Decidability and Decomposition in Process Algebras

Søren Christensen

Doctor of Philosophy

University of Edinburgh

1993

(Graduation Date: September 1993)

Abstract

This thesis is concerned with the question of obtaining decidable theories for behavioural equivalences on various models of (parallel) computation encompassing systems with infinitely many states. The models on which we concentrate are based on the process calculi BPA and BPP but we also consider labelled Petri nets. The equivalences which we are interested in are language equivalence, bisimulation equivalence and distributed bisimulation equivalence.

BPA (Basic Process Algebra) is provided by a standard calculus which admits of a general sequencing operator, along with atomic actions, choice and recursion. In contrast, BPP (Basic Parallel Processes) is provided by a standard calculus which admits of a simple parallel operator (full merge), along with atomic actions, choice and recursion. From the point of view of language equivalence we show that BPA and BPP are incomparable. This result is in part obtained through a pumping lemma for BPP. We furthermore show that the class of languages generated by BPP is included in a class of languages generated by labelled Petri nets as well as contained in the class of context-sensitive languages. Next we investigate into decidability of language equivalence on language classes related to BPP and Petri nets.

We then move on to bisimulation equivalence. We show that this equivalence is decidable on all of BPA, answering a question left open by Baeten, Bergstra and Klop. We also show that bisimulation equivalence is decidable on BPP and BPP_τ (where BPP_τ is similar to BPP except for its parallel operator which allows for synchronisation). By these results we have obtained a delicate line between decidable and undecidable theories for bisimilarity: by extending BPP_τ with the

operator of restriction we know that bisimilarity is undecidable. By relying on Jančar's recent result on the undecidability of bisimilarity on labelled Petri nets we further narrow the gap between decidable and undecidable theories: when extending BPP with a notion of forced synchronisation we know that bisimilarity is undecidable.

The decidability proofs of bisimulation equivalence on BPP and BPP_τ (obtained via the tableau technique) permit us further to present sound and complete equational theories for these process classes. As BPP and BPP_τ contain the regular processes these results may be seen as proper extensions of Milner's equational theory for regular processes.

In this thesis we shall also consider the problem of decomposing a process into the parallel composition of simpler processes. A particular aspect of decomposition involves decomposing uniquely into prime processes, i.e. those processes which cannot themselves be expressed as a parallel composition of other non-trivial processes. We shall present several unique decomposition results for bisimilarity and distributed bisimilarity on BPP and BPP_τ as well as subclasses thereof.

Finally, for distributed bisimulation equivalence we show decidability on the process classes BPP and BPP_τ. Again our proofs of decidability permit us to present sound and complete equational theories. These results may be seen as extensions of Castellani's equational theories for distributed bisimilarity on the recursion-free fragments of BPP and BPP_τ. Also for a fragment of BPP_τ where general summation is replaced by guarded summation shall we present an equational theory for distributed bisimilarity. The proof of completeness relies on the unique decomposition property admitted by this equivalence on BPP_τ.

Acknowledgements

I would first of all like to thank my supervisor Colin Stirling for his encouragement, guidance and support; had it not been for this I would not have written this thesis.

Thanks are also due to Faron Moller and Yoram Hirshfeld for numerous inspiring discussions. Without their help my Chapter 6 on deciding bisimulation equivalence for the process class BPP would have been a less attractive story. Thanks also go to Faron Moller and Hans Hüttel for reading an early draft of my thesis.

I also thank Aarhus University (Daimi) for their kind hospitality whenever I felt the need to visit Denmark. In particular, I thank Mogens Nielsen and Uffe Engberg for always being willing to listen to my ideas and problems, and for their illuminating comments.

Thanks also go to the staff, students and my friends in Edinburgh who made the experience of doing a thesis more enjoyable.

Finally, warm thanks to Lone for being there and for helping me stay clear during difficult periods.

***

The work presented in this thesis has been financially supported by a fellowship from Aarhus University and by a scholarship from the Danish Research Academy.

Declaration

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

Some of the material has been published as [Chr92], [CHS92], [CHM93] and [CHM93a].

Søren Christensen

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An important objective ever since the beginning of computer science has been the *verification* of systems correctness. A particular aspect of verification involves checking whether systems exhibit the *same* behaviour (according to some notion of behavioural equivalence). Such questions of verification are necessarily limited by the computational expressiveness of the systems being explored. For instance, checking whether a Turing machine halts is in general undecidable (see e.g. [HU79]). This thesis is concerned with the investigation into obtaining decidable theories of behavioural equivalences on various models of computation. We shall mostly concentrate on parallel models encompassing processes with infinite-state behaviour and thereby hopefully add to the understanding of reasoning about concurrent processes.

In the area of concurrency much attention has been devoted to the *algebraic approach* and a number of *process calculi* such as CCS [Mil89], ACP [BW90] and CSP [Hoa85] have emerged. The concept of *compositional reasoning* plays a fundamental rôle in these calculi: one describes a process by indicating how it is constructed from smaller but similar processes. It has become standard to represent the *behaviour* of processes in the style of *structural operational semantics*, an approach first proposed in [Plo81]. Underpinning this approach is the notion of a *labelled transition graph*, a fundamental and generally accepted model of computation.

Central to the algebraic approach is the notion of processes exhibiting the same behaviour and a number of *behavioural equivalences* have been suggested, e.g. those within the linear time–branching time spectrum of [Gla90]. The main rationale for these various equivalences has been to capture behavioural aspects such as deadlock, livelock and causality that standard language equivalence from formal language theory does not take into account. Notably, one equivalence, viz. *bisimulation equivalence* as defined by Milner and Park (see e.g. [Mil80,Par81]), has been extensively studied and now forms the foundation for the theory of CCS (see [Mil89]).

Finite-state systems have been extensively studied, both within formal language theory and within the theory of process calculi. Milner has in [Mil84] shown that finite-state systems correspond to the transition graphs for regular CCS processes, i.e. the recursively defined CCS processes over a signature of *action prefixing* and *choice*. For finite-state systems all studied equivalences are known to be decidable, and sound and complete equational theories exist for various of these, see e.g. [Sal66,Mil84,Koz91,Rab92]. Furthermore, several automated tools have been designed for the analysis of finite-state systems (see, e.g. [Mad92]). In fact, it is fair to postulate that the theory of finite-state systems and their equivalences is well-established.

The characterisation of finite-state behaviours of [Mil84] also shows that as soon as one moves beyond the constructors of regular CCS (also known as the dynamic constructors) recursively defined processes may have infinitely many states. This includes many realistic cases; in particular processes that are defined using various notions of *parallel composition* such as the asynchronous parallel operator of CCS. A natural question then is whether (parts of) the theory for finite-state behaviours can be extended to the infinite-state case, and recently researchers have been investigating this subject.

An example of systems with infinitely many states is that of the transition graphs of processes in the calculus BPA (Basic Process Algebra), also known as the class of *context-free* processes (see [BBK87]). Certainly language equivalence becomes undecidable here since, from the point of view of generating languages, BPA yields exactly the class of context-free languages. Furthermore, in [HT90, GH91] it is demonstrated that *all* of the standard behavioural equivalences, *except* bisimilarity, within the linear time–branching time spectrum of [Gla90] are undecidable over context-free processes. In [BBK87] (see also [Cau90,HS91,Gro91]) it is shown that bisimilarity is in fact decidable for the subclass of *normed* BPA (those processes which may terminate in a finite number of steps at any point during their execution). Furthermore, in [HS91] a sound and complete equational theory for normed BPA wrt bisimilarity has been designed.

Another example of the investigation of infinite-state systems (though not involving decidability of behavioural equivalences) concerns the question of *model-checking*, i.e. checking whether a given process satisfies a given formula in a suitable process logic such as the modal mu-calculus. For instance, the result of [BB92] shows decidability of model-checking fragments of the modal mu-calculus in the setting of normed BPA processes. Furthermore, in [BS90] a model-checker (based on the *tableau technique*) has been designed for the analysis of general systems, whether finite or infinite.

The class of context-free processes is provided by a standard process calculus which admits of a general *sequencing operator*, along with *atomic actions*, *choice* and *recursion*. However, of greater importance in process theory is the inclusion of some form of parallel combinator. Very little work, it seems, has been spent on exploring the decidability (or otherwise) of equivalences defined over process calculi which include parallel combinators within recursive definitions. Certainly with very little else one can express the power of Turing machines along with their undecidable problems (see e.g. [Tau89]). We shall thus face this challenge studying classes of recursively defined processes using various notions of parallelism and explore the possibilities of obtaining decidable theories for behavioural equivalences.

The behavioural equivalences in which we shall be interested are *language equivalence*, *bisimulation equivalence* and finally *distributed bisimulation equivalence*. All the behavioural equivalences within the linear time–branching time spectrum of [Gla90] (including language equivalence and bisimulation equivalence) interpret parallelism as nondeterministic interleaving. In contrast, distributed bisimulation equivalence as defined by Castellani and Hennessy (see [CH87, Cas88]) is a so-called noninterleaving equivalence that interprets concurrent processes in an operational setting which abandons the understanding of parallelism as nondeterministic interleaving. As such, it is *finer* than any of the equivalences within the linear time–branching time spectrum. We study distributed bisimilarity mainly because we deal with various models of computation focusing on parallelism and because we wish to investigate *properties* that are shared or divided between the two approaches of interleaving versus noninterleaving to the theory of concurrency (as for instance represented by bisimilarity and distributed bisimilarity).

In this thesis a particular approach to the decidability of behavioural equivalences on infinite-state systems involves decomposition of processes into "smaller" processes. We shall also see that in certain circumstances, a decision procedure may support a sound and complete equational theory, i.e. a formal system for deriving equivalences. We shall therefore concentrate on the three related issues of *decomposition*, *decidability* and *equational theories* in this thesis.

1.1 Decomposition

Decomposition may be seen as an example of the "divide and conquer" method. It has proven valuable in many areas, e.g. in program design (where a problem is divided into subproblems each of which is dealt with by a (recursive) procedure call), in logics (such as Hoare logic) and in the area of Petri nets (where for instance a net is decomposed into subnets each of which is analysed separately). The method of decomposition may support verification techniques, but may also support more theoretical work. It is the latter aspect which we explore in this thesis.

On deciding behavioural equivalences, decomposition is a key approach to make the infinite tractable. We shall explore various ways of decomposing pairs of equivalent processes into "smaller" pairs of equivalent processes such that only *finitely* many pairs of equivalent processes cannot be decomposed further. If achievable then this may provide us with a *finite representability* of the equivalence in question, a vital step towards obtaining decidability. For instance, in Chapter 4 we shall prove that bisimulation equivalence for any system of BPA processes is generated from a finite set of bisimilar pairs of processes using the usual laws for equational reasoning, i.e. the laws of reflexivity, symmetry and transitivity, together with the law of congruence wrt sequential composition. This will enable us to conclude that bisimilarity is decidable on BPA, and extends the result of [BBK87] beyond the normed case.

In a setting with parallel combinators, a particular aspect of decomposition involves decomposing *uniquely*, a question first addressed by Milner and Moller [MM90]. Unique decomposition is conceptually related to the notion of unique prime factorisation as it appears in algebra and is concerned with the question of finding a unique way of representing a given process as a parallel composition of other (simpler) processes. The simplest (non-trivial) processes are by analogy called *prime* processes.

Suppose $\times$ is a parallel operator on processes and $=$ a semantic equivalence on the class of processes considered. Further, suppose $\times$ is commutative, associative and has a unit (which we denote by $\mathbf{0}$) relative to the equivalence $=$. Then we say a process E is *prime* if, and only if, $E \neq \mathbf{0}$, and whenever $E = F \times G$ for processes F and G then $F = \mathbf{0}$ or $G = \mathbf{0}$.

If E is a process then the question of the unique decomposition of E can be stated as follows: does there exist a unique multiset (up to $=$) of prime processes $\{E_1, E_2 \ldots E_n\}$ such that

$$E = E_1 \times E_2 \times \cdots \times E_n.$$

In [MM90] Milner and Moller answer the question of unique decomposition affirmatively for finite processes (processes represented by finite acyclic transition graphs) with $\times$ being a purely interleaving operator and $=$ bisimulation equivalence. Two proofs of this result exist: the original proof by Milner and a simplified proof by Moller relying on a cancellation law admitted by finite processes.

In [MM90] Milner and Moller also note that unique decomposition does not hold for finite processes when $=$ is trace equivalence or failures equivalence (these observations are attributed to Hirshfeld and van Glabbeek respectively). Milner and Moller also note that wrt bisimulation equivalence finite-state processes (processes represented by finite transition graphs) may have no decomposition at all. For example, $X \stackrel{\text{def}}{=} aX$, i.e. the process that performs an a-action forever, is such a process.

Recently, Hirshfeld and Moller have discovered that Milner's original proof of unique decomposition for finite processes wrt bisimulation equivalence can be extended to a class of normed infinite-state behaviours [CHM93]. We shall give a full proof of this result in the thesis. As a simple consequence we obtain a *cancellation law* for normed processes, i.e. a law that allows one to remove common factors in bisimilar expressions. Interestingly enough, it remains an open question as to whether one can obtain the cancellation law without recourse to the property of unique decomposition. The validity of a cancellation law is important since it suggests a notion of decomposition of equivalent pairs, namely that given by cancelling out common factors.

When one moves beyond normed processes the property of unique decomposition does not hold wrt bisimulation equivalence. Our example of $X \stackrel{\text{def}}{=} aX$ is evidence of this. However, wrt distributed bisimulation equivalence the situation is different; we have been able to obtain unique decomposition for distributed bisimilarity on classes of infinite-state behaviours not necessarily being normed.

There are a number of reasons why the question of unique decomposition is interesting. For instance, it may have implications for the way processes can usefully be allocated for solving a problem. It may also be useful in the specification and verification of finite-state systems (see e.g. [GM92]). For instance, in verifying that $E = F$ one normally constructs the finite state space (if possible) and then uses an equivalence checker to calculate $E = F$. Often the state spaces of E and F are too large to be handled. But by first decomposing and then checking on the components one might avoid the state-space explosion.

In this thesis we shall use the property of unique decomposition in the decidability of process equivalences. If a class of processes has the property of unique decomposition wrt a particular equivalence then an appealing approach to checking this equivalence would be to decompose processes into primes and then check for equality among the primes. We shall see examples of this approach in the thesis. Also we shall use the property of unique decomposition wrt distributed bisimulation equivalence to obtain a sound and complete equational theory on a class of infinite-state behaviours. Unique decomposition shall play the rôle of normal form in obtaining the completeness proof.

1.2 Decidability of Behavioural Equivalences

One of the major problems addressed by this thesis is that of deciding whether processes exhibit the same behaviour. There are a number of existing approaches to the decidability (of behavioural equivalences or otherwise); in Chapter 2 we shall survey some of these techniques. In particular, we shall explore the use of *tableau systems*, a technique that has proven valuable in many areas. For instance, the tableau method has been used in the area of logics to show the validity of modal systems such as T, S4 and S5, as well as being templates for sound and complete axiomatisations (see [HC68]). In the area of automata and formal language theory the tableau method has been used in order to decide various equivalences; an example being the Korenjak-Hopcroft algorithm for deciding language equivalence of simple deterministic grammars (see [HK66,Cou83]). Recently, the tableau technique has been advocated by Stirling and Walker for model-checking the modal mu-calculus (see [SW89]), and it has also proven valuable for checking bisimilarity (see [HS91]).

For our purposes, a tableau system is a syntax driven, goal directed proof system: one replaces the goal of proving process E equal to process F by a set of (sufficient) subgoals to be established according to a finite number of *tableau rules*; the application of a particular rule is guided by the structure of processes. The tableau method provides a very natural procedure for checking bisimilarity: given two processes E and F one starts by building the essential part of a bisimulation relation by examining the possible moves performed by E and F.

We shall use the tableau method both for bisimilarity and distributed bisimilarity on various classes of processes. In particular, we shall prove that bisimulation equivalence is decidable on a class of recursively defined processes over a signature including *action prefixing*, *choice* and *parallel composition*. This class of processes is denoted BPP_τ (Basic Parallel Processes with τ) and is essentially pure CCS without *relabelling* and *restriction* (see [Mil89]). With this result we obtain a delicate line between decidable and undecidable theories for bisimulation equivalence: by extending BPP_τ with the combinator of restriction we know that bisimulation equivalence becomes undecidable as the Halting problem for Turing machines can be reduced to a bisimulation question. This result is essentially due to Milner but shown in detail by Taubner (see [Tau89]); we shall present the reduction in Chapter 2 following Taubner.

Having settled the bisimilarity problem for BPP_τ we then consider decidability for bisimulation equivalence on models of computation that are more expressive than BPP_τ. We show that by extending BPP_τ with a general synchronisation mechanism we obtain a calculus denoted SPP (Synchronised Parallel Processes) which is (at least) as expressive as labelled Petri nets. This result is obtained via a delicate decomposition of labelled Petri nets into simpler subnets expressible in BPP_τ. The processes of the subnets are then combined in parallel, and by relying on the general synchronisation mechanism designed we get the behaviour of the original net (up to bisimilarity). The question then is whether we can decide bisimilarity on the calculus SPP. However, by a recent result of Jančar showing undecidability of bisimilarity on labelled Petri nets [Jan93] we must conclude that bisimilarity is also undecidable on our extended calculus. This further narrows the gap between decidable and undecidable theories for bisimilarity since SPP is less expressive than the calculus BPP_τ extended with restriction.

Distributed bisimulation equivalence is also decidable on BPP_τ. Again, this result is obtained via the use of tableaux. However, in this case, the tableaux become particularly simple to analyse because of a strong cancellation law admitted by distributed bisimilarity. A natural question is whether we can extend the decidability result beyond BPP_τ. For instance, is distributed bisimilarity decidable on the extended calculus SPP (and thus on labelled Petri nets)? However, we encounter a serious problem here since we do not know how to obtain an appropriate definition of distributed bisimilarity on SPP. We shall discuss this point in Chapter 8 and also suggest possible solutions. The problem is in nature similar to a problem Castellani encountered when trying to define distributed bisimilarity on a calculus including the restriction operator of CCS (see [Cas88]).

1.3 Equational Theories

For our purposes, an equational theory is a formal system for deriving equivalences between processes. It consists of a set of axioms, i.e. equivalences that are accepted as universally true, together with the laws for equivalence (reflexivity, symmetry and transitivity) as well as the laws for substitutivity. Usually such theories are presented in a *natural deduction style* (see [Pra65]) yielding proof trees for deriving equivalences.

Milner proposed in [Mil84] a sound and complete equational theory for proving bisimulation equivalence between regular CCS processes, i.e. finite-state processes. The theory centres around a few laws for choice together with very natural rules for recursion essentially capturing that a process is equivalent to its unfolding and that recursive equations have unique solutions (up to bisimilarity).

One advantage of the tableau method is that it may support a sound and complete equational theory. In [HS91] Hüttel and Stirling have been able to extract an equational theory for bisimilarity on normed BPA from a corresponding tableau system. In the style of [HS91] we shall obtain a sound and complete equational theory for bisimulation equivalence on BPP_τ by appealing to a tableau system. As our calculus clearly contains the regular processes this result may be seen as a *proper* extension of Milner's result on regular CCS.

However, our equational theory is very different from Milner's as it involves *sequents*, i.e. relationships between assumptions (consisting of pairs of processes) and equations, and appeals to assumption introduction and elimination rules. We shall refer to such systems as *sequent-based* equational theories. In this thesis we do not explore the question of obtaining an equational theory of bisimilarity on BPP_τ using a system in the style of Milner's for finite-state behaviours.

We shall also present a sound and complete equational theory for distributed bisimilarity on BPP_τ (though involving the operator *left merge*). Again we shall appeal to a tableau system (for deciding distributed bisimilarity on BPP_τ) in obtaining the equational theory and therefore it is sequent-based.

Finally, wrt a subset of BPP_τ where general summation is replaced by *guarded summation* (see [Mil89]) we obtain a very simple equational theory for distributed bisimulation equivalence without recourse to the operator of left merge. Our theory is similar in style to Milner's equational theory for bisimilarity on regular CCS and the proof of completeness relies on the property of unique decomposition admitted by distributed bisimilarity on BPP_τ.

1.4 Layout of the Thesis

In the rest of this introduction we go through each of the remaining chapters indicating the contents.

In Chapter 2 we set the scene for the rest of the thesis: we define various process classes such as BPP, BPP_τ and BPA as well as introduce the model of labelled Petri nets. The calculus BPP is similar to BPP_τ except for its parallel operator which is *full merge* where synchronisation between processes is absent. We shall also define the semantic equivalences of language equivalence, bisimulation equivalence and distributed bisimulation equivalence. Finally, we shall introduce various notions and techniques used throughout the thesis such as the tableau method, equational theories and decompositions.

In Chapter 3 we study BPP from a formal language theoretic point of view; we shall compare BPP with standard language classes such as the context-free and the context-sensitive languages as well as languages generated from labelled Petri nets. We shall finally consider decidability questions for language equivalence on various classes of languages based on BPP as well as labelled Petri nets.

The result of [BBK87] demonstrates that bisimilarity is decidable on the class of *normed* BPA processes. In Chapter 4 we shall extend this result thus settling the question in the affirmative for *all* context-free processes. This chapter is based on the paper [CHS92] with Hüttel and Stirling. Our proof of decidability rests upon two semi-decision procedures: one for $\not\sim$, which is straightforward via the approximation technique for bisimilarity, and one for $\sim$ which relies on showing that the largest bisimulation relation is generated from a finite relation.

In Chapter 5 we shall consider the question of unique decomposition for the languages BPP and BPP_τ. After having presented the result of unique decomposition for bisimilarity on normed BPP due to Hirshfeld and Moller we extend the result to normed BPP_τ. Finally, we present the result of unique decomposition for distributed bisimulation equivalence on *all* of BPP and BPP_τ. Distributed bisimulation equivalence is a noninterleaving equivalence that takes into account the locality of processes. Therefore it might seem that the question of unique decomposition is an idle one: is it not the case that every process is its own unique decomposition wrt distributed bisimulation equivalence? We shall see that this is not the case; for instance, because of absorption laws (E is absorbed into F if $F + E = F$ where $+$ is a nondeterministic choice operator) unique decomposition plays an active rôle. However, to a certain extent the question of unique decomposition does degenerate wrt distributed bisimulation equivalence: every prime component of a process E is *contained* within E in a precise sense to be defined in Chapter 5.

In Chapter 6 we prove that bisimulation equivalence is decidable on the classes BPP and BPP_τ by appealing to tableau methods. We shall also discuss general conditions on the parallel operator for which a finite representability result for the largest bisimulation relation is achievable; as an example we consider the CSP related operator of $\|_A$. After having presented the tableau decision procedure we shall extract a sound and complete sequent-based equational theory for bisimulation equivalence on BPP and BPP_τ. Finally, we present a method very different from the tableau method for checking bisimilarity between *normed* processes of BPP_τ (as well as BPP). The method is inspired by Caucal [Cau90a] and delicately makes use of the unique decomposition result for bisimilarity on normed BPP_τ (and BPP). The method also opens up an analysis into the complexity of solving the problem, something we have not been able to perform wrt the tableau decision procedure.

In Chapter 7 we consider the questions addressed in Chapter 6 but now in the context of distributed bisimilarity. We prove that distributed bisimilarity is decidable on the two classes of BPP and BPP_τ. We present two different decision procedures: one is based on the tableau method, thus giving a very direct algorithm; and the other is based on the property of unique decomposition and is similar in style to the one we obtained for bisimilarity wrt normed processes. We shall also address the question of obtaining equational theories for distributed bisimilarity. We give a sound and complete equational theory for the class of BPP_τ (as well as BPP) processes extracted from the tableau decision procedure thus yielding a sequent-based theory. Finally, we provide an equational theory for distributed bisimilarity on a subclass of BPP_τ where general summation is replaced by *guarded summation*. Our theory is in style related to Milner's equational theory for regular CCS and relies on the property of unique decomposition admitted by distributed bisimilarity on BPP_τ.

In Chapter 8 we present a simplified version of Jančar's construction showing the undecidability of bisimilarity on labelled Petri nets. This is done by a reduction from the Halting problem (using two-counter machines). We then address the question of decomposing labelled Petri nets thereby obtaining a transformation from these nets into the calculus SPP. We shall finally discuss the difficulties by defining distributed bisimilarity on SPP.

In Chapter 9 we present a short conclusion of this thesis and give directions for further work.

Chapter 2

Background

In this chapter we lay the background for our work; we present various definitions and results that shall be used throughout the rest of this thesis. We define the model of *labelled transition graphs* as well as presenting a range of devices for generating labelled transition graphs such as *process algebras* and *labelled Petri nets*. We also define the semantic equivalences of *language equivalence*, *bisimulation equivalence* and *distributed bisimulation equivalence*. We begin with an exposition of the area of decidability thereby taking the opportunity to introduce central concepts and techniques.

2.1 Decidability

The *decision problem* for a property **P** concerns the existence of a mechanical test (a computational algorithm, an effective procedure) which applied to *any* object of the appropriate sort, *eventually* classifies that object correctly as a positive or a negative instance of that property. If such a mechanical test exists then we say that the property **P** is *solvable* or *decidable* or *recursive*, and the test is a *decision procedure* for the property **P**. If, on the other hand, no such test exists then we say that the property **P** is *unsolvable* or *undecidable*.

A *positive test* for **P** is a mechanical test which eventually classifies as positive all and only its positive instances. Similarly, a *negative test* is one which eventually classifies as negative all and only its negative instances. Clearly, if both a positive and a negative test for a property exist then, and only then, is the property solvable; for since any appropriate object will be either a positive or a negative instance, if both tests are available then they may be dovetailed to yield a decision procedure, and conversely, any test which correctly classifies any object as either a positive or a negative instance counts as both a positive and a negative test. If a property has either a positive or a negative test then we say that the positive or negative property (respectively) is *semi-decidable*.

In the following we shall discuss four main techniques for the establishment of the (semi-)decidability of a given property. Often a combination of these techniques is exploited in designing the decision algorithm. The first technique we discuss involves the method of *normal forms* which has been used in many areas. In general, the technique consists of transforming an instance I of a given property **P** into an equivalent normal form J which is tractable. Here equivalence between I and J simply means that I is a positive instance of **P** if, and only if, J is too.

In the area of logic, the technique of normal forms has been used to show some theories decidable. For instance, the theory of addition of natural numbers (Presburger arithmetic) was shown by Presburger [Pre29] to be decidable by transforming every formula into an equivalent one in normal form (not involving quantifiers and therefore also known as the technique of elimination of quantifiers) whose truth or falsehood can be directly ascertained. Another example from the area of logic involves the second-order theory of one successor (S1S). Here formulae are transformed into normal forms manageable by so-called Büchi automata and subsequently, the truth or falsehood of a formula is translated into an emptiness problem for the corresponding Büchi automaton (see [Buc62,Tho90]).

The technique of normal forms also plays an important rôle in the area of *replacement systems* such as semi-Thue systems, rewriting systems, graph grammars, the λ-calculus etc. The principal problem in this context is the *word problem*: given a replacement system and two objects, are the two objects equivalent, i.e. can one be transformed into the other by means of a finite number of applications of the rewrite rules? In general, this problem is undecidable (see for instance

[Jan88]). However, if the replacement system considered is canonical, i.e. confluent and well-founded, then irreducible objects may serve as normal forms and the word problem becomes decidable (by reducing to normal form and then comparing for identity).

A second technique shall be termed *generation from a recursive basis*. It is best illustrated by an example from the area of logic. Here it typically involves a (recursive) set of axioms AX for some theory T. If the axioms AX are complete for the theory then T becomes semi-decidable, i.e. if a formula A belongs to T then it can be asserted. For in these circumstances all members of T can be listed via an enumeration of all proofs as given by AX. Hence if $A \in T$ then a search through this list will eventually establish this fact.

In Chapter 4 of this thesis we shall see another example of the technique of generation from a recursive basis: we show that the bisimulation equivalence for the labelled transition graph determined by a system of context-free processes is generated from a finite (and hence recursive) set of pairs of bisimilar processes. This, as we shall see in Chapter 4, will lead to semi-decidability of bisimulation equivalence.

A third technique involves *reduction*, a method which has been used in many areas. Let **P** be a property known to be decidable and let **Q** be any property. Assume that we have a computable map from instances of **Q** to instances of **P** such that instances of **Q** are positive if, and only if, the corresponding (via the map) instances of **P** are positive. Under these circumstances **Q** has been *reduced* to **P** and decidability of **Q** follows directly from the decidability of **P**.

We have already mentioned one example of the method of reduction; the decidability of S1S is obtained via a reduction (using normal form as an intermediate step) to the emptiness problem for Büchi automata which is known to be decidable. Similarly, the second-order theory of two successors (S2S) can be shown to be decidable by a reduction to the emptiness problem for Rabin tree automata [Rab69]. Subsequently, a large number of theories of mathematical logic have been shown to be decidable via a reduction to S2S; we mention the monadic second-order theory of countable orderings [Tho90] and the monadic second-order theory of context-free graphs [MS85].

The method of reduction is also a powerful tool for proofs of undecidability. For if the property **Q** is known to be undecidable and it is reduced to **P** in a manner described above then the latter property must be undecidable as well. This aspect of the technique has been used to show a large number of properties from many areas to be undecidable. Almost all undecidability results may thus be rooted back to a few fundamental undecidable properties such as the Halting problem for Turing machines.

The last technique we wish to discuss involves the use of *tableau systems*. Such a system may be seen as a *goal directed* proof method; the goal of deciding an instance I of some property **P** is replaced by a number of sufficient subgoals $\mathrm{I}_1, \mathrm{I}_2 \ldots \mathrm{I}_n$ each of which in turn is replaced with new subgoals. The procedure for deciding I may thus be visualised as a labelled tree with root labelled I and nodes labelled with instances of subgoals.

Tableau systems arose in the area of logic with the work of Beth [Bet59] and Smullyan [Smu68] but have their origin in Gentzen [Gen35] as "upside down" Gentzen sequent systems. They have been used extensively for validity-checking in modal and intuitionistic logics as well as being templates for completeness proofs (see [Fit83]). The tableau systems considered in this area are *refutation* systems: a tableau construction is an attempt to refute a given formula; if it fails, the formula intuitively should be valid.

In more recent work Stirling has advocated the use of tableaux for showing relative truth as in model checking (see e.g. [SW89]). In Stirling's terminology tableaux are no longer refutations; a tableau construction is now an attempt to establish (directly) the property of interest. An advantage of the tableau method (which we shall rely on) is that it is an appropriate device for handling fixed points.

The tableau technique has recently been utilised by Hüttel and Stirling in the area of process algebras; they provided a tableau system to obtain decidability of bisimilarity for normed context-free processes [HS91]. Previous to this result there was a proof of the same result by Caucal [Cau90] using a very different technique. Caucal's technique may be characterised as belonging to the method of normal forms; the bisimilarity problem is transformed into a question of the word problem for a canonical semi-Thue system. Originally, the proof of decidability of bisimilarity for normed context-free processes was solved by Baeten, Bergstra and Klop [BBK87] using a technique very different from the two already mentioned; it may be characterised as using the method of generation from a recursive basis since they show that the transition graph for any normed context-free process is finitely generated.

Such questions of decidability in concurrency theory may be viewed as a natural continuation of similar questions in automata and formal language theory. The interest centres around various models of computation being more expressive than usual finite-state automata (such as context-free grammars, push-down automata, Turing machines and Petri nets), and the decidability of various behavioural equivalences including language equivalence but also stronger notions of equivalences which take into account for instance the notions of deadlock, livelock or causality.

Our work may be seen as a contribution to this field; we shall be looking at various models of computation (focusing on *parallelism*) encompassing models matching context-free grammars, Petri nets, Turing machines, and others in expressive power. We shall present a number of decidability results (and also some undecidability results) mainly for language equivalence, bisimulation equivalence and distributed bisimulation equivalence. The techniques which we use for our decidability results consort with the method of normal forms, the method of generation from a recursive basis and with the tableau method.

Our motivation for the study of such decidability questions is not initially manifested in any desire for practical applications. Often the algorithms considered consist of two semi-decision procedures (one for the positive test and one for the negative test) and therefore the complexity of solving the problems is not obvious to determine. In the circumstances where we are able to perform a complexity analysis we shall often not pursue it. Rather our motivations are: to draw boundaries between decidable and undecidable problems, and to investigate the properties of the models considered since a decidability result rests upon a deeper understanding of the structure of the model explored. In a broader perspective our work should be seen as part of the research into the question of whether there is a decidable general (non-trivial) theory of computation.

2.2 Labelled Transition Graphs

Underpining the fundamental concepts of *state* and *change of state* as for instance manipulated by a computer program is the notion of a *labelled transition graph* [Kel76], a commonly understood model of computation providing the basic operational semantics for a variety of languages including Milner's CCS [Mil89].

Definition 2.1 *A* labelled transition graph *is a triple* $\big(P, A, \{\xrightarrow{a} : a \in A\}\big)$ *consisting of a set of* states *or* processes P*, a set of* labels A *and a family of binary* transition relations $\{\xrightarrow{a}: a \in A\}$ *over* P*.*

For a labelled transition graph $\big(P, A, \{\xrightarrow{a} : a \in A\}\big)$, if $(E, F) \in \xrightarrow{a}$ we write $E \xrightarrow{a} F$ denoting the change of state from E to F under the performance of the action $a \in A$. If for all $a \in A$ there is no F such that $E \xrightarrow{a} F$ we write $E \not\rightarrow$. The reflexive, transitive closure $\{\xrightarrow{u}: u \in A^*\}$ of the transition relations is defined by $E \xrightarrow{\epsilon} E$ for all E (here and throughout the rest of this thesis we shall use ϵ to denote the empty string) and $E \xrightarrow{vw} F$ for $v, w \in A^*$ iff $E \xrightarrow{v} E'$ and $E' \xrightarrow{w} F$ for some $E' \in P$. If $E \xrightarrow{u} E'$ for some $u \in A^*$ we call E' a *successor state* of E.

Definition 2.2 *Let* $\big(P, A, \{\xrightarrow{a} : a \in A\}\big)$ *be a labelled transition graph and suppose* $E \in P$*. The language generated by* E*, denoted* $L(E)$*, is*

$$L(E) \quad = \quad \Big\{w \in A^* : E \xrightarrow{w} E' \not\rightarrow \textit{ for some process } E'\Big\}.$$

We say that two processes E *and* F *are* language equivalent*, written* $E =_L F$*, iff* $L(E) = L(F)$*.*

Notice that our definition of $L(E)$ is but one out of a number of possibilities. For instance, we could have considered only those words from E leading to a state belonging to a set of final states as is common in automata theory (see [HU79]).

We could also have allowed for the inclusion of infinite words as in the work of [Tho90].

One important implication of our choice will be that *non-terminating* states (states E for which there is no u, E' such that $E \xrightarrow{u} E'$ with $E' \not\rightarrow$) cannot contribute to the generation of languages. In contrast, a state E is *terminating* (or *normed*) if there exist u, E' such that $E \xrightarrow{u} E'$ with $E' \not\rightarrow$. A particular subclass of the terminating states are those states for which any successor state is terminating; such states are called *strongly terminating*. When considering language equivalence we may restrict attention to normed transition graphs, i.e. graphs for which any state is normed (and therefore also strongly terminating). This is a situation somewhat similar to the removal of useless productions in grammars generating languages (see [HU79]).

Language equivalence is rather coarse, i.e. it identifies many processes. Of greater interest in process theory are (questions regarding) stronger notions of equivalences which take into account for instance the notion of deadlock, livelock or causality. In particular *bisimulation equivalence*, as defined by Milner and Park (see e.g. [Mil80,Par81]), has been extensively studied and now forms the foundation for the study of CCS-like languages [Mil89].

Definition 2.3 *Let* $\left(P, A, \{\xrightarrow{a} : a \in A\}\right)$ *be a labelled transition graph. A binary relation* R *over* P *is a* bisimulation *if whenever* $(E, F) \in R$ *then for each* $a \in A$,

- *if* $E \xrightarrow{a} E'$ *then* $F \xrightarrow{a} F'$ *for some* F' *with* $(E', F') \in R$, *and*
- *if* $F \xrightarrow{a} F'$ *then* $E \xrightarrow{a} E'$ *for some* E' *with* $(E', F') \in R$.

Processes E *and* F *are* bisimilar, *written* $E \sim F$, *if they are related by some bisimulation.*

As shown in [Mil89] $\sim$ is the largest bisimulation and it is an equivalence relation, i.e. it satisfies the laws of reflexivity, symmetry and transitivity. Since $\sim$ is the largest bisimulation, when proving that two processes E and F are bisimilar it suffices to exhibit a bisimulation relating E and F.

A special subclass of labelled transition graphs in which we shall be interested are those which are *deterministic* in the following sense (taken from Milner [Mil89] and thus different from the classical notion of determinism [HU79]).

Definition 2.4 *A labelled transition graph* $\left(P, A, \{\xrightarrow{a} : a \in A\}\right)$ *is* deterministic *(up to bisimilarity) iff for all states* E *of* P *it follows that* $E \xrightarrow{a} F$ *and* $E \xrightarrow{a} G$ *implies* $F \sim G$.

We say that a state or process E is deterministic if it belongs to a deterministic labelled transition graph. Now it is a routine matter to show that the relation given by

$$\Big\{(E,F) : E =_L F \text{ and } E, F \text{ strongly terminating and deterministic}\Big\}$$

is a bisimulation relation. Since bisimilarity is contained in language equivalence we therefore conclude that bisimulation equivalence and language equivalence *coincide* on normed deterministic transition graphs. We shall return to this simple but important observation on a number of occasions throughout this thesis; for instance the techniques available for deciding bisimilarity become valid for language equivalence on normed deterministic transition graphs (and vice versa).

Certainly bisimilarity is decidable over finite-state transition graphs, i.e. transition graphs $\left(P, A, \{\xrightarrow{a} : a \in A\}\right)$ for which P is finite. For in these circumstances we may enumerate all finitely many binary relations over the state set P and search for a bisimulation among them containing the pair (E, F) that we wish to compare. If we find a bisimulation containing (E, F) then we conclude that $E \sim F$, and conversely, if $E \sim F$ then we will eventually find a bisimulation containing E and F.

If the labelled transition graphs considered have infinitely many states then the naive procedure described above for deciding bisimilarity does not work any more since there can be infinitely many binary relations and also bisimulations with infinitely many pairs. Hence, when dealing with transition graphs potentially having infinitely many states, deciding bisimilarity (if possible) requires other approaches. We shall in the following three subsections introduce a number of devices for generating labelled transition graphs, including graphs with infinitely many states.

2.2.1 Processes

We shall define a general language for states of labelled transition graphs together with rules for generating transitions. Our language will subsume known algebras such as BPA but will also contain a large number of CCS operators.

