belong to I_i, hence also to $\mathrm{Id}(A)$, or they are the sum of polynomials $p + q$ with both p and q in I_i, and hence also in $\mathrm{Id}(A)$, or they are of the form Pp where P is a power product and p belongs to $\mathrm{Id}(A)$. Since $\mathrm{Id}(A)$ is an ideal, it is closed under summation and multiplication with arbitrary polynomials therefore the fact that p and q are elements of $\mathrm{Id}(A)$ implies that the terms $p+q$ and Pp also belong to $\mathrm{Id}(A)$. Thus we have shown that every set I_i is a subset of $\mathrm{Id}(A)$ and we can come to the conclusion that $I(A) = \cup_{i \in \omega} I_i \subseteq \mathrm{Id}(A)$.

The other inclusion is proved in this way. We take a polynomial p from the ideal $\mathrm{Id}(A)$ which can be expressed as a finite sum $\lambda_1 p_1 + \ldots + \lambda_k p_k$, where λ_i are arbitrary polynomials from $\mathbb{F}_2[x_1, \ldots, x_n]$ and p_i are from the base set A. Since $I_0 = A$, clearly the polynomials p_i belong to I_0. Next, assume that the polynomials $\lambda_i = P_{i1} + P_{i2} + \ldots + P_{in_i}$, where P_{ij} are power products. Then the summand $\lambda_i p_i$ can be expressed as $P_{i1}p_i + P_{i2}p_i + \ldots + P_{in_i}p_i$ and from the definition of individual sets I_i, all $P_{ij}p_i$ belong to the set I_1 and the sum of multiples of p_i with the power products will certainly belong to the set I_{n_i}. Therefore the terms $\lambda_i p_i$ are contained in the sets I_{n_i} for the respective n_i. Since the set I_{i+1} contains

all binary sums of polynomials from I_i, the whole sum $\sum \lambda_i p_i$ must belong to I_N, where $N = n_1 + n_2 + \ldots + n_k$ (in fact that is an overestimate). Since $I_N \subseteq I(A)$, we have verified that $\mathrm{Id}(A) \subseteq I(A)$. $\square$

Now we can show that the polynomials from the ideal $I_\sim$ have a very special shape: they consist of an even number of power products and moreover, they can be paired up into bisimilar pairs of power products.

Proposition 5.6 *Let p be a polynomial. Then $p \in I_\sim$ if and only if there exist power products $P_1, \ldots, P_k, Q_1, \ldots, Q_k$ such that p can be expressed as $P_1 + Q_1 + \ldots + P_k + Q_k$ and $P_j \sim Q_j$ for all $1 \leq j \leq k$.*

Proof: We are going to use the inductive construction of $I_\sim$ as in the above **Lemma 5.5** with $I_0 = B_\sim$. One implication is easy. If a polynomial p can be expressed as $P_1 + Q_1 + \ldots + P_k + Q_k$ and $P_j \sim Q_j$ for all j then $P_j + Q_j$ belong to $B_\sim = I_0$ and hence p will certainly be in I_k.

The other direction will be proved by induction on I_i.

1. $p \in I_0$ iff $p \in B_\sim$ iff $p = P + Q$ and $P \sim Q$.

2. $p \in I_{i+1}$ if and only if $p \in I_i$ or $p = q + r$ with $q, r \in I_i$ or $p = Rq$ where $q \in I_i$ and R is a power product. The case of $p \in I_i$ is trivial so the proof falls into two parts:

   (a) Assume $p = q + r$, where $q = P_1 + Q_1 + \ldots + P_k + Q_k$ with $P_j \sim Q_j$ and $r = R_1 + S_1 + \ldots + R_l + S_l$ with $R_j \sim S_j$. The polynomial p can be expressed as $U_1 + V_1 + \ldots + U_m + V_m$, where each sum $U_i + V_i$ is either some $P_j + Q_j$ or $R_h + S_h$, in which case by induction hypothesis $U_i \sim V_i$, or it arises from some $P_j + Q_j$ and $R_h + S_h$ in the way that either $P_j = R_h$ or $P_j = S_h$ or $Q_j = R_h$ or $Q_j = S_h$.

   Assume that $P_j = R_h$ and thus $U_i + V_i = Q_j + S_h$. As $P_j \sim Q_j$ and $R_h \sim S_h$ then by the transitivity of $\sim$ also $Q_j \sim S_h$. All the other cases are symmetrical.

   (b) Assume $p = Rq$, where $q = P_1 + Q_1 + \ldots + P_k + Q_k$, $P_j \sim Q_j$. Then $p = RP_1 + RQ_1 + \ldots + RP_k + RQ_k$ and because strong bisimulation is closed under parallel composition also $RP_j \sim RQ_j$ for all j. That concludes the proof. $\square$

A simple consequence of the preceding proposition is that we only need to consider bases for $I_\sim$ which consist exclusively of two-term polynomials. If B is a basis of $I_\sim$ that contains a polynomial p with more than two terms (has to be of an even length) then, as p also belongs to $I_\sim$, it has to be of the form $P_1+Q_1+\ldots+P_k+Q_k$ with $P_j \sim Q_j$. It follows from the definitions of $B_\sim$ and $I_\sim$ that all sums $P_j + Q_j$ are in $I_\sim$. Hence the set B' which arises from B by replacing the polynomial p with the two-element sums $P_j + Q_j$ is another finite basis for $I_\sim$. However, this may not hold for other ideals which are not generated by the set $B_\sim$.

If P and Q are bisimilar then $P + Q$ certainly belongs to $B_\sim$. On the other hand, there might be a smaller bisimulation relation which contains (P, Q). We do not have any means of checking whether a set B generates the largest relation but we can find out when the subset of two-term polynomials of an ideal I corresponds to a bisimulation. That is captured in the following condition which will be denoted by $CC'(I)$.

Definition 5.7 *We say that an ideal I satisfies the CC' condition if every p from I can be expressed as $P_1 + Q_1 + \ldots + P_k + Q_k$ such that $P_j + Q_j \in I$ for all $1 \leq j \leq k$, and for all $\mu \in Act$*

- *for all $P_j \xrightarrow{\mu} P'_j$ there is $Q_j \xrightarrow{\mu} Q'_j$ such that $P'_j + Q'_j \in I$ and*
- *for all $Q_j \xrightarrow{\mu} Q'_j$ there is $P_j \xrightarrow{\mu} P'_j$ such that $P'_j + Q'_j \in I$.*

In the definition above we take all actions μ from the possibly infinite set of actions *Act* but in fact it suffices to consider only those actions that are used in the defining rules Δ of the algebra.