We presuppose a countably infinite set of *labels* $\Lambda_1 = \{a, b, c \ldots\}$ as well as a countably infinite set of *co-labels* $\bar{\Lambda}_1 = \{\bar{a}, \bar{b}, \bar{c} \ldots\}$ with the convention that $\bar{\bar{a}} = a$. Elements of $\Lambda_1 \cup \bar{\Lambda}_1$ are called *atomic actions* with a and $\bar{a}$ being *complementary* actions. Let τ be a distinguished element not in $\Lambda_1 \cup \bar{\Lambda}_1$ and define the set of *actions* Act to be $\Lambda_1 \cup \bar{\Lambda}_1 \cup \{\tau\}$. We extend the notation for complementary labels by decreeing that $\bar{\tau} = \tau$. Let $\mu, \nu \ldots$ range over Act. Finally, we assume a countably infinite set of *process variables* $Var = \{X, Y, Z \ldots\}$.

Our *process expressions* are given according to the following abstract syntax equations.

$$
\begin{array}{rcll}
E & ::= & \mathbf{0} & \text{(inaction)} \\
 & | & X & \text{(process variable, } X \in Var\text{)} \\
 & | & \mu E & \text{(action prefix, } \mu \in Act\text{)} \\
 & | & op(E, E) & \text{(binary process operator)}
\end{array}
$$

In general we let $E, F, G \ldots$ denote process expressions. The operator op shall range over the following binary operators all of which will be written in infix form.

- $(+)$: summation which given processes E and F returns $E + F$, a process that behaves either like E or like F.

- $(\cdot)$: sequential composition which given processes E and F returns $E \cdot F$, a process that behaves like E until termination whereupon it behaves like F.

- $(\|)$: merge which given processes E and F returns $E \| F$, a process that is capable of performing the actions of E and F, in the order which E and F would have performed them in, in an arbitrary interleaved fashion.

- ($\|\!\lfloor$) : left merge which given processes E and F returns $E\|\!\lfloor F$, a process that behaves like $E\|F$ however under the constraint that the first action observed must emanate from E.

- ($|$) : parallel composition which given processes E and F returns $E|F$, a process that behaves like $E\|F$ but also allows for E and F to synchronise on complementary actions; one offered by E, the other by F. The result of a synchronisation will be denoted τ.

- ($\lfloor$) : left merge which given processes E and F returns $E\lfloor F$, a process that behaves like $E|F$ however under the constraint that the first action observed must emanate from E.

Finally, to completely describe the informal behaviours of the different operators, $\mu \in Act$ is a unary operator which given a process E returns μE, a process that can perform μ and thereby evolve into a process that behaves like E. And $\mathbf{0}$ is a nullary operator which represents a process that is inactive, i.e. no actions can be observed and the process is always terminated.

We shall use parentheses to resolve ambiguities. However, to avoid too many parentheses we adopt the convention that the prefix operator and the sequential operator take precedence over the left merge operators which in turn take precedence over the operators of merge and parallel composition. Finally, the summation operator is preceded by all the other operators. Furthermore, to ease presentation we shall omit the symbol for sequential composition thus writing EF for $E \cdot F$ and also omit trailing $\mathbf{0}$'s from expressions, thus writing $\mu\mathbf{0}$ simply as μ. Finally, we shall write E^m to represent the expression given by combining m copies of E using either sequential composition, merge or parallel composition (which will be clear from the context).

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

$$\Delta \;=\; \left\{X_i \stackrel{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\right\},$$

where the X_i are distinct variables and the E_i are process expressions at most containing the variables $Var(\Delta) = \{X_1, X_2 \ldots X_n\}$. The variable X_1 is singled out as the *leading variable* and $X_1 \stackrel{\mathrm{def}}{=} E_1$ is called the *leading equation*. We shall restrict our attention to *guarded* systems of process equations.

Definition 2.5 *A process expression E is* guarded *iff every variable occurrence of E is within the scope of an action prefix. The family $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\}$ is guarded iff each E_i is guarded for $i = 1, 2 \ldots n$.*

Recursion, and therefore the possibility to describe infinite computations, is given by process variables and defining equations. In this respect we follow the tradition of the ACP-based algebras [BW90]. Recursion may also be defined by *recursive expressions* thus separating recursion from the definition facility. An example of this is CCS [Mil89] which includes a *fixed point* operator *fix*, that given a process E possibly involving a variable X, returns $fixX.E$ meaning that X (in E) is bound to E and thereby allowing for recursion; the effect of $fixX.E$ is similar to the introduction of the equation $X \stackrel{\mathrm{def}}{=} E$. We have chosen to work with defining equations rather than explicit recursive operators since it suits our purposes better. However, in Section 2.5.3 (and also in Chapter 7) we shall take the opportunity to introduce recursion by fixed point operators in a formal manner since we shall be looking at equational theories expressed via such an operator.

As has become standard in process calculi we shall present the semantics of our process language in the style of *structural operational semantics* (thus yielding labelled transition graphs), a natural method of presenting such semantics first proposed in [Plo81].

Any finite family Δ of guarded process equations determines a labelled transition graph; states are process expressions (with free variables over $Var(\Delta)$), the set of labels is given by Act and the transition relations are given as the least relations derived from the rules of Table 2–1. The rules are specified in an inferential style; if the sentence(s) mentioned above a line hold(s), then the sentence below the line must also hold. Strictly speaking, transitions are defined on process expressions relative to some family Δ and thus should be subscripted accordingly. However,

$$\frac{}{\mu E \xrightarrow{\mu} E} \qquad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'} \qquad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\frac{E \xrightarrow{\mu} E'}{X \xrightarrow{\mu} E'}(X \stackrel{\text{def}}{=} E \in \Delta) \qquad \frac{E \xrightarrow{\mu} E'}{EF \xrightarrow{\mu} E'F} \qquad \frac{isnil(E), F \xrightarrow{\mu} F'}{EF \xrightarrow{\mu} F'}$$

$$\frac{E \xrightarrow{\mu} E'}{E\|F \xrightarrow{\mu} E'\|F} \qquad \frac{F \xrightarrow{\mu} F'}{E\|F \xrightarrow{\mu} E\|F'} \qquad \frac{E \xrightarrow{\mu} E'}{E \lfloor\!\lfloor F \xrightarrow{\mu} E'\|F}$$

$$\frac{E \xrightarrow{\mu} E'}{E|F \xrightarrow{\mu} E'|F} \qquad \frac{F \xrightarrow{\mu} F'}{E|F \xrightarrow{\mu} E|F'} \qquad \frac{E \xrightarrow{\mu} E'}{E \lfloor F \xrightarrow{\mu} E'|F}$$

$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{\bar{a}} F'}{E|F \xrightarrow{\tau} E'|F'}$$

Table 2–1: Transition rules.

we shall leave the reader to infer the intended family.

For the presentation of the rules for sequential composition we introduce the predicate $isnil(E)$ indicating whether or not E is equivalent to the process $\mathbf{0}$ (wrt bisimilarity). For our calculus we can easily compute $isnil(E)$ by cases on the structure of E.

Definition 2.6 *The predicate isnil(E) is defined by cases on the structure of the process E as follows:*

- $isnil(\mathbf{0}) = true$,
- $isnil(\mu E) = false$,
- $isnil(E \lfloor\!\lfloor F) = isnil(E \lfloor F) = isnil(E)$,
- $isnil(E + F) = isnil(EF) = isnil(E\|F) = isnil(E|F) = isnil(E) \wedge isnil(F)$,
- $isnil(X) = isnil(E)$ *where* $X \stackrel{\text{def}}{=} E \in \Delta$.

Having interpreted processes as labelled transition graphs we may begin to investigate bisimilarity between processes by considering them as states of the corresponding transition graphs. For instance, we can now check that the predicate *isnil*() has the intended meaning.

Proposition 2.7 *$E \sim \mathbf{0}$ if, and only if, isnil(E).*

Proof: Straightforward by induction on the structure of E. ■

We shall rely on the well-established theory for bisimulation equivalence (see [Mil89]). For instance, bisimulation equivalence is a congruence relation wrt all the operators considered here, i.e. it is substitutive under equality (as given by bisimilarity). Moreover, any finite family Δ of guarded process equations has a unique solution up to bisimilarity. Notice that the restriction to guarded processes is essential for uniqueness of solutions to equations. Furthermore, by only considering guarded equations it is easy to verify that our processes generate *image-finite* transition graphs, i.e. graphs for which the set $\{F : E \xrightarrow{\mu} F\}$ is finite for each E and μ. This property is important for our later results on the decidability since it will provide us with semi-decidability of bisimulation inequivalence (see Chapter 4). Notice that if we allowed unguarded expressions we would not have image-finiteness; for example the process $X \stackrel{\text{def}}{=} a + a \| X$ generates a transition graph which is not image-finite.

In order to simplify our later analysis, we wish to identify several process expressions. A typical case is that we want $+$ as well as the parallel operators to be commutative and associative. We therefore define the following structural congruence over process expressions.

Definition 2.8 *Let $\equiv$ be the smallest congruence relation over process expressions such that the following laws hold:*

- *associativity and $\mathbf{0}$ as unit for sequential composition, and*
- *associativity, commutativity and $\mathbf{0}$ as unit for summation, merge and parallel composition.*

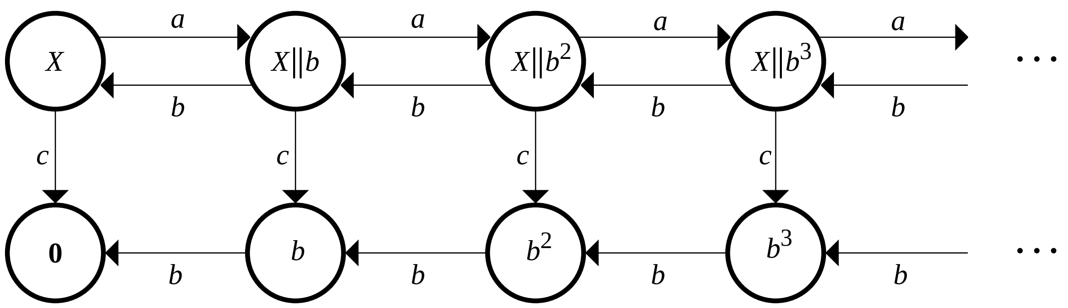

Figure 2–1: The transition graph for $\{X \stackrel{\mathrm{def}}{=} a(X\|b) + c\}$.

When inferring transitions we may ignore harmless **0**-components sitting in parallel. Thus not to be annoyed by such innocent matters we shall always assume that transitions have been inferred modulo the structural congruence $\equiv$. That is, we shall assume that transitions are inferred modulo the following inference rule of congruence.

$$\frac{E \equiv E' \quad E' \stackrel{\mu}{\longrightarrow} F' \quad F' \equiv F}{E \stackrel{\mu}{\longrightarrow} F}$$

We note that we can safely do so since any semantic equivalence that we are interested in satisfies the basic laws underlying $\equiv$.

Example 2.9 *Let Δ be the family $\{X \stackrel{\mathrm{def}}{=} a(X\|b) + c\}$. By the transition rules of Table 2–1 together with the congruence rule above, X generates the infinite-state transition graph of Figure 2–1.*

An important subclass of processes which we shall explore in later chapters consists of those processes being *normed.* As we have explained earlier, a process (state) is normed if it may terminate in a finite number of steps. The *norm* of a process is the least number of steps required for termination. Since we want the norm to be additive under the parallel composition operator (as well as the sequential operator and the merge operator) we must take care in defining it. For instance, both a and $\bar{a}$ have norm equal to one but $a|\bar{a}$ also has norm one since it may terminate in a single (synchronisation) step. We shall overcome the problem simply by counting τ-steps as two when computing the norm.

Definition 2.10 *Let length* : $Act^* \rightarrow I\!N$ *be a length function on strings defined by setting* $length(\epsilon) = 0$ *and*

$$length(\mu w) \;=\; \begin{cases} 1 + length(w) & \text{if } \mu \neq \tau \\ 2 + length(w) & \text{otherwise.} \end{cases}$$

The norm *of a process* E*, denoted* $\mathcal{N}(E)$*, is defined to be the least number of transitions (counting* τ*-steps as two) necessary to reach a deadlocked state. Formally, we have*

$$\mathcal{N}(E) \;=\; \min\left\{ length(w) : E \xrightarrow{w} E' \not\rightarrow, w \in Act^* \right\}.$$

A process E *is* normed *if* $\mathcal{N}(E) < \infty$*. If a process does not have finite norm then we say that it is* unnormed*.*

One important property of the norm of processes that we shall be using (especially in Chapter 5 on the subject of unique decomposition) is the property of being additive under sequential composition as well as under the parallel operators. This is expressed by the following proposition.

Proposition 2.11 *Let* E *and* F *be processes. We have* $\mathcal{N}(EF) = \mathcal{N}(E) + \mathcal{N}(F)$*,* $\mathcal{N}(E \| F) = \mathcal{N}(E) + \mathcal{N}(F)$ *as well as* $\mathcal{N}(E|F) = \mathcal{N}(E) + \mathcal{N}(F)$*.*

Proof: Certainly, we can assume that both E and F are normed; otherwise there is nothing to prove. We shall only consider proving that $\mathcal{N}(E|F) = \mathcal{N}(E) + \mathcal{N}(F)$ since the other cases follow from similar (but simpler) arguments.

Suppose $\mathcal{N}(E) = n$ and $\mathcal{N}(F) = m$. That is, there exists $v \in Act^*$ and process E' such that $E \xrightarrow{v} E' \not\rightarrow$ with $length(v) = n$ and there exists $w \in Act^*$ and process F' such that $F \xrightarrow{w} F' \not\rightarrow$ with $length(w) = m$. Thus $E|F \xrightarrow{v} E'|F \xrightarrow{w} E'|F' \not\rightarrow$ with $length(vw) = n + m$ and therefore $\mathcal{N}(E|F) \leq \mathcal{N}(E) + \mathcal{N}(F)$.

Suppose $E|F \xrightarrow{u} E'|F'$ for some $u \in Act^*$ and for some processes E' and F'. Then from u we can extract v and w (in the obvious way) such that $E \xrightarrow{v} E'$ and $F \xrightarrow{w} F'$. By induction on $length(u)$ we show $length(u) = length(v) + length(w)$.

If $length(u) = 0$ then certainly $u = v = w = \epsilon$ and therefore it follows that $length(u) = length(v) + length(w)$ as required. Thus assume that $length(u) \geq 1$.

Then for some $\mu \in Act$ and $u' \in Act^*$ we conclude that $u = \mu u'$. There are three cases to consider:

(i) Suppose $E|F \xrightarrow{\mu} E''|F \xrightarrow{u'} E'|F'$ for some process E'' and some $v' \in Act^*$ such that $E \xrightarrow{\mu} E'' \xrightarrow{v'} E'$ and $F \xrightarrow{w} F'$ with $v = \mu v'$. From the induction hypothesis we conclude that $length(u') = length(v') + length(w)$ and therefore $length(u) = length(\mu) + length(u') = length(\mu) + length(v') + length(w) = length(v) + length(w)$ as required.

(ii) Suppose $E|F \xrightarrow{\mu} E|F'' \xrightarrow{u'} E'|F'$ for some process F'' and some $w' \in Act^*$ such that $E \xrightarrow{v} E'$ and $F \xrightarrow{\mu} F'' \xrightarrow{w'} F'$ with $w = \mu w'$. Then the arguments are as in the above case.

(iii) Suppose $\mu = \tau$ and that $E|F \xrightarrow{\tau} E''|F'' \xrightarrow{u'} E'|F'$ for some processes E'', F'' and some a, v', w' of Act^* such that $E \xrightarrow{a} E'' \xrightarrow{v'} E'$ and $F \xrightarrow{\bar{a}} F'' \xrightarrow{w'} F'$ with $v = av'$ and $w = \bar{a}w'$. From the induction hypothesis we conclude that $length(u') = length(v') + length(w')$, and therefore it follows that $length(u) = 2 + length(u') = 2 + length(v') + length(w') = length(v) + length(w)$ as required.

Thus whenever $E|F \xrightarrow{u} E'|F' \not\rightarrow$ for some $u \in Act^*$ we have $length(u) \geq n + m$ and that completes the proof. ■

We now proceed by defining some subclasses of processes that shall have our attention throughout the rest of this thesis. The first class of processes is commonly known as Regular CCS.

Definition 2.12 *The class of* RCCS (Regular CCS) *process expressions consists of those expressions built over Var using only the operators of* $\mathbf{0}, \{a : a \in \Lambda_1\}$ *and* $(+)$. *A family of guarded process expressions* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ *defines* RCCS *processes iff each* E_i *is a guarded* RCCS *expression for* $i = 1, 2 \ldots n$.

RCCS corresponds to ordinary finite-state automata from formal language theory and has a well-established theory. For instance, bisimilarity (as well as any other studied equivalence) is decidable over RCCS and complete equational theories for various equivalences have been developed (see e.g. [Mil84,Rab92]). RCCS may be seen as our starting point; the motivation for our work has been to extend (parts of) the theory for RCCS beyond the case of finite-state processes. This has led to a focus on the following process classes.

Definition 2.13 *The class of recursive* $\mathrm{BPA}_{\mathbf{0}}$ (Basic Process Algebra with $\mathbf{0}$) *expressions consists of those process expressions built over Var using only the operators of* $\mathbf{0}, \{a : a \in \Lambda_1\}, (+)$ *and* $(\cdot)$. *A family* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ *of guarded process expressions defines* $\mathrm{BPA}_{\mathbf{0}}$ *processes iff each* E_i *is a guarded* $\mathrm{BPA}_{\mathbf{0}}$ *expression for* $i = 1, 2 \ldots n$.

Our calculus $\mathrm{BPA}_{\mathbf{0}}$ is *mutatis mutandis* the Dutch calculus BPA, commonly known as the class of context-free processes. The only differences concern our inclusion of the inactive process $\mathbf{0}$ as well as viewing $\{a : a \in \Lambda_1\}$ as operators; in BPA each $a \in \Lambda_1$ would represent a process, namely the process $a\mathbf{0}$. The only reason for including $\mathbf{0}$ and viewing $\{a : a \in \Lambda_1\}$ as operators is due to the fact that we thereby can make our exposition uniform, introducing a whole range of process classes within a general framework. Certainly regarding expressiveness there is no difference between BPA and $\mathrm{BPA}_{\mathbf{0}}$ and often we shall refer to $\mathrm{BPA}_{\mathbf{0}}$ as the class of context-free processes as well. However, we shall here mention one implication caused by adding $\mathbf{0}$ to BPA and that involves the search for an equational theory for the recursion-free fragment of $\mathrm{BPA}_{\mathbf{0}}$. From our understanding of the $\mathbf{0}$ process we certainly want a law like $E + \mathbf{0} = E$ to be valid. However, this in connection with the usual BPA distributivity of $\cdot$ over $+$ on the right, i.e. the law $(E + F)G = EG + FG$, gives for instance

$$ab \;=\; (a + \mathbf{0})b \;=\; ab + \mathbf{0}b \;=\; ab + b$$

which is certainly not the intention. In [Mol89] (which we refer to for a general discussion of the implications caused by adding the $\mathbf{0}$ process to BPA) these difficulties are dealt with by abandoning the general distributivity of $\cdot$ over $+$ on the right (requiring E and F in $(E + F)G = EG + FG$ to be different from $\mathbf{0}$) when designing an equational theory for the recursion-free fragment of $\mathrm{BPA}_{\mathbf{0}}$.

The other two classes of processes that we shall be interested in are BPP and BPP_τ defined as follows.

Definition 2.14 *The class of recursive* BPP (Basic Parallel Processes) *expressions consists of those process expressions built over Var using only the operators of* $\mathbf{0}, \{a : a \in \Lambda_1\}, (+), (\lfloor\lfloor)$ *and* $(\|)$. *Similarly, the class of recursive* BPP_τ (Basic Parallel Processes with τ) *expressions consists of those process expressions built over Var using only the operators of* $\mathbf{0}, \{\mu : \mu \in Act\}, (+), (\lfloor)$ *and* $(|)$. *A family of guarded process expressions* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ *defines* BPP (BPP_τ) *processes iff each* E_i *is a guarded* BPP (BPP_τ) *expression for* $i = 1, 2 \ldots n$.

Notice that the only difference between BPP and BPP_τ concerns parallelism; the pure interleaving operator $\|$ of BPP has been replaced by $|$ in BPP_τ thus allowing for processes to synchronise. In Chapter 3 we shall explore the class of languages generated by BPP processes; in particular, we shall show that this class is incomparable with the class of context-free languages, or equivalently, the class of languages generated by $\mathrm{BPA}_\mathbf{0}$ processes.

Of particular interest are the classes of *normed* $\mathrm{BPA}_\mathbf{0}$ $(\mathrm{BPP}, \mathrm{BPP}_\tau)$ processes. We say that a family Δ of process equations is normed iff every variable of $Var(\Delta)$ is normed, and by normed $\mathrm{BPA}_\mathbf{0}$ $(\mathrm{BPP}, \mathrm{BPP}_\tau)$ processes we then refer to processes defined by normed families of $\mathrm{BPA}_\mathbf{0}$ $(\mathrm{BPP}, \mathrm{BPP}_\tau)$ process equations. Example 2.9 shows a normed BPP process with $\mathcal{N}(X) = 1$. Notice that for normed $\mathrm{BPA}_\mathbf{0}$ $(\mathrm{BPP}, \mathrm{BPP}_\tau)$ any reachable state is normed. Hence these processes are not only normed but also strongly terminating.

We have left out a number of operators often seen in process algebras. Notably, the unary abstraction operators such as *restriction* and *relabelling* of CCS or *hiding* of CSP have been left out. There is a clear reason for this; some of our goals such as decidability of bisimulation equivalence would be impossible to achieve if our collection of process operators were extended with these operators. In fact, just extending BPP_τ with restriction makes bisimilarity undecidable as we shall demonstrate in Section 2.3 following the proof of Taubner (see [Tau89]).

We have also left out the CSP parallel operator denoted $\|$ in [Hoa85] (we shall denote it by $|||$ in order to distinguish it from full merge) as well as the related operator $\|_A$ where A is a set of labels. In [Hoa85] each process essentially consists of a pair (E, A) of a process expression E and a label set A such that the sort of E (the labels appearing in the description of E) is contained in A. The expression $(E, A)|||(F, B)$ stands for a process that executes E and F in parallel but under the constraint that E and F *must* synchronise on actions from $A \cap B$. And $\|_A$ is a parallel operator for the composition of (ordinary) processes; the expression $E\|_A F$ stands for the parallel execution of E and F under the constraint that E and F *must* synchronise on actions from A. In some sense the CSP operator $|||$ is a special case of $\|_A$ as the latter allows for the set of "synchronisation" actions to be chosen freely.

Since the operators of $|||$ and $\|_A$ include the possibility to *force* synchronisation between processes they are more expressive than the parallel operators of full merge and CCS parallel composition. We shall in Chapter 6 discuss the parallel operator $\|_A$ in relation to decidability of bisimilarity; in particular, we shall demonstrate decidability under certain (nontrivial) conditions imposed on the label set A and also demonstrate that bisimilarity is undecidable in the general case.

2.2.2 Labelled Petri Nets

In this thesis we are also interested in the model of *labelled place-transition nets* [Rei85], also commonly known as the model of *labelled Petri nets*. We shall mainly be interested in the model from the point of view of generating labelled transition graphs; it may as such be seen as a natural extension of BPP and BPP_τ regarding expressiveness; this shall be demonstrated in Chapter 3. By a *labelled Petri net system*, or simply a *net system*, we understand the following.

Definition 2.15 *A* Petri net $N = (S, T; F)$ *consists of two disjoint non-empty finite sets* S *(the* places*) and* T *(the* transitions*) together with a* flow relation $F : (S \times T) \cup (T \times S) \to \mathbb{N}$. *A* marking *for a net* N *is a map* $M : S \to \mathbb{N}$ *and a labelling for* N *is a map* $l : T \to A$ *where* A *is a set of* labels. *Labelling maps shall* *be extended to T^* in the usual way. Finally, a* labelled Petri net system *is a tuple (N, l, M_0) where N is a Petri net (called the* underlying *net), l is a labelling for N and M_0 is an initial marking for N.*

We shall denote nets by N and net systems by NS possibly subscripted. We use $p, p_0, p_1 \ldots$ to range over places and $t, t_0, t_1 \ldots$ to range over transitions with $x, y \ldots$ ranging over both places and transitions. Labelling maps are ranged over by $l, l_0, l_1 \ldots$ and finally $M, M_0, M_1 \ldots$ range over markings. In the following we assume that labels are drawn from Act, i.e. the set of actions defined in Section 2.2.1, and we denote by PN the class of such labelled Petri net systems.

Definition 2.16 *Let $N = (S, T; F)$ be a net. For $x \in S \cup T$, we let ${}^\bullet x$ denote the map $S \cup T \to I\!N$ given by ${}^\bullet x(y) = F(y, x)$, and similarly $x^\bullet(y) = F(x, y)$. Sometimes the notation ${}^\bullet x$ will be abused to mean the set $\{y : {}^\bullet x(y) > 0\}$ of* pre-places (pre-transitions) *of x and similarly for* post-places (post-transitions).

A net $N = (S, T; F)$ is commonly represented in graphical form by drawing circles for places, boxes for transitions and an arc from place p to transition t (or from transition t to place p) labelled $F(p, t)$ (or $F(t, p)$) which is called the *weight* for the arc; omitted weights are always assumed to represent unity. A net system $NS = (N, l, M_0)$ is represented graphically by drawing the net N, writing $l(t)$ inside the box for every transition t and by placing a number of *tokens* (represented by dots) corresponding to $M_0(p)$ inside the circle for every place p.

Example 2.17 *Let the net system $NS = (N, l, M_0)$ be given as follows. The underlying net N is $\big(\{p_1, p_2\}, \{t_{11}, t_{12}, t_{21}\}; F\big)$ with $F(p_1, t_{11}) = 1$, $F(t_{11}, p_1) = 1$, $F(t_{11}, p_2) = 1$, $F(p_2, t_{21}) = 1$, $F(t_{21}, p_2) = 2$ and $F(p_1, t_{12}) = 1$ (all other cases have zero weight). The labelling map is given as follows: $l(t_{11}) = a$, $l(t_{21}) = b$ and $l(t_{12}) = a$. Finally, the initial marking is given by $M_0(p_1) = 1$ and $M_0(p_2) = 0$. The graphical representation of the net system NS is presented in Figure 2–2.*

The *behaviour* or *token game* for a net system may be informally described as follows. A transition can *fire* if it is *enabled*. A transition is enabled if each of

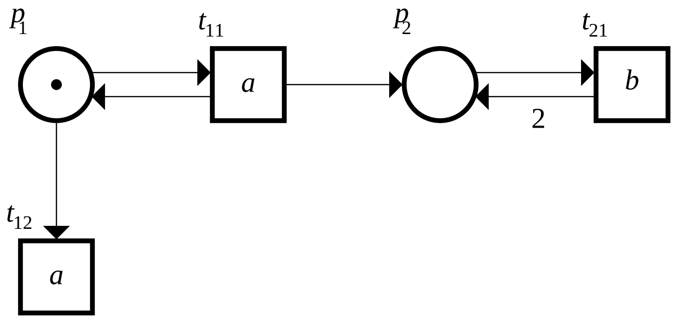

Figure 2–2: The net system of Example 2.17.

its *input places* (those with arcs pointing to the transition) contains at least the number of tokens specified by the weight of the corresponding arc. The firing of a transition t *changes* the marking by removing from each input place of t the number of tokens specified by the weight of the corresponding arc and then adding to each *output place* of t (those places that are pointed to by arcs from the transition) a number of tokens specified by the weight of the corresponding arc. In the next definition, formalising the token game, we use the notation of $\leq$, $-$ and $+$ on maps; these operations are assumed to be defined with their obvious meanings.

Definition 2.18 *Let $N = (S, T; F)$ be a net. A transition t is* enabled at *M, iff ${}^\bullet t \leq M$. We say M' is a* successor marking *of M via t, written $M[t{>}M'$ iff t is enabled at M and $M' = M - {}^\bullet t + t^\bullet$. The marking M' is* reachable *from M iff there exist a sequence $M = M_1, t_1, M_2, t_2 \ldots t_{n-1}, M_n = M'$ such that M_{i+1} is a successor marking of M_i via t_i (for $i = 1, 2 \ldots n-1$). By $R(M)$ we define the set of reachable markings from M.*

The reflexive, transitive closure $M[u{>}M'$, where $u \in T^*$, is defined by $M[\epsilon{>}M$ for every marking M and $M[vw{>}M'$ for $v, w \in T^*$ iff $M[v{>}M''$ and $M''[w{>}M'$ for some marking M''. If $M[u{>}M'$ then u is called a *firing sequence* from M to M'. We write $M[u{>}$ iff there exists a marking M' such that $M[u{>}M'$. Finally, by $M[{<}$ we denote the fact that no transition is enabled at M; such an M is called a *deadlock* (or simply *dead*) marking.

Based on the token game we associate to every net system $NS = (N, l, M_0)$ a labelled transition graph (also called the *case graph*), denoted C_{NS}, as follows. The states are given by $R(M_0)$, labels are drawn from Act and the transition relations $\{\xrightarrow{\mu} : \mu \in Act\}$ are given by $M \xrightarrow{\mu} M'$ for $M, M' \in R(M_0)$ iff there exists a transition t labelled μ of the underlying net such that $M[t{>}M'$.

A question about Petri nets that we shall be interested in is that of bisimilarity between net systems which may be formally defined as follows.

Definition 2.19 *Let $NS_i = (N_i, l_i, M_{0i})$ for $i = 1, 2$ be two net systems. We say that NS_1 and NS_2 are bisimilar, denoted $NS_1 \sim NS_2$, iff $M_{01} \sim M_{02}$ where the initial markings M_{01} and M_{02} are viewed as states of the case graphs for NS_1 and NS_2 respectively.*

Finally, a special subclass of PN, in which we shall be interested, is the class of net systems which are *deterministic* in the following sense.

Definition 2.20 *A net system $NS = (N, l, M_0)$ is* deterministic *iff the corresponding case graph C_{NS} is deterministic. (See Definition 2.4 for the notion of a deterministic labelled transition graph.)*

2.2.3 Push-Down Automata

The class of *push-down automata* generates exactly the context-free languages (see [HU79]). Thus from the point of view of generating languages push-down automata are just as expressive as the calculus $\mathrm{BPA}_\mathbf{0}$. However, from the point of view of generating labelled transition graphs there exist push-down automata whose transition graphs cannot be expressed as $\mathrm{BPA}_\mathbf{0}$ processes (up to bisimilarity) but not vice versa (see [Cau90a]). Hence wrt bisimilarity push-down automata yield a richer class of transition graphs compared to those of $\mathrm{BPA}_\mathbf{0}$. Let us now present the definition of a push-down automaton with the emphasis on the automaton as generating labelled transition graphs.

Definition 2.21 *A* push-down automaton *is a tuple $(Q, \Sigma, \Gamma, \delta)$ where Q is a finite set of* states, *Σ the* input alphabet, *Γ the* stack alphabet *and finally δ*

is a finite set of rules of the form $pA \xrightarrow{a} q\alpha$ *labelled by* $a \in \Sigma \cup \{\epsilon\}$*, and where* $p, q \in Q$*,* $A \in \Gamma$ *and* $\alpha \in \Gamma^*$*.*

The labelled transition graph generated by a push-down automaton $(Q, \Sigma, \Gamma, \delta)$ is given by $\Big(Q\Gamma^*, \Sigma \cup \{\epsilon\}, \Big\{ \xrightarrow{a}: a \in \Sigma \cup \{\epsilon\}\Big\}\Big)$ where the transitions $\xrightarrow{a}$ from $Q\Gamma^*$ to $Q\Gamma^*$ are deduced by closure of the rules δ under sequential composition. A transition $pA\beta \xrightarrow{a} q\alpha\beta$ indicates that in state p and with A on top of the stack, the push-down automaton may read the letter $a \in \Sigma \cup \{\epsilon\}$ and in doing so replace A by α on the stack while moving to state q. If the label is ϵ then this symbolises computation without reading any input symbol.

In this thesis we shall *not* investigate the class of labelled transition graphs generated from push-down automata. However, we have introduced them here for sake of perspective since they constitute a natural continuation of our work. For instance, transition graphs of push-down automata may be described (up to observational congruence, i.e. the version of bisimilarity that abstracts away from internal τ actions [Mil89]) as a class of processes of the form

$$\Big(E|F\Big) \setminus L$$

where E is a $\mathrm{BPA}_{\mathbf{0}}$ process (modelling the stack), F is an RCCS process (modelling the finite state-set) and L is a set of labels (consisting of the sorts of E and F) which have been abstracted away using the restriction operator of CCS. If any progress could be achieved on deciding bisimilarity for such processes then it would probably constitute a major step towards solving the following long-standing open question (see [HU79]).

Question: *Is language equivalence decidable on deterministic push-down automata?*

Formally, we say a push-down automaton $(Q, \Sigma, \Gamma, \delta)$ is deterministic if the rules δ satisfy that whenever $pA \xrightarrow{\epsilon} q\alpha$ then there is no rule of the form $pA \xrightarrow{a} r\beta$ for $a \neq \epsilon$ and if $pA \xrightarrow{a} q\alpha$ and $pA \xrightarrow{a} r\beta$ for $a \in \Sigma \cup \{\epsilon\}$ then $q = r$ and $\alpha = \beta$. Thus a deterministic push-down automaton yields a deterministic transition graph in the sense of Definition 2.4. Since bisimilarity and language equivalence coincide on normed deterministic transition graphs, the techniques developed in this thesis for deciding bisimulation equivalence are also applicable when dealing with such transition graphs and might offer new directions on solving the above mentioned question.

2.3 A Calculus with Full Turing Power

In this section we shall explore a calculus obtained by extending BPP_τ with the operator of *restriction* from CCS [Mil89]. We shall demonstrate that such an algebra has full Turing power, i.e. Turing machines may be simulated by processes of the calculus. Consequently, we shall be able to conclude that bisimilarity is undecidable on BPP_τ extended with restriction.

Let us begin by formally introducing the operator of restriction. It is a unary operator denoted $\backslash L$ for finite subsets $L \subseteq \Lambda_1 \cup \bar{\Lambda}_1$; given a process E, it returns $E \setminus L$, a process which behaves like E under the constraint that none of its transitions may be labelled with actions from L or $\bar{L}$, where $\bar{L}$ is the set of complements of the labels in L. Notice that we do not allow τ-actions to be restricted. We assume that BPP_τ is extended with $\backslash L$ and likewise assume that the transition rules for the operators of BPP_τ are extended with the rule for restriction given by

$$\frac{E \xrightarrow{\mu} E'}{E \setminus L \xrightarrow{\mu} E \setminus L}(\mu \notin L \cup \bar{L}),$$

and call the resulting process algebra for $\mathrm{BPP}_{\mathrm{r}}$. In order to show that our new calculus has full Turing power we shall consider *two-counter machines* [Min67]. A two-counter machine (denoted here by K possibly subscripted) is a finite-state program, using counters `c1` and `c2`, of $m \in \mathbb{N}$ labelled statements

$$\begin{array}{lll}
\mathtt{s}_1 & : & \mathtt{Com}_1 \\
\mathtt{s}_2 & : & \mathtt{Com}_2 \\
\vdots & & \\
\mathtt{s}_{m-1} & : & \mathtt{Com}_{m-1} \\
\mathtt{s}_m & : & \mathtt{halt}
\end{array}$$

where each command $\texttt{Com}_h$ $(h = 1, 2 \dots m-1)$ is either a so-called type I command of the form: `cj:=cj+1; goto` $\texttt{s}_k$, or a so-called type II command of the form: `if cj=0 then goto` $\texttt{s}_{k1}$ `else cj:=cj-1; goto` $\texttt{s}_{k2}$. For a type II command we refer to the `then` case as the *zero branch* and the `else` case as the *non-zero* branch.

Execution of a two-counter machine K (with given input values on the counters) begins with the command $\texttt{Com}_1$ and proceeds through the sequence of labelled statements, looping back and forth as indicated by the `goto` constructions. The execution stops iff the last (`halt`) instruction is encountered. It is well-known (see e.g. [Min67]) that a Turing machine can be simulated by a two-counter machine; in particular, it is in general undecidable whether a given two-counter machine on given input values halts.

We shall use two-counter machines on a number of occasions throughout this thesis. For instance, in Chapter 3 as well as in Chapter 8 we shall show how labelled Petri nets may simulate two-counter machines (in a weak sense, since Petri nets do not have full Turing power, lacking the crucial ability to test if a counter is zero). However, here we show that $\mathrm{BPP_r}$ has full Turing power; we do this (following Taubner [Tau89]) by simulating two-counter machines. That is, given a two-counter machine K with counters `c1` and `c2` initialised to the values $n1$ and $n2$ of $I\!N$ respectively we construct a process E_K of the form

$$E_K \;=\; \Big(C_1|C_2|F\Big) \setminus L \qquad (\star)$$

that simulates the two-counter machine. Here C_1 and C_2 play the rôles of the counters `c1` and `c2`. They shall be initialised to contain the values $n1$ and $n2$ respectively. The expression F is a finite-state process, i.e. a process of RCCS, which simulates the labelled statements of K by communicating with the counters C_1 and C_2. Finally, L consists of all the actions of C_1, C_2 and F, and we restrict these actions at the outermost level of the process thereby forcing F to communicate with the counters C_1 and C_2.

The crucial part of the above process expression $(\star)$ involves the possibility to build counters. In order to specify counters formally let $\sigma : Act^* \to (\Lambda_1 \cup \bar{\Lambda}_1)^*$ be

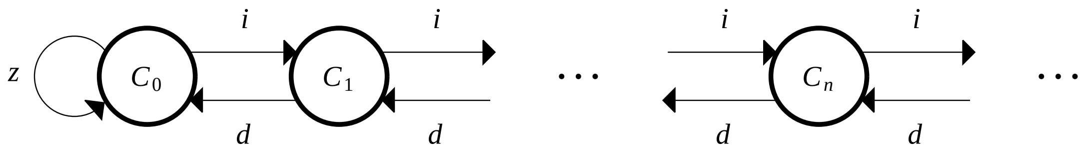

Figure 2–3: The specification of a counter.

the operator on strings which abstracts away from τ-actions, i.e. $\sigma(\epsilon) = \epsilon$ and

$$\sigma(\mu w) \;=\; \begin{cases} \mu\sigma(w) & \text{if } \mu \neq \tau \\ \sigma(w) & \text{otherwise.} \end{cases}$$

We now say that any process E satisfying

$$\Big\{\sigma(w) : w \in Act^*, E \xrightarrow{w} \Big\} \;=\; \Big\{w : C_n \xrightarrow{w} \Big\},$$

where C_n is the n'th state of the labelled transition graph of Figure 2–3, represents a counter having value n. Here the labels z, i and d stand for (test for) zero, increment and decrement respectively.

A process of the form $\big\{X = i(X|d)\big\}$ almost satisfies the specification of a counter; the labelled transition graph for X would be the same as the transition graph of Figure 2–3 obtained by removing the arrow labelled z at state C_0. However, it is exactly this ability to test for zero which is so important and which requires additional expressiveness such as that obtained via the restriction operator. Following Taubner [Tau89] we define counters in our calculus via the family Δ_{zid} of equations consisting of

$$\begin{aligned} X_1 &\stackrel{\text{def}}{=} zX_1 + i\Big((X_2|aX_1) \setminus \{a, \bar{a}\}\Big) \\ X_2 &\stackrel{\text{def}}{=} d\bar{a} + i\Big((X_3|bX_2) \setminus \{b, \bar{b}\}\Big) \\ X_3 &\stackrel{\text{def}}{=} d\bar{b} + i\Big((X_2|aX_3) \setminus \{a, \bar{a}\}\Big). \end{aligned}$$

In order to be able to state that Δ_{zid} defines counters (of any value) we introduce the notion of a *context*. We define $P_1 = ([\;]|aX_1) \setminus \{a, \bar{a}\}$ representing a context with a single placeholder symbolised by $[\;]$; the term $P_1(E)$ stands for the process expression obtained by substituting E for $[\;]$ in P_1, i.e. the expression

$(E|aX_1) \setminus \{a, \bar{a}\}$. We now introduce a sequence of contexts starting with P_1 as given above and continuing with

$$\begin{aligned} P_{2n} &= P_{2n-1}\Big(([\ ]|bX_2) \setminus \{b, \bar{b}\}\Big) \\ P_{2n+1} &= P_{2n}\Big(([\ ]|aX_3) \setminus \{a, \bar{a}\}\Big) \end{aligned}$$

for $n \geq 1$. We have the following proposition whose proof may be found in [Tau89].

Proposition 2.22 ([Tau89]) *The process X_1 represents a counter with value 0, and the process $P_{2n+1}(X_2)$ represents a counter with value $2n+1$ while $P_{2n+2}(X_3)$ represents a counter with value $2n+2$ ($n = 0, 1 \ldots$).*

We may now complete the description of the process expression E_K of ($\star$). We take a copy $\Delta_{z_1 i_1 d_1}$ of our family Δ_{zid} where z, i and d have been replaced by z_1, i_1 and d_1 respectively. We then define C_1 to be the process expression (based on $\Delta_{z_1 i_1 d_1}$) which represents a counter with value $n1 \in I\!N$. In a similar fashion we take a copy $\Delta_{z_2 i_2 d_2}$ and define the process expression C_2 to represent a counter with value $n2 \in I\!N$. The finite-state process F will consists of m guarded process equations as follows: $\{Y_1 \stackrel{\text{def}}{=} E_1, Y_2 \stackrel{\text{def}}{=} E_2 \ldots Y_{m-1} \stackrel{\text{def}}{=} E_{m-1}, Y_m \stackrel{\text{def}}{=} \mathbf{0}\}$ where for each $h = 1, 2 \ldots m-1$ the expression E_h is $i_j Y_k$ if the h'th statement of the two-counter machine is $\mathtt{s}_h\mathtt{:cj:=cj+1;\ goto\ s}_k$ while E_h is $z_j Y_{k1} + d_j Y_{k2}$ if the h'th statement is $\mathtt{s}_h\mathtt{:if\ cj=0\ then\ goto\ s}_{k1}\mathtt{\ else\ cj:=cj-1;\ goto\ s}_{k2}$. Finally, the set L of labels which is abstracted away at the outermost level of E_K is given by $\{z_j, d_j, i_j, \bar{z}_j, \bar{d}_j, \bar{i}_j : j = 1, 2\}$.

It is now clear that E_K offers the choices of the corresponding two-counter machine K, and the enforced synchronisation of F with the counters C_1 and C_2 ensures that only those choices taken by the two-counter machine can be performed. We therefore conclude that $\mathrm{BPP_r}$ yields a calculus having full Turing power since two-counter machines can be simulated. Notice that computation steps of the two-counter machine K are represented by internal τ-actions of the corresponding process E_K. From this observation we have the following.

Theorem 2.23 *Bisimilarity is undecidable on* $\mathrm{BPP_r}$.

Proof: Let K be an arbitrary two-counter machine (with initial values $n1$ and $n2$ on the counters `c1` and `c2` respectively). Suppose E_K is the corresponding $\mathrm{BPP_r}$ process expression simulating K. Now consider the process X given by the single equation $\{X \stackrel{\mathrm{def}}{=} \tau X\}$. We have $E_K \sim X$ if, and only if, K diverges, and have therefore reduced the divergence problem for two-counter machines to a question of bisimilarity between processes of $\mathrm{BPP_r}$. We must conclude that bisimilarity is undecidable on $\mathrm{BPP_r}$. ■

2.4 Distributed Bisimulation Equivalence

We now introduce the last of the semantic equivalences in which we shall be interested. It is the so-called *distributed bisimulation equivalence* as described by Castellani and Hennessy (see e.g. [CH87,Cas88]). Distributed bisimilarity is a congruence that takes into account the distribution of processes in space. It is finer than bisimilarity (for instance, it does not obey the *expansion law* of bisimilarity [Mil89]) and is suggested as a way of interpreting concurrent processes in an operational setting that abandons the understanding of parallelism as nondeterministic interleaving.

Originally, distributed bisimilarity was defined (and its properties investigated) on a process algebra corresponding to the recursion-free fragment of BPP_τ. Here we shall extend this definition to deal with the whole of BPP_τ (and hence also BPP). As a step towards defining distributed bisimilarity we have to recast the operational semantics of our processes of BPP_τ and BPP in the setting of *distributed labelled transition graphs*. Such graphs are similar to ordinary labelled transition graphs as defined in Section 2.2 but with the important difference that transitions give rise to a *compound residual* consisting of a *local* component and a *concurrent* component. The description reflects a view of processes as distributed in space; intuitively the local component records the *location* at which the action of the transition is observed whereas the concurrent component records the part of the process being separated from the local component.

It is not our intention to introduce the general definition of distributed la-

$$\frac{}{\mu E \xrightarrow{\mu} (E, \mathbf{0})} \qquad \frac{E \xrightarrow{\mu} (E_1, E_2)}{X \xrightarrow{\mu} (E_1, E_2)} (X \stackrel{\text{def}}{=} E \in \Delta)$$

$$\frac{E \xrightarrow{\mu} (E_1, E_2)}{E + F \xrightarrow{\mu} (E_1, E_2)} \qquad \frac{F \xrightarrow{\mu} (F_1, F_2)}{E + F \xrightarrow{\mu} (F_1, F_2)}$$

$$\frac{E \xrightarrow{\mu} (E_1, E_2)}{E \| F \xrightarrow{\mu} (E_1, E_2 \| F)} \qquad \frac{F \xrightarrow{\mu} (F_1, F_2)}{E \| F \xrightarrow{\mu} (F_1, E \| F_2)}$$

$$\frac{E \xrightarrow{\mu} (E_1, E_2)}{E \lfloor\!\lfloor F \xrightarrow{\mu} (E_1, E_2 \| F)} \qquad \frac{E \xrightarrow{\mu} (E_1, E_2)}{E \lfloor F \xrightarrow{\mu} (E_1, E_2 | F)}$$

$$\frac{E \xrightarrow{\mu} (E_1, E_2)}{E | F \xrightarrow{\mu} (E_1, E_2 | F)} \qquad \frac{F \xrightarrow{\mu} (F_1, F_2)}{E | F \xrightarrow{\mu} (F_1, E | F_2)}$$

$$\frac{E \xrightarrow{a} (E_1, E_2) \quad F \xrightarrow{\bar{a}} (F_1, F_2)}{E | F \xrightarrow{\tau} (E_1 | F_1, E_2 | F_2)}$$

Table 2–2: Distributed transition rules.

belled transition graphs as it should be clear from the above discussion how such graphs are formed. Rather we proceed directly to the definition of the operational semantics of BPP and BPP$_\tau$ processes using such graphs.

Any finite family Δ of guarded BPP or BPP$_\tau$ process equations determines a distributed labelled transition graph; states and labels are given as usual, and the transition relations are given as the least relations derived from the rules of Table 2–2. Again, transitions are defined on process expressions relative to some family Δ so strictly speaking transitions should be subscripted accordingly. However, again we leave the reader to infer the intended family. We shall not comment further on the rules of Table 2–2 but refer to [Cas88] for a thorough explanation.

As for ordinary transition relations we shall assume that distributed transition relations are inferred modulo the rule of congruence given by

$$\frac{E \equiv E' \quad E' \xrightarrow{\mu} (E_1', E_2') \quad E_1' \equiv E_1 \quad E_2' \equiv E_2}{E \xrightarrow{\mu} (E_1, E_2)},$$

in order for instance not to be annoyed by innocent $\mathbf{0}$-components sitting in parallel. Again we note that we can safely do so since the notion of distributed bisimilarity (which we define now) satisfies the basic laws underlying $\equiv$. The definition of distributed bisimulation equivalence is closely related to that for bisimilarity. However, the relation will now be based on the information contained in the individual residuals. Thus a distributed bisimulation relates local and concurrent components separately.

Definition 2.24 *A binary relation R over process expressions is a* distributed bisimulation *if whenever $(E, F) \in R$ then for each $\mu \in Act$,*

- $E \xrightarrow{\mu} (E_1, E_2)$ *implies* $F \xrightarrow{\mu} (F_1, F_2)$ *such that* $(E_i, F_i) \in R$ *for* $i = 1, 2$,
- $F \xrightarrow{\mu} (F_1, F_2)$ *implies* $E \xrightarrow{\mu} (E_1, E_2)$ *such that* $(E_i, F_i) \in R$ *for* $i = 1, 2$.

Processes E and F are distributed bisimilar*, written $E \sim_d F$, if they are related by some distributed bisimulation.*

We shall rely on the theory of distributed bisimilarity as established in [Cas88]. For instance, it is an equivalence relation as well as a congruence relation wrt all the process combinators of BPP and BPP_τ. Moreover, it is strictly contained in $\sim$ and there exists a sound and complete axiomatisation for $\sim_d$ on the recursion-free fragments of BPP and BPP_τ (see e.g. [Cas88]).

2.5 Further Notions

In this final section of our chapter on background we introduce further notions and techniques which shall be used throughout the thesis. We define *normal forms* for our various process classes. We also introduce the *tableau technique* by applying it to the process class RCCS. We then discuss the notion of *equational theories* in the area of process algebra; we shall compare the equation-based approach (as for instance presented in [Mil84]) with the sequent-based approach as supported by the tableau technique (see e.g. [HS91]). Finally, we shall introduce the notions of *self-bisimulation* and *decomposition* which are central concepts regarding our techniques for proving decidability.

2.5.1 Normal Forms

We shall be interested in comparing, using bisimilarity or any other equivalence, two families of guarded equations Δ_1 and Δ_2. For instance, wrt bisimilarity we write $\Delta_1 \sim \Delta_2$ to indicate that the leading variables of Δ_1 and Δ_2 are bisimilar. Often one of Δ_1 or Δ_2 serves as a *normal form* for the other family. We now introduce the notion of normal form for the various process classes that we have defined wrt the two semantic equivalences of bisimilarity and distributed bisimilarity.

Definition 2.25 *A finite family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ of guarded* RCCS *process equations is defined to be in* normal form *if every expression E_i is of the form*

$$E_i \equiv \sum_{j=1}^{n_i} a_{ij} X_{ij}$$

where for each $j = 1, 2 \ldots n_i$ we have $X_{ij} \in Var(\Delta)$.

For the above normal form (as well as the other normal forms we introduce shortly) we shall recognise the empty sum as $\mathbf{0}$, and ignore the ordering of expressions in sums, hence defining the notion of normal form modulo associativity and commutativity of summation.

It is straightforward to show that any finite family Δ of guarded RCCS process equations can be transformed effectively into a family Δ' in normal form such that bisimilarity is preserved, i.e. $\Delta \sim \Delta'$. Notice that for any family Δ of RCCS processes in normal form, the set of reachable states from any $X \in Var(\Delta)$ will be a subset of $Var(\Delta)$ and hence finite.

Next we describe a normal form for families Δ of guarded $\mathrm{BPA}_{\mathbf{0}}$ process equations which has been called *Greibach Normal Form (GNF)* by analogy with context-free grammars in GNF (see e.g. [HU79]). For any finite family Δ of $\mathrm{BPA}_{\mathbf{0}}$ process equations we denote by $\mathit{Var}(\Delta)^*$ the set of sequences of variables from $\mathit{Var}(\Delta) = \{X_1, X_2 \ldots X_n\}$ and let lower-case Greek letters $\alpha, \beta \ldots$ range over $\mathit{Var}(\Delta)^*$. Each $\alpha \in \mathit{Var}(\Delta)^*$ denotes a $\mathrm{BPA}_{\mathbf{0}}$ process by viewing juxtaposition as sequential composition. We recognise the empty sequence as $\mathbf{0}$, and ignore the grouping of variables, hence identifying processes denoted by elements of $\mathit{Var}(\Delta)^*$ up to associativity of sequential composition.

Definition 2.26 *A finite family* $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\}$ *of guarded* $\mathrm{BPA}_{\mathbf{0}}$ *process equations is defined to be in* Greibach Normal Form (GNF) of degree d *if every expression* E_i *is of the form*

$$E_i \quad \equiv \quad \sum_{j=1}^{n_i} a_{ij}\alpha_{ij}$$

where for each $j = 1, 2 \ldots n_i$ *we have* $\alpha_{ij} \in \mathit{Var}(\Delta)^*$ *such that* $\mathit{length}(\alpha_{ij}) \leq d$.

In [BBK87a] (see also [Hüt91]) it is shown that any finite family Δ of guarded BPA process equations can be effectively presented in GNF of degree 2. The same proof can easily be adapted to the setting of $\mathrm{BPA}_{\mathbf{0}}$. We state this important result without further comments.

Proposition 2.27 *If* Δ *is a finite family of guarded* $\mathrm{BPA}_{\mathbf{0}}$ *process equations we can* effectively *construct another finite family* Δ' *of* $\mathrm{BPA}_{\mathbf{0}}$ *process equations in* GNF *of degree 2 such that* $\Delta \sim \Delta'$.

Notice that if Δ is a guarded family of $\mathrm{BPA}_{\mathbf{0}}$ equations in GNF then the set of reachable states from any $\alpha \in \mathit{Var}(\Delta)^*$ will be a subset of $\mathit{Var}(\Delta)^*$ and in general infinite.

We finally present the notion of normal form for the two process classes of BPP and BPP_τ wrt the two semantic equivalences of distributed bisimilarity and bisimilarity. For distributed bisimilarity we call our normal form *standard form* while for bisimilarity it is called *full standard form*. To ease presentation we only define the notions of standard form and full standard form in the setting of BPP_τ. The corresponding definitions in the setting of BPP are easily obtained; one replaces the parallel combinator by merge and only allows actions to be drawn from Λ_1.

For any finite family Δ of BPP_τ process equations we denote by $Var(\Delta)^{\otimes}$ the set of finite multisets over $Var(\Delta) = \{X_1, X_2 \ldots X_n\}$ and let lower-case Greek letters $\alpha, \beta \ldots$ range over elements of $Var(\Delta)^{\otimes}$. Each such α denotes a BPP_τ process by forming the product of the elements of α, i.e. by combining the elements of α in parallel using the parallel operator. We recognise the empty product as $\mathbf{0}$, and we ignore the ordering of variables in products, hence identifying processes denoted by elements of $Var(\Delta)^{\otimes}$ up to associativity and commutativity of parallel composition. By $size(\alpha)$ we denote the cardinality of α.

Definition 2.28 *A finite family* $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ *of* BPP_τ *process equations is defined to be in* standard form of degree d *iff every expression* E_i *is of the form*

$$E_i \;\equiv\; \sum_{j=1}^{n_i} (\mu_{ij}\alpha_{ij}) \lfloor \beta_{ij}$$

where for each $j = 1, 2 \ldots n_i$ *it follows that* $\alpha_{ij}, \beta_{ij} \in Var(\Delta)^{\otimes}$ *such that we have* $size(\alpha_{ij}), size(\beta_{ij}) \leq d$*.*

Notice that a family Δ of BPP_τ process equations in standard form is not guarded in the usual sense. However, we shall call such a family guarded if it satisfies the following. For $X, Y \in Var(\Delta)$ we define $X \rightarrow Y$ to mean that X's defining expression has a summand of the form $(\mu\alpha)\lfloor\beta$ such that β contains Y. We let $\rightarrow^+$ be the transitive closure of $\rightarrow$ and define Δ to be *guarded* iff there is no variable $X \in Var(\Delta)$ such that $X \rightarrow^+ X$. So, for instance, $\{X = (aX)\lfloor X\}$ is not a guarded family, while $\{X = (aX)\lfloor Y, Y = (bX)\lfloor \mathbf{0}\}$ is.

We have the following important proposition showing that any guarded family Δ of BPP_τ process equations can be effectively presented in guarded standard form (of degree 3) while preserving distributed bisimilarity. As the proof of this result is rather involved and hence might disturb the flow of presentation we have chosen to give it in Appendix A.

Proposition 2.29 *Given any finite family of guarded* BPP_τ *equations* Δ *we can* effectively *construct another finite family of* BPP_τ *equations* Δ' *in guarded standard form of degree 3 such that* $\Delta \sim_d \Delta'$.

We proceed by defining the notion of full standard form.

Definition 2.30 *A finite family* $\Delta = \{X_i \overset{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\}$ *of guarded* BPP_τ *process equations is defined to be in* full standard form of degree d *iff every expression* E_i *is of the form*

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

where for each $j = 1, 2 \ldots n_i$ *we have* $\alpha_{ij} \in Var(\Delta)^{\otimes}$ *such that* $size(\alpha_{ij}) \leq d$.

Notice the close correspondence between full standard form for BPP_τ and Greibach Normal Form for $\mathrm{BPA}_\mathbf{0}$. The only difference concerns the interpretation of the α_{ij}'s; in the setting of BPP_τ each such expression represents a *parallel* composition while in the setting of $\mathrm{BPA}_\mathbf{0}$ it represents a *sequential* composition. Also notice that the normal form for RCCS correspond to GNF of degree 1 as well as full standard form of degree 1.

In [Cas88] it is remarked that particularly one axiom emphasises the difference between distributed bisimilarity and bisimilarity, viz. the axiom $(\mu E)\lfloor F = \mu(E|F)$ which is valid wrt bisimilarity but *not* wrt distributed bisimilarity. Using this axiom together with Proposition 2.29 we immediately see that any finite family Δ of guarded BPP_τ equations can be effectively presented in full standard form of degree 6 while preserving bisimilarity. However, as shown in Appendix A, any family Δ of BPP_τ process equations in full standard form of degree d for some d may effectively be transformed into full standard form of degree 2 while preserving bisimilarity. We therefore conclude the following.

Proposition 2.31 *Given any finite family of guarded* BPP_τ *equations* Δ *we can* effectively *construct another finite family of* BPP_τ *equations* Δ' *in full standard form of degree 2 in which* $\Delta \sim \Delta'$.

Notice that if Δ is a guarded family of BPP_τ equations in full standard form, the set of reachable states from any $\alpha \in \mathit{Var}(\Delta)^{\otimes}$ will be a subset of $\mathit{Var}(\Delta)^{\otimes}$ and in general infinite.

Our various normal forms will become important in the decidability of bisimilarity and distributed bisimilarity. For instance, the question of deciding bisimilarity is the question whether or not $\Delta_1 \sim \Delta_2$ for two families of guarded process equations Δ_1 and Δ_2. However, due to our normal form propositions we can assume without loss of generality that both Δ_1 and Δ_2 are in normal form (whether it is GNF or full standard form). Furthermore, we shall only prove decidability of bisimilarity or distributed bisimilarity within a *single* family Δ of process equations in normal form. We can do so since we can let Δ be the disjoint union of Δ_1 and Δ_2 that we are comparing (with suitable renamings of variables, if necessary).

We finally note that for the proof of decidability it is not important that our normal forms involve specific degrees (2 for bisimilarity and 3 for distributed bisimilarity); just the fact that we can obtain normal forms for some degree is sufficient. However, often we shall assume that our processes are presented in normal form with the least possible degree since it simplifies some of our analysis.

2.5.2 The Tableau Method

For our purposes, the tableau method consists of a syntax driven, goal directed proof system; one replaces the goal of proving E equal to F by a set of sufficient subgoals to be established. The replacement of goals by subgoals follows from a collection of *tableau rules* constituting the tableau system. The application of a particular rule is guided by the structure of processes. The rules of the tableau system are built around equations $E = F$ where E and F are processes of the appropriate type. Each rule has the form

$$\frac{E = F}{E_1 = F_1 \qquad \cdots \qquad E_n = F_n}$$

possibly with side conditions. The premise of the rule represents the goal to be achieved (for instance that $E \sim F$) whereas the consequents represent (sufficient) subgoals to be established.

REC	$\dfrac{X = Y}{E = F}$	where $X \stackrel{\text{def}}{=} E, Y \stackrel{\text{def}}{=} F \in \Delta$
SUM	$\dfrac{\sum_{i=1}^{n} a_i X_i = \sum_{j=1}^{m} b_j Y_j}{\{a_i X_i = b_{f(i)} Y_{f(i)}\}_{i=1}^{n} \qquad \{b_j Y_j = a_{g(j)} X_{g(j)}\}_{j=1}^{m}}$	where $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ $g : \{1, \ldots, m\} \to \{1, \ldots, n\}$
PREFIX	$\dfrac{aX = aY}{X = Y}$	

Table 2–3: Rules of the tableau system for $\sim$ on RCCS.

Let us illustrate the tableau method by demonstrating the well-known result that bisimilarity is decidable on finite-state processes, i.e. the calculus RCCS. We fix a finite family $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ of such process equations assumed to be in normal form. We are interested in deciding for any $X, Y \in \mathit{Var}(\Delta)$ whether or not $X \sim Y$, and we do this using the tableau method.

A *tableau* for $X = Y$ is a maximal proof tree whose root is labelled $X = Y$ and where the labelling of immediate successors of a node is determined according to the rules of the tableau system presented in Table 2–3. We adopt some terminology for tableaux. Tableaux are denoted by T (and also by $T(E = F)$ to indicate the label of the root). Paths through tableaux are denoted by π and nodes are denoted by $\mathbf{n}$ (with roots also denoted by $\mathbf{r}$) possibly with subscripts. If a node $\mathbf{n}$ has label $E = F$ we write $\mathbf{n} : E = F$.

In building tableaux the rules are only applied to nodes that are not *terminal*. A terminal node can either be *successful* or *unsuccessful*. In our setting of RCCS a successful terminal node $\mathbf{n}$ is one labelled $E = E$, or $E = F$ such that there is another node *above* $\mathbf{n}$ in the tableau (and an application of PREFIX in between) also labelled $E = F$. An unsuccessful terminal node is one labelled $aX = bY$ such that $a \neq b$, or $aX = \mathbf{0}$ or $\mathbf{0} = bY$. A tableau is successful if, and only if, all terminal nodes are successful; otherwise it is unsuccessful.

Along with any tableau system are three important properties, viz. *finiteness*, *completeness* and *soundness*. The property of finiteness concerns tableaux which in all circumstances must be shown to be finite if the tableau system is to constitute a decision procedure. For our example of RCCS we have finiteness of tableaux as expressed via the following proposition.

Proposition 2.32 *Every tableau for $X = Y$ is finite.*

The proof of this proposition is trivial since for a tableau to be infinite there must be an infinite path π (following from König's Lemma) at which we know that REC has been applied infinitely often. But the labels of nodes at which REC is applied are all drawn from $Var(\Delta) \times Var(\Delta)$ of which there are only finitely many; hence along π there are infinitely many successful terminals and therefore it must terminate. Thus there cannot exist infinite tableaux.

In other cases the finiteness argument is much harder to establish. For instance, in [HS91], Hüttel and Stirling presented a tableau system for deciding bisimilarity on normed context-free processes. In this case labels of nodes may be drawn from $Var(\Delta)^* \times Var(\Delta)^*$ of which there are infinitely many. And in Chapter 6 we shall present a tableau system for deciding bisimilarity on BPP_τ (as well as BPP) processes. In this case labels of nodes may be drawn from $Var(\Delta)^\otimes \times Var(\Delta)^\otimes$ of which there again are infinitely many. In these circumstances the finiteness arguments become rather delicate, ultimately relying on a *decomposition* of pairs of processes into "smaller" pairs according to some appropriate well-founded ordering on pairs of processes.

The other two important properties of a tableau system are *completeness* and *soundness* which in the setting of RCCS read as follows.

Theorem 2.33 (Completeness) *If $X \sim Y$ then there exists a successful tableau with root labelled $X = Y$.*

The proof of completeness follows immediately by showing that each of the tableau rules of Table 2–3 are *forward sound* in the sense that if the antecedent relates bisimilar processes then it is possible to find a set of consequents relating bisimilar processes. For in these circumstances we may build a tableau such that every node $\mathbf{n} : E = F$ of the tableau satisfies $E \sim F$ (provided the label of the root relates bisimilar processes). Clearly such a tableau is bound to be successful.

Theorem 2.34 (Soundness) *If $T(X = Y)$ is a successful tableau for $X = Y$ then $X \sim Y$.*

For our example of RCCS the proof of soundness is rather trivial: a successful tableau $T(X = Y)$ for $X = Y$ essentially constitutes a bisimulation relation containing (X, Y); one can easily show that the collection of labels of any node of $T(X = Y)$ is a bisimulation relation. This fact once more illustrates that bisimulations for finite-state systems are themselves finite.

Normally the soundness argument does not follow as easily as in the case of RCCS. For instance, in the setting of normed context-free processes, tableaux cannot in general constitute bisimulation relations. This follows from the fact that bisimulations may be infinite whereas tableaux always are finite (see [HS91]). Due to the same reasons we also know that the tableau system presented in Chapter 6 for BPP and BPP_τ cannot in general construct bisimulation relations. In these cases one may instead show that successful tableaux constitute some *essential part* of a bisimulation. For instance bisimulations may be generated (using the notion of self-bisimulation which we introduce in Section 2.5.4) from the finite basis consisting of labels of nodes of the tableau system considered.

Let us end this subsection by concluding that bisimilarity is decidable on RCCS (as based on the tableau system). Given a finite family Δ of RCCS processes in normal form, and X and Y of $\mathit{Var}(\Delta)$ that we wish to compare, start listing tableaux for $X = Y$. Each tableau is finite and there are only finitely many tableaux since any tableau can at most have height $3n^2$ where n is the number of variables of $Var(\Delta)$. Now start searching for a successful tableau; if one is found then answer "yes" to the question of $X \sim Y$; otherwise answer "no". By soundness and completeness of that tableau system we know that this algorithm always gives the correct answer.

2.5.3 Equational Theories

When proving that two processes E and F are bisimilar it suffices to construct a bisimulation relation containing the two processes. As illustrated in the previous section, a particular approach to this involves the use of tableaux. Another technique for proving that two processes are bisimilar consists in characterising bisimilarity via an *equational theory*, i.e. a formal system for deriving equivalences between processes. An equational theory consists of a set of axioms together with the laws for equivalence (reflexivity, symmetry and transitivity) as well as the laws for substitutivity. Usually such theories are presented in a *natural deduction style* [Pra65] yielding proof trees for deriving equivalences.

Let us illustrate the use of equational theories by presenting two different systems for bisimilarity wrt the class of finite-state processes RCCS. The first system we present constitutes (a part of) Milner's equational theory for finite-state behaviours (see [Mil84]) whereas the other system is supported by the tableau system of the previous subsection and as such related in style to the equational theory of [HS91] for normed context-free processes.

Milner's equational theory centres around a few very elegant laws for recursion which is given by the explicit fixed point constructor *fix* we informally introduced in Section 2.2.1. We therefore consider the class RCCS of finite-state processes being defined using this operator and shall introduce it formally now. So the operators of RCCS consist of prefix and summation (as before) together with the operator *fix* for recursion. Given a process E and a variable X, the recursive expression $fixX.E$ binds the variable X in E; we say a variable X is *free* in an expression E if it is not bound by a fixed point operator. We let $fv(E)$ denote the set of free variables of E and call E *closed* if $fv(E) = \emptyset$. Furthermore, we say a recursive expression $\mathit{fix}X.E$ is *guarded* provided X is guarded in E (in the usual sense) and call an expression E guarded iff every subexpression of E which is a recursion is guarded.

In order to simplify our presentation we shall restrict attention to closed and guarded expressions and denote by RCCS these expressions (the equational theory of [Mil84] also deals with expressions not necessarily closed or guarded).

The operational semantics for RCCS is again defined via a labelled transition graph. The transition rules for the operators of prefix and summation are as usual. For the fixed point operator we have the inference rule

$$\frac{E\{\mathit{fix}X.E/X\} \stackrel{a}{\longrightarrow} E'}{\mathit{fix}X.E \stackrel{a}{\longrightarrow} E'},$$

where $E\{F/X\}$ denotes the expression obtained by substituting F for each free occurrence of X in E, renaming bound variables as necessary. We call $E\{\mathit{fix}X.E/X\}$ the *unfolding* of the recursion $\mathit{fix}X.E$.

The equational theory for bisimilarity on RCCS is denoted $\mathfrak{R}$ and is presented in Table 2–4. We have left out the laws for equivalence as well as the laws for substitutivity. The axioms of $\mathfrak{R}$ consist of the abelian monoid laws for summation together with idempotency for summation. Finally, the two laws for recursion essentially express that a recursive process is equal to its unfolding and that guarded equations have unique solutions. By $\mathfrak{R} \vdash E = F$ we shall denote the fact that E and F are *provably equal* using the equational theory $\mathfrak{R}$. That is, if $\mathfrak{R} \vdash E = F$ then there exists a proof tree with root labelled $E = F$, leaves being instances of the axioms, and where the children of a node are determined by an application of one of the inference rules.

Given an equational theory we may begin to generate equivalences by building proofs. It is important to be sure that any equivalences which we can possibly generate are going to be valid wrt the semantic equivalence in question. This is referred to as *soundness*. For our equational theory $\mathfrak{R}$ soundness is expressed by the following theorem.

Theorem 2.35 (Soundness) *If* $\mathfrak{R} \vdash E = F$ *then* $E \sim F$.

Axioms

S1	$E + (F + G)$	$=$	$(E + F) + G$	S2	$E + F$	$=$	$F + E$
S3	$E + \mathbf{0}$	$=$	E	S4	$E + E$	$=$	E

Recursion

R1 $\quad \mathit{fix}X.E = E\{\mathit{fix}X.E/X\}$

R2 $\quad \dfrac{E = F\{E/X\}}{E = \mathit{fix}X.F}$(X guarded in F)

Table 2–4: Equational theory $\mathfrak{R}$ for $\sim$ on RCCS.

The proof of soundness may be found in [Mil84]. Usually, such a proof is not hard, involving for each law the construction of an appropriate bisimulation relation. The only law that needs some attention is R2 which may be shown valid by proving that guarded recursive equations have unique solutions (up to bisimilarity).

In contrast to soundness we would like that any two processes which are semantically equivalent can be proven equal using the equational theory in question. This is referred to as *completeness*. For our equational theory $\mathfrak{R}$ completeness is expressed by the following theorem.

Theorem 2.36 (Completeness) *If $E \sim F$ then $\mathfrak{R} \vdash E = F$.*

Usually a completeness proof is much more challenging compared to the proof of soundness. In [Mil84] the proof of completeness is obtained via three results; one result states that if E and F are bisimilar processes with E provably satisfying a certain kind of equation set S_1 while F provably satisfies another such equation set S_2 then both E and F provably satisfy a single equation set S (the equation sets serve as a kind of normal form). A second result states that every expression E provably satisfies an equation set S while the last result states that if two processes E and F satisfy the same equation set S then they may be proven equal within $\mathfrak{R}$. In Chapter 7 we shall introduce this proof style in a formal manner since wrt distributed bisimilarity on a subset of BPP_τ we shall prove a completeness result for an equational theory (presented in the style of $\mathfrak{R}$) .

We refer to equational theories such as $\mathfrak{R}$ as *equation-based* theories since the elements of the system involve (unconditional) equations. The system of [Mil84] (of which $\mathfrak{R}$ is a subset) is an example of an equation-based theory. The result of [Mil84] has later been extended by Milner to deal with the weaker equivalence of *observational congruence* (see [Mil89a]). Another example of an equation-based theory consists of Castellani's system for distributed bisimilarity on the recursion-free fragments of BPP and BPP_τ (see [Cas88]).

A tableau system for deciding bisimilarity such as that of [HS91] or the one we presented in Section 2.5.2 also supports an equational theory; for all that is required is a collection of axioms and inference rules which enables one to derive the roots of successful tableaux. In these circumstances, the equational theory is built around *sequents* of the form $\Gamma \vdash_\Delta E = F$ where Γ is a finite set of *assumptions* consisting of pairs of processes, and E and F are expressions of the appropriate type (with recursion given by defining equations relative to Δ and *not* via explicit fixed point constructors). We shall refer to such systems as *sequent-based* equational theories.

Let us illustrate the use of sequent-based equational theories by presenting such a system for RCCS as based on the tableau system of Section 2.5.2. So we now assume that the class of RCCS is given via the original definition involving defining equations for recursion. We assume a fixed family Δ of such equations and suppose it is presented in normal form. As explained, the equational theory is built around sequents of the form $\Gamma \vdash_\Delta E = F$ where (in the case of RCCS) Γ is a finite set of assumptions of the form $X = Y$, and E and F are RCCS expressions. The semantic interpretation of a sequent $\Gamma \vdash_\Delta E = F$, denoted $\Gamma \models_\Delta E = F$, is as follows: if $X \sim Y$ for all $X = Y \in \Gamma$, then $E \sim F$. As Δ will remain fixed throughout, we shall omit its subscripted appearance, thus writing $\vdash$ for $\vdash_\Delta$ and $\models$ for $\models_\Delta$. Also we shall omit empty assumption sets, thus writing $\vdash E = F$ and $\models E = F$ for $\emptyset \vdash E = F$ and $\emptyset \models E = F$ respectively. Notice that the relationship $\models E = F$ reduces to $E \sim F$.

The sequent-based equational theory is presented in Table 2–5; again we have omitted the laws for equivalence and substitutivity. The axioms are (as before) the abelian monoid laws for summation together with idempotency for summation but now formulated as sequents. The laws for recursion differ considerably from the recursion laws of $\mathfrak{R}$. Here R1 is an *assumption introduction* rule underpinning the rôle of the assumption list Γ; and R2 is an *assumption elimination* rule which represents a form of fixed point induction.

A proof of $\Gamma \vdash E = F$, which we shall denote by this sequent, consists of a proof tree with root labelled $\Gamma \vdash E = F$, instances of the axioms as leaves and where the children of a node are determined by an application of one of the inference rules.

As for the equation-based theory the properties of vital interest are soundness and completeness. The soundness of our sequent-based equational system is expressed via the following theorem.

Theorem 2.37 (Soundness) *If $\Gamma \vdash E = F$ then $\Gamma \models E = F$. In particular, if $\vdash E = F$ then $E \sim F$.*

The proof of the above theorem is not hard. It may be obtained by *reductio ad absurdum*, assuming that all pairs of Γ relate bisimilar processes but $E \not\sim F$, and then showing that there can be no proof of the sequent $\Gamma \vdash E = F$.

Finally, completeness of our sequent-based equational theory is expressed by the following theorem.

Theorem 2.38 (Completeness) *If $E \sim F$ then $\vdash E = F$.*

Notice, that this actually states *weak* completeness since the assumption list is empty. The proof of completeness relies on the tableau system presented in Section 2.5.2; given processes E and F (with $E \sim F$) one constructs a successful

Axioms

S1	$\Gamma \vdash E + (F + G) = (E + F) + G$	S2	$\Gamma \vdash E + F = F + E$
S3	$\Gamma \vdash E + \mathbf{0} = E$	S4	$\Gamma \vdash E + E = E$

Assumption introduction and elimination

R1 $\Gamma, X = Y \vdash X = Y$

R2 $$\frac{\Gamma, X = Y \vdash E = F}{\Gamma \vdash X = Y} (X \stackrel{\text{def}}{=} E, Y \stackrel{\text{def}}{=} F \in \Delta)$$

Table 2–5: A sequent-based equational theory for $\sim$ on RCCS.

tableau for $E = F$ and show that for any node $\mathbf{n} : G = H$ of this tableau it follows that Recnodes($\mathbf{n}$) $\vdash G = F$. Here Recnodes($\mathbf{n}$) refers to the collection of labels of nodes above $\mathbf{n}$ at which Rec is applied. Notice that Recnodes($\mathbf{r}$) $\vdash E = F$ reduces to $\vdash E = F$.

Whether one prefers an equation-based theory or a sequent-based theory is perhaps a matter of taste. One might argue in favour of the equation-based theory due to its elegant and natural laws for recursion. On the other hand, the advantage of the sequent-based theory is that it offers a very natural and direct method of presenting proofs (via the corresponding tableau system). Using equation-based theories (such as Milner's for RCCS) it is less straightforward how proofs are built up. Finally, in favour of the sequent-based theory is the fact that it has proven applicable to a wider range of process classes such as normed context-free processes (see [HS91]) and also BPP$_\tau$ as we shall demonstrate in Chapter 6; it is not known how to obtain the corresponding systems using equation-based theories.

In this thesis we shall see some examples of equational theories for the two semantic equivalences of bisimilarity and distributed bisimilarity. As explained, in Chapter 6 we shall present a sequent-based theory for bisimilarity on BPP_τ (as well as BPP). In Chapter 7 we shall present the corresponding sequent-based theory for distributed bisimilarity. Also in Chapter 7 we shall present an equation-based theory for distributed bisimilarity on a subclass of BPP_τ where general summation is replaced by *guarded summation.*

2.5.4 Self-bisimulations

As mentioned in Section 2.5.2, in the case of normed context-free processes or in the case of BPP_τ, a successful tableau cannot in general constitute a bisimulation relation since the latter may be infinite whereas tableaux always are finite. However, a successful tableau will represent a *self-bisimulation*, i.e. a relation whose closure under congruence with some operator (sequential composition for context-free processes and parallel composition for BPP_τ) is a bisimulation relation.