It is easy to show that if $CC'(I)$ for an ideal I then the set $\mathcal{R} = \{(P, Q) \mid P+Q \in I\}$ is a strong bisimulation: we consider a pair (P, Q) such that $P+Q$ is in the ideal I. For any move $P \xrightarrow{\mu} P'$ there is a response $Q \xrightarrow{\mu} Q'$ with $P'+Q'$ in the ideal I, therefore also the pair (P', Q') is in the set $\mathcal{R} = \{(P, Q) \mid P+Q \in I\}$. The same is true about all moves of Q hence the set $\mathcal{R}$ is closed under expansion and forms a bisimulation relation.

Lemma 5.8 *For an ideal I, if $CC'(I)$ then the set $\mathcal{R} = \{(P, Q) \mid P + Q \in I\}$ is a strong bisimulation relation.*

This condition however does not have to be finite since it applies to all elements of a given ideal, as opposed to pairs $P + Q$ in the case of CC. Therefore we are looking for a connection between the CC' condition for ideals and the (possibly)

finite condition CC for bases. If we have an ideal I with a basis B and CC' holds for the ideal I then for any sum $P+Q$ from B and every move $P \xrightarrow{\mu} P'$, there must be a response $Q \xrightarrow{\mu} Q'$ with $P'+Q' \in I = \mathrm{Id}(B)$, and the same for transitions of Q, so the basis B satisfies CC.

Lemma 5.9 *Assume that I is an ideal with a basis B. Then $CC'(I)$ implies $CC(B)$.*

More importantly, the other direction is true as well, that is if CC holds for a (finite) basis B then the ideal $\mathrm{Id}(B)$ satisfies CC' (and hence gives rise to a strong bisimulation relation). That will enable us to replace a potentially infinite test with a finite one. We will prove that in the following proposition:

Proposition 5.10 $CC(B) \Longrightarrow CC'(Id(B))$.

Proof: Again we will make use of **Lemma 5.5** and assume that the ideal $\mathrm{Id}(B)$ can be expressed inductively with $I_0 = B$. Then it suffices to prove that for every i, every polynomial p from I_i can be expressed as $P_1+Q_1+\ldots+P_k+Q_k$ such that all P_j+Q_j belong to $\mathrm{Id}(B)$ and have an expansion in $\mathrm{Id}(B)$. That will be done by induction on i.

1. For $i=0$, $I_0 = B$ and $CC(B)$ implies that all pairs $P+Q$ from B have an expansion in B, hence also in $\mathrm{Id}(B)$.

2. $p \in I_{i+1}$ if and only if $p \in I_i$ or $p = q+r$ with $q, r \in I_i$ or $p = Rq$ with $q \in I_i$ and R a power product. The case of $p \in I_i$ is trivial and we will consider the two following cases:

   (a) Assume that $p = q+r$, where $q = P_1+Q_1+\ldots+P_k+Q_k$, $r = R_1+S_1+\ldots+R_l+S_l$ and q and r satisfy the claim. $p = U_1+V_1+\ldots+U_m+V_m$, where each U_i+V_i is either some P_j+Q_j or R_h+S_h, in which case we can apply the induction hypothesis, or U_i+V_i arises from some P_j+Q_j and R_h+S_h so that either $P_j = R_h$ or $P_j = S_h$ or $Q_j = R_h$ or $Q_j = S_h$.

   Assume that $P_j = R_h$ and $U_i+V_i = Q_j+S_h$. If $U_i = Q_j \xrightarrow{\mu} U$ then by induction hypothesis $P_j \xrightarrow{\mu} P$ so that $U+P \in \mathrm{Id}(B)$. As $P_j = R_h$ there has to be a move $S_h = V_i \xrightarrow{\mu} S$ such that $P+S \in \mathrm{Id}(B)$. As the ideal $\mathrm{Id}(B)$ is closed under addition then both $U_i+V_i = Q_j+S_h$ and $U+S \in \mathrm{Id}(B)$. Moves of V_i and all the other cases are symmetrical.

(b) Assume that $p = Rq$, $q \in I_i$.

We may assume that $q = P_1 + Q_1 + \ldots + P_k + Q_k$ so that all $P_j + Q_j \in \mathrm{Id}(B)$. Let $p = P'_1 + Q'_1 + \ldots + P'_k + Q'_k = RP_1 + RQ_1 + \ldots + RP_k + RQ_k$. Since all $P_j + Q_j \in \mathrm{Id}(B)$ then also all $RP_j + RQ_j \in \mathrm{Id}(B)$ and hence it remains to check the moves:

$P'_j \xrightarrow{\mu} P'$: $P'_j = RP_j$ so either R makes a move $R \xrightarrow{\mu} R'$ and $P' = R'P_j$ or P_j makes a move $P_j \xrightarrow{\mu} P$ and $P' = RP$. In the former case we use the fact that $P_j + Q_j \in \mathrm{Id}(B_0)$ hence also $R'P_j + R'Q_j \in \mathrm{Id}(B)$.

In the latter case by induction hypothesis there is a corresponding move $Q_j \xrightarrow{\mu} Q$ so that $P + Q \in \mathrm{Id}(B)$ and since $\mathrm{Id}(B)$ is closed under multiplication also $RP + RQ \in \mathrm{Id}(B)$. Similarly for moves of Q'_j. $\square$

5.4 Semidecision procedure

Before we describe the decision procedure we will show how we can use the CC condition for ideal bases to demonstrate semidecidability of strong bisimilarity for BPP. This in fact suffices to demonstrate decidability as BPP are image-finite processes and for the class of image-finite processes we can easily construct a semidecision procedure for non-bisimilarity (cf. **Chapter 2**).

The procedure is based on a simple principle of enumerating all finite bases that contain the input pair and testing the CC condition for them. If we at some point come across a set B which satisfies CC then, as a consequence of the theorems presented above, B must be a basis of a bisimulation that relates the input pair. Hence we can stop the procedure with confirmation of bisimilarity.

Here follows an informal sketch of the semidecision procedure which halts with a positive answer if and only if the two input processes P and Q are strongly bisimilar. We choose any effective ordering of finite sets of pairs that contain $P + Q$ starting with the singleton set $\{P + Q\}$.