The notion of self-bisimulation was introduced by Caucal in [Cau90] in the context of sequential composition. Let us introduce the notion of self-bisimulation in the context of any binary process operator *op*. In Chapter 4 we shall replace *op* by sequential composition while in Chapter 6 we replace *op* by parallel composition. For the introduction of self-bisimulations, the notion of a least congruence wrt composition by *op* plays a central rôle.

Definition 2.39 *Let R be any binary relation over processes. We denote by $R^{\rightarrow}$ the least precongruence wrt composition by op containing R. By $R^{\leftrightarrow}$ we then denote the symmetric closure of $R^{\rightarrow}$ and finally by $R^{\leftrightarrow^*}$ we denote the reflexive and transitive closure of $R^{\leftrightarrow}$. Thus $R^{\leftrightarrow^*}$ is the least congruence wrt composition by op containing R.*

Now a self-bisimulation is a bisimulation up to congruence wrt composition by *op* as expressed by the following definition.

Definition 2.40 *A binary relation R over processes is called a* self-bisimulation *(wrt composition by op) iff $(E, F) \in R$ implies for all $\mu \in Act$ that*

- *if* $E \xrightarrow{\mu} E'$ *then* $F \xrightarrow{\mu} F'$ *for some* F' *with* $(E', F') \in R^{\leftrightarrow^*}$, *and*
- *if* $F \xrightarrow{\mu} F'$ *then* $E \xrightarrow{\mu} E'$ *for some* E' *with* $(E', F') \in R^{\leftrightarrow^*}$.

An important property of a self-bisimulation which we shall rely on is the following stating that a self-bisimulation is a *witness* for bisimilarity. We shall prove the result in the context of parallel composition; for a proof of the corresponding result in the context of sequential composition we refer to [Cau90].

Proposition 2.41 *If* R *is a self-bisimulation (wrt parallel composition) then we have* $R^{\leftrightarrow^*} \subseteq \sim$.

Proof: We show that $R^{\leftrightarrow^*}$ is a bisimulation relation from which the result follows since $\sim$ is the largest bisimulation. Let $(E, F) \in R^{\leftrightarrow^*}$ and suppose $E \xrightarrow{\mu} E'$ for some μ, E'. We must find an F' with $F \xrightarrow{\mu} F'$ such that $(E', F') \in R^{\leftrightarrow^*}$. Now, if $(E, F) \in R^{\leftrightarrow^*}$ then we can find a sequence $E = E_1 R^{\leftrightarrow} E_2 \cdots E_{k-1} R^{\leftrightarrow} E_k = F$. We proceed by induction on k.

For $k = 1$ the result follows trivially since E and F are identical. For $k = 2$ we must have either $(E, F) \in R^{\rightarrow}$ or $(F, E) \in R^{\rightarrow}$. Assume without loss of generality that $(E, F) \in R^{\rightarrow}$. Then by the definition of the least precongruence, there exists $(E_0, F_0) \in R$ and processes G and H such that $E = G|E_0|H$ and $F = G|F_0|H$. Suppose now that $G \xrightarrow{\mu} G'$ such that $E \xrightarrow{\mu} G'|E_0|H$. Then we have $F \xrightarrow{\mu} G'|F_0|H$ with $(G'|E_0|H, G'|F_0|H) \in R^{\leftrightarrow^*}$ as required. In a similar fashion we may deal with moves from H or moves due to a synchronisation between G and H. So suppose $E_0 \xrightarrow{\mu} E_0'$ such that $E \xrightarrow{\mu} G|E_0'|H$. Since R is a self-bisimulation we can find F_0' such that $F_0 \xrightarrow{\mu} F_0'$ and $(E_0', F_0') \in R^{\leftrightarrow^*}$. But then $F \xrightarrow{\mu} G|F_0'|H$ with $(G|E_0'|H, G|F_0'|H) \in R^{\leftrightarrow^*}$ as required. In a similar fashion we may deal with moves from E due to a synchronisation between E_0 and one of G or H.

Now assume the required property holds for k and consider the sequence of length $k+1$ consisting of $E = E_1 R^{\leftrightarrow} E_2 \cdots E_k R^{\leftrightarrow} E_{k+1} = F$. If $E \xrightarrow{\mu} E'$ then from the induction hypothesis we conclude that $E_k \xrightarrow{\mu} E_k'$ for some process E_k' with

$(E', E'_k) \in R^{\leftrightarrow^*}$. Again (since $E_k \xrightarrow{\mu} E'_k$) we conclude from the induction hypothesis that $F \xrightarrow{\mu} F'$ for some F' with $(E'_k, F') \in R^{\leftrightarrow^*}$. But this trivially gives the required relationship $(E', F') \in R^{\leftrightarrow^*}$ by transitivity of $R^{\leftrightarrow^*}$.

By symmetry we now conclude that $R^{\leftrightarrow^*}$ is a bisimulation relation and that completes the proof. ■

As a simple consequence of the above proposition we have the following (which also applies to the sequential case).

Corollary 2.42 *$E \sim F$ if, and only if, there is a self-bisimulation R such that $(E, F) \in R$.*

Proof: Clearly $\sim$ is a self-bisimulation. Conversely, by Proposition 2.41, if R is a self-bisimulation such that $(E, F) \in R$ then $E \sim F$ since $R^{\leftrightarrow^*}$ is a bisimulation relation. ■

In Chapter 4 and also in Chapter 6 we shall obtain results corresponding to the above corollary but for *finite* self-bisimulations. In these circumstances we shall demonstrate that semi-decidability of bisimilarity follows simply by searching for the *finite* witness to bisimilarity which a finite self-bisimulation constitutes.

2.5.5 Decompositions

Since bisimulations may be infinite on context-free processes as well as on BPP_τ (and BPP), when proving that tableaux are finite or when proving that bisimulations can be generated from *finite* self-bisimulations we must ultimately *decompose* pairs of bisimilar processes into "smaller" pairs of bisimilar processes such that there are only finitely many pairs that cannot be decomposed further.

Our notion of decomposition in the setting of BPA_0 concerns bisimilar processes over $Var(\Delta)^*$ where Δ is a family of BPA_0 process equations in GNF.

Definition 2.43 *Suppose $X\alpha, Y\beta \in Var(\Delta)^*$. When $X\alpha \sim Y\beta$ we say that the pair $(X\alpha, Y\beta)$ is* decomposable *if X and Y are normed and there is a $\gamma \in Var(\Delta)^*$ such that*

- *if* $\mathcal{N}(X) \leq \mathcal{N}(Y)$ *then* $\alpha \sim \gamma\beta$ *and* $X\gamma \sim Y$, *and*
- *if* $\mathcal{N}(Y) \leq \mathcal{N}(X)$ *then* $\gamma\alpha \sim \beta$ *and* $X \sim Y\gamma$.

In the case of normed context-free processes, the important property underpinning decidability of bisimilarity (as for instance established in [Cau90]) is the fact that *any* pair $(X\alpha, Y\beta)$ of bisimilar processes is decomposable. This may be demonstrated using the following important *cancellation* property due to Caucal (see [Cau88]).

Proposition 2.44 *For normed* BPA_0 *processes* E, F *and* G, *if* $EG \sim FG$ *then* $E \sim F$.

Now suppose $X\alpha, Y\beta \in \mathit{Var}(\Delta)^*$ are normed bisimilar processes. Furthermore suppose that $\mathcal{N}(X) \leq \mathcal{N}(Y)$ and that β is not empty. Since X is normed we may find a (shortest) terminating sequence $X \xrightarrow{w} \mathbf{0}$. Thus $X\alpha \xrightarrow{w} \alpha$ and therefore $Y\beta \xrightarrow{w} \gamma\beta$ for some γ with $\alpha \sim \gamma\beta$. Using substitutivity we get $X\gamma\beta \sim Y\beta$ which from the above proposition implies $X\gamma \sim Y$ thus showing that the pair $(X\alpha, Y\beta)$ is decomposable. Consequently, in the case of normed context-free processes, bisimulation equivalence is then generable from a finite self-bisimulation consisting of bisimilar pairs of the form (X, α), and this leads to decidability.

However, in the presence of unnormed processes, the cancellation property is no longer valid; a simple counter example is the family $\{X \stackrel{\mathrm{def}}{=} aX, Y \stackrel{\mathrm{def}}{=} a\}$ since $YYX \sim YX$ but clearly $YY \not\sim Y$. Furthermore, there can be bisimilar pairs $(X\alpha, Y\beta)$, for X and Y normed, which are not decomposable. For instance, wrt the family $\{X \stackrel{\mathrm{def}}{=} a + b, Y \stackrel{\mathrm{def}}{=} aZ + b, Z \stackrel{\mathrm{def}}{=} a, V \stackrel{\mathrm{def}}{=} aV\}$ we have X and Y normed, and $XV \sim YV$, but (XV, YV) is not decomposable. However, in Chapter 4 we shall demonstrate that in this case there are only finitely many different (wrt bisimilarity) pairs which are not decomposable, from which we shall obtain our decidability result.

In the setting of BPP_τ our notion of decomposition concerns bisimilar pairs of processes over $\mathit{Var}(\Delta)^{\otimes}$ where Δ is a family of BPP processes in full standard form. In previously published work (see [CHM93]) the notion of decomposition was derived from the following *cancellation* property whose proof we shall present in Chapter 5.

Proposition 2.45 *For normed* BPP_τ *processes* E, F *and* G*, if* $E|G \sim F|G$ *then* $E \sim F$.

Inspired by the above proposition we define bisimilar pairs (α, β) to be decomposable if $\alpha \equiv \alpha_1|\alpha_2$ and $\beta \equiv \beta_1|\beta_2$ such that $\alpha_i \not\equiv \mathbf{0}$, $\beta_i \not\equiv \mathbf{0}$ and $\alpha_i \sim \beta_i$ for $i = 1, 2$. Using the above proposition we show in [CHM93] that there are only finitely many pairs which are not decomposable, and this leads to decidability of bisimilarity in the normed case.

The cancellation property is not valid in the presence of unnormed processes; a simple counter example is again the family $\{X \stackrel{\mathrm{def}}{=} aX, Y \stackrel{\mathrm{def}}{=} a\}$ since $Y|Y|X \sim Y|X$ but clearly $Y|Y \not\sim Y$. Furthermore (in contrast to the situation for context-free processes) there can be infinitely many (different) non-decomposable bisimilar pairs in the case of unnormed processes.

Example 2.46 *Consider the family* $\{X \stackrel{\mathrm{def}}{=} aX, Y \stackrel{\mathrm{def}}{=} aZ, Z \stackrel{\mathrm{def}}{=} b\}$. *We have for all* $n \in \mathbb{N}$ *that* $X|Y^n \sim X|Z^n$ *with* $(X|Y^n, X|Z^n)$ *not being decomposable. Notice that for all* $n, m \in \mathbb{N}$ *with* $n \neq m$ *we have* $X|Y^n \not\sim X|Y^m$ *(as well as* $X|Z^n \not\sim X|Z^m$*).*

However, in Chapter 6 we shall demonstrate that by introducing a new notion of decomposition (which was discovered by Hirshfeld) our decidability result of [CHM93] can be extended beyond the normed case.

Chapter 3

Language Classes

In this chapter we shall explore the class of languages generated by processes of BPP; we denote this class by L(BPP). We shall compare L(BPP) with standard language classes such as the context-free languages (or equivalently, the class of languages generated by BPA_0) and the context-sensitive languages. We shall also compare L(BPP) with a class of languages generated by labelled Petri nets. Furthermore, we shall examine closure properties for L(BPP) and finally we consider decidability questions for language equivalence on L(BPP) and related language classes, notably various classes of languages generated by labelled Petri nets.

3.1 Comparison of Language Classes

The class of languages which we will be interested in throughout this chapter is formally defined as follows.

Definition 3.1 *The class* L(BPP) *consists of languages generated by processes of* BPP. *That is, L is a language of* L(BPP) *if there exists a finite family Δ of guarded* BPP *process equations such that $L = L(X_1)$ where X_1 is the leading variable of Δ.*

Notice that our full standard form for BPP processes wrt bisimilarity (see Section 2.5.1) also serves as a normal form wrt language equivalence since $\sim$ is contained in $=_L$. Thus whenever L is a language of L(BPP) we may assume that it is generated by a family of BPP processes in full standard form.

In this chapter we shall also be interested in labelled Petri nets as devices for generating languages, i.e. we shall view Petri nets in terms of the firing sequences that may occur. Such languages began with the work of Baker [Bak72], Hack [Hac75], Peterson [Pet74] and was continued by Starke [Sta78], Jantzen [Jan86] and others. There exist various ways of defining Petri net languages. We shall concentrate on the following two classes.

Definition 3.2 *Let $NS = (N, l, M_0)$ be a labelled Petri net system and M_f a distinguished marking called a* final *marking. We define*

- $L(NS, M_f) = \Big\{ l(w) : \exists w \in T^* \text{ such that } M_0[w{>}M_f \Big\}$, *and*
- $T(NS) = \Big\{ l(w) : \exists w \in T^* \text{ such that } M_0[w{>} \Big\}$.

$L(NS, M_f)$ is called the *terminal* language of NS (with final marking M_f) while $T(NS)$ is the *trace* language of NS and thus prefix closed. By L(PN) we denote the class of all terminal languages generated by net systems of PN and T(PN) for the class of all trace languages generated by net systems of PN. In [Hac75] it is shown that T(PN) $\subsetneq$ L(PN). Furthermore, by L($\mathrm{PN_d}$) we denote the subclass of L(PN) by restricting attention to deterministic net systems. Likewise for T($\mathrm{PN_d}$).

There exist other ways of defining Petri net languages; notably we could consider the labelled firing sequences from the initial marking leading to a dead marking. Such languages are commonly known as *deadlock* Petri net languages [Jan86]. However, as shown in [PP85], the class of such languages defines exactly L(PN).

Let us begin by comparing L(BPP) with the class of context-free languages which we denote by CFL. We shall in the following demonstrate that these two language classes are *incomparable*. First we consider the case of L(BPP) $\not\subseteq$ CFL. That is, we seek a process E of BPP such that $L(E)$ is not a context-free language. Let E be X where X is given by the family

$$\Big\{ X \stackrel{\mathrm{def}}{=} a(b\|c\|X + b\|c) + b(a\|c\|X + a\|c) + c(a\|b\|X + a\|b) \Big\}.$$

Then $L(E)$ consists of all nonempty strings over the alphabet $\{a, b, c\}$ which contain equal numbers of a's, b's and c's. Suppose that $L(E)$ is context-free. As the class of context-free languages is closed under intersection with regular languages we must have that $L(E) \cap a^*b^*c^* = \{a^nb^nc^n : n > 0\}$ is a context-free language. However, via the pumping lemma admitted by CFL's, it is easy to show that $\{a^nb^nc^n : n > 0\}$ cannot be context-free. Hence, neither can $L(E)$ be context-free.

Next we turn to the case of CFL $\not\subseteq$ L(BPP). To show this we have found it useful to first obtain a *pumping lemma* for the class L(BPP). We need additional definitions and a helpful lemma before proving the pumping lemma.

Suppose Δ is a finite family of guarded BPP process equations in full standard form of degree 2. Let $Var(\Delta) = \{X_1, X_2 \dots X_n\}$. For $X, Y \in Var(\Delta)$ define $X \to Y$ iff for $X \stackrel{\text{def}}{=} \sum_{i=1}^{m} a_i\alpha_i \in \Delta$ there exists α_i such that $Y \in \alpha_i$. Let $\to^+$ be the transitive closure of $\to$. Finally, for $X \in Var(\Delta)$, let $R(X)$ denote the set $\{Y \in Var(\Delta) : X \to^+ Y\}$ and extend this definition to multisets of $Var(\Delta)^\otimes$ in the obvious way.

Lemma 3.3 *Let Δ be a finite family of guarded* BPP *process equations in full standard form of degree 2. If for all variables $X \in Var(\Delta)$ we have $X \notin R(X)$ then for any $\alpha \in Var(\Delta)^\otimes$ it follows that*

$$\forall u \in \Lambda_1^* : u \in L(\alpha) \textit{ implies } length(u) \leq m(\alpha)$$

for the constant $m(\alpha)$ given by $(2^n - 1)size(\alpha)$ where n is the number of variables of $Var(\Delta)$.

Proof: For a proof it is enough to give an upper bound on the length of words generated from a single variable $X \in Var(\Delta)$. Notice that we have total freedom in the way steps from X are interleaved in order to provide an upper bound on words from $L(X)$.

For $\alpha, \beta \in Var(\Delta)^\otimes$ we define $\alpha \stackrel{w}{\rightsquigarrow} \beta$ iff *each* of the variable instances of α has participated with exactly *one* step in the derivation of w and no other variables have participated. In these circumstances we have $length(w) = size(\alpha)$ and $size(\beta) \leq 2size(\alpha)$.

Suppose $u \in L(X)$, i.e. we may assume that $X \xrightarrow{u} \mathbf{0}$. Let v be a permutation of u such that $v = v_1 v_2 \cdots v_k$ with $X = \alpha_0 \overset{v_1}{\rightsquigarrow} \alpha_1 \overset{v_2}{\rightsquigarrow} \cdots \overset{v_{k-1}}{\rightsquigarrow} \alpha_{k-1} \overset{v_k}{\rightsquigarrow} \mathbf{0}$. For this sequence we have $R(\alpha_i) \subsetneq R(\alpha_{i-1})$ for $i = 1, 2 \ldots k$ and therefore $k \leq n$. We also have $size(\alpha_i) \leq 2size(\alpha_{i-1})$ for $i = 1, 2 \ldots k$ which gives $size(\alpha_i) \leq 2^i$. As $length(v_i) = size(\alpha_{i-1})$ it follows that $length(v_i) \leq 2^{i-1}$. We may thus conclude that $length(u) = length(v) = \sum_{i=1}^{n} length(v_i) \leq 2^n - 1$. Now, let $m(\alpha)$ be the constant given by the expression $(2^n - 1)size(\alpha)$ and we get for all $u \in L(\alpha)$ that $length(u) \leq m(\alpha)$ as required. ∎

Lemma 3.4 (Pumping lemma) *Let L be any language of* L(BPP)*. Then there exists a constant m such that if u is a word of L with $length(u) > m$ then there exist x, y and z of Λ_1^* such that*

- *$u = xz$,*
- *$length(y) \geq 1$, and*
- *$\forall i \geq 0 : xy^i z \in L$.*

Proof: As L is a language of L(BPP) there exists a finite family Δ of BPP equations in full standard form of degree 2 such that L is given by $L(X)$ where X is the leading variable of Δ. We may furthermore assume that Δ is normed since from the point of view of generating languages we are only interested in *terminating* sequences. Let m be the constant $2^n - 1$ where n is the number of variables of $Var(\Delta)$. Suppose u is a word of L with $length(u) > m$. From Lemma 3.3 it follows that there exists $Y \in Var(\Delta)$ with $Y \in R(Y)$ such that Y appears in a derivation of the word u. Hence, for some $x, z \in \Lambda_1^*$ with $u = xz$ we may write $X \xrightarrow{x} \beta \| Y \xrightarrow{z} \mathbf{0}$ for some $\beta \in Var(\Delta)^{\otimes}$. But if $Y \in R(Y)$ then for some $y \in \Lambda_1^*$ with $length(y) \geq 1$ we have $Y \xrightarrow{y} Y$ since Δ is normed, and therefore for all $i \geq 0$ it follows that $xy^i z \in L$ as required. ∎

Using the above pumping lemma we now show that CFL $\nsubseteq$ L(BPP). Certainly, the language $L = \{a^n b^n : n > 0\}$ is a member of CFL since $\{A \rightarrow aAb \mid ab\}$ is a context-free grammar generating L. Suppose L is a language of L(BPP) and let

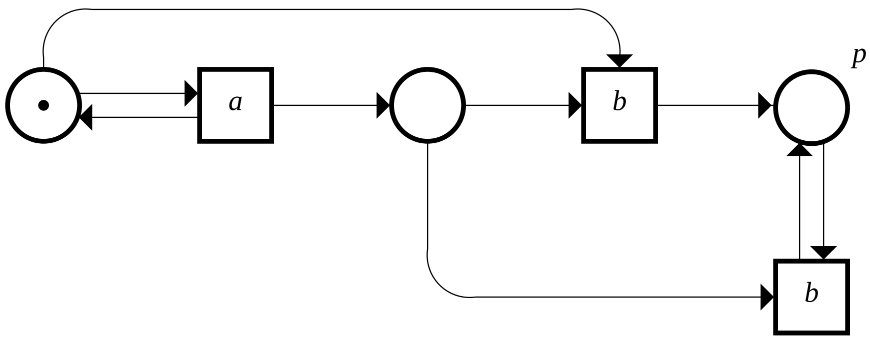

Figure 3–1: A net system generating the language $\{a^n b^n : n > 0\}$.

m be the constant of Lemma 3.4. Consider the word $a^m b^m \in L$. Clearly, it is impossible for $a^m b^m$ to be divided into x and z such that a nonempty word can be pumped with the resulting word still belonging to L. Thus, we must conclude that L cannot be a language of L(BPP).

We now proceed by comparing L(BPP) with the class of terminal Petri net languages, i.e. the class L(PN). In the following we shall demonstrate the strict inclusion of L(BPP) $\subsetneq$ L(PN). Firstly, strictness follows by showing that the context-free language $L = \{a^n b^n : n > 0\}$ belongs to L(PN).

Example 3.5 *Consider the net system of Figure 3–1. It follows readily that the terminal language (with the final marking having a single token in place p while all other places are empty) of this net system is* $\{a^n b^n : n > 0\}$ *which is a language not belonging to* L(BPP)*.*

For the inclusion of L(BPP) in L(PN) we first show the somewhat stronger result that any process of BPP may be represented (up to bisimilarity) as the labelled transition graph of a corresponding labelled Petri net system.

Suppose $\Delta = \{X_i \stackrel{\text{def}}{=} E_i : i = 1, 2 \ldots n\}$ is a family of BPP processes in full standard form of degree 2, i.e. each expression E_i takes the form $\sum_{j=1}^{n_i} a_{ij}\alpha_{ij}$ such that $size(\alpha_{ij}) \leq 2$ for all i, j. Associated with Δ is a Petri net $N_\Delta = (S_\Delta, T_\Delta; F_\Delta)$ constructed as follows. The set of places S_Δ is given by $\{p_i : i = 1, 2 \ldots n\}$ and the set of transitions T_Δ is given by $\{t_{ij} : i = 1, 2 \ldots n, j = 1, 2 \ldots n_i\}$. Finally, the flow relation F_Δ is given by $F_\Delta(p_i, t_{ij}) = 1$ and

$$F_\Delta(t_{ij}, p_k) \;=\; \begin{cases} 1 & \text{if } X_k \in \alpha_{ij} \text{ and occurs only once,} \\ 2 & \text{if } X_k \in \alpha_{ij} \text{ and occurs twice,} \end{cases}$$

while F_Δ is defined to be zero in all other cases. To every $\alpha \in Var(\Delta)^{\otimes}$ we obtain a marking M_α of N_Δ given by $M_\alpha(p_i) = k_i$ iff X_i occurs exactly k_i times in α; in particular, M_{X_1} is the marking that puts a single token in p_1 and leaves all other places empty. We may now associate to Δ the net system $NS_\Delta = (N_\Delta, l_\Delta, M_{X_1})$ with labelling map l_Δ given by $l_\Delta(t_{ij}) = a_{ij}$ for all i, j.

Example 3.6 *Let Δ be the family $\{X_1 \stackrel{\text{def}}{=} a(X_1 \| X_2) + a, X_2 \stackrel{\text{def}}{=} b(X_2 \| X_2)\}$. The net system associated with Δ is exactly that of Example 2.17 of Chapter 2 (see page 32).*

Proposition 3.7 *Let Δ be a family of* BPP *equations in full standard form of degree 2. We then have $X_1 \sim M_{X_1}$ where X_1 is the leading variable of Δ and M_{X_1} the initial marking for NS_Δ viewed as a state of the corresponding case graph.*

Proof: It is a routine matter to show that $\left\{(\alpha, M_\alpha) : \alpha \in Var(\Delta)^{\otimes}\right\}$ constitute a bisimulation relation. ∎

Suppose Δ is a family of BPP process equations in full standard form of degree 2 and NS_Δ the associated net system. Since from Proposition 3.7 we have $X_1 \sim M_{X_1}$ where X_1 is the leading variable of Δ while M_{X_1} is the initial marking for NS_Δ it is not hard to demonstrate that $L(X_1) = L(NS, M_z)$ where M_z is defined to be the *zero marking*, i.e. the marking that satisfies $M_z(p) = 0$ for every place p of NS_Δ. This together with Example 3.5 leads us to conclude that L(BPP) $\subsetneq$ L(PN).

Since L(BPP) is incomparable with the class of context-free languages and furthermore included in L(PN), it follows that there exist languages of L(PN) which are not context-free. In [Pet74] it is shown that the context-free language given by $\left\{ww^r : w \in \{a, b\}^*\right\}$, where w^r is the word w listed in reversed order, cannot be a language of L(PN). Hence we conclude that CFL and L(PN) are also incomparable. Furthermore, in [Jan86] it is shown that L(PN) is strictly included

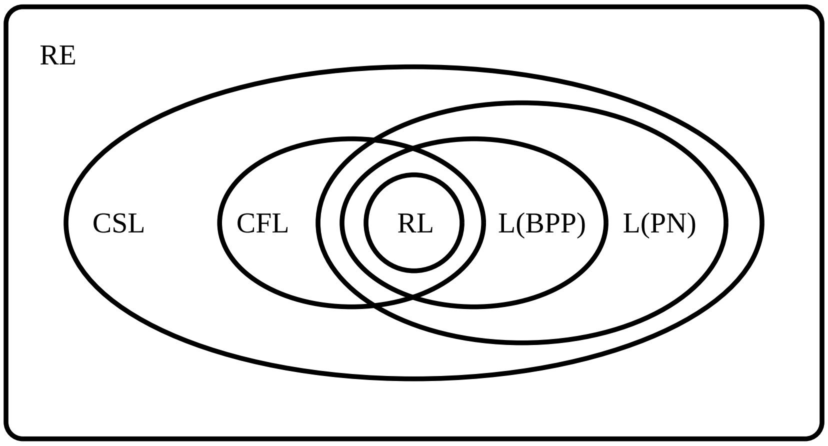

Figure 3–2: Comparison of the language classes (modulo the empty word).

in the class of context-sensitive languages which we denote by CSL. Hence the comparison of this section can be summarised by Figure 3–2 where RE stands for the class of recursive enumerable languages while RL stands for the class of regular languages.

Notice that we may characterise RL as the class of languages generated by RCCS processes and that CFL may be characterised as the class of languages generated by BPA_0 processes. Finally, we may say that RE is the class of languages recognised by BPP_τ extended with restriction since this calculus has full Turing power (see Section 2.3).

From Example 3.5 and Proposition 3.7 it follows that the model PN is a proper extension of the process algebra BPP (up to bisimilarity). Certainly, PN is also an extension of the process algebra BPP_τ: given a family Δ of BPP_τ process equations in full standard form of degree 2 we may construct an associated net system NS_Δ such that the result corresponding to Proposition 3.7 is valid.

The construction of the net system NS_Δ is similar to the one we presented for BPP processes however slightly more complicated in order to cater for the possibility that processes synchronise on complementary actions. Rather than giving the formal definition of the net system NS_Δ we shall illustrate the construction by an example.

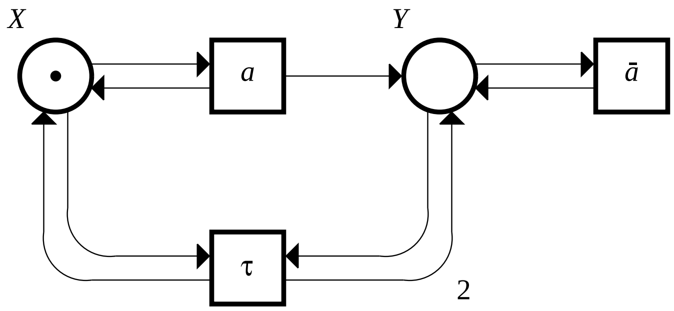

Figure 3–3: The net system associated with the family Δ of Example 3.8.

Example 3.8 *Let the* BPP_τ *family* Δ *be given by* $\{X \stackrel{\mathrm{def}}{=} a(X|Y), Y \stackrel{\mathrm{def}}{=} \bar{a}Y\}$. *The associated net* NS_Δ *is shown in Figure 3–3.*

The inclusion of L(BPP) in the class of context-sensitive languages was deduced from the two inclusions L(BPP) $\subsetneq$ L(PN) and L(PN) $\subsetneq$ CSL. However, it may be illustrative to prove directly that L(BPP) is included in CSL; this may show the "degree of context-sensitivity" inherent in languages of L(BPP). We end this section by performing a construction giving the inclusion of L(BPP) in CSL directly.

Suppose Δ is a guarded family of BPP process equations in full standard form of degree 2. Associated with the family Δ is a context-sensitive grammar denoted $G_\Delta = (V_\Delta, T_\Delta, P_\Delta, S_\Delta)$ and defined as follows. The set of non-terminals V_Δ is given by $\left\{A_X : X \in \mathit{Var}(\Delta)\right\}$ together with a distinguished start symbol S_Δ. The set of terminals T_Δ is given by the syntactic sort of Δ and the set of productions P_Δ is given as follows.

- For all X and Y of $\mathit{Var}(\Delta)$ we have $A_X A_Y \to A_Y A_X$ (which we shall denote the *permutation productions*).

- If $X \stackrel{\mathrm{def}}{=} \sum_{i=1}^{n} a_i\alpha_i \in \Delta$ then $A_X \to a_1 A_{\alpha_1} \mid \cdots \mid a_n A_{\alpha_n}$ where in general A_α denotes $A_{X_1} A_{X_2} \cdots A_{X_m}$ for α consisting of $X_1, X_2 \ldots X_m$ listed in an arbitrary order.

- $S_\Delta \to A_X$ where X is the leading variable of Δ.

Introducing the notation A_α might seem careless since the actual sequence it denotes clearly depends on the order in which the elements of α are listed; A_α denotes many different sequences of $(V_\Delta)^*$. However, due to the permutation productions each of them can be obtained from the others by a sequence of derivations only involving the permutation productions. In this sense we let A_α be a generic name for all the possible sentences of $(V_\Delta)^*$ based on α. Notice that the permutation productions are the only productions which prevents G_Δ from being context-free.

Proposition 3.9 *Let L be a language of the class* L(BPP). *Then L is also context-sensitive.*

Proof: As L is a language of L(BPP) there exists a finite family Δ of BPP process equations in full standard form of degree 2 such that $L = L(X)$ where X is the leading variable of Δ. Let $G_\Delta = (V_\Delta, T_\Delta, P_\Delta, S_\Delta)$ be the context-sensitive grammar associated with Δ. We claim that $L = L(G_\Delta)$.

We first consider the case of $L \subseteq L(G_\Delta)$. By induction on the length of u we show that if $\alpha \xrightarrow{u} \beta$ then $A_\alpha \stackrel{*}{\Rightarrow} uA_\beta$ from which the required result readily follows. If *length*$(u) = 1$ then $u = a$ for some $a \in \Lambda_1$. Since $\alpha \xrightarrow{a} \beta$ we must have $\alpha \equiv \alpha' \| Y$ such that $a\gamma$ is a summand of the defining equation for Y and $\beta \equiv \alpha' \| \gamma$. Clearly, we have $A_\alpha \stackrel{*}{\Rightarrow} A_Y A_{\alpha'} \Rightarrow aA_\gamma A_{\alpha'} \stackrel{*}{\Rightarrow} aA_\beta$.

Now, suppose that *length*$(u) > 1$. Then we may write $u = av$ for some a, v such that we have $\alpha \xrightarrow{a} \gamma \xrightarrow{v} \beta$. By induction it follows that $A_\gamma \stackrel{*}{\Rightarrow} vA_\beta$. From the above case we now conclude that $A_\alpha \stackrel{*}{\Rightarrow} aA_\gamma \stackrel{*}{\Rightarrow} avA_\beta$ as required.

We now turn to the case of $L(G_\Delta) \subseteq L$. First of all, if $uA_\alpha \stackrel{*}{\Rightarrow} uaA_\beta$ then we have $\alpha \xrightarrow{a} \beta$ as can easily be shown from the definition of G_Δ. Secondly, if $u \in L(G_\Delta)$ then there exists a derivation of u in which each sentential form is taken from $(T_\Delta)^*(V_\Delta)^*$. Again, this is a straightforward consequence from the definition of G_Δ. Hence, suppose that $u = a_1 a_2 \cdots a_k \in L(G_\Delta)$ and let this word be generated by the sequence $S_\Delta \stackrel{*}{\Rightarrow} a_1 A_{\alpha_1} \stackrel{*}{\Rightarrow} a_1 a_2 A_{\alpha_2} \stackrel{*}{\Rightarrow} \cdots \stackrel{*}{\Rightarrow} a_1 a_2 \cdots a_k$. From the above observation we then get $X \xrightarrow{a_1} \alpha_1 \xrightarrow{a_2} \alpha_2 \xrightarrow{a_3} \cdots \xrightarrow{a_k} \mathbf{0}$ and therefore we have $u \in L$ as required. ∎

3.2 Closure Properties for L(BPP)

If a class of languages is closed under a particular operation then it is called a *closure property* for this class of languages. Closure properties are important as a method of characterising language classes. They may also be useful for proving or disproving membership of languages to particular classes; we demonstrated an example of this in the previous section. Closure properties for standard language classes such as CFL and CSL may be found in [HU79]. Closure properties for language classes of Petri nets have been studied in e.g. [Hac75,Sta78,Jan86]. Here we shall turn to the class of L(BPP) and demonstrate that it is closed under very few of the usual operators.

Proposition 3.10 L(BPP) *is closed under union and shuffle.*

Proof: Suppose L_1 and L_2 are languages of L(BPP). Thus, let Δ_1 and Δ_2 be finite families of BPP equations in full standard form of degree 2 such that $L_i = L(X_i)$ where X_i is the leading variable for Δ_i ($i = 1, 2$). Suppose E_1, respectively E_2, is the defining expression for X_1, respectively X_2.

For closure under union we form a new family of BPP equations $\Delta_1 \cup \Delta_2$ (assuming disjoint variables, using renaming if necessary) together with the new leading equation $X \stackrel{\text{def}}{=} E_1 + E_2$ for a new variable X. Clearly, we have $L_1 \cup L_2 = L(X)$ thus showing that L(BPP) is closed under union.

Let shuffle(L_1, L_2) denote the *shuffle* of the two languages L_1 and L_2 (see [HU79] for a definition). For closure under shuffle let the family Δ be $\Delta_1 \cup \Delta_2$ together with a new leading equation $X \stackrel{\text{def}}{=} E_1 \| E_2$ for a new variable X. It is clear that shuffle$(L_1, L_2) = L(X)$ thus showing that the class L(BPP) is also closed under shuffle. ■

Notice that CFL is not closed under shuffle (see for instance [HU79]). As in the case of context-free languages, L(BPP) is not closed under intersection or complementation.

Proposition 3.11 L(BPP) *is not closed under intersection or complementation.*

Proof: Clearly $L = \left\{w \in \{a,b\}^* : |w|_a = |w|_b > 0\right\}$, where $|w|_a$ denotes the number of occurrences of a in the word w, is a language of L(BPP): if X has the defining equation $\left\{X \stackrel{\text{def}}{=} a(b\|X + b) + b(a\|X + a)\right\}$ then $L = L(X)$. We now have $L \cap a^*b^* = \{a^nb^n : n > 0\}$ and therefore L(BPP) cannot be closed under intersection since $\{a^nb^n : n > 0\}$ is not a member of L(BPP) as was demonstrated in Section 3.1.

From DeMorgan's law, i.e. $A \cap B = (A^c \cup B^c)^c$ where A^c denotes the complementation of A, it also follows that L(BPP) cannot be closed under complementation. ■

Notice from the proof of the above proposition that L(BPP) is not even closed under intersection with regular languages (contrary to the situation for CFL). The intersection of languages from L(BPP) with regular languages yields rather expressive languages as shall be demonstrated in Section 3.3.1.

Unlike the situation for CFL we have that L(BPP) is not closed under concatenation, substitution or homomorphism.

Proposition 3.12 L(BPP) *is not closed under concatenation, substitution or homomorphism.*

Proof: Let Δ be a finite family of BPP process equations in full standard form of degree 2. Suppose $L(X)$, where X is the leading variable of Δ, is nonregular. Then there must exist a word u of $L(X)$ such that in a derivation of u we have $X \xrightarrow{v} \alpha\|\beta \xrightarrow{w} \mathbf{0}$ where $u = vw$ and α and β are nontrivial processes. For if this is not the case then certainly $L(X)$ is regular.

As we have seen in the proof of Proposition 3.11, the language L given by $\left\{w \in \{a,b\}^* : |w|_a = |w|_b > 0\right\}$ belongs to L(BPP). By using the pumping lemma admitted by regular languages it follows that L is not regular. Suppose L(BPP) is closed under concatenation. Let $c \in \Lambda_1$ with $a \neq c$ and $b \neq c$. By assumption $L\{c\}$ must be a (nonregular) language of L(BPP). Thus let Δ be a finite family of BPP equations in full standard form of degree 2 such that $L(X) = L\{c\}$ where X is the leading variable of Δ. From the above remark we must have $uc \in L(X)$

such that $X \xrightarrow{v} \alpha\|\beta \xrightarrow{wc} \mathbf{0}$ for $u = vw$ and nontrivial processes α and β. One of these two processes must provide c, say α. Then we obtain for nonempty words w_1 and w_2 with $w = w_1w_2$ that $X \xrightarrow{v} \alpha\|\beta \xrightarrow{w_1c} \beta \xrightarrow{w_2} \mathbf{0}$ contradicting the fact that $L(X) = L\{c\}$. Hence $L\{c\}$ cannot be a language of L(BPP).

From the above it also follows that L(BPP) lacks closure under substitution; for let a particular substitution replace the symbol 1 by L and 0 by $\{c\}$. Since $\{10\} \in$ L(BPP) and $\{10\}$ under this substitution gives $L\{c\}$ we get our result.