Proposition 5.11 *The algorithm presented in Fig. 5.2 halts if and only if the input pair P and Q are strongly bisimilar.*

Proof: We need to verify that the semidecision procedure is correct. Assume we are given a pair of processes P, Q such that $P \sim Q$. Then $P + Q$ is contained in the set $B_\sim$. By **Hilbert's Basis Theorem**, the ideal $I_\sim$ is generated by some

1. Put $B = \{P + Q\}$.
2. Check whether $CC(B)$.
3. If the condition holds then output $P \sim Q$ else enumerate another finite set B containing the sum $P + Q$ and go to 2.

Figure 5.2: Semidecision procedure for $\sim$ of BPP

finite set B. It is easy to see that $CC'(I_\sim)$ and hence also $CC(B)$ for any finite basis B of $I_\sim$. The Caucal-like condition can be verified since checking it for B means checking ideal membership finitely many times and ideal membership is decidable using Gröbner bases. We can effectively generate all finite sets so eventually we will generate this B, verify that $CC(B)$ and output $P \sim Q$.

On the other hand, assume we are given a pair of processes P, Q such that $P \nsim Q$. That means there is no bisimulation relation containing the pair (P, Q). We generate B so that it contains the sum $P + Q$. Therefore the Caucal-like condition will always fail for B because otherwise by **Proposition 5.10** $\mathrm{Id}(B)$ would correspond to a bisimulation relating P and Q. Hence the procedure will never give a positive answer in the case of non-bisimilarity. $\square$

5.5 Decision procedure

The principle of the decision procedure is to start from the set consisting of the input pair $P+Q$ and gradually construct a finite basis of a bisimulation (if P and Q are bisimilar) by adding new pairs which are appropriate derivatives of P and Q. This approach is closely related to Hirshfeld's bisimulation trees (see **Chapter 2**). The finiteness of this approach is guaranteed by the finite branching of BPP and the fact that every increasing chain of ideals has a finite length (**Theorem 5.13**).

Now we will explain the basic idea of the decision procedure together with some new notation. Starting from the basis $B_0 = \{P + Q\}$, we assume that we have constructed a basis B such that $\neg CC(B)$. Then there exist $P + Q \in B$, μ and $P \xrightarrow{\mu} P'$ such that for all $Q \xrightarrow{\mu} Q'$, the sum $P' + Q'$ is not in the ideal $\mathrm{Id}(B)$ (or, symmetrically, there exists $Q \xrightarrow{\mu} Q'$ such that $P' + Q' \notin \mathrm{Id}(B)$ for all $P \xrightarrow{\mu} P'$) and we will say that *CC fails on* $P + Q$. We define the set of failed pairs as

- $\mathcal{F}(B) = \{P + Q \in B \mid CC \textit{ fails on } P + Q\}$

and for $P+Q \in \mathcal{F}(B)$ and an action μ we define

- $\mathcal{P}(P+Q,\mu) = \{P' \mid P \xrightarrow{\mu} P' \wedge \forall Q \xrightarrow{\mu} Q'.\ P'+Q' \notin \mathrm{Id}(B)\}$, and
- $\mathcal{Q}(P+Q,\mu) = \{Q' \mid Q \xrightarrow{\mu} Q' \wedge \forall P \xrightarrow{\mu} P'.\ P'+Q' \notin \mathrm{Id}(B)\}$,

where $\mathcal{P}(P+Q,\mu)$ is the set of μ-derivatives of P for which there is no μ-derivative of Q such that the sum of the two derivatives belongs to $\mathrm{Id}(B)$. The latter set $\mathcal{Q}(P+Q,\mu)$ is a symmetric analogue for Q.

If we want to maintain bisimilarity we need to modify the failed basis B. For each $P' \in \mathcal{P}(P+Q,\mu)$ we will choose some $Q \xrightarrow{\mu} Q'$ and add $P'+Q'$ to B. We will do that symmetrically for every $Q' \in \mathcal{Q}(P+Q,\mu)$ as well. This is explained in Step 4 of Fig. 5.3 in more detail. We will do that for all pairs $P+Q \in \mathcal{F}(B)$. Therefore each new basis B' obtained as a result of this one-step nondeterministic modification of B will satisfy this condition: for each $P+Q \in \mathcal{F}(B)$, for each action μ and each $P' \in \mathcal{P}(P+Q,\mu)$, there will be a derivation $Q \xrightarrow{\mu} Q'$ such that $P'+Q' \in \mathrm{Id}(B')$ (and symmetrically for all processes $Q' \in \mathcal{Q}(P+Q,\mu)$). We continue with the modified basis B'.

It can happen that for some $P \xrightarrow{\mu} P'$ there is no response $Q \xrightarrow{\mu} Q'$. In that case we clearly cannot maintain bisimilarity with the chosen basis and hence we will stop with B and consider it an unsuccessful leaf. On the other hand, if we have arrived at a basis B such that $CC(B)$ then we know that B is a basis of a bisimulation that relates the two input processes. Hence we can output confirmation of bisimilarity and stop. The outline of the decision procedure is given in Fig. 5.3.

It is important that we define the sets B_μ nondeterministically which ensures that each possible new sum $P'+Q'$ will be considered and appear in some new basis constructed in Step 4.

Example 5.12 We will demonstrate the algorithm on a simple BPP algebra. We assume the process variables X, Z, B and C together with the transition rules

$$X \xrightarrow{a} X\|B \qquad B \xrightarrow{b} X \qquad Z \xrightarrow{a} Z\|C \qquad C \xrightarrow{b} X$$

The variables X and Z are obviously strongly bisimilar and we will step by step follow the computation of the algorithm on the input X, Z. We will express process terms as power products and will be omitting the symbol $\|$. In the context of polynomial algebra we have the indeterminates X, Z, B, C given in this order. Hence a power product will be any term $X^i Z^j B^k C^l$ for natural numbers i, j, k, l.

1. Input a pair P, Q.

2. Put $B = \{P + Q\}$.

3. Check $CC(B)$.

   If $CC(B)$ then $CC'(\mathrm{Id}(B))$ and so $\mathrm{Id}(B)$ represents a bisimulation relating P and Q, hence stop with a positive reply.

   If $\neg CC(B)$ then go to 4.

4. If there is a pair $P + Q \in \mathcal{F}(B)$ such that $P \not\sim_1 Q$ then stop without a reply else define a new basis in this way:

   $B' := B$

   For every pair $P + Q$ from $\mathcal{F}(B)$ and every μ,

   - for every $P' \in \mathcal{P}(P + Q, \mu)$ choose $Q \xrightarrow{\mu} Q'$ and put $B' := B' \cup \{P' + Q'\}$
   - for every $Q' \in \mathcal{Q}(P + Q, \mu)$ choose $P \xrightarrow{\mu} P'$ and put $B' := B' \cup \{P' + Q'\}$.

   Put $B := B'$ and go to 3.

Figure 5.3: Decision procedure for $\sim$ of BPP

For example, the term XB is a power product and the term $XB + ZC$ is a polynomial. We will also use distributivity in the form $(P + Q)R = PR + QR$.

We initialise the procedure with the step $B_0 := \{X + Z\}$. This set will clearly fail CC because X can do $X \xrightarrow{a} XB$ and Z can perform $Z \xrightarrow{a} ZC$ but $XB + ZC$ cannot be generated from B_0. Hence we add the sum to B_0 and put $B_1 = \{X + Z, XB + ZC\}$, and check CC for this new basis. Now we only need to check the moves of the term $XB + ZC$. There are four of them in total:

1. $XB \xrightarrow{a} XB^2$
2. $XB \xrightarrow{b} X^2$
3. $ZC \xrightarrow{a} ZC^2$
4. $ZC \xrightarrow{b} XZ$

For both actions a and b, there is a single transition available from XB, resp. ZC, so the matching derivatives of XB and ZC can be paired up as $XB^2 + ZC^2$ and $X^2 + XZ$. We actually find out that now we are already finished since we can obtain $X^2 + XZ$ as $(X + Z)X$ and we can generate $XB^2 + ZC^2$ in this way: $(XB + ZC)B + (X + Z)BC + (XB + ZC)C = XB^2 + ZBC + XBC + ZBC + XBC + ZC^2 = XB^2 + ZC^2$. Hence the basis B_1 satisfies CC and we will conclude that the two processes are strongly bisimilar. $\square$

In order to prove the correctness of the decision procedure we will need the following theorem [61] which is a simple consequence of **Hilbert's Basis Theorem**:

Theorem 5.13 *Let $I_1, I_2, I_3, \ldots$ be a sequence of ideals such that $I_i \subseteq I_{i+1}$ for every i. Then there exists an n such that $I_n = I_{n+i}$ for every i. (Or, every strictly increasing sequence of ideals is finite.)*

We will sketch a proof of the theorem. Assume that we have an infinite increasing sequence of ideals $I_1 \subseteq I_2 \subseteq \ldots I_n \subseteq \ldots$. It is not difficult to verify that the union I of all ideals I_n is also an ideal which, by **Hilbert's Basis Theorem**, is also finitely generated. Therefore there exists a set $B = \{p_1, \ldots p_k\}$ such that $I = \mathrm{Id}(B)$. But each p_i belongs to I and hence to some I_{n_i}, and so the whole set B is a subset of I_N, where N is the maximum of n_i. Thus $I \subseteq I_N$ and from that follows that $I = I_N = I_{N+1} = I_{N+i}$ for every i. Finally we can establish the correctness of the algorithm presented above.

Theorem 5.14 *The decision procedure will always stop for any given input pair P, Q and the output will be affirmative if and only if $P \sim Q$.*

Proof: The proof consists of two parts. First we show that the procedure is finite.

The computation of the procedure can be described by a tree whose branching corresponds to the non-deterministic choices made by the algorithm. The nodes of the tree are either labelled by some finite set B or they are leaves. Leaves are either successful - if we manage to find a basis of a bisimulation, or unsuccessful - if we find out that $P \not\sim_1 Q$ for some pair $P + Q$. First we are going to show that all successor nodes of a node B can be constructed in finite time.

Suppose we are at a node B. B contains a finite number of pairs $P + Q$. For any $P+Q$ and $P \xrightarrow{\mu} P'$ we have to check if there is a $Q \xrightarrow{\mu} Q'$ so that $P'+Q' \in \mathrm{Id}(B)$, and vice versa. There are only finitely many possible derivatives P' of P and Q' of Q and the condition $P' + Q' \in \mathrm{Id}(B)$ can be decided using the method of Gröbner bases. Hence we can check $CC(B)$ in finite time.

If $\neg CC(B)$ then we proceed to Step 4 in which we construct all possible successors of B. Here for every failed pair $P + Q$ and every μ we proceed as follows: for every $P' \in \mathcal{P}(P + Q, \mu)$ we choose some $Q \xrightarrow{\mu} Q'$ and add $P' + Q'$ to B, and symmetrically for every $Q' \in \mathcal{Q}(P + Q, \mu)$. As P and Q are finitely branching there is only a finite number of successors of every node and hence the tree is finitely branching.

It remains to show that all branches are of finite lengths. There are three types of nodes in the tree, successful leaves, unsuccessful leaves and nodes labelled by finite bases. Obviously a branch containing a leaf is finite. If there was an infinite branch $B_0, B_1, B_2, \ldots$ in the tree then each B_{i+1} would be constructed from a node B_i by adding at least one new pair $P' + Q'$ to B_i that would not belong to $\mathrm{Id}(B_i)$. Trivially $P' + Q' \in \mathrm{Id}(B_{i+1})$ and so this branch would generate an increasing chain of ideals $\mathrm{Id}(B_0) \subset \mathrm{Id}(B_1) \subset \mathrm{Id}(B_2) \subset \ldots$. From the statement of **Theorem 5.13** we know that every such chain must be finite and hence there cannot be any infinite branch in the tree.

We can now deduce that the tree determined by the algorithm is finite. We know that at each step there is only finite branching and also each branch is of finite length. As a consequence of König's Lemma we obtain that the constructed tree must be finite.

Correctness is a straightforward consequence of finiteness. If there is a branch in the tree that finishes with a successful leaf then that means we have found a set

B containing $P + Q$ such that $CC(B)$ and by **Proposition 5.10** that implies that the ideal $\mathrm{Id}(B)$ constitutes a witnessing bisimulation for P and Q.