We shall also use the language L to show that L(BPP) cannot be closed under homomorphism. Let $h : \{a, b\} \to \{0, 1\}$ be the homomorphism defined by $h(a) = 00$ and $h(b) = 11$. Suppose $h(L)$ is a language of L(BPP) and let Δ be a family of BPP process equations in full standard form of degree 2 such that $h(L) = L(X)$ where X is the leading variable of Δ. We have for all $m > 0$ that $1^{2m}0^{2m} \in h(L)$. Therefore for big enough m there is a derivation

$$X \xrightarrow{1^{2m-1}} \alpha\|\beta \xrightarrow{1} \gamma \xrightarrow{0^{2m}} \mathbf{0}$$

where both α and β are nontrivial. Otherwise it is impossible to perform *exactly* $2m$ copies of 0 (for all $m > 0$) from the expression $\alpha\|\beta$. But then we also obtain the word $1^{2m-1}010^{2m-1}$ of $L(X)$, hence $h(L) = L(X)$ cannot be the case. We therefore conclude that L(BPP) cannot be closed under homomorphism. ■

In conclusion, L(BPP) is closed under only a few of the standard operators on languages. It seems that a language class like L(BPP) has had very little attention in the area of formal language theory; when shuffle is considered it is always in connection with concatenation or sequencing. In [Gis81] Gischer studies closure properties for a language class including concatenation as well as shuffle. Essentially he shows that such a class has the closure properties of L(BPP) as well as those of CFL.

3.3 Decidability Questions for Languages

It is a well-known fact that language equivalence is decidable on the class of regular languages. Moreover, as soon as one moves beyond regular languages to the class of context-free languages then language equivalence becomes undecidable (see e.g. [HU79]). We shall now present well-known as well as some new results on the decidability of language equivalence for L(BPP), L(PN) and related language classes.

3.3.1 On Deciding Language Equivalence for L(BPP)

An obvious question is whether language equivalence is decidable on L(BPP). We know that for the subclass of normed and deterministic BPP (for the definition of deterministic processes see Chapter 2, Definition 2.4) language equivalence is decidable: this equivalence coincides with bisimilarity on normed and deterministic BPP and in Chapter 6 we show that bisimilarity is decidable on the class of BPP processes.

For some time we tried to settle the problem of deciding language equivalence (either in the positive or in the negative) for the full class L(BPP). However, we were not able to provide an answer to this problem. A related question we have been investigating is deciding language equivalence for the class L(BPP$\cap$RL) which consists of languages obtained by the intersection of languages from L(BPP) with regular languages. In this subsection we show that $=_L$ is undecidable on L(BPP$\cap$RL). However, a very recent result by Hirshfeld demonstrates that language equivalence is undecidable on L(BPP) as well[1]. Thus, in light of this result our proof of undecidability of $=_L$ on L(BPP$\cap$RL) is rendered somewhat superfluously. Still we carry out the proof partly because the reduction we shall perform offers a characterisation of the nature of languages from L(BPP).

Our reduction is based on ideas from [Gra79] where a reduction is performed from the unsolvability of equivalence between a special class of transducer automata in order to show the unsolvability of some Petri net language problems. The automata in question are so-called *ϵ-free nondeterministic generalised sequential*

[1] The proof by Hirshfeld (see [Hir93]) relies on observing that $L(E_2) \subseteq L(E_1)$ if, and only if, $E_1 + E_2 =_L E_1$ and then showing language inclusion to be undecidable on L(BPP). This, in turn, is achieved by a reduction from the Halting problem and relies on a construction somewhat related to Jančar's construction for proving bisimilarity to be undecidable on labelled Petri nets.

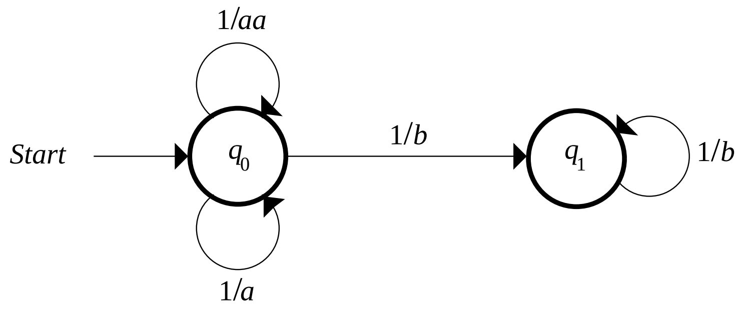

Figure 3–4: The ϵ-free NGSM of Example 3.14.

machines defined as follows.

Definition 3.13 *An* ϵ-free nondeterministic generalised sequential machine*, abbreviated* ϵ-free NGSM*, is a 5-tuple* $M = (Q, \Sigma, \Gamma, \delta, q_0)$ *where* Q *is a finite set of states,* Σ *and* Γ *the* input alphabet *and* output alphabet *respectively,* δ *a mapping from* $Q \times \Sigma \times Q$ *to finite subsets of* Γ^+ *and* q_0 *the* initial state*.*

For an ϵ-free NGSM $M = (Q, \Sigma, \Gamma, \delta, q_0)$ we shall write $M : \Sigma \rightarrow \Gamma$ to indicate the input and output alphabets.

Example 3.14 *Let* $M = \big(\{q_0, q_1\}, \{1\}, \{a, b\}, \delta, q_0\big)$ *be an* ϵ-free NGSM *with the map* δ *defined by* $\delta(q_0, 1, q_0) = \{a, aa\}$*,* $\delta(q_1, 1, q_1) = \{b\}$*,* $\delta(q_0, 1, q_1) = \{b\}$*, and* $\delta(q_1, 1, q_0) = \emptyset$*.*

We may draw an ϵ-free NGSM as a finite automaton with an edge labelled u/v from state p to state q if $v \in \delta(p, u, q)$. The diagram for the ϵ-free NGSM of Example 3.14 is shown in Figure 3–4.

We extend the domain of δ to $Q \times \Sigma^+ \times Q$ by

$$\delta(p, xy, r) \quad = \quad \bigcup_{q \in Q} \delta(p, x, q)\delta(q, y, r)$$

for all $p, r \in Q$ and $x, y \in \Sigma^+$.

Definition 3.15 *Given an* ϵ-free NGSM $M = (Q, \Sigma, \Gamma, \delta, q_0)$, *we define a* transduction function T_M *from* Σ^+ *to finite subsets of* Γ^+ *by*

$$\forall x \in \Sigma^+ : T_M(x) = \bigcup_{r \in Q} \delta(q_0, x, r).$$

We say that two ϵ-free NGSM $M, M' : \Sigma \to \Gamma$ *are* equivalent *iff* $T_M = T_{M'}$.

From [Gri68,Iba77] we have the following.

Theorem 3.16 *The equivalence problem for* ϵ-free NGSM *is undecidable if either the input alphabet or the output alphabet contains at least two elements.*

Thus for instance the equivalence problem between ϵ-free NGSM's with input alphabet $\{1\}$ and an output alphabet Γ with at least two elements is undecidable. We shall use this fact to show that language equivalence is undecidable on L(BPP$\cap$RL).

Given an ϵ-free NGSM $M : \{1\} \to \Gamma$ we define

$$L_M \quad = \quad \left\{ w\$^k : w \in T_M(1^k), k > 0 \right\},$$

where $\$^k$ denotes a sequence of \$'s with length k (likewise for 1^k) and \$ is a symbol not in Γ. It is now straightforward to show the following.

Lemma 3.17 *Let* $M, M' : \{1\} \to \Gamma$ *be two* ϵ-free NGSM. *We then have* $T_M = T_{M'}$ *if, and only if,* $L_M = L_{M'}$.

Given an ϵ-free NGSM $M : \{1\} \to \Gamma$ we *effectively* construct a BPP process E_M such that $L_M = L(E_M) \cap \Gamma^+\$^+$. Hence from Lemma 3.17 we will have (effectively) reduced the equivalence problem for ϵ-free NGSM's with input alphabet $\{1\}$ and output alphabet Γ to language equivalence on L(BPP$\cap$RL) and therefore by Theorem 3.16 the latter is undecidable (if Γ contains at least two elements).

Let $M = (Q, \{1\}, \Gamma, \delta, q_0)$ be given. Associated with M is a finite family Δ_M of BPP process equations defined as follows. Suppose $Q = \{q_0, q_1 \ldots q_n\}$. Then let the variables of Δ_M be $\{X_0, X_1 \ldots X_n\}$. If for states q_i and q_j of Q we have $\delta(q_i, 1, q_j) = \{w_1, w_2 \ldots w_k\}$ then we define the BPP expression

$$E_{ij} \;=\; \sum_{l=1}^{k} w_l(X_j \| \$) + \sum_{l=1}^{k} w_l \$.$$

By convention, if $k = 0$ then $E_{ij} = \mathbf{0}$. We understand the prefix form in the above expression as a *sequence* of prefixes based on the alphabet Γ and thus forming the word w_l. Hence w_l is *not* a symbol of the sort of Δ_M. For each process variable $X_i \in \mathit{Var}(\Delta_M)$ we now obtain a defining equation given by

$$X_i \;\stackrel{\text{def}}{=}\; \sum_{j=0}^{n} E_{ij}.$$

That completes the construction of Δ_M. We let E_M be X_0, the leading variable of Δ_M.

Lemma 3.18 *Let $M : \{1\} \to \Gamma$ be an ϵ-free* NGSM *and suppose E_M is the associated* BPP *process as constructed above. We have $L_M = L(E_M) \cap \Gamma^+\$^+$.*

Proof: We first prove that $L_M \subseteq L(E_M) \cap \Gamma^+\$^+$. Thus let $u\$^k \in L_M$, i.e. for some state q_i of M we have $u \in \delta(q_0, 1^k, q_i)$. By induction on k we prove that $u\$^k \in L(E_M)$ and also that we have the derivation $E_M \xrightarrow{u} X_i \| \$^k$. From this the required result follows since $u\$^k \in \Gamma^+\$^+$. (By a mismatch of notation, $\$^k$ both represents a sequence of k \$'s and also a parallel combination of k \$'s.)

Suppose $k = 1$. Since $u\$ \in L_M$ we must have $u \in \delta(q_0, 1, q_i)$ for some state q_i. By construction of Δ_M the defining expression for X_0 has summands $u\$$ and also $u(X_i \| \$)$. Therefore we have $u\$ \in L(E_M)$ and $E_M \xrightarrow{u} X_i \| \$$ as required.

Suppose $k > 1$. We then have $u = vw$ for some strings v and w such that $v \in \delta(q_0, 1^{k-1}, q_i)$ and $w \in \delta(q_i, 1, q_j)$ for some states q_i and q_j of Q. By induction we have $E_M \xrightarrow{v} X_i \| \$^{k-1}$. From the construction of Δ_M it also follows that the defining expression for the variable X_i contains the two summands $w\$$ and $w(X_j \| \$)$. Therefore we get $u\$^k \in L(E_M)$ and $E_M \xrightarrow{u} X_j \| \$^k$ as required.

Next we consider the case $L(E_M) \cap \Gamma^+\$^+ \subseteq L_M$. Suppose $u \in L(E_M) \cap \Gamma^+\$^+$. Due to the regular language $\Gamma^+\$^+$ we know that u is of the form $v\$^k$ for some

$v \in \Gamma^+$ and $k > 0$. Moreover, from the construction of Δ_M we must be able to find the derivation sequence

$$X_0 \xrightarrow{v_1} X_{i_1} \| \$ \xrightarrow{v_2} \cdots \xrightarrow{v_k} X_{i_k} \| \$^k,$$

where $v = v_1 v_2 \cdots v_k$. By induction on k we show that $v \in \delta(q_0, 1^k, q_{i_k})$ from which we have $u \in L_M$ as is required.

Suppose $k = 1$. Then we have $X_0 \xrightarrow{v_1} X_{i_1} \| \$$. Thus $v_1(X_{i_1} \| \$)$ is a summand of the defining expression for X_0. But then we have $v_1 \in \delta(q_0, 1, q_{i_1})$ by definition of Δ_M.

Suppose $k > 1$. By induction we have $v_1 v_2 \ldots v_{k-1} \in \delta(q_0, 1^{k-1}, q_{i_{k-1}})$. We also have $X_{i_{k-1}} \| \$^{k-1} \xrightarrow{v_k} X_{i_k} \| \$^k$ and by definition of Δ_M this can only be the case provided $v_k \in \delta(q_{i_{k-1}}, 1, q_{i_k})$. Hence it follows that $v = v_1 v_2 \ldots v_k \in \delta(q_0, 1^k, q_{i_k})$ as required. ■

Notice the rôle of the regular language $\Gamma^+\$^+$ in the above proof; it is simply used to control the order of the occurrences of the symbol \$ emitted by the process E_M in order to count the steps of the corresponding automaton M. In a sense this summarises the nature of L(BPP); it offers the possibility to count but cannot control the order of occurrences of symbols in words being generated.

Let us end this subsection by stating the undecidability result which has been established.

Corollary 3.19 *Language equivalence is undecidable on* L(BPP$\cap$RL).

Proof: By Lemma 3.17 and Lemma 3.18 we have effectively reduced the (unsolvable) equivalence problem for ϵ-free NGSM with unary input alphabet to a question of language equivalence on L(BPP$\cap$RL) and thus by Theorem 3.16 the latter is undecidable. ■

3.3.2 Decidability Questions for Petri Net Languages

Petri net languages were extensively studied in the seventies both in terms of closure properties and in terms of decidability questions. We shall summarise some of the decidability results in this section as well as present a result for the class $\mathrm{L}(\mathrm{PN}_{\mathrm{d}})$ based on a reduction from the Halting problem (using two-counter machines).

Hack proved in [Hac75a] that the equivalence problem for T(PN), i.e. the problem of deciding whether $T(NS_1) = T(NS_2)$ for arbitrary net systems NS_1 and NS_2, is reducible to the reachability problem (which is the problem of deciding whether a given marking can be reached from the initial marking) provided the net systems in question are unlabelled (or equivalently labelled by injective labelling maps). The reachability problem was later settled in the affirmative by Mayr [May81,May84] (see also [Kos82,Mul84]).

Hack's construction for reducing the equivalence problem for T(PN) to the reachability problem is readily adapted to the case of deterministic labelled Petri net systems, thus we may conclude the following.

Proposition 3.20 *The equivalence problem for* $\mathrm{T}(\mathrm{PN}_{\mathrm{d}})$ *is decidable.*

In [Mil89] it is demonstrated that bisimulation equivalence and trace equivalence coincide on deterministic processes. From the above proposition we therefore conclude that bisimilarity is decidable on deterministic net systems. In Chapter 8 we shall demonstrate (following a construction of Jančar) that bisimilarity becomes *undecidable* when moving beyond deterministic net systems to the full model PN.

Hack proved in [Hac75a] that the equivalence problem for T(PN) is undecidable by a reduction from Hilbert's tenth problem. Since T(PN) is included in L(PN) we also obtain undecidability of the equivalence problem for L(PN).

Proposition 3.21 *The equivalence problems for* T(PN) *and* L(PN) *are undecidable.*

We end this section by carrying out a reduction from the Halting problem to the equivalence problem for $\mathrm{L}(\mathrm{PN}_{\mathrm{d}})$ thus showing the latter to be undecidable. We shall rely on two-counter machines, which we formally introduced in Chapter 2, for our reduction. The construction we now present is to some extent similar to a construction carried out by Araki and Kasami (see [AK77]) although for different purposes.

It is clear that all the operations of a two-counter machine can be simulated by a Petri net (the program of the two-counter machine is simulated by a finite-state subnet and each counter is simulated by an unbounded place, i.e. a place potentially containing arbitrarily many tokens) except for the crucial test for zero. In order to match the expressive power of two-counter machines (and thereby also the full power of Turing machines) the model of Petri nets has been extended in various ways (see e.g. [Hac75a]); notably so-called *inhibitor arcs*, which only connect places to transitions, have been considered. The interpretation of such arcs are as follows: a transition that has an incoming inhibitor arc is enabled iff the corresponding place is empty.

Instead of relying on inhibitor arcs to simulate two-counter machines *correctly* we shall simulate such machines by ordinary Petri net systems in a *weak* sense, allowing for the zero branch of a type II command to be executed even though the corresponding counter is not zero. This idea has already been used for instance by Hack (see [Hac75]) and also by Araki and Kasami (see [AK77]).

Let K be a two-counter machine with counters `c1` and `c2`. Assume that the counters have been initialised to contain the values $n1$ and $n2$ of $I\!N$ respectively. We construct a deterministic net system $NS_1 = (N_1, l_1, M_{01})$ based on K as follows. The underlying net N_1 has places p_{c1} and p_{c2} corresponding to the counters `c1` and `c2`. Furthermore, to each statement $\mathtt{s}_i\mathtt{:Com}_i$ of K we introduce a place p_i; a token in place p_i shall indicate that the *control* is at statement $\mathtt{s}_i$, i.e. $\mathtt{Com}_i$ is the next command about to be simulated. Finally, N_1 includes a distinguished place denoted p.

For each type I statement of the form $\mathtt{s}_i\mathtt{:cj:=cj+1;\ goto\ s}_k$ we introduce a transition t_i whose connection to the places are as indicated in Figure 3–5(A). We let t_i be labelled by some $a_i \in Act$. For each type II statement of the form $\mathtt{s}_i\mathtt{:if\ cj=0\ then\ goto\ s}_{k1}\mathtt{\ else\ cj:=cj-1;\ goto\ s}_{k2}$ we introduce three transitions t_i^{nz}, t_i^{z} and t_i^{zc} with labels b_i, c_i and c_i of Act respectively. The connection of these transitions to the places are as indicated in Figure 3–5(B). For the last

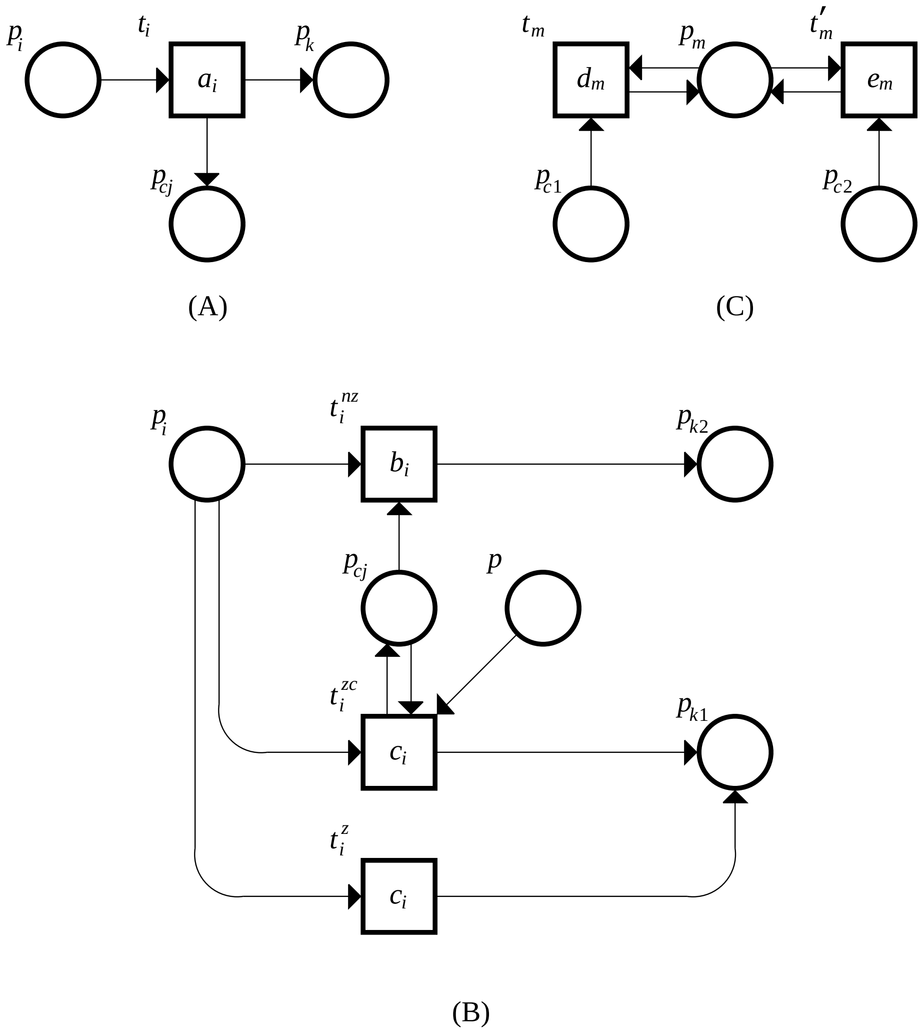

Figure 3–5: The parts of the net constructed for Lemma 3.22.

statement $\mathtt{s}_m$:`halt` we introduce transitions t_m and t'_m labelled d_m and e_m of Act respectively and connected to the places as indicated in Figure 3–5(C).

Finally, the initial marking M_{01} puts tokens in p_{c1} and p_{c2} corresponding to the initial values $n1$ and $n2$ for the counters `c1` and `c2` of the two-counter machine, and puts a single token in p_1 (the *control token*) and also a single token in p while all other places are empty.

A firing of transition t_i of Figure 3–5(A) correctly simulates the corresponding type I statement by adding a token to the place representing the counter in question while removing the control token from place p_i to p_k. Likewise, a firing of t_i^{nz} of Figure 3–5(B) correctly simulates the non-zero branch of the corresponding type II statement by removing a token from the place representing the counter in question while transferring the control token from place p_i to p_{k2}. However, a firing of transition t_i^z of Figure 3–5(B), which is supposed to simulate the zero branch of the corresponding type II statement, may occur even though the place representing the counter in question contains tokens. Whenever such a situation occurs then the two-counter machine K is simulated wrongly; we say that the net system has *cheated* on the zero branch of a type II command. The transition t_i^{zc} of Figure 3–5(B) is also supposed to simulate the zero branch of a type II command. But whenever t_i^{zc} fires, it *must* have been a simulation that cheated since the transition is only enabled provided the place representing the corresponding counter is not empty.

Finally, the subnet of Figure 3–5(C) is included in order to empty the counters once the place p_m (corresponding to the last statement of the two-counter machine) has been reached.

Now let $NS_2 = (N_2, l_2, M_{02})$ be a copy of NS_1 except that NS_2 does not contain place p, transitions t_i^{zc} and all adjacent arcs. Thus the only difference between NS_1 and NS_2 regards the possibility for NS_1 to fire a transition t_i^{zc} which *guarantees* a cheating simulation. We have the following.

Lemma 3.22 *Let M_{f1} be the final marking for NS_1 which has a single token in p_m while all other places are empty. Similarly, let M_{f2} be the final marking for*

NS_2 which has a single token in p_m while all other places are empty. We have $L(NS_1, M_{f1}) \neq L(NS_2, M_{f2})$ if, and only if, K halts.

Proof: First suppose that the two-counter machine K halts on the initial values $n1$ and $n2$. Consider the firing sequence $M_{02}[u\!>\!M_2$ of NS_2 with $M_2(p_m) = 1$ and such that u is a *correct* simulation of the two-counter machine, i.e. whenever an instance of t_i^z fired along u then the place p_{cj} representing the corresponding counter was empty. We may now remove the tokens from the two places representing the counters thus obtaining a word $l_2(u)d_m^{m1}e_m^{m2} \in L(NS_2, M_{f2})$ for some $m1$ and $m2$ of $I\!N$ corresponding to the number of tokens on p_{c1} and p_{c2} at marking M_2. Clearly such a word cannot be generated by NS_1 since any word of $L(NS_1, M_{f1})$ must include an instance of the transition t_i^{zc} in order to remove the token at place p. But such a transition obviously gives rise to a simulation that cheats which is assumed not to be the case.

Next suppose that the two-counter machine K does not halt on the initial values $n1$ and $n2$. Let $l_1(u) \in L(NS_1, M_{f1})$ for some firing sequence u from M_{01} to M_{f1}. This sequence must contain at least (and therefore exactly) one instance of t_i^{zc} which we now replace by t_i^z thus obtaining a firing sequence v of NS_2 with $l_1(u) = l_2(v)$. It is readily seen that $M_{02}[v\!>\!M_{f2}$ and therefore $l_2(v) \in L(NS_2, M_{f2})$ giving the inclusion $L(NS_1, M_{f1}) \subseteq L(NS_2, M_{f2})$. For the other inclusion suppose $l_2(v) \in L(NS_2, M_{f2})$ where v is some firing sequence from M_{02} to M_{f2}. Then v must include a transition t_i^z giving rise to a simulation that cheats, i.e. the place representing the corresponding counter is not empty at the marking where t_i^z fires (since K does not halt). Replace a single such occurrence of t_i^z along v by t_i^{zc} thus obtaining a firing sequence u with $l_1(u) = l_2(v)$. Moreover, u is a firing sequence of NS_1 from M_{01} to M_{f1} and therefore $l_1(u) \in L(NS_1, M_{f1})$. Hence we have the inclusion $L(NS_2, M_{f2}) \subseteq L(NS_1, M_{f1})$ as well and this completes the proof. ■

Notice the important rôle played by the final marking M_{f1} of NS_1; by requiring the place p to be empty in order for a word to be generated we *force* an instance of t_i^{zc} to fire, i.e. we force a simulation that cheats.

We note that the net systems constructed are deterministic; NS_2 is clearly so since it is labelled by an injective labelling map. On the other hand, the labelling map of NS_1 assigns the *same* label to the transitions t_i^z and t_i^{zc}, and might therefore cause nondeterminism. However, it is not hard to see that whenever t_i^z and t_i^{zc} may fire at a marking M then the resulting markings are bisimilar.

Corollary 3.23 *The equivalence problem for* $\mathrm{L}(\mathrm{PN_d})$ *is undecidable.*

Proof: By Lemma 3.22 we have effectively reduced the Halting problem for two-counter machines to the equivalence problem for $\mathrm{L}(\mathrm{PN_d})$. ∎

One final remark before we end this chapter. Hack proved in [Hac75a] that for unlabelled Petri nets (or equivalently, Petri nets labelled with injective labelling maps) the equivalence problem for the class of terminal languages is reducible to the reachability problem and therefore decidable. By the above result we see that as soon as the labelling map is not injective (even though the resulting Petri nets are still deterministic) the equivalence problem becomes undecidable.

Chapter 4

Decidability of $\sim$ for all Context-Free Processes

In [BBK87] Baeten, Bergstra and Klop proved the remarkable result that bisimulation equivalence is decidable on the class of *normed* context-free processes. Their proof is rather lengthy and hard to grasp; it ultimately relies on showing a periodicity for any transition graph generated from normed context-free processes. Caucal presented in [Cau90] a more elegant (and shorter) proof of the same result utilising rewrite techniques. Finally, in [HS91] Hüttel and Stirling presented yet another proof of the decidability result by appealing to the tableau method. The tableau based approach also supports a sound and complete sequent-based equational theory for normed context-free processes (see [HS91,Hüt91]).

One remaining question to be answered is whether bisimulation equivalence is decidable for the full class of context-free processes. In this chapter we answer this question in the affirmative, using a technique inspired by Caucal's proof of the decidability of language equivalence for simple algebraic grammars (see [Cau86]). This chapter is based on the paper [CHS92] with Hüttel and Stirling.

In the first section we introduce an alternative characterisation of bisimulation equivalence, namely via a sequence of approximations, which will enable us to conclude semi-decidability of bisimulation inequivalence on the class of guarded context-free processes. Thus we only need to consider semi-decidability of bisimulation equivalence in order to establish our result. This is achieved in the following (and final) section through a finite representability result; here the emphasis is on *decomposition* of pairs of bisimilar processes into "smaller" pairs of bisimilar processes such that only finitely many interesting pairs of bisimilar processes cannot be decomposed further.

4.1 Semi-decidability of $\not\sim$

In order to establish our decidability result we shall consider an alternative characterisation of bisimulation equivalence which will enable us to deduce semi-decidability of bisimulation inequivalence. The alternative characterisation is given via a sequence of approximations.

Definition 4.1 *The sequence of bisimulation approximations $\{\sim_n\}_{n=0}^{\infty}$ is defined inductively by $E \sim_0 F$ for all processes E and F, and $E \sim_{n+1} F$ iff for each label a we have that*

- *$E \xrightarrow{a} E'$ implies $F \xrightarrow{a} F'$ for some F' with $E' \sim_n F'$, and*
- *$F \xrightarrow{a} F'$ implies $E \xrightarrow{a} E'$ for some E' with $E' \sim_n F'$.*

It is a standard result, see [Mil89] for instance, that for any image-finite labelled transition graph (we remind the reader that these are the transition graphs for which $\{F : E \xrightarrow{a} F\}$ is finite for each state E and label a):

$$\sim \;\; = \;\; \bigcap_{n=0}^{\infty} \sim_n .$$

Certainly for image-finite transition graphs the n-equivalence problem (whether or not $E \sim_n F$) is decidable: $\sim_0$ is decidable and if $\sim_n$ is decidable then $\sim_{n+1}$ becomes decidable since checking $E \sim_{n+1} F$ only requires examining membership of $\sim_n$ for finitely many pairs of processes (as E and F are image-finite). Hence, for image-finite labelled transition graphs bisimulation inequivalence is semi-decidable via the simple procedure which seeks the least n such that $E \not\sim_n F$; if $E \not\sim F$ is the case then eventually we will find an n such that $E \not\sim_n F$.

Clearly, the transition graph for any family Δ of guarded context-free processes is image-finite. Thus, due to the above observation, we only need to establish semi-decidability of bisimulation equivalence in order to claim our result. Our proof of this relies on showing that $\sim$ is generated from a *finite* self-bisimulation.

4.2 Semi-decidability of $\sim$

We assume a fixed family Δ of BPA_0 process equations in Greibach Normal Form (see Section 2.5.1 for a definition). The bisimulation equivalence problem is whether or not $\alpha \sim \beta$ where $\alpha, \beta \in Var(\Delta)^*$. According to the observation of the previous section, it is enough to establish semi-decidability of bisimulation equivalence. The proof of this (inspired by [Cau86,Cau88,Cau90]) relies on showing that there is a *finite* self-bisimulation which generates the bisimulation relation $\sim$ on $Var(\Delta)^*$.

We formally introduced the notion of a self-bisimulation in Section 2.5.4 and also pointed out that such a relation constitutes a *witness* for bisimilarity, i.e. if R is a self-bisimulation on $Var(\Delta)^*$ then $R^{\leftrightarrow^*} \subseteq \sim$. Here our aim is to show that there exists a *finite* self-bisimulation R on $Var(\Delta)^*$ such that $R^{\leftrightarrow^*} = \sim$. From this it clearly follows that $\alpha \sim \beta$ if, and only if, there exists a finite self-bisimulation R such that $(\alpha, \beta) \in R$. Once this is established we shall be able to show semi-decidability of bisimulation equivalence.

Let us at this point explain the intuition. Since our proof of $R^{\leftrightarrow^*} = \sim$ is non-constructive, i.e. we prove the existence of R but cannot construct it (not even provide an upper bound on the size of R), it is not immediately clear how the semi-decidability result is established. However, if the property of being self-bisimilar itself is semi-decidable then we shall have our result since in these circumstances we may *dovetail* the procedure for checking self-bisimilarity with the search through all finite binary relations on $Var(\Delta)^*$; if $\alpha \sim \beta$ then we shall eventually find a self-bisimulation containing the pair (α, β), at which point we know that $\alpha \sim \beta$.

In order to demonstrate that the property of being self-bisimilar is semi-decidable it is clearly enough to show that membership of $R^{\leftrightarrow^*}$ for any finite binary relation R on $Var(\Delta)^*$ is semi-decidable. This is indeed the case and may be illustrated by analogy with semi-decidability of validity in the area of logic. If a given theory T is completely axiomatised by a finite set of axioms AX then validity becomes semi-decidable since one may start by building proofs (using the axioms AX) in some order. If the formula F is valid then eventually a proof will be built for F, at which point we know that F is valid. In a similar fashion, if (α, β) is given and we want to check whether $(\alpha, \beta) \in R^{\leftrightarrow^*}$, then viewing R as axioms, we may start building "proofs", i.e. equivalences, by using the usual rules for equational reasoning (reflexivity, symmetry and transitivity) together with the rules for congruence wrt sequential composition. If $(\alpha, \beta) \in R^{\leftrightarrow^*}$ then eventually we shall have a proof for $\alpha = \beta$, at which point we know that $(\alpha, \beta) \in R^{\leftrightarrow^*}$.

Caucal obtained in [Cau90] decidability of bisimilarity on normed context-free processes by showing $R^{\leftrightarrow^*} = \sim$ for a finite self-bisimulation R. However, in this case R is of a simple nature; viewing R as a rewrite system, it is confluent and well-founded, i.e. $R^{\leftrightarrow^*}$ is decidable, and there are only finitely many different relations R to be considered (see also the discussion in Section 2.5.5). Thus in these circumstances decidability of $\sim$ follows from a single procedure searching for the self-bisimulation satisfying $(\alpha, \beta) \in R^{\leftrightarrow^*}$ where $\alpha, \beta \in Var(\Delta)^*$ are the processes we wish to compare.

4.2.1 Decomposition for Context-Free Processes

In order to show $R^{\leftrightarrow^*} = \sim$ for a finite self-bisimulation we shall be interested in decomposing pairs of bisimilar processes (α, β) into a sequence of "smaller" pairs $(\alpha_1, \beta_1) \dots (\alpha_m, \beta_m)$ with $\alpha_i \sim \beta_i$ for each $i = 1, 2 \dots m$ such that there are only finitely many interesting pairs of bisimilar processes that cannot be decomposed further. The notion of decomposition that we shall explore was formally introduced in Section 2.5.5 and we refer the reader to Definition 2.43. However, further definitions and some preliminary results are needed.

We divide the variable set $Var(\Delta)$ into disjoint subsets as follows: V_0 denote the set $\{X \in Var(\Delta) : X \text{ is normed}\}$ and $V_1 = Var(\Delta) \setminus V_0$. The following example contains two variables X and Y with X being unnormed while Y is normed. Thus in this case we have $V_0 = \{Y\}$ and $V_1 = \{X\}$.

Example 4.2 *In the family of* $\mathrm{BPA}_{\mathbf{0}}$ *equations* $\Delta = \{X \stackrel{\text{def}}{=} aX, Y \stackrel{\text{def}}{=} c + aX\}$ *the variable* X *is not normed since there is no* $w \in \Lambda_1^*$ *such that* $X \xrightarrow{w} \mathbf{0}$ *whereas* Y *is normed with* $\mathcal{N}(Y) = 1$.

A straightforward consequence of the definition of having a norm is the following: if $X \in V_1$ then $\alpha X \beta \sim \alpha X$ for any $\beta \in \mathit{Var}(\Delta)^*$. Therefore we can assume that our fixed family of $\mathrm{BPA}_{\mathbf{0}}$ equations in Greibach Normal Form

$$\Delta \;=\; \left\{X_i \stackrel{\text{def}}{=} \sum_{j=1}^{n_i} a_{ij}\alpha_{ij} : i = 1, 2 \ldots n\right\}$$

has the property that each $\alpha_{ij} \in (V_0^* V_1) \cup V_0^*$.

The next lemma is crucial for our result that only finitely many interesting pairs of bisimilar processes cannot be decomposed further. The lemma was first proved by Stirling. The proof presented here is somewhat simplified and is due to Moller.

Lemma 4.3 *For any* $\alpha, \beta \in \mathit{Var}(\Delta)^*$, *if* $\alpha \not\sim \beta$ *then there are at most finitely many different* $\gamma \in \mathit{Var}(\Delta)^*$ *(up to bisimilarity) such that* $\alpha\gamma \sim \beta\gamma$.

Proof: We proceed by contradiction thus assuming that there are infinitely many different $\gamma \in \mathit{Var}(\Delta)^*$ (up to bisimilarity) such that $\alpha\gamma \sim \beta\gamma$. By induction on m we show that $\alpha \sim_m \beta$ for all m.

Clearly $\alpha \sim_0 \beta$ by definition of $\sim_0$. Assume $m = 1$ and that $\alpha \xrightarrow{a} \alpha'$ for some $a \in \Lambda_1$ and $\alpha' \in \mathit{Var}(\Delta)^*$. We must show that $\beta \xrightarrow{a} \beta'$ for some $\beta \in \mathit{Var}(\Delta)^*$. But if this is not the case then $\alpha\gamma \sim \beta\gamma$ can only be satisfied if $\mathit{isnil}(\beta)$. However, then we must have $\alpha\gamma \sim \gamma$ and by uniqueness of solutions to guarded equations there can only be one γ (up to bisimilarity) satisfying $\alpha\gamma \sim \gamma$. By symmetry we now conclude that $\alpha \sim_1 \beta$.

For the induction step we argue as follows. We have $\alpha\gamma \sim \beta\gamma$ for infinitely many different γ (up to bisimilarity). Assume $\alpha \xrightarrow{a} \alpha'$. For each of the infinitely many γ we then have $\alpha\gamma \xrightarrow{a} \alpha'\gamma$ and therefore $\beta\gamma \xrightarrow{a} \beta'\gamma$ for some β' with

$\alpha'\gamma \sim \beta'\gamma$. Since $\{\beta' : \beta \xrightarrow{a} \beta'\}$ is a finite set we can find β' with $\beta \xrightarrow{a} \beta'$ such that $\alpha'\gamma \sim \beta'\gamma$ for infinitely many different γ (up to bisimilarity). Thus by induction we conclude that $\alpha' \sim_m \beta'$. By symmetry we now have $\alpha \sim_{m+1} \beta$ and this completes the proof. ∎

We say that the pairs $(X\alpha, Y\beta)$ and $(X\alpha_1, Y\beta_1)$ are *distinct* when $\alpha \not\sim \alpha_1$ or $\beta \not\sim \beta_1$. The next surprising result shows that there are only finitely many distinct pairs $(X\alpha, Y\beta)$ that are not decomposable.

Lemma 4.4 *For any $X, Y \in Var(\Delta)$, any binary relation R over $(V_0^*V_1) \cup V_0^*$ of the form*

$$\Big\{(X\alpha, Y\beta) : X\alpha \sim Y\beta \textit{ and } (X\alpha, Y\beta) \textit{ is not decomposable}\Big\}$$

such that all pairs are distinct, is finite.

Proof: First of all, if both X and Y belong to V_1 then R contains just one member. Secondly, assume one of them is in V_1, without loss of generality let this be X. As Y is normed let $\mathcal{N}(Y) = n$. Therefore $Y \xrightarrow{w} \mathbf{0}$ for some w of length n. But there are only finitely many γ such that $X \xrightarrow{w} \gamma$. If R were infinite containing pairs $(X, Y\beta_i)$ for all i then we could find a β_j such that $\beta_j \sim \beta_k$ for infinitely many k which would contradict distinctness. Finally, assume that both $X, Y \in V_0$ and without loss of generality let $\mathcal{N}(X) \le \mathcal{N}(Y)$ with $\mathcal{N}(X) = n$. Consider a word w of length n such that $X \xrightarrow{w} \mathbf{0}$. Since $X\alpha_i \sim Y\beta_i$ for all $(X\alpha_i, Y\beta_i) \in R$ we must have $Y \xrightarrow{w} \gamma$ for some γ. But then consider the set $B = \{\gamma : Y \xrightarrow{u} \gamma, length(u) = n\}$ which is finite. Suppose R contains infinitely many distinct pairs. Then for some $\gamma \in B$, since $\alpha_i \sim \gamma\beta_i$, it must be the case that $X\gamma\beta_i \sim Y\beta_i$ for infinitely many different β_i (up to bisimilarity). But this is impossible by Lemma 4.3, as $X\gamma \not\sim Y$ follows from the assumption that the pairs are not decomposable. ∎

4.2.2 Finite Representability of $\sim$

We are now almost in a position to prove our main theorem, which relies on an induction based on the norm of sequences of variables, denoted by $\mathcal{N}'(\alpha)$, and defined for every $\alpha \in (V_0^*V_1) \cup V_0^*$ as follows:

$$\mathcal{N}'(\alpha X) \;=\; \begin{cases} \mathcal{N}(\alpha X) & \text{if } X \in V_0 \\ \mathcal{N}(\alpha) & \text{otherwise.} \end{cases}$$

We let $\sqsubseteq$ denote the ordering on $(V_0^* V_1) \cup V_0^* \times (V_0^* V_1) \cup V_0^*$ defined as follows: $(\alpha_1, \alpha_2) \sqsubseteq (\beta_1, \beta_2)$ iff $\max\left\{\mathcal{N}'(\alpha_1), \mathcal{N}'(\alpha_2)\right\} \leq \max\left\{\mathcal{N}'(\beta_1), \mathcal{N}'(\beta_2)\right\}$. By $\sqsubset$ we denote the ordering defined by $(\alpha_1, \alpha_2) \sqsubset (\beta_1, \beta_2)$ iff $(\alpha_1, \alpha_2) \sqsubseteq (\beta_1, \beta_2)$ and furthermore $\max\left\{\mathcal{N}'(\alpha_1), \mathcal{N}'(\alpha_2)\right\} \neq \max\left\{\mathcal{N}'(\beta_1), \mathcal{N}'(\beta_2)\right\}$. Clearly $\sqsubset$ is well-founded, i.e. the ordering has no infinite decreasing chain.

Theorem 4.5 *There is a finite binary relation R over $(V_0^* V_1) \cup V_0^*$ such that $R^{\leftrightarrow^*} = \sim$.*

Proof: We define R as the union of two finite relations R_1 and R_2. R_1 is a largest binary relation over V_0^* of the form $\left\{(X, \alpha) : X \sim \alpha\right\}$ and R_2 is a largest binary relation over $(V_0^* V_1) \cup V_0^*$ of the form

$$\left\{(X\alpha, Y\beta) : X\alpha \sim Y\beta \text{ and } (X\alpha, Y\beta) \text{ is not decomposable}\right\}$$

such that all pairs in R_2 are distinct. Moreover, we assume *minimal* elements wrt our ordering $\sqsubseteq$, i.e. if $(X\alpha, Y\beta) \in R_2$ is *not* distinct from $(X\alpha', Y\beta')$ then we have $(\alpha, \beta) \sqsubseteq (\alpha', \beta')$. Notice that both R_1 and R_2 are finite; finiteness of R_1 follows from the fact that there are only finitely many elements of V_0^* with a given finite norm (norm is additive under sequential composition) and finiteness of R_2 follows from Lemma 4.4. Thus R is finite.

We now want to show that $R^{\leftrightarrow^*} = \sim$. As $R \subseteq \sim$ and $\sim$ is a congruence wrt sequential composition we immediately have $R^{\leftrightarrow^*} \subseteq \sim$. So we consider proving $\sim \subseteq R^{\leftrightarrow^*}$ and proceed by induction on $\sqsubset$. Let $X\alpha \sim Y\beta$ for processes $X\alpha$ and $Y\beta$ of $(V_0^* V_1) \cup V_0^*$. There are two cases:

(i) Suppose that $(X\alpha, Y\beta)$ is not decomposable. Then by the maximality of R_2 we have $(X\alpha', Y\beta')$ in R_2 such that $(\alpha', \beta') \sqsubseteq (\alpha, \beta)$ with $\alpha \sim \alpha'$ and $\beta \sim \beta'$. If $X, Y \in V_0$ then clearly $(\alpha', \beta') \sqsubseteq (\alpha, \beta) \sqsubset (X\alpha, Y\beta)$ from which we have $(\alpha, \alpha') \sqsubset (X\alpha, Y\beta)$ and $(\beta, \beta') \sqsubset (X\alpha, Y\beta)$. By the induction hypothesis it follows that $(\alpha, \alpha') \in R^{\leftrightarrow^*}$ and $(\beta, \beta') \in R^{\leftrightarrow^*}$ from which we get

$(X\alpha, Y\beta) \in R^{\leftrightarrow^*}$ as required. If $X \in V_1$ and $Y \in V_0$ then we get $\alpha = \alpha' = \epsilon$ and therefore $X \sim Y\beta$. As $Y \in V_0$ it follows that $\mathcal{N}'(\beta') \leq \mathcal{N}'(\beta) < \mathcal{N}'(Y\beta)$ hence we have $(\beta, \beta') \sqsubset (X, Y\beta)$ which by the induction hypothesis implies $(\beta, \beta') \in R^{\leftrightarrow^*}$. But then $(X, Y\beta) \in R^{\leftrightarrow^*}$ as required. Finally, if we have $X, Y \in V_1$ then $\alpha = \alpha' = \epsilon$ and also $\beta = \beta' = \epsilon$. Hence we have $(X, Y) \in R_2$ from which $(X, Y) \in R^{\leftrightarrow^*}$ clearly follows.

(ii) Suppose $(X\alpha, Y\beta)$ is decomposable. By the definition of decomposability it follows that $X, Y \in V_0$. Assume without loss of generality that we have γ such that $\gamma\alpha \sim \beta$ and $X \sim Y\gamma$. As X is normed and $X \sim Y\gamma$ clearly $\mathcal{N}'(\gamma\alpha) < \mathcal{N}'(X\alpha)$. Similarly, as Y is normed, we also have $\mathcal{N}'(\beta) < \mathcal{N}'(Y\beta)$ and therefore $(\gamma\alpha, \beta) \sqsubset (X\alpha, Y\beta)$ from which $(\gamma\alpha, \beta) \in R^{\leftrightarrow^*}$ follows by the induction hypothesis. As $X \sim Y\gamma$ with $X \in V_0$ we have $(X, Y\gamma) \in R_1$ from the maximality of R_1. But then $(X\alpha, Y\beta) \in R^{\leftrightarrow^*}$ as required.

This completes the proof. ∎

A simple consequence of the above theorem is the following.

Corollary 4.6 *Let $\alpha, \beta \in (V_0^* V_1) \cup V_0^*$. Then $\alpha \sim \beta$ if, and only if, there exists a* finite *self-bisimulation on $(V_0^* V_1) \cup V_0^*$ containing the pair (α, β).*

Proof: The finite relation R of Theorem 4.5 is clearly a self-bisimulation. If $\alpha \sim \beta$ then, since $R^{\leftrightarrow^*} = \sim$, we conclude that $R \cup \{(\alpha, \beta)\}$ is a self-bisimulation as well. Conversely, if $R \cup \{(\alpha, \beta)\}$ is a finite self-bisimulation the $\alpha \sim \beta$ since any self-bisimulation is a witness for bisimilarity. ∎

We now show that the above corollary is sufficient for semi-decidability of $\sim$. For given any finite binary relation R on $(V_0^* V_1) \cup V_0^*$ it is semi-decidable whether it is a self-bisimulation. The procedure consists in defining a derivation or proof system: the axioms are the pairs in R, and the rules are congruence rules for sequential composition together with the usual equivalence rules. Consequently, for each n let $D_n(R)$ be the finite set of pairs (α, β) which are derivable within n steps of the proof system.

Definition 4.7 *A finite binary relation R on $(V_0^* V_1) \cup V_0^*$ is called an* n-self-bisimulation *iff $\alpha R \beta$ implies for all $a \in \Lambda_1$ that*

- *if $\alpha \xrightarrow{a} \alpha'$ then $\beta \xrightarrow{a} \beta'$ for some β' with $(\alpha', \beta') \in D_n(R)$, and*
- *if $\beta \xrightarrow{a} \beta'$ then $\alpha \xrightarrow{a} \alpha'$ for some α' with $(\alpha', \beta') \in D_n(R)$.*

For each n clearly it is decidable whether a finite binary relation R on $(V_0^* V_1) \cup V_0^*$ is an n-self-bisimulation. Moreover, if R is a finite self-bisimulation then for some n it is an n-self-bisimulation.

We now complete the proof that bisimulation equivalence is semi-decidable using a dovetailing technique (compare [Cau86]). Let $R_0, R_1 \dots R_i \dots$ be an effective enumeration list of all finite binary relations on $(V_0^* V_1) \cup V_0^*$ and let $g : I\!N \rightarrow I\!N^2$ be an effective bijection. To check whether $\alpha \sim \beta$, for each $n \geq 0$ in turn consider the pair $(i, j) = g(n)$: if $(\alpha, \beta) \in R_i$ then test if R_i is a j-self-bisimulation. Consequently, if $\alpha \sim \beta$ this must be established at the n'th stage of this procedure for some n. The decidability result is now established.

Theorem 4.8 *Bisimulation equivalence is decidable for all guarded context-free processes.*

As the proof of decidability involves two semi-decision procedures it is not obvious how to determine the complexity of solving this problem. Moreover it does not provide us with an intuitive technique for deciding bisimilarity as does the tableau method in [HS91] which also has the advantage of providing us with a way of extracting a sound and complete (sequent-based) equational theory for normed context-free processes. It is still an open question how to extend this result to the full class of context-free processes.

Chapter 5

Unique Decompositions

In this chapter we deal with the question of expressing a given process as a parallel composition of other (simpler) processes. We present a proof of unique decomposition for the class of normed BPP processes wrt bisimulation equivalence. The proof technique is essentially due to Milner and the observation that this technique can be employed on the class of normed BPP processes is due to Hirshfeld and Moller (see [CHM93]). As a consequence we obtain a cancellation law for the class of normed BPP processes wrt bisimulation equivalence. We then proceed by extending the unique decomposition result to allow synchronisation between processes, thus replacing the pure interleaving operator ($\|$) with parallel composition ($|$). Again as a consequence we obtain a cancellation law wrt bisimulation equivalence. We shall use the property of unique decomposition in Chapter 6 in order to show decidability of bisimilarity on normed BPP$_\tau$ (as well as BPP): we shall check bisimilarity between processes by first decomposing and then comparing the individual prime components.

We know that unique decomposition does not hold for bisimilarity wrt unnormed processes. For distributed bisimilarity the situation is different; we prove a unique decomposition result for *all* of BPP wrt distributed bisimulation equivalence. We shall also prove a theorem of degeneration, i.e. a theorem stating that every prime component is contained (up to distributed bisimilarity) in the process being decomposed. Finally, these results will be extended to BPP$_\tau$. We shall in

Chapter 7 use the property of unique decomposition to show decidability of distributed bisimilarity on BPP_τ (as well as BPP). Also in Chapter 7 we shall use the property to obtain a sound and complete equational theory for distributed bisimilarity on a subset of BPP_τ where general summation is replaced by guarded summation.

5.1 Unique Decomposition for normed BPP wrt $\sim$

We begin by showing unique decomposition for bisimilarity on normed BPP processes. As mentioned, the proof is to a large extent similar to Milner's proof of unique decomposition for finite processes, i.e. processes with finite acyclic transition graphs (see [Mol89]).

Definition 5.1 *A process E of* BPP *is* prime *(wrt bisimilarity) iff $E \not\sim \mathbf{0}$ and whenever $E \sim F \| G$ we have that either $F \sim \mathbf{0}$ or $G \sim \mathbf{0}$.*

Example 5.2 *Let $a, b \in \Lambda_1$. Then a is prime whereas $ab + ba$ is not since this process is bisimilar to $a \| b$.*

For our proof, we rely on the following easily verified properties of the norm of processes (see Definition 2.10 for the notion of norm of processes).

Lemma 5.3 *Let E and F be processes of* BPP*. Then*

- *$E \sim F$ implies $\mathcal{N}(E) = \mathcal{N}(F)$,*
- *$E \not\sim \mathbf{0}$ and F normed implies $\mathcal{N}(E\|F) > \mathcal{N}(F)$, and*
- *E normed implies $E \xrightarrow{a} F$ for some $a \in \Lambda_1$ and F such that $\mathcal{N}(E) > \mathcal{N}(F)$.*

Theorem 5.4 (Unique Decomposition) *Any normed process E of* BPP *can be expressed uniquely, up to $\sim$, as a product (via the combinator $\|$) of primes.*

Proof: That any normed process E of BPP can be expressed as a product of primes is straightforward: if $E \sim \mathbf{0}$, it is equal to the empty product; if E is prime, it is equal to the singleton product, namely itself; otherwise $E \sim F \| G$ where $F, G \not\sim \mathbf{0}$ are of lesser norm, so by induction on norms, each of F and G can be expressed as a product of primes:

$$F \sim F_1 \| F_2 \| \cdots \| F_m, \qquad G \sim G_1 \| G_2 \| \cdots \| G_n.$$

Then E can be expressed as a product of primes itself by:

$$E \sim F \| G \sim F_1 \| F_2 \| \cdots \| F_m \| G_1 \| G_2 \| \cdots \| G_n.$$

The proof that this decomposition is unique proceeds by induction on $\mathcal{N}(E)$.

Suppose that $E \sim F$, but that E and F are distinct decompositions into products of primes given by

$$\begin{aligned} E &\equiv A_1^{k_1} \| A_2^{k_2} \| \cdots \| A_n^{k_n}, \\ F &\equiv A_1^{l_1} \| A_2^{l_2} \| \cdots \| A_n^{l_n}, \end{aligned}$$

where the A_i's are distinct primes (that is, $i \neq j$ implies $A_i \not\sim A_j$), and that $k_i, l_i \geq 0$.

Assume that all processes G with $\mathcal{N}(G) < \mathcal{N}(E) = \mathcal{N}(F)$ have a unique decomposition into a product of primes, and let $\exp(A, G)$ be the *exponent* of prime A (the number of times A appears) in the unique decomposition of such a G.

Let m be chosen such that $k_m \neq l_m$, and that $\mathcal{N}(A_j) > \mathcal{N}(A_m)$ implies that $k_j = l_j$; that is, A_m is a maximal-sized (wrt norm) prime appearing in the decomposition of E or F in which the exponents differ. Without loss of generality, we can assume that $k_m > l_m$ (otherwise exchange the rôles of E and F). The proof now proceeds by cases on the possible forms of the decomposition of E.

First suppose E is a power of a single prime: $E \equiv A_m^{k_m}$. First of all, if E is prime (that is, $k_m = 1$), then from $E \sim F$, we have that F is prime, and since $k_m > l_m$, we have that $F \equiv A_j$ for some j different from m. But then $A_m \sim A_j$, contradicting the distinctness assumption on the A_i's. Hence we must have that $k_m > 1$.

We now want to show that $l_m > 0$ leads to a contradiction. So suppose $l_m > 0$. Then we can do $F \xrightarrow{a} F'$ for some $a \in \Lambda_1$ and F' with $\mathcal{N}(F') < \mathcal{N}(F)$, and have a unique decomposition for F' in which $\exp(A_m, F') = l_m - 1$. But whenever we have $E \xrightarrow{a} E'$ for some $a \in \Lambda_1$ and E' with $\mathcal{N}(E') < \mathcal{N}(E)$ then it follows that $\exp(A_m, E') = k_m - 1 > l_m - 1$. Therefore, there is no E' with $E \xrightarrow{a} E'$ such that $E' \sim F'$ contradicting the assumption that $E \sim F$. We are thus lead to conclude that $l_m = 0$.

Next we want to show that $l_m = 0$ also leads to a contradiction. From the maximality constraint in the definition of m it follows that if $l_i > 0$ then (because $k_i = 0$) we must have $\mathcal{N}(A_i) \leq \mathcal{N}(A_m)$. Hence whenever $F \xrightarrow{a} F'$ for some $a \in \Lambda_1$ and F' with $\mathcal{N}(F') < \mathcal{N}(F)$ then F' has a unique decomposition in which $\exp(A_m, F') = 0$. However, as $k_m > 1$, we can do $E \xrightarrow{a} E'$ for some $a \in \Lambda_1$ and E' with $\mathcal{N}(E') < \mathcal{N}(E)$, where E' has a unique decomposition in which $\exp(A_m, E') = k_m - 1 > 0$. Therefore, there is no process F' with $F \xrightarrow{a} F'$ such that $E' \sim F'$ contradicting the assumption that $E \sim F$.

Next suppose E is not a power of a single prime, i.e. there exists j different from m such that $k_j > 0$.

Let $a \in \Lambda_1$ and G be chosen such that $E \xrightarrow{a} G$ with $\mathcal{N}(G) < \mathcal{N}(E)$, and whenever $E \xrightarrow{b} E'$ for some $b \in \Lambda_1$ and E' with $\mathcal{N}(E') < \mathcal{N}(E)$, then (since E' and G have unique decompositions) $\exp(A_m, E') \leq \exp(A_m, G)$. That is, $E \xrightarrow{a} G$ is chosen to be a norm reducing move which results in G having a maximal number of prime A_m in its unique decomposition. By the existence of at least one copy of A_j in the decomposition of E we have $\exp(A_m, G) \geq k_m$. We want to show that F cannot match the move $E \xrightarrow{a} G$.

Let $F \xrightarrow{a} F'$ for some F' with $\mathcal{N}(F') < \mathcal{N}(F)$; such an F' must exist, assuming $E \sim F$. Then there exists a prime A_s of F and process H such that $A_s \xrightarrow{a} H$ with $\mathcal{N}(H) < \mathcal{N}(A_s)$, where

$$F \xrightarrow{a} F' \quad \equiv \quad A_1^{l_1} \| \cdots \| A_s^{l_s - 1} \| \cdots \| A_n^{l_n} \| H.$$

If $F' \sim G$ then we must have $\exp(A_m, F') = \exp(A_m, G) \geq k_m > l_m$ which implies that $\exp(A_m, H) > 0$. Therefore $\mathcal{N}(A_s) > \mathcal{N}(H) \geq \mathcal{N}(A_m)$, so by the maximality constraint in the definition of m, we have that $k_s = l_s$ (and $s \neq m$). However now, since $k_s = l_s > 0$, we also have $E \xrightarrow{a} E'$ for some E' with $\mathcal{N}(E') < \mathcal{N}(E)$ such that $\exp(A_m, E') = k_m + \exp(A_m, H) > l_m + \exp(A_m, H) = \exp(A_m, F')$. Hence we have by the choice of G, $\exp(A_m, G) \geq \exp(A_m, E') > \exp(A_m, F')$ and therefore there cannot exists an F' with $F \xrightarrow{a} F'$ such that $F' \sim G$ contradicting the assumption that $E \sim F$. This completes the proof. ■

An important corollary of this theorem is.

Corollary 5.5 (Cancellation Law) *For normed processes E, F and G of* BPP *it follows that $E \| G \sim F \| G$ implies $E \sim F$.*

Proof: Suppose that E is decomposed into $A_1, A_2 \ldots A_n$, that F is decomposed into $B_1, B_2 \ldots B_m$ and finally that G is decomposed into $C_1, C_2 \ldots C_k$. Then $E \| G$ has the decomposition:

$$A_1 \| A_2 \| \cdots \| A_n \| C_1 \| C_2 \| \cdots \| C_k,$$

and $F \| G$ has the decomposition:

$$B_1 \| B_2 \| \cdots \| B_m \| C_1 \| C_2 \| \cdots \| C_k.$$

As $E \| G \sim F \| G$ we have from the theorem of unique decomposition that the above two decompositions must be identical. But then also $A_1 \| A_2 \| \cdots \| A_n$ and $B_1 \| B_2 \| \cdots \| B_m$ must be identical decompositions, hence $E \sim F$ as required. ■

5.2 Unique Decomposition for normed BPP_τ wrt $\sim$

We now extend the result of the previous section thus showing that unique decomposition also holds for the class of normed BPP_τ, i.e. replacing the full merge with parallel composition. The setup to the theorem is as in the previous section; notably, the definition of prime is as before with $\|$ being replaced by $|$. However, to facilitate the proof we need further definitions and some preliminary results.

Let E be a process of BPP_τ. If $E \xrightarrow{\mu} E'$ such that $\mu \neq \tau$ then we call this move *visible*; otherwise it is called *invisible*. If $E \xrightarrow{\tau} E'$ and this transition has been inferred without using the inference rule for synchronisation then the move is referred to as an *inactive* synchronisation; otherwise it is an *active* synchronisation. So for instance $\tau E \xrightarrow{\tau} E$ represents an inactive synchronisation while $aE|\bar{a}F \xrightarrow{\tau} E|F$ represents an active synchronisation.

As in the previous section our proof of unique decomposition proceeds by induction on the norm of processes. One important property on norm which we shall utilise is that it is additive under parallel composition (see Chapter 2, Proposition 2.11). A simple consequence of this result is the following.

Proposition 5.6 *Let* E, F *be normed processes of* BPP_τ. *If* $\mathcal{N}(E'|F') < \mathcal{N}(E|F)$ *then* $\mathcal{N}(E') < \mathcal{N}(E)$ *or* $\mathcal{N}(F') < \mathcal{N}(F)$.

Proof: By contradiction, suppose that $\mathcal{N}(E') \geq \mathcal{N}(E)$ and also $\mathcal{N}(F') \geq \mathcal{N}(F)$. But then $\mathcal{N}(E'|F') = \mathcal{N}(E') + \mathcal{N}(F') \geq \mathcal{N}(E) + \mathcal{N}(F) = \mathcal{N}(E|F)$ since norm is additive under parallel composition. However, this contradicts the assumption that $\mathcal{N}(E'|F') < \mathcal{N}(E|F)$. ∎

The proof of unique decomposition furthermore relies on the following easily verified properties of the norm of processes.

Lemma 5.7 *Let* E *and* F *be processes of* BPP_τ. *Then we have the following.*

- $E \sim F$ *implies* $\mathcal{N}(E) = \mathcal{N}(F)$.
- E *normed implies* $E \xrightarrow{\mu} F$ *for some* $\mu \in Act$ *and* F *with* $\mathcal{N}(F) < \mathcal{N}(E)$.
- *If* E *is normed and* $E \xrightarrow{a} F$ *for some* $a \in Act$ *and* F *with* $\mathcal{N}(F) < \mathcal{N}(E)$ *then* $\mathcal{N}(E) = \mathcal{N}(F) + 1$.
- *Suppose* E *is normed. Furthermore assume for all* $a \in Act$ *and* F *that* $E \xrightarrow{a} F$ *implies* $\mathcal{N}(F) \geq \mathcal{N}(E)$. *Then there exists a process* G *such that* $E \xrightarrow{\tau} G$ *with* $\mathcal{N}(E) = 2 + \mathcal{N}(G)$.

We are now ready to prove the theorem of unique decomposition. The proof is to a large extent similar to the unique decomposition proof of the previous section. However, in our case analysis we must take more care since processes may now synchronise.

Theorem 5.8 (Unique Decomposition) *Let E be a normed process of* BPP_τ. *Then E can be expressed uniquely, up to $\sim$, as a parallel composition of primes.*

Proof: That any normed process E of BPP_τ can be expressed as a parallel composition of primes is straightforward. The proof that this decomposition is unique proceeds by induction on $\mathcal{N}(E)$.

Suppose that $E \sim F$, but that E and F are distinct decompositions into products of primes given by

$$\begin{aligned} E &\equiv A_1^{k_1} | A_2^{k_2} | \cdots | A_n^{k_n}, \\ F &\equiv A_1^{l_1} | A_2^{l_2} | \cdots | A_n^{l_n}, \end{aligned}$$

where the A_i's are distinct primes (that is, $i \neq j$ implies $A_i \not\sim A_j$), and that $k_i, l_i \geq 0$.

By induction, all processes G with $\mathcal{N}(G) < \mathcal{N}(E) = \mathcal{N}(F)$ have a unique decomposition into a parallel composition of primes. Let $\exp(A, G)$ be the *exponent* of prime A (the number of times A appears) in the unique decomposition of such a G.

Let m be chosen such that $k_m \neq l_m$, and that $\mathcal{N}(A_j) > \mathcal{N}(A_m)$ implies that $k_j = l_j$. Thus A_m is a maximal-sized (wrt norm) prime appearing in the decomposition of E or F in which the exponents differ. Without loss of generality, we can assume that $k_m > l_m$ (otherwise exchange the rôles of E and F). The proof proceeds now by cases on the possible forms of the decomposition of E.

First suppose E is a power of a single prime: $E \equiv A_m^{k_m}$. First of all, if E is prime, i.e. $k_m = 1$, then from $E \sim F$ we know that F is prime, and since $k_m > l_m$, we have that $F \equiv A_j$ for some j with $j \neq m$. But then $A_m \sim A_j$ contradicting the distinctness assumption on the A_i's. Hence we must have that $k_m > 1$.

We now want to show that $l_m > 0$ leads to a contradiction. So suppose $l_m > 0$. If there exists $a \in Act$ and G such that $A_m \xrightarrow{a} G$ with $\mathcal{N}(G) < \mathcal{N}(A_m)$ then from the assumption that $l_m > 0$ it follows that $F \xrightarrow{a} F'$ for some F' with $\mathcal{N}(F') < \mathcal{N}(F)$ and $\exp(A_m, F') = l_m - 1$. But whenever we have $E \xrightarrow{a} E'$ for some $a \in Act$ and E' with $\mathcal{N}(E') < \mathcal{N}(E)$ then $\exp(A_m, E') = k_m - 1 > l_m - 1$.

Therefore, there is no E' with $E \xrightarrow{a} E'$ such that $E' \sim F'$ contradicting the assumption that $E \sim F$.

Now, suppose there is no $a \in Act$ and G such that $A_m \xrightarrow{a} G$ is a norm reducing move. Then for some G we have $A_m \xrightarrow{\tau} G$ with $\mathcal{N}(G) < \mathcal{N}(A_m)$. From the assumption that $l_m > 0$ we get $F \xrightarrow{\tau} F'$ for some F' with $\mathcal{N}(F') < \mathcal{N}(F)$ and $\exp(A_m, F') = l_m - 1$. Suppose $E \xrightarrow{\tau} E'$ for some E' with $\mathcal{N}(E') < \mathcal{N}(E)$. From Proposition 5.6 we know that the τ-move from E cannot be the result of an active synchronisation between two copies of A_m. Thus whenever $E \xrightarrow{\tau} E'$ for some E' with $\mathcal{N}(E') < \mathcal{N}(E)$ we have $\exp(A_m, E') = k_m - 1 > l_m - 1$. Therefore, there is no E' with $E \xrightarrow{\tau} E'$ such that $E' \sim F'$ contradicting the assumption that $E \sim F$. We are thus lead to conclude that $l_m = 0$

Next we want to show that $l_m = 0$ also leads to a contradiction. From the maximality constraint in the definition of m it follows that if $l_i > 0$ then (because $k_i = 0$) we have $\mathcal{N}(A_i) < \mathcal{N}(A_m)$. Thus, whenever $F \xrightarrow{\mu} F'$ for some $\mu \in Act$ and F' with $\mathcal{N}(F') < \mathcal{N}(F)$ we know that F' has a unique decomposition in which $\exp(A_m, F') = 0$. However, as $k_m > 1$, we have that $E \xrightarrow{\mu} E'$ for some $\mu \in Act$ and E' with $\mathcal{N}(E') < \mathcal{N}(E)$ where E' has a unique decomposition in which $\exp(A_m, E') = k_m - 1 > 0$. Therefore, there is no F' with $F \xrightarrow{\mu} F'$ such that $E' \sim F'$ contradicting the assumption that $E \sim F$.

Next suppose that E is not a power of a single prime, i.e. there exists j different from m such that $k_j > 0$.

Suppose first that A_j can perform a visible move while reducing norm. Let $a \in Act$ and G be chosen such that $E \xrightarrow{a} G$ with $\mathcal{N}(G) < \mathcal{N}(E)$ and whenever we have $E \xrightarrow{b} E'$ for some $b \in Act$ and process E' with $\mathcal{N}(E') < \mathcal{N}(E)$ then $\exp(A_m, G) \geq \exp(A_m, E')$. By the assumption on A_j we have $\exp(A_m, G) \geq k_m$. We want to show that F cannot match the move $E \xrightarrow{a} G$.

Let $F \xrightarrow{a} F'$ for some F' with $\mathcal{N}(F') < \mathcal{N}(F)$; such an F' must exists, assuming $E \sim F$. Then there exist a prime A_s of F and process H_s such that $A_s \xrightarrow{a} H_s$ with $\mathcal{N}(H_s) < \mathcal{N}(A_s)$ and

$$F \xrightarrow{a} F' \;\equiv\; A_1^{l_1} | \cdots | A_s^{l_s - 1} | \cdots | A_n^{l_n} | H_s.$$

If $F' \sim G$ then we must have $\exp(A_m, F') = \exp(A_m, G) \geq k_m > l_m$ which implies that $\exp(A_m, H_s) > 0$. Therefore $\mathcal{N}(A_s) > \mathcal{N}(H_s) \geq \mathcal{N}(A_m)$ and from the maximality constraint on m it follows that $k_s = l_s$ (and $m \neq s$). However now, since $k_s = l_s > 0$, we also have $E \xrightarrow{a} E'$ for some E' with $\mathcal{N}(E') < \mathcal{N}(E)$ such that $\exp(A_m, E') = k_m + \exp(A_m, H_s) > l_m + \exp(A_m, H_s) = \exp(A_m, F')$. Hence we have $\exp(A_m, G) \geq \exp(A_m, E') > \exp(A_m, F')$ and therefore there cannot exists F' with $F \xrightarrow{a} F'$ such that $F' \sim G$ contradicting the assumption that $E \sim F$.

Next suppose that no prime factor of E different from A_m can perform a visible move while reducing norm. Let G be such that $E \xrightarrow{\tau} G$ with $\mathcal{N}(E) = 2 + \mathcal{N}(G)$ and whenever $E \xrightarrow{\tau} E'$ with $\mathcal{N}(E) = 2 + \mathcal{N}(E')$ then $\exp(A_m, G) \geq \exp(A_m, E')$. From the assumption on primes different from A_m and Lemma 5.7 we know that such a G exists. Moreover, we have $\exp(A_m, G) \geq k_m$. As before we want to show that F cannot match the move $E \xrightarrow{\tau} G$.

Let $F \xrightarrow{\tau} F'$ for some F' with $\mathcal{N}(F') = \mathcal{N}(G)$; such an F' must exists, assuming $E \sim F$. Suppose first of all that the τ-move from F originates from a single prime factor. That is, there exist a prime factor A_s of F and process H_s such that $A_s \xrightarrow{\tau} H_s$ with $\mathcal{N}(A_s) = 2 + \mathcal{N}(H_s)$ and

$$F \xrightarrow{\tau} F' \quad \equiv \quad A_1^{l_1}|\cdots|A_s^{l_s-1}|\cdots|A_n^{l_n}|H_s.$$

If $F' \sim G$ then we must have $\exp(A_m, F') = \exp(A_m, G) \geq k_m > l_m$ which implies that $\exp(A_m, H_s) > 0$. Therefore $\mathcal{N}(A_s) > \mathcal{N}(H_s) \geq \mathcal{N}(A_m)$ and from the maximality constraint on m it follows that $k_s = l_s$ (and $m \neq s$). However now, since $k_s = l_s > 0$, we also have $E \xrightarrow{\tau} E'$ for some E' with $\mathcal{N}(E) = 2 + \mathcal{N}(E')$ such that $\exp(A_m, E') = k_m + \exp(A_m, H_s) > l_m + \exp(A_m, H_s) = \exp(A_m, F')$. Hence we have $\exp(A_m, G) \geq \exp(A_m, E') > \exp(A_m, F')$ and therefore there cannot exists F' with $F \xrightarrow{\tau} F'$ such that $F' \sim G$ contradicting the assumption that $E \sim F$.

Finally, suppose for some F' that $F \xrightarrow{\tau} F'$ with $\mathcal{N}(F') = \mathcal{N}(G)$ and that the τ-move from F originates from an active synchronisation between two primes of F. We therefore have primes A_s and A_t of F and processes H_s and H_t such that $A_s \xrightarrow{a} H_s$ and $A_t \xrightarrow{\bar{a}} H_t$ for some $a \in Act$ with

$$F \xrightarrow{\tau} F' \quad \equiv \quad A_1^{l_1}|\cdots|A_s^{l_s-1}|\cdots|A_t^{l_t-1}|\cdots|A_n^{l_n}|H_s|H_t.$$

We have $\mathcal{N}(F) = 2 + \mathcal{N}(F')$ and therefore from Lemma 5.7 we may deduce the two inequalities $\mathcal{N}(H_s) < \mathcal{N}(A_s)$ and $\mathcal{N}(H_t) < \mathcal{N}(A_t)$. As $\exp(A_m, G) \geq k_m$ and $k_m > l_m$ we must have either $\exp(A_m, H_s) > 0$ or $\exp(A_m, H_t) > 0$. Without loss of generality assume that $\exp(A_m, H_s) > 0$ is the case. Then it follows that $\mathcal{N}(A_s) > \mathcal{N}(H_s) \geq \mathcal{N}(A_m)$ and therefore from the maximality constraint on m we conclude that $k_s = l_s$ (and $s \neq m$). But then there exists a prime factor of E different from A_m which is capable of performing a visible move and thereby reduce norm. This contradicts the assumption on the prime factors of E and completes the proof. ■

An important corollary of this theorem is.

Corollary 5.9 (Cancellation Law) *For normed processes E, F and G of* BPP$_\tau$ *it follows that $E|G \sim F|G$ implies $E \sim F$.*

Proof: As in the proof of Corollary 5.5. ■

Notice that the unique decomposition theorem for bisimilarity is not valid when one moves beyond normedness. For instance, the process $X \stackrel{\mathrm{def}}{=} aX$ has no decomposition at all since $X \sim X|X$. However, as we show in the next section, the situation is different for distributed bisimilarity.

5.3 Unique Decomposition for BPP wrt $\sim_d$

We now show validity of a unique decomposition theorem for distributed bisimilarity on *all* of BPP. Our proof cannot rely on norm as an inductive measure any more. Instead our proof is based on an induction using the *initial degree of parallelism* of processes as the measure.

Processes are (as usual) defined relative to some family Δ of finitely many guarded process equations. Normally, we shall omit mentioning the family when confusion cannot arise.

Definition 5.10 *Let E be a process of* BPP. *By structural induction on E we define the* initial degree of parallelism *of E, denoted $\mathcal{D}(E)$, to be*

- $\mathcal{D}(\mathbf{0}) = \mathcal{D}(aF) = 0,$
- $\mathcal{D}(F_1 + F_2) = \max\{\mathcal{D}(F_1), \mathcal{D}(F_2)\},$
- $\mathcal{D}(F_1 \| F_2) = \begin{cases} \mathcal{D}(F_2) & \textit{if } F_1 \sim_d \mathbf{0} \\ \mathcal{D}(F_1) & \textit{if } F_2 \sim_d \mathbf{0} \\ 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) & \textit{otherwise} \end{cases}$
- $\mathcal{D}(F_1 \lfloor\!\lfloor F_2) = \begin{cases} 0 & \textit{if } F_1 \sim_d \mathbf{0} \\ \mathcal{D}(F_1 \| F_2) & \textit{otherwise} \end{cases}$
- $\mathcal{D}(X) = \mathcal{D}(F)$ *where* $X \stackrel{\text{def}}{=} F \in \Delta$.

Notice that $\mathcal{D}(E)$ is finite for all BPP processes since equations of Δ are guarded. We shall prove a number of technical results concerning $\mathcal{D}$ showing that it is a proper measure on which to base our proof of unique decomposition. For the proofs we shall use inductive arguments based on the measure $\mathcal{C}$ given by

$$\begin{array}{lcl@{\qquad}lcl} \mathcal{C}(\mathbf{0}) & = & 0 & \mathcal{C}(F_1 + F_2) & = & 1 + \mathcal{C}(F_1) + \mathcal{C}(F_2) \\ \mathcal{C}(aF) & = & 0 & \mathcal{C}(F_1 \| F_2) & = & 1 + \mathcal{C}(F_1) + \mathcal{C}(F_2) \\ \mathcal{C}(X) & = & 1 + \mathcal{C}(F) & \mathcal{C}(F_1 \lfloor\!\lfloor F_2) & = & 1 + \mathcal{C}(F_1) + \mathcal{C}(F_2) \end{array}$$

where $X \stackrel{\text{def}}{=} F \in \Delta$. As equations of Δ are guarded we observe that $\mathcal{C}$ is well-defined, i.e. $\mathcal{C}(E)$ is finite for all BPP processes E. The requirements on $\mathcal{C}$ that we need in our proofs are

$$\begin{array}{lcl@{\qquad}lcl} \mathcal{C}(F_1) & < & \mathcal{C}(F_1 + F_2) & \mathcal{C}(F_1) & < & \mathcal{C}(F_1 \| F_2) \\ \mathcal{C}(F_2) & < & \mathcal{C}(F_1 + F_2) & \mathcal{C}(F_2) & < & \mathcal{C}(F_1 \| F_2) \\ \mathcal{C}(F) & < & \mathcal{C}(X) & \mathcal{C}(F_1) & < & \mathcal{C}(F_1 \lfloor\!\lfloor F_2) \end{array}$$

where $X \stackrel{\text{def}}{=} F \in \Delta$. By inspection these requirements are clearly seen to be satisfied. With the measure $\mathcal{C}$ in hand we are now ready to prove our required technical results concerning the initial degree of parallelism.

Lemma 5.11 *Let E be a process of* BPP. *If $E \xrightarrow{a} (E_1, E_2)$ for some $a \in \Lambda_1$ and E_1, E_2 such that $E_2 \not\sim_d \mathbf{0}$ then $\mathcal{D}(E) > \mathcal{D}(E_2)$.*

Proof: We proceed by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = F_1 \| F_2$. Assume without loss of generality that $F_1 \xrightarrow{a} (F_{11}, F_{12})$ for processes F_{11} and F_{12} such that $E \xrightarrow{a} (F_{11}, F_{12} \| F_2)$, i.e. $E_1 \equiv F_{11}$ and $E_2 \equiv F_{12} \| F_2$. There are now three cases to consider.

(i) Suppose $F_2 \sim_d \mathbf{0}$. By assumption we have $F_{12} \| F_2 \not\sim_d \mathbf{0}$ hence $F_{12} \not\sim_d \mathbf{0}$ must be the case. Therefore by induction we have $\mathcal{D}(F_1) > \mathcal{D}(F_{12})$ which implies $\mathcal{D}(E) = \mathcal{D}(F_1) > \mathcal{D}(F_{12}) = \mathcal{D}(F_{12} \| F_2)$ as required.