On the other hand, if P and Q are bisimilar then, as in Step 4 we consider all possible extensions of the current node, there will be at least one branch in the tree that consists of a chain of finite subsets of $B_\sim$. Since all branches are finite at one point we will find a B so that $CC(B)$ and stop the procedure. □

We can conclude that the procedure that we have described will always halt and give a correct answer.

5.6 Discussion

The method we have presented in this chapter constitutes a new connection between process algebra and classical algebra of polynomials. Much effort has been devoted to solving problems from classical algebra using computers which gave rise to the field of computer algebra. By connecting processes with polynomials we open up new possibilities for developing techniques that would deal with decidability problems, perhaps for a wider range of processes.

There are several directions we might follow now. The first is to try to optimise the presented algorithm in order to obtain a more efficient decision procedure. The current state of art decision procedure for strong bisimilarity of BPP is not even primitive recursive which contrasts with the sequential counterpart of BPP, where for BPA-processes there is an algorithm running in doubly exponential time. The complexity of the presented (nondeterministic) algorithm for BPP mainly depends on two factors. The first is the computational complexity of the ideal membership test which is at least exponential. The second factor concerns the maximal length of a branch constructed during the computation. Unfortunately, we do not use any constructive bound on the maximal length; rather, the argument that each branch eventually reaches an end is an application of **Hilbert's Basis Theorem**. It applies to particular sequences of ideals and we do not know whether there exists any efficient bound on the maximal length of such sequences.

The other direction one might follow is to apply this technique to other process algebras, or equivalences other than strong bisimulation. An example might be applying this technique to strong bisimilarity of BPA-processes. There are 'reasonable' algorithms deciding the strong bisimilarity on basic process algebras, however it might be interesting to find a way of expressing BPA-processes by special polynomials and then proceeding in a similar fashion to BPP.

Lastly, we might try to apply polynomial methods to weak bisimilarity. That

would involve encoding processes as infinite polynomials (power series) for which an equivalent of the **Hilbert's Basis Theorem** still holds ([39], [44]). That might serve as a tool that will represent infinite branching thus making manipulation with a potentially infinite number of derived processes feasible.

Chapter 6

Conclusions and further work

In this thesis we studied decidability of equivalences on simple process algebras, namely strong and weak bisimilarity on Basic Process Algebras and Basic Parallel Process Algebras. We also examined some related issues, such as computational complexity of the decision problems and structural properties of weak bisimulation.

6.1 Strong bisimilarity

We have mentioned on several occasions that strong bisimilarity is decidable for BPA-processes and BPP. There exist polynomial time decision procedures for normed BPA-processes and BPP which is a satisfactory result. For general BPA-processes, the current best decision procedure runs in estimated doubly exponential time. In the case of general BPP, there is much room for improvement. Without further assumptions we cannot even say whether the existing decision procedures have a primitive recursive upper bound.

We have attempted to improve the current situation. We presented a new decision technique which made use of a connection between Basic Parallel Processes and polynomials in the polynomial ring over the two-element field $\mathbb{F}_2$. We were able to construct a decision procedure which used in a substantial way the algorithm for testing polynomial ideal membership.

However, we have not been able to pin down the computational complexity of the presented algorithm. The finiteness is ensured by a theoretical argument and it is not clear whether some concrete upper bound can be obtained. It seems to be worthwhile pursuing further analysis and optimisation of this method, as well as comparison with other techniques, namely Hirshfeld's bisimulation trees. We might also try to develop this technique to deal with other process algebras and other equivalences. For instance, we might attempt to encode processes as infinite

polynomials (power series) and hence attempt to construct a decision procedure for weak bisimilarity.

Another option is to consider sequential composition, that is BPA-processes. We might try to encode them as non-commutative polynomials and attempt to construct a decision procedure for BPA-processes based on a similar principle of ideal membership test. That might provide a uniform way of deciding bisimilarity for both algebras in the framework of classical algebra of polynomials.

Lastly, we may want to concentrate on lower bounds on the complexity of deciding strong bisimilarity. So far there is no indication that there may not exist a polynomial time decision procedure. We have unsuccessfully tried to show that the complexity has to be at least exponential by reducing some hard problems from language and automata theory to bisimilarity. With the new connection to polynomial ideals there arise new possibilities for further research to the problem of lower bounds.

6.2 Weak bisimilarity

There is a wider spectrum of open problems concerning weak bisimilarity. The most important question is that of decidability. We know that for the restricted subclass of totally normed BPA-processes and BPP, weak bisimilarity is decidable. We also know that for BPP, there exists a semidecision procedure. Unfortunately, the other cases remain unanswered. In the pursuit of answers to these questions we also considered related problems of hardness of the decision problem and structural properties of weak bisimulation.

6.2.1 Hardness results

Since we still do not know in general whether weak bisimilarity is decidable, hardness results have to be interpreted as lower bounds on a decision procedure that might exist. We have obtained two kinds of results. The first is NP-hardness of weak bisimilarity for totally normed BPA-processes and BPP. Totally normed processes seem to be an analogue of normed processes with respect to strong bisimilarity. Hence we can view this result as a comparison of the two equivalences, and we can deduce that deciding weak bisimilarity seems to be harder than deciding strong bisimilarity.

The other result applies to BPA-processes. We have demonstrated a reduction from a PSPACE-complete problem to weak bisimilarity of BPA-processes. That implies that any decision procedure which would decide weak bisimilarity for

Basic Process Algebras would be PSPACE-hard.

A direction for further research would be to improve the hardness results. It seems rather likely that to decide weak bisimilarity may require at least exponential time although as yet there is no evidence that would support this conjecture. In particular, the status of weak bisimilarity on BPA is quite puzzling. It might be the case that it is actually undecidable. That would be an interesting contrast to strong bisimilarity where it appears that deciding bisimilarity for BPA-processes is easier than for BPP.

6.2.2 Ordinal characterisation

We investigated an alternative approach to weak bisimulation which consists in defining a non-increasing sequence of weak bisimulation approximants $\approx_\alpha$ that converge at weak bisimulation. These relations were labelled with ordinal numbers and we were searching for the least ordinal number at which convergence occurs. We considered Basic Process Algebras and Basic Parallel Process Algebras separately. To state that precisely, we were looking for the least α such that for every BPA, resp. BPPA, if the sequence of approximants converges at β then $\beta \leq \alpha$.