(ii) Suppose $F_{12} \sim_d \mathbf{0}$. By assumption we have $F_{12} \| F_2 \not\sim_d \mathbf{0}$ and therefore we must have $F_2 \not\sim_d \mathbf{0}$. But from $F_1 \not\sim_d \mathbf{0}$ and $F_2 \not\sim_d \mathbf{0}$ we now get by definition of $\mathcal{D}$ that $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) > \mathcal{D}(F_2) = \mathcal{D}(F_{12} \| F_2)$ as required.

(iii) Suppose $F_{12} \not\sim_d \mathbf{0}$ and $F_2 \not\sim_d \mathbf{0}$. By the induction hypothesis $\mathcal{D}(F_1) > \mathcal{D}(F_{12})$ hence $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) > 1 + \mathcal{D}(F_{12}) + \mathcal{D}(F_2) = \mathcal{D}(F_{12} \| F_2)$ as required.

This completes the case. ∎

Lemma 5.12 *Let E be a process of* BPP. *Then we have*

$$\mathcal{D}(E) = 0 \textit{ iff } \forall a \in \Lambda_1, \forall E_1, E_2 \in \text{BPP} : E \xrightarrow{a} (E_1, E_2) \textit{ implies } E_2 \sim_d \mathbf{0}.$$

Proof: Follows readily by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We omit the details. ∎

Lemma 5.13 *Let E be a process of* BPP. *If $\mathcal{D}(E) > 0$ then there exist $a \in \Lambda_1$ and E_1, E_2 such that $E \xrightarrow{a} (E_1, E_2)$ with $\mathcal{D}(E) = 1 + \mathcal{D}(E_2)$. Moreover, $E_2 \not\sim_d \mathbf{0}$.*

Proof: We proceed by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = F_1 \| F_2$. We must have one of the following three situations.

(i) Suppose $F_1 \sim_d \mathbf{0}$. Then by definition we have $\mathcal{D}(E) = \mathcal{D}(F_2)$ and therefore $\mathcal{D}(F_2) > 0$. Thus by the induction hypothesis we have $F_2 \xrightarrow{a} (F_{21}, F_{22})$ for some $a \in \Lambda_1$ and some processes F_{21} and F_{22} such that $\mathcal{D}(F_2) = 1 + \mathcal{D}(F_{22})$ with $F_{22} \not\sim_d \mathbf{0}$. But then we also have the move $E \xrightarrow{a} (F_{21}, F_{22} \| F_1)$ with $\mathcal{D}(E) = \mathcal{D}(F_2) = 1 + \mathcal{D}(F_{22}) = 1 + \mathcal{D}(F_{22} \| F_1)$ as required. Furthermore, we have $F_{22} \| F_1 \not\sim_d \mathbf{0}$.

(ii) Suppose $F_2 \sim_d \mathbf{0}$. Then the arguments are as above.

(iii) Suppose $F_1 \not\sim_d \mathbf{0}$ and $F_2 \not\sim_d \mathbf{0}$. By definition $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2)$. Suppose first that $\mathcal{D}(F_1) = \mathcal{D}(F_2) = 0$. From $F_1 \not\sim_d \mathbf{0}$ we can find $a \in \Lambda_1$ and processes F_{11}, F_{12} such that $F_1 \xrightarrow{a} (F_{11}, F_{12})$. By Lemma 5.12 we must have $F_{12} \sim_d \mathbf{0}$. Thus for $E \xrightarrow{a} (F_{11}, F_{12} \| F_2)$ we have $\mathcal{D}(E) = 1 + \mathcal{D}(F_{12} \| F_2)$ since $\mathcal{D}(F_{12} \| F_2) = 0$. Moreover, we have $F_{12} \| F_2 \not\sim_d \mathbf{0}$. Now, assume that for $i = 1$ or $i = 2$ we have $\mathcal{D}(F_i) > 0$. Assume without loss of generality that $\mathcal{D}(F_1) > 0$. From the induction hypothesis it follows that there exists $a \in \Lambda_1$ and processes F_{11}, F_{12} such that we have the move $F_1 \xrightarrow{a} (F_{11}, F_{12})$ with $\mathcal{D}(F_1) = 1 + \mathcal{D}(F_{12})$ and $F_{12} \not\sim_d \mathbf{0}$. Then we have $E \xrightarrow{a} (F_{11}, F_{12} \| F_2)$ with $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) = 1 + 1 + \mathcal{D}(F_{12}) + \mathcal{D}(F_2) = 1 + \mathcal{D}(F_{12} \| F_2)$ where the last equality follows from the fact that $F_{12} \not\sim_d \mathbf{0}$ and $F_2 \not\sim_d \mathbf{0}$.

This completes the case. ■

Lemma 5.14 *Let E and F be processes of* BPP. *If $E \sim_d F$ then $\mathcal{D}(E) = \mathcal{D}(F)$.*

Proof: Suppose E and F are processes with $E \sim_d F$. First of all, if $\mathcal{D}(E) = 0$ then from Lemma 5.12 and the fact that $E \sim_d F$ it follows that $\mathcal{D}(F) = 0$. Similarly, if $\mathcal{D}(F) = 0$ we also get $\mathcal{D}(E) = 0$. Therefore assume that $\mathcal{D}(E), \mathcal{D}(F) > 0$. We proceed by induction on $\max\{\mathcal{D}(E), \mathcal{D}(F)\}$.

As $\mathcal{D}(E) > 0$ it follows from Lemma 5.13 that for some $a \in \Lambda_1$ and some processes E_1 and E_2 we have $E \xrightarrow{a} (E_1, E_2)$ with $\mathcal{D}(E) = 1 + \mathcal{D}(E_2)$ and $E_2 \not\sim_d \mathbf{0}$. From $E \sim_d F$ we have $F \xrightarrow{a} (F_1, F_2)$ for some processes F_1 and F_2 with $E_1 \sim_d F_1$ and $E_2 \sim_d F_2$. Hence from the induction hypothesis we get $\mathcal{D}(E_2) = \mathcal{D}(F_2)$ and therefore we have $\mathcal{D}(E) = 1 + \mathcal{D}(E_2) = 1 + \mathcal{D}(F_2) < 1 + \mathcal{D}(F)$ where the last inequality follows from Lemma 5.11 since $F_2 \not\sim_d \mathbf{0}$.

From $\mathcal{D}(F) > 0$ we get in a similar manner the inequality $\mathcal{D}(F) < 1 + \mathcal{D}(E)$. Now, from the two inequalities $\mathcal{D}(E) < 1 + \mathcal{D}(F)$ and $\mathcal{D}(F) < 1 + \mathcal{D}(E)$ it readily follows that $\mathcal{D}(E) = \mathcal{D}(F)$ as required. ■

We now have sufficient properties concerning the measure $\mathcal{D}$ and are almost in a position to prove the theorem of unique decomposition. However, our proof relies on a cancellation law for BPP processes admitted by distributed bisimulation equivalence, and in order to obtain a proof of this we need the following technical result.

Lemma 5.15 *For* BPP *processes E, F and G, if $G \xrightarrow{a} (G_1, G_2)$ for some $a \in \Lambda_1$ and G_1, G_2 such that $E\|G \sim_d F\|G_2$ then there exist F_1, F_2 and G_3 such that $F \xrightarrow{a} (F_1, F_2)$ with $G_1 \sim_d F_1$ and $E\|G_3 \sim_d F_2\|G_3$.*

Proof: The proof proceeds by induction on $\mathcal{D}(G)$. From $G \xrightarrow{a} (G_1, G_2)$ we have $E\|G \xrightarrow{a} (G_1, E\|G_2)$. Therefore from $E\|G \sim_d F\|G_2$ we must have one of the following two situations.

(i) Suppose for F_1, F_2 that $F \xrightarrow{a} (F_1, F_2)$ such that $F\|G_2 \xrightarrow{a} (F_1, F_2\|G_2)$ with $G_1 \sim_d F_1$ and $E\|G_2 \sim_d F_2\|G_2$. But this is exactly what we are required to prove.

(ii) Suppose for G_{21}, G_{22} that $G_2 \xrightarrow{a} (G_{21}, G_{22})$ with $F\|G_2 \xrightarrow{a} (G_{21}, F\|G_{22})$ where $G_1 \sim_d G_{21}$ and $E\|G_2 \sim_d F\|G_{22}$. From Lemma 5.11 (since $G_2 \not\sim_d \mathbf{0}$) it follows that $\mathcal{D}(G) > \mathcal{D}(G_2)$. Hence by induction we have $F \xrightarrow{a} (F_1, F_2)$ for some processes F_1, F_2 and G_3 such that $G_{21} \sim_d F_1$ and $E\|G_3 \sim_d F_2\|G_3$. And from $G_1 \sim_d G_{21}$ and $G_{21} \sim_d F_1$ we get $G_1 \sim_d F_1$ as required.

This completes the proof. ■

Lemma 5.16 (Cancellation Law) *For processes E, F and G of* BPP *it follows that $E\|G \sim_d F\|G$ implies $E \sim_d F$*

Proof: Let E, F and G be given such that $E\|G \sim_d F\|G$. A sufficient condition for $E \sim_d F$ will be to show that the relation R given by

$$R \;=\; \Big\{(E,F) : \text{there exists } G \text{ such that } E\|G \sim_d F\|G\Big\}$$

is a distributed bisimulation relation.

Let $(E,F) \in R$. Thus there exists G with $E\|G \sim_d F\|G$. Suppose for some $a \in \Lambda_1$ and processes E_1, E_2 that $E \xrightarrow{a} (E_1, E_2)$. Then $E\|G \xrightarrow{a} (E_1, E_2\|G)$. As $E\|G \sim_d F\|G$ we have one of the following two cases.

(i) Suppose for F_1, F_2 that $F \xrightarrow{a} (F_1, F_2)$ such that $F\|G \xrightarrow{a} (F_1, F_2\|G)$ with $E_1 \sim_d F_1$ and $E_2\|G \sim_d F_2\|G$. Then it clearly follows that $(E_1, F_1) \in R$ and also $(E_2, F_2) \in R$ as required.

(ii) Suppose for G_1, G_2 that $G \xrightarrow{a} (G_1, G_2)$ with $F\|G \xrightarrow{a} (G_1, F\|G_2)$ where $E_1 \sim_d G_1$ and $E_2\|G \sim_d F\|G_2$. From Lemma 5.15 it then follows that there exist processes F_1, F_2 and G_3 such that we have the move $F \xrightarrow{a} (F_1, F_2)$ with $G_1 \sim_d F_1$ and $E_2\|G_3 \sim_d F_2\|G_3$. Thus $(E_2, F_2) \in R$ and from $E_1 \sim_d G_1$ together with $G_1 \sim_d F_1$ we also have $(E_1, F_1) \in R$ as required.

By symmetry we now conclude that R is a distributed bisimulation relation. ∎

We are now ready to prove the theorem of unique decomposition. The definition of primes is as in Section 5.1 (relative to distributed bisimulation equivalence).

Theorem 5.17 (Unique Decomposition) *Let E be a process of* BPP. *Then E can be expressed uniquely, wrt distributed bisimulation equivalence, as a product (using the combinator $\|$) of primes.*

Proof: That any process E of BPP can be expressed as a product of primes wrt distributed bisimilarity is straightforward; it follows by induction on the initial degree of parallelism of E. The proof that this decomposition is unique proceeds by induction on $\mathcal{D}(E)$.

Suppose that $E \sim_d F$ and that E and F are decompositions into primes given by

$$\begin{aligned} E &\equiv A_1\|A_2\| \cdots \| A_n\|C, \\ F &\equiv B_1\|B_2\| \cdots \|B_m\|C. \end{aligned}$$

That is, the two decompositions have a common prime factor. By the cancellation law we get $A_1\|A_2\|\cdots\|A_n \sim_d B_1\|B_2\|\cdots\|B_m$. From the induction hypothesis we then conclude that $A_1\|A_2\|\cdots\|A_n$ and $B_1\|B_2\|\cdots\|B_m$ are identical decompositions. But then the prime decompositions for E and F are also identical.

Finally, suppose that $E \sim_d F$ and that E and F are decompositions into primes with no prime factor in common:

$$E \equiv A_1\|A_2\| \cdots \| A_n,$$
$$F \equiv B_1\|B_2\| \cdots \|B_m,$$

where for all i, j it follows that $A_i \not\sim_d B_j$.

First of all, if $n = 1$ or $m = 1$ then both E and F are prime and therefore $n = m = 1$. But then $A_1 \sim_d B_1$ contradicting the distinctness assumption on the prime factors of E and F. Thus $n, m > 1$.

Without loss of generality assume that $\mathcal{D}(A_1) \leq \mathcal{D}(A_i), \mathcal{D}(B_j)$ for all i, j. That is, A_1 is the prime component with least initial degree of parallelism. Suppose $A_1 \xrightarrow{a} (G_1, G_2)$ for some G_1, G_2 such that $E \xrightarrow{a} (G_1, G_2\|A_2\|\cdots\|A_n)$. As $E \sim_d F$ we have without loss of generality $B_1 \xrightarrow{a} (H_1, H_2)$ for some H_1, H_2 such that $F \xrightarrow{a} (H_1, H_2\|B_2\|\cdots\|B_m)$ with $G_2\|A_2\|\cdots\|A_n \sim_d H_2\|B_2\|\cdots\|B_m$. Suppose G_2 is decomposed into the primes $C_1, C_2 \ldots C_k$ and H_2 is decomposed into the primes $D_1, D_2 \ldots D_l$. Then $G_2\|A_2\|\cdots\|A_n$ has the decomposition:

$$C_1\|C_2\|\cdots\|C_k\|A_2\|A_3\|\cdots\|A_n,$$

and $H_2\|B_2\|\cdots\|B_m$ has the decomposition:

$$D_1\|D_2\|\cdots\|D_l\|B_2\|B_3\|\cdots\|B_m.$$

By Lemma 5.11 it follows that $\mathcal{D}(E) > \mathcal{D}(G_2\|A_2\|\cdots\|A_n) = \mathcal{D}(H_2\|B_2\|\cdots\|B_m)$. Hence from the induction hypothesis we know that the above two decompositions must be identical. Thus B_2 must equal one of the prime factors $C_1, C_2 \ldots C_k$ or $A_2, A_3 \ldots A_n$. Again from Lemma 5.11 it follows that $\mathcal{D}(A_1) > \mathcal{D}(G_2)$ (assuming $G_2 \not\sim_d \mathbf{0}$; otherwise the required contradiction follows readily). We also have $\mathcal{D}(G_2) \geq \mathcal{D}(C_i)$ for $i = 1 \ldots k$. By the choice of A_1 we have $\mathcal{D}(B_2) \geq \mathcal{D}(A_1)$ and therefore B_2 cannot equal one of $C_1, C_2 \dots C_k$ as equal processes must have the same initial degree of parallelism. Hence B_2 must equal one of $A_2, A_3 \dots A_n$ contradicting the distinctness assumption on the prime factors of E and F. This completes the proof. ∎

We now investigate to which extent the question of unique decomposition of processes wrt distributed bisimulation equivalence *degenerate*. For any process E of BPP we have just proved that E can be described uniquely, up to $\sim_d$, as a product (using the combinator $\|$) of primes $A_1, A_2 \dots A_n$ as follows:

$$E \;\; \sim_d \;\; A_1 \| A_2 \| \cdots \| A_n.$$

The question is whether the primes $A_1, A_2 \dots A_n$ already are *contained* within the process E. A suggestion for containment is the following definition.

Definition 5.18 *Let $\Subset$ be a relation on* BPP *processes defined by $E \Subset F$ iff*

- *$E \sim_d F$, or*
- *$F = F_1 + F_2$ and $E \Subset F_i$ for $i = 1$ or $i = 2$, or*
- *$F = F_1 \| F_2$ and $E \Subset F_i$ for $i = 1$ or $i = 2$, or*
- *$F = F_1 \lfloor\!\lfloor F_2$ and $E \Subset F_1$ or ($F_1 \not\sim \mathbf{0}$ and $E \Subset F_2$), or*
- *$F = Y$ where $Y \stackrel{\mathrm{def}}{=} G \in \Delta$ and $E \Subset G$.*

Note that the relation $\Subset$ is well-defined as equations of Δ are guarded.

The theorem of degeneration relies on the following technical result.

Lemma 5.19 *Let E and F be processes of* BPP*. Suppose that F is prime. If for some $a \in \Lambda_1$ and E_1, E_2 we have $E \stackrel{a}{\longrightarrow} (E_1, E_2)$ with $F \Subset E_2$ then $F \Subset E$.*

Proof: The proof proceeds by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = G_1 \| G_2$. Assume without loss of generality that $G_1 \stackrel{a}{\longrightarrow} (G_{11}, G_{12})$ for some G_{11} and G_{12} such that $E \stackrel{a}{\longrightarrow} (G_{11}, G_{12} \| G_2)$, i.e. $E_1 \equiv G_{11}$ and $E_2 \equiv G_{12} \| G_2$. By assumption we have $F \Subset G_{12} \| G_2$. There are now three cases to consider.

(i) Suppose $F \Subset G_{12}$. Then by induction we have $F \Subset G_1$ and therefore also $F \Subset E$ as required.

(ii) Suppose $F \Subset G_2$. Then we clearly have $F \Subset E$.

(iii) Suppose $F \sim_d G_{12} \| G_2$. As F is prime we have either $G_{12} \sim_d \mathbf{0}$ or $G_2 \sim_d \mathbf{0}$. But if $G_2 \sim_d \mathbf{0}$ then we must have $F \Subset G_{12}$ (the first case applies) and if $G_{12} \sim_d \mathbf{0}$ then we must have $F \Subset G_2$ (the second case applies).

This completes the case. ■

Theorem 5.20 (Degeneration) *Let E be a process of* BPP*. Assume E is decomposed into $A_1, A_2 \ldots A_n$. Then $A_i \Subset E$ for $i = 1 \ldots n$.*

Proof: If $n = 0$ or $n = 1$ there is nothing to prove. Hence assume that $n > 1$. We proceed by induction on $\mathcal{D}(E)$.

Suppose we have the move $A_1 \xrightarrow{a} (F_{11}, F_{12})$ for some F_{11} and F_{12} such that $A_1 \| A_2 \| \cdots \| A_n \xrightarrow{a} (F_{11}, F_{12} \| A_2 \| \cdots \| A_n)$ and let $E \xrightarrow{a} (E_1, E_2)$ for some E_1, E_2 be the matching move of E with $F_{11} \sim_d E_1$ and $F_{12} \| A_2 \| \cdots \| A_n \sim_d E_2$. Suppose F_{12} is decomposed into the primes $B_1, B_2 \ldots B_k$. Then the prime decomposition of E_2 must be

$$B_1 \| B_2 \| \cdots \| B_k \| A_2 \| A_3 \| \cdots \| A_n.$$

From Lemma 5.11 (since $E_2 \not\sim_d \mathbf{0}$) we have $\mathcal{D}(E) > \mathcal{D}(E_2)$. Thus from the induction hypothesis we conclude that $A_i \Subset E_2$ for $i = 2 \ldots n$. From Lemma 5.19 it then follows that $A_i \Subset E$ for $i = 2 \ldots n$. By letting one of the other prime factors perform a move it also follows that $A_1 \Subset E$ and this completes the proof. ■

Example 5.21 *The process $(a + b) \| a + a \| a$ is not prime being distributed bisimilar to $(a + b) \| a$ where $a + b$ and a both are prime. But each of the prime factors a and $a + b$ can be found in $(a + b) \| a + (a \| a)$ in the sense of $\Subset$.*

Notice that the theorem of degeneration does not hold in the case of bisimulation: $ab + ba$ is not prime wrt bisimulation equivalence being bisimilar to $a \| b$. But neither a nor b is contained within $ab + ba$.

5.4 Unique Decomposition for BPP_τ wrt $\sim_d$

In the final section of this chapter we shall extend the result of the previous section to BPP_τ thus allowing for synchronisation between processes. The setup to the theorem of unique decomposition is as in the previous section. The proof is based on induction using the initial degree of parallelism of processes of BPP_τ which we again denote by $\mathcal{D}$. The definition of $\mathcal{D}$ is exactly as in Section 5.3 with $\|$ and $\lfloor\!\lfloor$ being replaced by $|$ and $\lfloor$ respectively to cater for BPP_τ processes. We shall not bother the reader by defining $\mathcal{D}$; we refer to Definition 5.10.

As in Section 5.3 we need a number of results concerning the initial degree of parallelism $\mathcal{D}$. These results are obtained by induction using a simple measure on the structural complexity of processes of BPP_τ. As in Section 5.3 we shall denote this measure by $\mathcal{C}$. The definition and requirements that $\mathcal{C}$ must satisfy are exactly as in Section 5.3 with $\|$ and $\lfloor\!\lfloor$ being replaced by $|$ and $\lfloor$ respectively to cater for BPP_τ processes.

We now lift the properties concerning the measure $\mathcal{D}$ as established in Section 5.3 to the setting of BPP_τ. However, in order not to repeat ourselves we shall refer to the corresponding proofs of Section 5.3 whenever possible and thus in our case analysis we often only consider the case of synchronisation.

Lemma 5.22 *Let E be a process of* BPP_τ. *If for some $\mu \in Act$ and E_1, E_2 we have $E \xrightarrow{\mu} (E_1, E_2)$ with $E_2 \not\sim_d \mathbf{0}$ then $\mathcal{D}(E) > \mathcal{D}(E_2)$.*

Proof: We proceed by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = F_1|F_2$. From $E \xrightarrow{\mu} (E_1, E_2)$ we have three cases: either the μ-move originates from F_1, or F_2, or $\mu = \tau$ and the move originates from a synchronisation between F_1 and F_2. The arguments for the first two cases are exactly as in the proof of Lemma 5.11. We only consider the third case.

Suppose $\mu = \tau$ and $F_1 \xrightarrow{a} (F_{11}, F_{12}), F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ for some $a \in Act$ and processes F_{11}, F_{12}, F_{21} and F_{22} such that $F_1|F_2 \xrightarrow{\tau} (F_{11}|F_{21}, F_{12}|F_{22})$, i.e. we have $E_1 \equiv F_{11}|F_{21}$ and $E_2 \equiv F_{12}|F_{22}$. First of all, suppose that $F_{12} \sim_d \mathbf{0}$. As $E_2 \not\sim_d \mathbf{0}$

we must have $F_{22} \not\sim_d \mathbf{0}$ and therefore from the induction hypothesis it follows that $\mathcal{D}(F_2) > \mathcal{D}(F_{22})$. Hence $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) > \mathcal{D}(F_{22}) = \mathcal{D}(F_{12}|F_{22})$ as required. Secondly, suppose that $F_{22} \sim_d \mathbf{0}$. Then the arguments are exactly as in the first case. Finally, suppose that $F_{12} \not\sim_d \mathbf{0}$ and $F_{22} \not\sim_d \mathbf{0}$. From the induction hypothesis we have $\mathcal{D}(F_1) > \mathcal{D}(F_{12})$ and also $\mathcal{D}(F_2) > \mathcal{D}(F_{22})$. Therefore we have $\mathcal{D}(E) = 1 + \mathcal{D}(F_1) + \mathcal{D}(F_2) > 1 + \mathcal{D}(F_{12}) + \mathcal{D}(F_{22}) = \mathcal{D}(F_{12}|F_{22})$ as required. ■

Lemma 5.23 *Let E be a process of* BPP_τ. *Then we have*

$$\mathcal{D}(E) = 0 \textit{ iff } \forall \mu \in Act, \forall E_1, E_2 \in \mathrm{BPP}_\tau : E \xrightarrow{\mu} (E_1, E_2) \textit{ implies } E_2 \sim_d \mathbf{0}.$$

Proof: Follows readily by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We omit the details. ■

Lemma 5.24 *Let E be a process of* BPP_τ. *If $\mathcal{D}(E) > 0$ then there exist $\mu \in Act$ and E_1, E_2 such that $E \xrightarrow{\mu} (E_1, E_2)$ with $\mathcal{D}(E) = 1 + \mathcal{D}(E_2)$. Moreover, $E_2 \not\sim_d \mathbf{0}$.*

Proof: Can readily be obtained from the proof of Lemma 5.13. ■

Lemma 5.25 *Let E and F be processes of* BPP_τ. *If $E \sim_d F$ then $\mathcal{D}(E) = \mathcal{D}(F)$.*

Proof: Can readily be obtained from the proof of Lemma 5.14. ■

As in Section 5.3 the proof of unique decomposition relies on a cancellation law for BPP_τ processes wrt distributed bisimulation equivalence. Before proving the cancellation law we need the following technical results.

Lemma 5.26 *Let E be a process of* BPP_τ. *If for some $a \in Act$ and processes E_1, E_2, E_{21} and E_{22} we have $E \xrightarrow{a} (E_1, E_2)$ and $E_2 \xrightarrow{\bar{a}} (E_{21}, E_{22})$ then we can also obtain $E \xrightarrow{\tau} (E_1|E_{21}, E_{22})$.*

Proof: The proof proceeds by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = F_1|F_2$. Assume without loss of generality that F_1 performs the move $F_1 \xrightarrow{a} (F_{11}, F_{12})$ for some processes F_{11} and F_{12} such that $F_1|F_2 \xrightarrow{a} (F_{11}, F_{12}|F_2)$, i.e. $E_1 \equiv F_{11}$ and $E_2 \equiv F_{12}|F_2$. By assumption we have $F_{12}|F_2 \xrightarrow{\bar{a}} (E_{21}, E_{22})$. There are now two cases to consider.

(i) Suppose for some G_1, G_2 that F_{12} performs the move $F_{12} \xrightarrow{\bar{a}} (G_1, G_2)$ such that $F_{12}|F_2 \xrightarrow{\bar{a}} (G_1, G_2|F_2)$, i.e. $E_{21} \equiv G_1$ and $E_{22} \equiv G_2|F_2$. We now have $F_1 \xrightarrow{a} (F_{11}, F_{12})$ and $F_{12} \xrightarrow{\bar{a}} (G_1, G_2)$ thus by induction it follows that we have the move $F_1 \xrightarrow{\tau} (F_{11}|G_1, G_2)$. But then also $E \xrightarrow{\tau} (F_{11}|G_1, G_2|F_2)$ as required.

(ii) Suppose for some F_{21}, F_{22} that F_2 performs the move $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ such that $F_{12}|F_2 \xrightarrow{\bar{a}} (F_{21}, F_{12}|F_{22})$, i.e. $E_{21} \equiv F_{21}$ and $E_{22} \equiv F_{12}|F_{22}$. We now have $F_1 \xrightarrow{a} (F_{11}, F_{12})$ and also $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ from which we conclude that $E \xrightarrow{\tau} (F_{11}|F_{21}, F_{12}|F_{22})$ as required.

This completes the case. ■

Lemma 5.27 *For processes E, F and G of* BPP_τ*, if $G \xrightarrow{\mu} (G_1, G_2)$ for some $\mu \in Act$ and G_1, G_2 such that $E|G \sim_d F|G_2$ then there exist F_1, F_2 and G_3 such that $F \xrightarrow{\mu} (F_1, F_2)$ with $G_1 \sim_d F_1$ and $E|G_3 \sim_d F_2|G_3$.*

Proof: The proof proceeds by induction on $\mathcal{D}(G)$. Since $G \xrightarrow{\mu} (G_1, G_2)$ we conclude that $E|G \xrightarrow{\mu} (G_1, E|G_2)$. As $E|G \sim_d F|G_2$ we know that $F|G_2$ is capable of performing a matching μ-move. We have one of the following three possibilities: either the μ-move originates from F, or G_2, or $\mu = \tau$ and the τ-move is performed via a synchronisation between F and G_2. Here we only consider the last case as the arguments for the first two cases readily can be obtained from the proof of Lemma 5.15 of Section 5.3.

Suppose $\mu = \tau$ and $F \xrightarrow{a} (F_1, F_2), G_2 \xrightarrow{\bar{a}} (G_{21}, G_{22})$ for some $a \in Act$ and F_1, F_2, G_{21} and G_{22} such that $F|G_2 \xrightarrow{\tau} (F_1|G_{21}, F_2|G_{22})$ with $G_1 \sim_d F_1|G_{21}$ and $E|G_2 \sim_d F_2|G_{22}$. Especially, we have $G_2 \xrightarrow{\bar{a}} (G_{21}, G_{22})$ such that $E|G_2 \sim_d F_2|G_{22}$. Now, from Lemma 5.22 we have $\mathcal{D}(G) > \mathcal{D}(G_2)$ (as $G_2 \not\sim_d \mathbf{0}$) and therefore from the induction hypothesis we have $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ for some processes F_{21}, F_{22} and G_3 such that $G_{21} \sim_d F_{21}$ and $E|G_3 \sim_d F_{22}|G_3$. But from $F \xrightarrow{a} (F_1, F_2)$ and $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ it follows by Lemma 5.26 that $F \xrightarrow{\tau} (F_1|F_{21}, F_{22})$. Moreover, we have $E|G_3 \sim_d F_{22}|G_3$. We just need to show that $G_1 \sim_d F_1|F_{21}$. But from $G_1 \sim_d F_1|G_{21}$ and $G_{21} \sim_d F_{21}$ we have $G_1 \sim_d F_1|F_{21}$ as required. ■

Lemma 5.28 (Cancellation Law) *For processes E, F and G of* BPP_τ *it follows that $E|G \sim_d F|G$ implies $E \sim_d F$.*

Proof: Let E, F and G be processes of BPP_τ such that $E|G \sim_d F|G$. A sufficient condition for $E \sim_d F$ will be to show that the relation

$$R \;=\; \Big\{(E, F) : \text{there exists } G \text{ such that } E|G \sim_d F|G\Big\}$$

is a distributed bisimulation relation.

Suppose $(E, F) \in R$. Hence there exists a process G such that $E|G \sim_d F|G$. Suppose for some $\mu \in Act$ and processes E_1 and E_2 that $E \xrightarrow{\mu} (E_1, E_2)$. Then $E|G \xrightarrow{\mu} (E_1, E_2|G)$. As $E|G \sim_d F|G$ we know that $F|G$ is capable of performing a matching μ-move. We have one of the following three possibilities: either the μ-move originates from F, or G, or $\mu = \tau$ and the τ-move is the result of a synchronisation between F and G. The arguments for the first two cases can readily be obtained from the proof of the cancellation law of Section 5.3, i.e. Lemma 5.16. Thus we only consider the last case.

Suppose $\mu = \tau$ and $F \xrightarrow{a} (F_1, F_2), G \xrightarrow{\bar{a}} (G_1, G_2)$ for some $a \in Act$ and processes F_1, F_2, G_1 and G_2 such that $F|G \xrightarrow{\tau} (F_1|G_1, F_2|G_2)$. Then $E_1 \sim_d F_1|G_1$ and $E_2|G \sim_d F_2|G_2$. Especially, we have $G \xrightarrow{\bar{a}} (G_1, G_2)$ with $E_2|G \sim_d F_2|G_2$. Thus from Lemma 5.27 we get $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ for processes F_{21}, F_{22} and G_3 such that $G_1 \sim_d F_{21}$ and $E_2|G_3 \sim_d F_{22}|G_3$. But then $F \xrightarrow{a} (F_1, F_2)$ and $F_2 \xrightarrow{\bar{a}} (F_{21}, F_{22})$ and therefore from Lemma 5.26 we conclude that $F \xrightarrow{\tau} (F_1|F_{21}, F_{22})$. We now have $E_2|G_3 \sim_d F_{22}|G_3$, i.e. $(E_2, F_2) \in R$. Hence we only need to demonstrate that $(E_1, F_1|F_{21}) \in R$. But this follows from $E_1 \sim_d F_1|G_1$ and $G_1 \sim_d F_{21}$. By symmetry we now conclude that R is a distributed bisimulation equivalence. ■

We are now ready to state and prove the theorem of unique decomposition. The definition of primes is as in Section 5.1 but defined relative to distributed bisimulation equivalence and by replacing $\|$ with $|$.

Theorem 5.29 (Unique Decomposition) *Let E be a process of* BPP_τ. *Then E can be expressed uniquely, up to $\sim_d$, as a parallel composition of primes.*

Proof: That any process E of BPP_τ can be expressed as a parallel composition of primes wrt $\sim_d$ is straightforward; it follows by induction on the initial degree of parallelism of E. The proof that this decomposition is unique proceeds by induction on $\mathcal{D}(E)$.

Suppose that $E \sim_d F$ and that E and F are decompositions into primes given by

$$E \equiv A_1|A_2| \cdots | A_n|C,$$
$$F \equiv B_1|B_2| \cdots |B_m|C.$$

That is, the two decompositions have a common prime factor. By the cancellation law we get $A_1|A_2|\cdots|A_n \sim_d B_1|B_2|\cdots|B_m$. From the induction hypothesis we then conclude that $A_1|A_2|\cdots|A_n$ and $B_1|B_2|\cdots|B_m$ are identical decompositions. But then the prime decompositions for E and F are also identical.

Finally, suppose that $E \sim_d F$ and that E and F have decompositions into primes with no prime factor in common as follows:

$$E \equiv A_1|A_2| \cdots | A_n,$$
$$F \equiv B_1|B_2| \cdots |B_m,$$

where for all i,j it follows that $A_i \not\sim_d B_j$. First of all, if $n = 1$ or $m = 1$ then both E and F are prime and therefore $n = m = 1$. But then $A_1 \sim_d B_1$ contradicting the distinctness assumption on the primes of E and F. Thus $n, m > 1$.

Without loss of generality assume that $\mathcal{D}(A_1) \leq \mathcal{D}(A_i), \mathcal{D}(B_j)$ for all i,j. Suppose $A_1 \xrightarrow{\mu} (G_{11}, G_{12})$ for some G_{11}, G_{12} such that $E \xrightarrow{\mu} (G_{11}, G_{12}|A_2|\cdots|A_n)$. As $E \sim_d F$ we know that F is capable of performing a matching μ-move. We have one of the following two cases: either the μ-move from F originates from a single prime factor of F, or $\mu = \tau$ and F performs the τ-move via a synchronisation between two prime factors of F. The arguments for the first case can readily be obtained from the proof of unique decomposition of Section 5.3, i.e. Theorem 5.17. Thus we only consider the second case.

Suppose $\mu = \tau$ and that F always matches the move of A_1 by a synchronisation between two prime factors of F. All actions performed by A_1 must be τ-actions;

otherwise the first case applies. We choose $A_1 \xrightarrow{\tau} (G_{11}, G_{12})$ for some G_{11}, G_{12} such that $E \xrightarrow{\tau} (G_{11}, G_{12}|A_2|\cdots|A_n)$ with $\mathcal{D}(E) = 1 + \mathcal{D}(G_{12}|A_2|\cdots|A_n)$. If we have $\mathcal{D}(A_1) > 0$ then this choice is possible from Lemma 5.24 and if $\mathcal{D}(A_1) = 0$ then it follows from Lemma 5.23 since $G_{12} \sim_d \mathbf{0}$ must be the case.

Now, suppose F matches the move $E \xrightarrow{\tau} (G_{11}, G_{12}|A_2|\cdots|A_n)$ by a synchronisation between two prime components. We may without loss of generality assume that $B_1 \xrightarrow{a} (H_{11}, H_{12})$ and $B_2 \xrightarrow{\bar{a}} (H_{21}, H_{22})$ for some $a \in Act$ and processes H_{11}, H_{12}, H_{21} and H_{22} such that $F \xrightarrow{\tau} (H_{11}|H_{21}, H_{12}|H_{22}|B_3|\cdots|B_m)$ with $G_{12}|A_2|\cdots|A_n \sim_d H_{12}|H_{22}|B_3|\cdots|B_m$. From Lemma 5.22 we observe that we cannot have $H_{12} \not\sim_d \mathbf{0}$ and $H_{22} \not\sim_d \mathbf{0}$ since otherwise we would have

$$\mathcal{D}(G_{12}|A_2|\cdots|A_n) \quad \neq \quad \mathcal{D}(H_{12}|H_{22}|B_3|\cdots|B_m).$$

Assume without loss of generality that $H_{12} \sim_d \mathbf{0}$ is the case. We therefore consider the move $F \xrightarrow{a} (H_{11}, H_{12}|B_2|\cdots|B_m)$ and know that E must be able to match it. Assume without loss of generality that $A_2 \xrightarrow{a} (G_{21}, G_{22})$ for some processes G_{21} and G_{22} such that we have the move $E \xrightarrow{a} (G_{21}, A_1|G_{22}|A_3|\cdots|A_n)$ with $H_{11} \sim_d G_{21}$ and $H_{12}|B_2|\cdots|B_m \sim_d A_1|G_{22}|A_3|\cdots|A_n$. Since we have

$$\mathcal{D}(A_1|G_{22}|A_3|\cdots|A_n) \quad = \quad \mathcal{D}(H_{12}|B_2|\cdots|B_m) \quad < \quad \mathcal{D}(E)$$

(follows from Lemma 5.22) we may conclude from the induction hypothesis that $H_{12}|B_2|\cdots|B_m$ and $A_1|G_{22}|A_3|\cdots|A_n$ have identical prime decompositions. As $H_{12} \sim_d \mathbf{0}$ it follows that A_1 must equal one of $B_2, B_3 \ldots B_m$ contradicting the distinctness assumption on the primes of E and F. This completes the proof. ■

Just as in Section 5.3 we shall prove a theorem of degeneration. If E of BPP_τ is decomposed into the primes $A_1, A_2 \ldots A_n$ then we shall prove that $A_i \Subset E$ for each $i = 1 \ldots n$. The definition of $\Subset$ is exactly as in Section 5.3 with $\|$ and $\lfloor\!\lfloor$ being replaced by $|$ and $\lfloor$ respectively; we refer to Definition 5.18.

First we need the following technical result.

Lemma 5.30 *Let E and F be processes of* BPP_τ*. Suppose that F is prime. If for some $\mu \in Act$ and E_1, E_2 we have $E \xrightarrow{\mu} (E_1, E_2)$ with $F \Subset E_2$ then $F \Subset E$.*

Proof: We proceed by case analysis on the structure of E using induction on $\mathcal{C}(E)$. We only demonstrate the most interesting case of $E = G_1|G_2$. Since $E \xrightarrow{\mu} (E_1, E_2)$ we have three possibilities as follows: either the μ-move originates from G_1, or from G_2, or $\mu = \tau$ and the τ-move of E is the result of a synchronisation between G_1 and G_2. We only consider the last case here as the arguments for the first two cases readily are obtained form the proof of Lemma 5.19 of Section 5.3.

Thus suppose $\mu = \tau$ and $G_1 \xrightarrow{a} (G_{11}, G_{12}), G_2 \xrightarrow{\bar{a}} (G_{21}, G_{22})$ for some $a \in Act$ and processes G_{11}, G_{12}, G_{21} and G_{22} such that $G_1|G_2 \xrightarrow{\tau} (G_{11}|G_{21}, G_{12}|G_{22})$, i.e. $E_1 \equiv G_{11}|G_{21}$ and $E_2 \equiv G_{12}|G_{22}$. By assumption we have $F \Subset G_{12}|G_{22}$. There are three cases to consider.