We managed to find some lower bounds on these ordinal numbers. For BPA, we demonstrated that $\alpha \geq \omega^\omega$, and for BPPA, we showed that $\alpha \geq \omega \cdot 2$. We established these lower bounds by means of examples. We examined two conjectures, that for BPA, $\alpha = \omega^\omega$, and for BPPA, $\alpha = \omega \cdot 2$. Unfortunately, the only upper bound that we have been able to produce so far is rather large. For both classes of algebras, convergence has to occur at the level ω_1. This is a straightforward consequence of a cardinality argument. However, for BPA there may be a way of settling down this upper bound by trying to prove an analogous statement for particular preorders from which the conjecture would easily follow.

Following the method of semideciding weak bisimilarity for BPP we managed to show decidability of each individual approximants $\approx_n$ for BPP. That has an interesting consequence which is semidecidability of $\not\approx_\omega$. That leaves room for the possibility of $\approx_\omega$ and $\approx_{\omega+n}$ being decidable for every n which would result in a semidecision procedure for $\not\approx$.

We may also investigate Milner approximants $\approx^M_\alpha$. We know that $\approx^M_n$ are undecidable for every $n > 1$, it may be the case that $\not\approx^M_n$ might be semidecidable. Then we might investigate the conjecture that $\approx = \approx^M_\omega$. If these conjectures were both proved to be true then they might be combined together to produce a semidecision procedure for non-bisimilarity.

Appendix A

This appendix is devoted to the proof of **Lemma 3.7** from **Chapter 3**.

Lemma 3.7

1. *for every* $\alpha \in On$, $\approx^M_\alpha \subseteq \approx_\alpha \subseteq \approx^s_\alpha$
2. *for every* $\alpha, \beta \in On$, $\alpha < \beta \Rightarrow \approx_\beta \subseteq \approx_\alpha$
3. *for every* $\alpha \in On$, $\approx \subseteq \approx_\alpha$
4. *if there is an* α *such that* $\approx_\alpha = \approx_{\alpha+1}$ *then for all* $\beta \geq \alpha$, $\approx_\alpha = \approx_\beta = \approx$
5. $\approx = \bigcap_{\alpha \in On} \approx_\alpha$
6. *for BPA and BPPA,* $\approx = \approx_{\omega_1}$

Proof: In order to prove these claims we will need the full power of transfinite induction. We recall that to verify that some property P holds for the class On we have to test three cases: the base case $P(0)$, the successor case $P(\alpha) \Rightarrow P(\alpha+1)$ and the limit case $(\forall \alpha < \lambda.\ P(\alpha)) \Rightarrow P(\lambda)$.

We have expressed the properties 2. to 6. in terms of $\approx_\alpha$ and that is how we will prove them. However, the properties remain valid even for the other approximants $\approx^s_\alpha$ and $\approx^M_\alpha$. If we take any two approximants $\approx_\alpha$ and $\approx_\beta$ then clearly $\approx_\beta \subseteq \approx_\alpha$ iff for every pair of processes P and Q, $P \approx_\beta Q$ implies that $P \approx_\alpha Q$. Hence we will prove the inclusions in terms of the latter implication.

1. We will show that for every α, $\approx^M_\alpha \subseteq \approx_\alpha$. The proof that $\approx_\alpha \subseteq \approx^s_\alpha$ then follows along the same lines. The claim is straightforward for $\alpha = 0$. Assuming that $\approx^M_\alpha \subseteq \approx_\alpha$, we will show that $\approx^M_{\alpha+1} \subseteq \approx_{\alpha+1}$. If $P \approx^M_{\alpha+1} Q$ then for every sequence of moves $P \stackrel{w}{\Longrightarrow} P'$ there is a response $Q \stackrel{w}{\Longrightarrow} Q'$ with $P' \approx^M_\alpha Q'$. Hence also for every single move $P \stackrel{\mu}{\Longrightarrow} P'$ there exists a derivation $Q \stackrel{\mu}{\Longrightarrow} Q'$ with

$P' \approx^M_\alpha Q'$. From our assumption follows that also $P' \approx_\alpha Q'$ and since this is true for all moves of P and Q, we can conclude that $P \approx_{\alpha+1} Q$.

The argument for a limit ordinal λ goes as follows. The approximant $\approx^M_\lambda$ is taken to be $\bigcap_{\alpha<\lambda} \approx^M_\alpha$ and we assume that for every $\alpha < \lambda$, $\approx^M_\alpha \subseteq \approx_\alpha$. Hence also the intersection $\approx^M_\lambda \subseteq \approx_\alpha$ for every $\alpha < \lambda$ and we can conclude that $\approx^M_\lambda \subseteq \bigcap_{\alpha<\lambda} \approx_\alpha = \approx_\lambda$.

2. The approximant $\approx_0$ is the universal binary relation hence all other approximants must be included in it. The next step is to show that if for all $\alpha < \beta$, $\approx_\beta \subseteq \approx_\alpha$ then also for all $\alpha < \beta + 1$, $\approx_{\beta+1} \subseteq \approx_\alpha$. We take $P \approx_{\beta+1} Q$ and a move $P \stackrel{\mu}{\Longrightarrow} P'$. Then there exists a response $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx_\beta Q'$. From the induction hypothesis $P' \approx_\alpha Q'$ for any $\alpha < \beta$ and we can conclude that $P \approx_\alpha Q$ for $\alpha < \beta + 1$. The limit case is straightforward because we can express $\approx_\lambda = \bigcap_{\alpha<\lambda} \approx_\alpha$. Then, trivially, $\approx_\lambda \subseteq \approx_\alpha$ for any $\alpha < \lambda$.

3. Again this statement trivially holds for $\approx_0$ as it is the universal relation. Next we will verify that if $\approx \subseteq \approx_\alpha$ then also $\approx \subseteq \approx_{\alpha+1}$. For every $P \approx Q$ and $P \stackrel{\mu}{\Longrightarrow} P'$ there exists a $Q \stackrel{\mu}{\Longrightarrow} Q'$ such that again, $P' \approx Q'$. Then we can claim that $P' \approx_\alpha Q'$ and as a consequence we obtain that the original P and Q are related at $\approx_{\alpha+1}$. For a limit λ, if $P \approx Q$ and $P \stackrel{\mu}{\Longrightarrow} P'$ then there is $Q \stackrel{\mu}{\Longrightarrow} Q'$ such that $P' \approx Q'$ and hence $P' \approx_\alpha Q'$ for any $\alpha < \lambda$. Hence, $P \approx_\lambda Q$.

4. First we verify that if $\approx_\alpha = \approx_{\alpha+1}$ then for every $\alpha < \beta$, $\approx_\alpha = \approx_\beta$. To do that it suffices to prove that if $\approx_\alpha \subseteq \approx_{\alpha+1}$ then $\approx_\alpha \subseteq \approx_\beta$ for all $\alpha < \beta$. The other implication that ensures equality of all the approximants follows from claim 2.

We start with the fact that if $\approx_\alpha \subseteq \approx_\beta$ then also $\approx_\alpha \subseteq \approx_{\beta+1}$. We assume $P \approx_\alpha Q$ and a transition $P \stackrel{\mu}{\Longrightarrow} P'$. Since $\approx_\alpha \subseteq \approx_{\alpha+1}$, we also have that $P \approx_{\alpha+1} Q$ and so there exists a matching transition $Q \stackrel{\mu}{\Longrightarrow} Q'$ such that $P' \approx_\alpha Q'$. As $\approx_\alpha \subseteq \approx_\beta$, we have that $P' \approx_\beta Q'$ and we can put these facts together to deduce that $P \approx_{\beta+1} Q$. For a limit ordinal λ, if for every $\beta < \lambda$ $\approx_\alpha \subseteq \approx_\beta$ then as $\approx_\lambda = \bigcap_{\beta<\lambda} \approx_\beta$, we can conclude that $\approx_\alpha \subseteq \approx_\lambda$.

It remains to be proved that if there is an α such that $\approx_\alpha = \approx_{\alpha+1}$ then $\approx_\alpha = \approx$. The proof of this claim relies on the fact that $\approx$ is the maximal weak bisimulation and hence includes all other weak bisimulations. Therefore it suffices to show that if $\approx_\alpha = \approx_{\alpha+1}$ then $\approx_\alpha$ is a weak bisimulation. We presuppose processes P and Q such that $P \approx_\alpha Q$. It follows from our assumption that also $P \approx_{\alpha+1} Q$ and so for any move $P \stackrel{\mu}{\Longrightarrow} P'$ there is a matching response $Q \stackrel{\mu}{\Longrightarrow} Q'$ with $P' \approx_\alpha Q'$. The same also holds for any moves of Q and we can conclude that $\approx_\alpha$ is closed

under expansion and thus forms a weak bisimulation relation. That implies that $\approx_\alpha \subseteq \approx$ and finally, $\approx_\alpha = \approx$.

5. The proof of the fact that $\approx = \bigcap_{\alpha \in On} \approx_\alpha$ involves arguments from fixed-point theory and it is an analogue of **Proposition 2.6**.

6. In order to demonstrate this claim we need to use some extra property of BPA and BPPA. Without loss of generality we fix a BPA (Σ^*, Δ). The set of variables Σ is finite and so the set of processes which corresponds to the free monoid Σ^* is countable. Each approximant $\approx_\alpha$ is a subset of $\Sigma^* \times \Sigma^*$ and hence also countable. The approximants form a non-increasing sequence of countable sets. We can use the property of such sequences that says that in every such sequence, there must be an $\alpha < \omega_1$ such that $\approx_\alpha = \approx_\beta$ for all $\alpha < \beta$. Then by applying claim 4. above, we deduce that for this $\alpha < \omega_1$, $\approx_\alpha = \approx_{\omega_1} = \approx$. The proof would work analogously for BPPA.

Bibliography

[1] Balcar B. and Štěpánek P. **Set theory (in Czech)**. Academia/Praha, 1986.

[2] Baeten J.C.M., Bergstra J.A. and Klop J.W. *Decidability of bisimulation equivalence for processes generating context-free languages*, in Proceedings of PARLE'87, LNCS 259, 1987.

[3] Buchberger B. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal (in German)*, Ph.D. Thesis, University of Innsbruck, Math. Inst., 1965.

[4] Burkart O., Caucal D and Steffen B. *An elementary bisimulation decision procedure for arbitrary context-free processes*, in Aachener Informatik - Berichte 94–28, 1994.

[5] Caucal D. *Graphes canoniques de graphes algébriques*, in Informatique Théorique et Applications (RAIRO) 24(4), 339–352, 1990.

[6] Christensen S. *Decidability and Decomposition in Process Algebras*, Ph.D. Thesis, University of Edinburgh, 1993.

[7] Christensen S., Hirshfeld Y. and Moller F. *Decomposability, Decidability and Axiomatisability for Bisimulation Equivalence on Basic Parallel Processes*, in Proceedings of LICS, IEEE Computer Society Press 386–396, 1993.

[8] Christensen S., Hirshfeld Y. and Moller F. *Bisimulation Equivalence is Decidable for Basic Parallel Processes*, in Proceedings of CONCUR 93, LNCS 715, 143–157, 1993.

[9] Christensen S., Hirshfeld Y. and Moller F. *Decidable subsets of CCS*, in The Computer Journal 37, No.4, 1994.

[10] Christensen S., Hüttel H. and Stirling C. *Bisimulation equivalence is decidable for all context-free processes*, in Proceedings of CONCUR 92, LNCS 630, 138–147, 1992.

[11] Christensen S., Hüttel H. and Stirling C. *Bisimulation equivalence is decidable for all context-free processes*, in Information and Computation 121(2), 143–148, 1995.

[12] Davenport J.H., Siret Y. and Tournier E. **Computer Algebra**, Academic Press, 1993.

[13] Dickson L.E. *Finiteness of the odd perfect and primitive abundant numbers with distinct factors*, in American Journal of Mathematics 35, 413–422, 1913.

[14] Eilenberg S. and Schützenberger M.P. *Rational sets in commutative monoids*, in Journal of Algebra 13, 173–191, 1969.

[15] Esparza J. *Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes*, in Fundamenta Informaticae 31, 13–26, 1997.

[16] Esparza J. *Petri nets, commutative context-free grammars and basic parallel processes*, in Fundamentals of Computation Theory 95, LNCS 965, Springer Verlag, 221–232, 1995.

[17] Garey M.R. and Johnson D.S. **Computers and Intractability: A Guide to the Theory of NP-completeness**. Freeman, San Francisco, 1979.

[18] Geddes K.O., Czapor S.R. and Labahn G. **Algorithms for Computer Algebra**, Kluwer Academic Publishers, 1992.

[19] Ginsburg S. and Spanier E.H. *Semigroups, Presburger formulas and languages*, in Pacific Journal of Mathematics 16, 285–296, 1966.

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

[21] Halmos P.R. **Naive Set Theory**, Van Nostrand Reinhold Company, 1960.

[22] Hayden S. and Kennison J.F. **Zermelo-Fraenkel Set Theory**, Charles E. Merrill Publishing Company, Columbus, Ohio, 1968.

[23] Hennessy M. and Milner R. *Algebraic laws for nondeterminism and concurrency*, in Journal of the ACM 32, 137–161, 1985.

[24] Hilbert D. and Bernays P. **Grundlagen der Mathematik**, Edward Brothers, Inc., Aan Arbor, Michigan, 1944.

[25] Hinman P.G. **Recursion-Theoretic Hierarchies**, Springer-Verlag, Berlin Heidelberg New York, 1978.

[26] Hirshfeld Y. *Petri Nets and the Equivalence Problem*, in Proceedings of CSL'93, LNCS 832, 165–174, 1993.

[27] Hirshfeld Y. *Deciding equivalences in simple Process Algebras*, in LFCS Report Series, University of Edinburgh, 1994.

[28] Hirshfeld Y. *Bisimulation trees and the decidability of weak bisimulations*, in Proceedings of INFINITY'96, Steffen B., Caucal D. (Eds.), ENTCS 5, 1996.

[29] Hirshfeld Y., Jerrum M. and Moller F. *A polynomial-time algorithm for deciding bisimulation equivalence of normed Basic Parallel Processes*, in LFCS Report Series, University of Edinburgh, 1994.

[30] Hirshfeld Y., Jerrum M. and Moller F. *A polynomial-time algorithm for deciding bisimulation equivalence of normed Basic Parallel Processes*, in Math. Struct. in Comp. Science 6, 251–259, Cambridge University Press, 1996.

[31] Hirshfeld Y., Jerrum M. and Moller F. *A polynomial algorithm for deciding bisimilarity of normed context-free processes*, in LFCS Report Series, University of Edinburgh, 1994.

[32] Hirshfeld Y. and Moller F. *Decidability results in automata and process theory*, in Logics for Concurrency, Moller F., Birtwistle G. (Eds.), LNCS 1043, 1996.

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

[34] Hüttel H. *Decidability, Behavioural Equivalences and Infinite Transition Graphs*, PhD Thesis, University of Edinburgh, 1991.

[35] Hüttel H. *Silence is golden: branching bisimilarity is decidable for context-free processes*, in LFCS Report Series, University of Edinburgh, 1991.

[36] Hüttel H. *Undecidable equivalences for Basic Parallel Processes*, in LFCS Report Series, University of Edinburgh, 1993.

[37] Hüttel H. and Stirling C. *Actions speak louder than words: Proving bisimilarity for context-free processes*, in Proceedings of the 6th annual IEEE symposium on logic in computer science (LICS), 376–386, Amsterdam, 1991.

[38] Huynh D.T. and Tian L. *Deciding bisimilarity of normed context-free processes is in* Σ_2^p, in Journal of Theoretical Computer Science 123, 183–197, 1994.

[39] Jacobson N. **Basic Algebra I, II**, W.H. Freeman and Company, 1980.

[40] Jančar P. *Decidability Questions for Bisimilarity of Petri Nets and Some Related Problems*, in Proceedings of STACS'94, LNCS 775, 581–592, 1995.

[41] Jančar P. *Undecidability of Bisimilarity for Petri Nets and Some Related Problems*, in Journal of Theoretical Computer Science 148, 281–301, 1995.

[42] Karp R.M. *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Miller R.E. and Thatcher J.W. (Eds.), 85–103, Plenum Press, New York, 1972.

[43] Koppenhagen U. and Mayr E.W. *An Optimal Algorithm for Constructing the Reduced Gröbner Basis of Binomial Ideals*, in Proceedings of ISSAC'96, ACM Press, 55–62, 1996.

[44] Lang S. **Algebra**, 3rd edition, Addison-Wesley, 1993.

[45] Levy A. **Basic Set Theory**, Springer-Verlag, Berlin, 1979.

[46] Mayr E.W. and Meyer A.R. *The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals*, in Advances in Mathematics 46, 305–329, 1982.

[47] McAloon K. *Petri nets and large finite sets*, in Theoretical Computer Science 32, North-Holland, 1984.

[48] Milner R. **A Calculus of Communicating Systems**, LNCS 92, Springer-Verlag, 1980.

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

[50] Milner R. and Moller F. *Unique decomposition of processes*, in Journal of Theoretical Computer Science 107, 357–363, 1993.

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

[52] Moller F. Axioms for Concurrency, PhD Thesis, University of Edinburgh, 1989.

[53] Moller F. *Infinite Results*, in Proceedings of CONCUR'96, Montanari U., Sassone V. (Eds.), LNCS 1119, 1996.

[54] Papadimitriou C.H. **Computational Complexity**, Addison-Wesley, 1994.

[55] Park D.M.R. *Concurrency and Automata on Infinite Sequences*, in Theoretical Computer Science, 5th GI-Conference, P. Deussen (Ed.), LNCS 104, 168–183, 1981.

[56] Presburger M. *Über die Vollständigkeit eines gewissen Systems der Arithmetic ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt*, Sprawozdanie z I Kongresu Matematykow Krajow Slowianskych, Warsaw, 92–101, 1930.

[57] Sénizergues G. *Decidability of bisimulation equivalence for equational graphs of finite out-degree*, in Proceedings of FOCS'98, IEEE, 120–129, 1998.

[58] Stirling C. *Decidability of bisimulation equivalence for normed pushdown processes*, in Proceedings of CONCUR'96, Montanari U. and Sassone V. (Eds.), LNCS 1119, 217–232, 1996.

[59] Stříbrná J. *Decidability of strong bisimulation of basic parallel processes using Hilbert's basis theorem*, in Proceedings of INFINITY'97, Moller F. (Ed.), ENTCS 9, 1997.

[60] Stříbrná J. *Hardness results for weak bisimilarity of simple process algebras*, in Proceedings of MFCS'98 Workshop on Concurrency, ENTCS 18, 1998.

[61] van der Waerden B.L. **Algebra II**, Ungar, New York, 1970.

[62] Winkler F. **Polynomial Algorithms in Computer Algebra**, Springer-Verlag, 1996.