(i) Suppose $F \Subset G_{12}$. Then by induction we have $F \Subset G_1$ and therefore also $F \Subset E$ as required.

(ii) Suppose $F \Subset G_{22}$. Then the arguments are as in the above case.

(iii) Suppose $F \sim_d G_{12}|G_{22}$. As F is prime we have either $G_{12} \sim_d \mathbf{0}$ or $G_{22} \sim_d \mathbf{0}$. But if $G_{22} \sim_d \mathbf{0}$ then we must have $F \Subset G_{12}$ (the first case applies) and if $G_{12} \sim_d \mathbf{0}$ then we must have $F \Subset G_{22}$ (the second case applies).

This completes the case. ∎

Theorem 5.31 (Degeneration) *Let E be a process of* BPP$_\tau$. *Assume E is decomposed into the primes $A_1, A_2 \ldots A_n$. Then $A_i \Subset E$ for $i = 1 \ldots n$.*

Proof: The arguments are readily obtained from the proof of Theorem 5.20 of Section 5.3. ∎

Chapter 6

Decidability of $\sim$ for Basic Parallel Processes

In this chapter we show that bisimilarity is decidable on the two process classes of BPP and BPP_τ. We shall first give a proof of decidability using the *tableau decision technique*. However, we shall also provide decidability via the technique described in Chapter 4, i.e. via a finite representability result for bisimilarity. This technique furthermore allows us to discuss other notions of parallelism and in particular to state sufficient conditions under which bisimilarity may be decidable.

An advantage of the tableau decision method is that we can extract sound and complete sequent-based equational theories for bisimulation equivalence on our process classes of BPP and BPP_τ. As our processes clearly include the regular processes, i.e. the calculus RCCS, this result can be seen as a proper extension to Milner's sound and complete equational theory for bisimulation equivalence on regular processes (see [Mil84]).

In previously published work (see [CHM93]) we explored the subclasses of *normed* and also *live* BPP (a process is live if it can never perform an infinite number of identical actions in succession uninterrupted). For these subclasses we obtained proofs of decidability of bisimulation equivalence utilising the tableau method. In order to obtain our result we necessarily had to show that every tableau was finite. This, in turn, was achieved through a decomposition of pairs of bisimilar processes, and the soundness of this decomposition required validity of the cancellation law wrt full merge (see the discussion in Section 2.5.5). Hence the restriction to normed and live BPP; we have seen in Chapter 5 that such a law is valid in the case of normed processes and the corresponding law in the case of live processes was proved in [MM90].

In [CHM93] we also presented a sound and complete sequent-based equational theory for bisimulation equivalence. The theory was extracted from the tableau method and in the case of proving completeness we relied on deriving *roots* of successful tableaux. Thus we could only obtain completeness on the subclass of normed and live BPP although it seemed feasible expecting the equational theory to be complete wrt the whole calculus. Indeed, as discovered by Hirshfeld, the arguments of decidability presented in [CHM93] just required a different notion of decomposition of pairs of bisimilar processes (which is valid on all, not necessarily normed or live processes) in order to be extendable to the whole calculus BPP. This also allowed us to extend the proof of completeness of the equational theory to the whole of BPP.

Our decidability result relies on a search for a successful tableau. We can demonstrate that there are only finitely many tableaux thus guaranteeing that the search eventually stops. However, we have not been able to provide an upper bound on the size of tableaux and hence we have not been able to estimate the complexity of solving the bisimilarity problem.

The tableau method provides a very natural procedure for checking bisimilarity: given two processes E and F one tries to build a self-bisimulation containing the pair (E, F) by examining the possible moves from E and F. By a delicate use of the result of unique decomposition (Chapter 5) we shall present a very different procedure for checking bisimilarity between normed processes. The method is inspired by Caucal [Cau90] and has also some similarity with the technique presented in Chapter 4. Essentially, it consists in decomposing expressions into primes and then checking for syntactical identity. The algorithm also allows for a more precise complexity bound although we shall not take the opportunity to perform such an analysis in details.

Rather than presenting our results both in the setting of BPP and BPP$_\tau$ we have chosen only to concentrate on BPP$_\tau$; in the setting of BPP the results follows from similar arguments. Along the way we shall underline the difference between BPP and BPP$_\tau$ as manifested in the proof of decidability and also in the equational theory.

6.1 A Tableau Decision Method for $\sim$ on BPP$_\tau$

In this section we fix a finite family $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\}$ of guarded BPP$_\tau$ equations in full standard form of degree 2. We are interested in deciding for any α and β of $Var(\Delta)^\otimes$ whether $\alpha \sim \beta$ is the case or not. The procedure for checking $\alpha \sim \beta$ is based on the *tableau decision method* as for instance utilised by Hüttel and Stirling in [HS91]. We refer to Section 2.5.2 for a general introduction to the tableau technique; in particular, we introduced some terminology for tableaux which we intend to use in this chapter.

A *tableau* for $\alpha = \beta$ is a maximal proof tree whose root is labelled $\alpha = \beta$ and where the labelling of immediate successors of a node is determined according to the rules of the tableau system presented in Table 6–1. For the presentation of rule Rec we introduce the notation unf(α) to mean the *unfolding* of α given as follows (assuming that $\alpha \equiv Y_1|\cdots|Y_m$):

$$\begin{aligned} \mathrm{unf}(\alpha) \quad = \quad & \sum\Big\{\mu(Y_1|\cdots|\alpha_i|\cdots|Y_m) : \mu\alpha_i \in Y_i\Big\} \\ & + \sum\Big\{\tau(Y_1|\cdots|\alpha_i|\cdots|\alpha_j|\cdots|Y_m) : a\alpha_i \in Y_i,\ \bar{a}\alpha_j \in Y_j,\ i < j\Big\}, \end{aligned}$$

where the notation $\mu\alpha \in Y$ is introduced to mean that $\mu\alpha$ is a summand of the defining equation for Y (as given by Δ). Notice that unf(α) essentially consists of a substitution (replacing variables of α with their defining expressions) followed by an application of the *expansion law* for bisimilarity (see [Mil89]).

We shall identify BPP$_\tau$ expressions in our tableaux up to the structural congruence $\equiv$, i.e. up to the laws of associativity, commutativity and **0**-absorption of sum and parallel composition.

In building tableaux the rules are only applied to nodes that are not *terminal*.

REC	$$\frac{\alpha = \beta}{\mathrm{unf}(\alpha) = \mathrm{unf}(\beta)}$$	
SUM	$$\frac{\sum_{i=1}^{n} \mu_i\alpha_i = \sum_{j=1}^{m} \nu_j\beta_j}{\{\mu_i\alpha_i = \nu_{f(i)}\beta_{f(i)}\}_{i=1}^{n} \qquad \{\nu_j\beta_j = \mu_{g(j)}\alpha_{g(j)}\}_{j=1}^{m}}$$	where $f : \{1,\ldots,n\} \to \{1,\ldots,m\}$ $g : \{1,\ldots,m\} \to \{1,\ldots,n\}$
PREFIX	$$\frac{\mu\alpha = \mu\beta}{\alpha = \beta}$$	
SUBL	$$\frac{\alpha\vert\gamma = \delta}{\beta\vert\gamma = \delta}$$	if there is a dominated node labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \succ \beta$
SUBR	$$\frac{\delta = \alpha\vert\gamma}{\delta = \beta\vert\gamma}$$	if there is a dominated node labelled $\alpha = \beta$ or $\beta = \alpha$ with $\alpha \succ \beta$

Table 6–1: Rules of the tableau system for $\sim$ on BPP_τ.

$$
\begin{array}{lccc}
\textsc{Rec} & & \dfrac{\alpha = \beta}{\sum \mu_i \alpha_i = \sum \nu_i \beta_i} & \\
\textsc{Sum} & & \overline{\hphantom{\mu_1\alpha_1 = \mu_1\beta_1 \quad \cdots \quad \mu_n\alpha_n = \mu_n\beta_n}} & \\
\textsc{Prefix} & \dfrac{\mu_1 \alpha_1 = \mu_1 \beta_1}{\alpha_1 = \beta_1} \quad \cdots \quad \dfrac{\mu_n \alpha_n = \mu_n \beta_n}{\alpha_n = \beta_n} & & \textsc{Prefix}
\end{array}
$$

Figure 6–1: A basic step.

A terminal node can either be *successful* or *unsuccessful*. A successful terminal node is one labelled $\alpha = \alpha$, while an unsuccessful terminal node is one labelled either $\mu\alpha = \nu\beta$ such that $\mu \neq \nu$, or $\mu\alpha = \mathbf{0}$, or $\mathbf{0} = \nu\beta$. A tableau is successful if, and only if, all terminal nodes are successful; otherwise it is unsuccessful.

Tableaux are built from *basic steps*. A basic step for $\mathbf{n} : \alpha = \beta$ (which is called a *basic node*) consists of an application of Rec to $\mathbf{n} : \alpha = \beta$ followed (possibly) by an application of Sum followed by an application of Prefix to each of its consequents. See Figure 6–1 for the schema of a basic step. A basic step represents a set of single transition steps in the operational semantics: for each consequent $\mathbf{n}_i : \alpha_i = \beta_i$ we have $\alpha \xrightarrow{\mu_i} \alpha_i$ and $\beta \xrightarrow{\mu_i} \beta_i$.

If tableaux were constructed only using basic steps then in general they would be infinite since labels of nodes can become arbitrarily large due to unfoldings. Thus we need a method of *decomposing* labels of nodes into "smaller" labels. This is the purpose of the two rules SubL and SubR. In order to facilitate the use of these rules we shall rely on the following well-founded ordering.

Definition 6.1 *By $\prec$ we denote the well-founded ordering on $Var(\Delta)^\otimes$ given as follows:*

$$X_1^{k_1} | \cdots | X_n^{k_n} \quad \prec \quad X_1^{l_1} | \cdots | X_n^{l_n}$$

iff there exists j such that $k_j < l_j$ and for all $i < j$ we have $k_i = l_i$.

It is straightforward to show that $\prec$ is well-founded. We shall furthermore rely on the fact that $\prec$ is *total* in the sense that for any $\alpha, \beta \in Var(\Delta)^\otimes$ such that $\alpha \not\equiv \beta$ we have $\alpha \prec \beta$ or $\beta \prec \alpha$. Also we shall rely on the fact that $\alpha \prec \beta$ implies

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X_1 = X_2}{a(X_1|X_4) = aX_3}\;\textsc{Rec}}{X_1|X_4 = X_3}\;\textsc{Prefix}}{X_2|X_4 = X_3}\;\textsc{SubL}}{a(X_3|X_4) + \bar{a}X_2 + \tau X_3 = a(X_3|X_4) + \bar{a}X_2 + \tau(X_2|X_4)}\;\textsc{Rec}}{\dfrac{a(X_3|X_4) = a(X_3|X_4)}{X_3|X_4 = X_3|X_4}\;\textsc{Prefix} \qquad \dfrac{\bar{a}X_2 = \bar{a}X_2}{X_2 = X_2} \qquad \dfrac{\dfrac{\tau X_3 = \tau(X_2|X_4)}{X_3 = X_2|X_4}\;\textsc{Prefix}}{X_3 = X_3}\;\textsc{SubR}}\;\textsc{Sum}
$$

Figure 6–2: A successful tableau for $X_1 = X_2$ (Example 6.2).

$\alpha|\gamma \prec \beta|\gamma$ as well as $\alpha \prec \beta|\gamma$ for any $\gamma \in \mathit{Var}(\Delta)^{\otimes}$. All these properties are easily seen to hold for $\prec$.

When building tableaux basic nodes might *dominate* other basic nodes; we say a basic node $\mathbf{n} : \alpha|\gamma = \delta$ or $\mathbf{n} : \delta = \alpha|\gamma$ dominates any node $\mathbf{n}' : \alpha = \beta$ or $\mathbf{n}' : \beta = \alpha$ which appears *above* $\mathbf{n}$ in the tableau in which $\alpha \succ \beta$ and to which rule Rec is applied. Whenever a basic node dominates a previous one, we apply one of the Sub rules to reduce the expressions before applying the Rec rule. If both rules are applicable at the same time then arbitrarily choose one of them to be applied. Notice that the side condition for the Sub rules is a condition on tableaux and not on the particular rules.

Example 6.2 *Let Δ be the following family in full standard form of degree 2: $\{X_1 \stackrel{\mathrm{def}}{=} a(X_1|X_4), X_2 \stackrel{\mathrm{def}}{=} aX_3, X_3 \stackrel{\mathrm{def}}{=} a(X_3|X_4) + \bar{a}X_2 + \tau(X_2|X_4), X_4 \stackrel{\mathrm{def}}{=} \bar{a}\}$. A successful tableau for $X_1 = X_2$ is presented in Figure 6–2.*

Lemma 6.3 *Every tableau for $\alpha = \beta$ is finite. Furthermore, there is only a finite number of tableaux for $\alpha = \beta$.*

Proof: Let $T(\alpha = \beta)$ be a tableau with root labelled $\alpha = \beta$. It can only be infinite if there exists an infinite path as every node has finite branching degree. Hence suppose π is such an infinite path starting at the root $\mathbf{r} : \alpha = \beta$. The path π can only be infinite if it contains infinitely many basic nodes to which the tableau rule REC is applied. This is due to the well-foundedness of the ordering $\prec$ on $Var(\Delta)^{\otimes}$ which is decreased through applications of the SUB rules. Thus from the path π we can form an infinite sequence S of nodes $\{\mathbf{n}_i : \alpha_i = \beta_i\}_{i=1}^{\infty}$ by collecting (in order of appearance) the basic nodes along π to which the rule REC is applied. Hence $\mathbf{n}_1 : \alpha_1 = \beta_1$ represents the root, $\mathbf{n}_2 : \alpha_2 = \beta_2$ represents the second node along π at which REC is applied, and so on.

An expression α can be viewed as a vector $\tilde{v}$ of $\mathbb{N}^n$: the value of the i'th coordinate of $\tilde{v}$, denoted $\tilde{v}(i)$, indicates the number of occurrences of variable X_i in α. Thus we can represent the sequence S by an infinite sequence of vectors $\{\tilde{u}_i\}_{i=1}^{\infty}$ where $\tilde{u}_i \in \mathbb{N}^{2n}$ for all i. The first n coordinates represent α_i and the last coordinates represent β_i.

Consider the infinite sequence $\{\tilde{u}_i(1)\}_{i=1}^{\infty}$ consisting of all the first coordinates of vectors of the sequence S. If this sequence has an upper bound we extract from S an infinite sequence S_1 of vectors $\{\tilde{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\tilde{v}_i$ remains constant throughout S_1. If the sequence $\{\tilde{u}_i(1)\}_{i=1}^{\infty}$ does not have an upper bound we extract from S an infinite sequence S_1 of vectors $\{\tilde{v}_i\}_{i=1}^{\infty}$ with the property that the first coordinate of $\tilde{v}_i$ is nondecreasing, i.e. $\tilde{v}_i(1) \leq \tilde{v}_j(1)$ whenever $i \leq j$. Continuing in this fashion we arrive at an infinite sequence S_{2n} of vectors $\{\tilde{w}_i\}_{i=1}^{\infty}$ with the property that all coordinate sequences are nondecreasing. But then every node in this sequence is dominated by every node after it, so the rule REC cannot be applied to any of these nodes, as a SUB rule is applicable.

For the proof of the second part we say a *partial* tableau for $\alpha = \beta$ is a finite proof tree with root labelled $\alpha = \beta$ and where the labelling of immediate successors of a node are determined according to the tableau rules of Table 6–1. Thus a tableau is a special kind of partial tableaux where each leaf is terminal. Next we note that there are only finitely many partial tableaux with a given height since any tableau rule at most introduces finitely many new nodes. Now if there were an infinite number of tableaux there must be an infinite sequence of partial tableaux, each of which being derived from the previous by the application of some rule to the node most recently introduced. But then this sequence provides a partial tableau which can be extended in the infinite which by the first part of this proof cannot be. ■

We now proceed by showing the soundness and completeness of the tableau system.

Theorem 6.4 (Completeness) *If $\alpha \sim \beta$ then there exists a successful tableau with root labelled $\alpha = \beta$.*

Proof: Suppose $\alpha \sim \beta$. If we can construct a tableau $T(\alpha = \beta)$ for $\alpha = \beta$ with the property that any node $\mathbf{n} : E = F$ of $T(\alpha = \beta)$ satisfies $E \sim F$, then by Lemma 6.3 this construction must terminate and each terminal will be successful. Thus the tableau itself will be successful.

We can construct such a $T(\alpha = \beta)$ if we verify that each rule of the tableau system is *forward sound* in the sense that if the antecedent as well as all nodes above relate bisimilar processes then it is possible to find a set of consequents relating bisimilar processes. It is easily verified that the rules are indeed forward sound in this sense. Notice in particular that the SUB rules are forward sound because bisimilarity is a congruence wrt parallel composition and that rule REC reflects the expansion law for parallel composition (see [Mil89]). ■

The proof of soundness of the tableau system relies on the alternative characterisation of bisimulation equivalence given by a sequence of approximations $\{\sim_n\}_{i=0}^{\infty}$ as described in Chapter 4 and valid when transition graphs are image-finite. Certainly, BPP_τ processes yield image-finite transition graphs.

Theorem 6.5 (Soundness) *If there is a successful tableau for $\alpha = \beta$ then $\alpha \sim \beta$.*

Proof: Suppose $T(\alpha = \beta)$ is a tableau for $\alpha = \beta$, and that $\alpha \not\sim \beta$. We shall construct a maximal path $\pi = \{\mathbf{n}_i : E_i = F_i\}$ through this tableau starting at the root $\alpha = \beta$ in which $E_i \not\sim F_i$ for each i. Hence the terminal node of this path cannot be successful, so there can be no successful tableau for $\alpha = \beta$.

While constructing π, we shall at the same time construct the sequence of integers $\{m_i \ : \ E_i \not\sim_{m_i} F_i$ and $E_i \sim_j F_i$ for all $j < m_i\}$. We shall also prove along the way that this sequence is nonincreasing, and strictly decreasing through applications of the rule PREFIX.

Given $\mathbf{n}_i : E_i = F_i$ and m_i, we get $\mathbf{n}_{i+1} : E_{i+1} = F_{i+1}$ and m_{i+1} according to the following cases.

(i) If REC is applied to $\mathbf{n}_i$, then the consequent is $\mathbf{n}_{i+1}$ and $m_{i+1} = m_i$.

(ii) If SUM is applied to $\mathbf{n}_i$, then we know that there must be some consequent $\mathbf{n}_{i+1} : E_{i+1} = F_{i+1}$ of node $\mathbf{n}_i$ with $E_{i+1} \not\sim_{m_i} F_{i+1}$. Therefore there is a $m_{i+1} \leq m_i$ such that $E_{i+1} \not\sim_{m_{i+1}} F_{i+1}$ and for all $j < m_{i+1}$, $E_{i+1} \sim_j F_{i+1}$.

(iii) If PREFIX is applied to $\mathbf{n}_i$, then the consequent is $\mathbf{n}_{i+1}$ and $m_{i+1} = m_i - 1$.

(iv) If SUBL is applied to $\mathbf{n}_i : E_i = F_i$ then $E_i = F_i$ must be of the form $\gamma|\zeta = \eta$ with dominated node $\mathbf{n}_j : \gamma = \delta$ $(\gamma \succ \delta)$. Since between $\mathbf{n}_i$ and $\mathbf{n}_j$ there must have been an intervening application of the rule PREFIX, we must have that $m_i < m_j$. We take the node $\mathbf{n}_{i+1} : \delta|\zeta = \eta$, and show that we have some valid $m_{i+1} \leq m_i$, that is, that $\delta|\zeta \not\sim_{m_i} \eta$. But this follows from $\gamma \sim_{m_i} \delta$ and $\gamma|\zeta \not\sim_{m_i} \eta$. The arguments for the other possible applications of the SUB rules are identical.

That the above conditions hold of the resulting path is now clear. ■

Our proof of the above theorem (which rely on using the bisimulation approximations) could also have been established using the notion of self-bisimulation (see Section 2.5.4 for an introduction to self-bisimulations). For given a successful tableau $T(\alpha = \beta)$ for $\alpha = \beta$, let R be the binary relation on $Var(\Delta)^{\otimes}$ consisting of labels of basic nodes of $T(\alpha = \beta)$. It is then possible to show that R constitutes a self-bisimulation. Furthermore, from $R^{\leftrightarrow^*} \subseteq \sim$ (which any self-bisimulation on $Var(\Delta)^{\otimes}$ satisfies) we would clearly have $\alpha \sim \beta$ as is required for the proof of soundness.

We are now in a position to infer the decidability of bisimulation equivalence on BPP_τ processes. In order to decide the validity of $\alpha = \beta$ we simply start listing tableaux for $\alpha = \beta$ and stop with answer "yes" if a successful tableau has been found. If we list all of the finite number of finite tableaux (systematically, so that we recognise when they have all been listed) and fail to discover a successful one, then we answer "no". By soundness and completeness of the tableau system we know that this procedure will always give the right answer. Thus the decidability result is established.

Theorem 6.6 *Bisimulation equivalence is decidable on the class of* BPP_τ *processes.*

Certainly, the above theorem also holds for the class BPP. The arguments are more or less the same; the major modification concerns the tableau rule REC since in the case of BPP we need not take into account the possibility of parallel expressions performing synchronisations. We state the result without further comments.

Corollary 6.7 *Bisimulation equivalence is decidable on the class of* BPP *processes.*

Notice that by Theorem 6.6 we have obtained a delicate line between theories of decidability and undecidability for bisimulation equivalence: by extending BPP_τ with the combinator of *restriction* [Mil89] we know that bisimulation equivalence becomes undecidable (see Section 2.3). This further emphasises the importance (and descriptive power) of restriction as a combinator.

6.1.1 Finite Representability of $\sim$

In Chapter 4 we obtained semi-decidability of bisimilarity on context-free processes through a finite representability result; we proved that the largest bisimulation relation $\sim$ on $Var(\Delta)^*$, where Δ is a finite family of BPA_0 equations in GNF, is generable from a finite self-bisimulation. In a similar fashion, we shall now prove that the largest bisimulation relation $\sim$ on $Var(\Delta)^\otimes$, where Δ is a finite family of BPP_τ equations in full standard form, can be represented by some finite self-bisimulation on $Var(\Delta)^\otimes$. This will demonstrate semi-decidability (and therefore decidability since processes of BPP_τ generate image-finite transition graphs) of bisimilarity on BPP_τ. Furthermore, by presenting this technique we shall be able to state sufficient conditions on the parallel operator for which such a finite representability result is possible.

As before we assume a fixed family Δ of BPP_τ equations in full standard form and let $\sim$ denote the largest bisimulation relation on $Var(\Delta)^{\otimes}$. We are interested in showing that there exists a finite self-bisimulation R on $Var(\Delta)^{\otimes}$ such that $R^{\leftrightarrow^*} = \sim$. To do this we shall decompose bisimilar pairs (α, β) into "smaller" bisimilar pairs $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \ldots (\alpha_m, \beta_m)$ such that there are only finitely many bisimilar pairs that cannot be decomposed further. The notion of decomposition on $Var(\Delta)^{\otimes} \times Var(\Delta)^{\otimes}$ that we shall exploit relies on the ordering $\prec$ (see Definition 6.1) and it formally defined as follows.

Definition 6.8 *A pair (α, β) of bisimilar processes over $Var(\Delta)^{\otimes}$ is* decomposable *if we have $\alpha \equiv \alpha_1|\alpha_2$ (or $\beta \equiv \beta_1|\beta_2$) such that there exists $\gamma \in Var(\Delta)^{\otimes}$ with $\gamma \prec \alpha_2$, $\gamma \sim \alpha_2$ and $\alpha_1|\gamma \sim \beta$ (or $\gamma \prec \beta_2$, $\gamma \sim \beta_2$ and $\alpha \sim \beta_1|\gamma$).*

Notice the apparently redundant requirement of $\alpha_1|\gamma \sim \beta$ (or $\alpha \sim \beta_1|\gamma$) in the above definition: from $\gamma \sim \alpha_2$ (or $\gamma \sim \beta_2$) we have $\alpha_1|\gamma \sim \beta$ (or $\alpha \sim \beta_1|\gamma$) since $\sim$ is a congruence wrt parallel composition. However, this requirement is part of our definition of the concept of decomposition due to generality since wrt other notions of parallelism (e.g. CSP parallelism) bisimilarity is no longer a congruence.

Lemma 6.9 *Any binary relation R over $Var(\Delta)^{\otimes}$ of the form*

$$R \;=\; \Big\{(\alpha, \beta) : \alpha \not\equiv \beta, \alpha \sim \beta \text{ and } (\alpha, \beta) \text{ not decomposable}\Big\}$$

is finite.

Proof: By contradiction suppose that R as given above is infinite. As in the proof of Lemma 6.3, i.e. the proof of finiteness of tableaux, we may find elements $(\alpha, \beta) \in R$ and $(\alpha|\gamma, \beta|\delta) \in R$. We have $\alpha \not\equiv \beta$ and therefore $\alpha \prec \beta$ or $\beta \prec \alpha$ since $\prec$ is total. Assume without loss of generality that $\beta \prec \alpha$ is the case. As $\sim$ is a *congruence* wrt parallel composition we have from $\alpha \sim \beta$ that $\beta|\gamma \sim \beta|\delta$. But this shows that the pair $(\alpha|\gamma, \beta|\delta)$ is decomposable. We now have our required contradiction and must conclude that R is finite. ∎

We are almost in a position to show that $\sim$ can be represented by a finite self-bisimulation. To facilitate the proof of this we shall rely on the binary ordering $\sqsubset$ on $Var(\Delta)^{\otimes} \times Var(\Delta)^{\otimes}$ defined by

$$(\alpha, \beta) \sqsubset (\alpha', \beta') \quad \text{iff} \quad \alpha \prec \alpha' \text{ or } (\alpha \equiv \alpha' \text{ and } \beta \prec \beta').$$

Since $\prec$ is well-founded we clearly have well-foundedness of $\sqsubset$. Therefore our proof of the next theorem can be based on induction using this ordering.

Theorem 6.10 *There exists a finite binary relation R over $Var(\Delta)^{\otimes}$ such that $R^{\leftrightarrow^*} = \sim$.*

Proof: Let R_1 be the relation $\{(X, X) : X \in Var(\Delta)\}$ and let R_2 be a largest binary relation over $Var(\Delta)^{\otimes}$ of the form

$$R_2 \;=\; \Big\{(\alpha, \beta) : \alpha \not\equiv \beta, \alpha \sim \beta \text{ and } (\alpha, \beta) \text{ not decomposable}\Big\}.$$

Clearly R_1 is finite and finiteness of R_2 follows from Lemma 6.9. Hence the relation $R = R_1 \cup R_2$ is finite.

We aim to show that $R^{\leftrightarrow^*} = \sim$. Since $R \subseteq \sim$ and $\sim$ is an equivalence as well as a congruence wrt parallel composition we have $R^{\leftrightarrow^*} \subseteq \sim$. Therefore we only need to consider $\sim \subseteq R^{\leftrightarrow^*}$. We proceed by induction on the ordering $\sqsubset$. Let $\alpha, \beta \in Var(\Delta)^{\otimes}$ and assume that $\alpha \sim \beta$. If $\alpha \equiv \beta$ then from R_1 we clearly have $(\alpha, \beta) \in R^{\leftrightarrow^*}$. Hence suppose that $\alpha \not\equiv \beta$. If (α, β) is not decomposable then we have $(\alpha, \beta) \in R_2$ from maximality of R_2 and therefore $(\alpha, \beta) \in R^{\leftrightarrow^*}$ as required. Finally suppose that (α, β) is decomposable. Assume without loss of generality that $\alpha \equiv \alpha_1 | \alpha_2$ and that there exists $\gamma \in Var(\Delta)^{\otimes}$ such that $\gamma \prec \alpha_2$, $\alpha_2 \sim \gamma$ and $\alpha_1 | \gamma \sim \beta$. We clearly have $(\alpha_1 | \gamma, \beta) \sqsubset (\alpha_1 | \alpha_2, \beta)$ and therefore by induction $(\alpha_1 | \gamma, \beta) \in R^{\leftrightarrow^*}$. We also have $(\gamma, \alpha_2) \sqsubset (\alpha_1 | \alpha_2, \beta)$ which again by induction implies $(\gamma, \alpha_2) \in R^{\leftrightarrow^*}$. But from $(\alpha_1 | \gamma, \beta) \in R^{\leftrightarrow^*}$ together with $(\gamma, \alpha_2) \in R^{\leftrightarrow^*}$ we have $(\alpha, \beta) \in R^{\leftrightarrow^*}$ as required. ∎

Corollary 6.11 *Let $\alpha, \beta \in Var(\Delta)^{\otimes}$. Then $\alpha \sim \beta$ if, and only if, there exists a finite self-bisimulation on $Var(\Delta)^{\otimes}$ containing the pair (α, β).*

Proof: The finite relation R of Theorem 6.10 is clearly a self-bisimulation. If $\alpha \sim \beta$ then from $R^{\leftrightarrow^*} = \sim$ we conclude that $R \cup \{(\alpha, \beta)\}$ is a self-bisimulation as well. Conversely, if $R \cup \{(\alpha, \beta)\}$ is a finite self-bisimulation then $\alpha \sim \beta$ since any self-bisimulation is a witness for bisimilarity (see Proposition 2.41, page 58). ■

As in Chapter 4 the above corollary is sufficient for semi-decidability of bisimilarity via the procedure that searches for a finite self-bisimulation containing the pair (α, β) that we wish to compare. We refer the reader to Chapter 4, Section 4.2 for the details. Since bisimulation inequivalence is semi-decidable we may conclude (once more) that bisimilarity is decidable on BPP$_\tau$.

Let us at this point state sufficient conditions on the parallel operator used in the description of Δ in order to obtain the result corresponding to Corollary 6.11 and hence semi-decidability of $\sim$. Thus we consider a finite family Δ of process equations in full standard form but replace the CCS parallel operator $|$ with any other parallel operator and want to find sufficient conditions satisfied by this parallel operator in order to obtain $R^{\leftrightarrow^*} = \sim$ where $\sim$ is the largest bisimulation on $Var(\Delta)^{\otimes}$ and R is a finite self-bisimulation on $Var(\Delta)^{\otimes}$.

Clearly the parallel operator must be associative and commutative in order for the notation $Var(\Delta)^{\otimes}$ to make sense. Now, the crucial step in obtaining $R^{\leftrightarrow^*} = \sim$ for a *finite* self-bisimulation relies on showing that any binary relation on $Var(\Delta)^{\otimes}$ of the form

$$R \;=\; \Big\{(\alpha, \beta) : \alpha \not\equiv \beta, \alpha \sim \beta \text{ and } (\alpha, \beta) \text{ not decomposable}\Big\}$$

is finite. In our proof of Lemma 6.9 stating this fact for CCS parallel composition we only used the property that bisimilarity is a congruence wrt parallel composition.

Hence, whenever bisimilarity satisfies the laws of associativity and commutativity and furthermore is a congruence for the parallel operator used in the description of Δ then the largest bisimulation relation on $Var(\Delta)^{\otimes}$ is represented by a finite self-bisimulation and hence bisimilarity is semi-decidable. If we in addition know that bisimulation inequivalence is semi-decidable (as in the circumstances where Δ generates an image-finite transition graph) then we may conclude that bisimilarity is decidable.

As an example let us consider the parallel operator $\|_A$ which we informally introduced in Chapter 2. It is an operator related to the parallel combinator of CSP [Hoa85] and has been used by for instance Taubner (see [Tau89]). Here we assume that A is a set of labels from Λ_1. So in fact we are considering a family $\{\|_A\}_{A \subseteq \Lambda_1}$ of parallel operators. For each $A \subseteq \Lambda_1$, $\|_A$ is a binary operator for parallel composition of processes; given processes E and F then $E\|_A F$ is a process that behaves like $E\|F$ (for our full merge operator $\|$) however under the constraint that E and F *must* synchronise on actions from A while all other actions are performed autonomously. This is described by the transition rules

$$\frac{E \xrightarrow{b} E'}{E\|_A F \xrightarrow{b} E'\|_A F}(b \notin A) \qquad \frac{F \xrightarrow{b} F'}{E\|_A F \xrightarrow{b} E\|_A F'}(b \notin A)$$

$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E\|_A F \xrightarrow{a} E'\|_A F'}(a \in A)$$

Notice that $\|_\emptyset$ has the same meaning as the full merge operator $\|$.

Certainly $\|_A$ is associative and commutative. Furthermore, it is not hard to show that from $E \sim F$ we have $E\|_A G \sim F\|_A G$ for any G. Hence bisimilarity is a congruence wrt the operator $\|_A$. Here it is important that both parallel operators are indexed with the *same* set of labels as otherwise it is easy to find a counter example to the congruence property.

Now, if Δ is a finite family of process equations in full standard form such that every instance of the parallel operator is indexed by the same set of labels then the largest bisimulation relation $\sim$ on $Var(\Delta)^{\otimes}$ is represented by a finite self-bisimulation and therefore we may conclude that bisimulation equivalence is semi-decidable. Since such a family generates an image-finite transition graph we also conclude that bisimulation inequivalence is semi-decidable. Therefore bisimulation equivalence is decidable on $Var(\Delta)^{\otimes}$.

A natural question is how expressive a calculus based on the operator $\|_A$ is. Interestingly enough, within such a language it is possible to build counters (see Section 2.3 for the formal specification of a counter).

Example 6.12 *Let A consists of the singleton $\{z\}$. Then consider the family $\Delta = \left\{U \stackrel{\text{def}}{=} zU + i(V\|_A U), V \stackrel{\text{def}}{=} dW, W \stackrel{\text{def}}{=} zW\right\}$. It is not hard to show that for any $n \in I\!N$ the process $V^n\|_A U$ represents a counter with value n.*

By relying on the above example we may simulate two-counter machines if we allow the set of labels on the parallel operators to vary: any two-counter machine K can be simulated by an expression of the form $E_K = (C_1\|_{A_1}E)\|_{A_2}C_2$ where (for $j = 1, 2$) C_j is a counter (as given by the above example) using actions $A_j = \{z_j, d_j, i_j\}$, and E is a finite-state process communicating with the counters. Thus allowing the set of labels to vary on the parallel operators we arrive at a calculus with full Turing power. It furthermore follows that on such a calculus bisimilarity is undecidable: let K be a two-counter machine and $E_K = (C_1\|_{A_1}E)\|_{A_2}C_2$ the process simulating K. Next form a new process $F_K = (C_1\|_{A_1}F)\|_{A_2}C_2$ which is identical to E_K except from the fact that F may perform a new action \$ (not part of E) when F reaches the final (`halt`) statement and thus detecting whether or not K halts. We clearly have $E_K \not\sim F_K$ if, and only if, K halts.

There exist other notions of parallel composition for which bisimilarity is not a congruence. An example is the CSP binary parallel operator $|||$ where processes must synchronise on actions shared by the two processes (We informally introduced this operator in Chapter 2, see page 30). We shall in Chapter 8 see another example of a parallel operator for which bisimilarity is not a congruence. Hence in these circumstances we cannot use the technique described here for decidability of bisimilarity. If fact, the notion of parallelism that we shall consider in Chapter 8, which consists of an extension of BPP using a general synchronisation mechanism, will provide us with undecidability of bisimilarity. Thus by the result of this subsection we know that bisimilarity cannot be a congruence on this process calculus.

The technique we have used in this subsection (as well as in Chapter 4) for deciding bisimilarity is very different from the tableau technique which we utilised in the previous section. We believe that the technique is rather powerful and may be used to prove decidability of other behavioural equivalences; of particular interest are the weak bisimulation equivalence $\approx$ and the observational congruence $=$ (see [Mil89]). We shall in Chapter 9 return to this point in some details. However one advantage of the tableau decision technique is that it supports a sound and complete equational theory. This is the subject of the following section.

6.2 An Equational Theory for $\sim$ on BPP_τ

We now describe a sound and complete sequent-based equational theory for bisimilarity on BPP_τ processes. We restrict attention to processes in full standard form of degree 2. Let Δ be a finite family of such processes. The theory shall be parameterised by Δ and consists of axioms and inference rules that enable one to derive the root of successful tableaux.

The theory is in spirit similar to that offered in [HS91,Hüt91,CHM93] and is built around sequents of the form $\Gamma \vdash_\Delta E = F$ where Γ is a finite set of *assumptions* of the form $\alpha = \beta$, and E and F are BPP_τ expressions (see also the discussion of sequent-based equational theories in Section 2.5.3). The semantical interpretation of a sequent $\Gamma \vdash_\Delta E = F$, denoted $\Gamma \models_\Delta E = F$, is as follows: if $\alpha \sim \beta$ for all $(\alpha = \beta) \in \Gamma$, then $E \sim F$. As Δ will remain fixed throughout, we shall omit its subscripted appearance, thus writing $\vdash$ for $\vdash_\Delta$ and $\models$ for $\models_\Delta$. Also we shall omit empty assumption sets, thus writing $\vdash E = F$ and $\models E = F$ for $\emptyset \vdash E = F$ and $\emptyset \models E = F$ respectively. Notice that the relationship $\models E = F$ reduces to $E \sim F$.

The axioms and inference rules are presented in Table 6–2. We have standard inference rules for equivalence (E1–E3) as well as congruence (C1–C3). We also have standard axioms for summation (S1–S4) together with standard axioms for parallelism (P1–P3); notably we have associativity and commutativity for parallel composition. Finally, we have two rules characteristic for this equational theory: R1 is an assumption introduction rule underpinning the rôle of the assumption list Γ; and R2 is an assumption elimination rule, and represents a form of fixed point induction. The special form of R2 has been dictated by the rule Rec of the tableau system presented in Table 6–1. Notice that we do not have an explicit expansion law, as it is incorporated in the assumption elimination rule R2.

Equivalence

E1 $\Gamma \vdash E = E$

E2 $\dfrac{\Gamma \vdash F = E}{\Gamma \vdash E = F}$

E3 $\dfrac{\Gamma \vdash E = F \qquad \Gamma \vdash F = G}{\Gamma \vdash E = G}$

Congruence

C1 $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash \mu E = \mu F}$

C2 $\dfrac{\Gamma \vdash E_1 = F_1 \qquad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 + E_2 = F_1 + F_2}$

C3 $\dfrac{\Gamma \vdash E_1 = F_1 \qquad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 | E_2 = F_1 | F_2}$

Axioms

S1 $\Gamma \vdash E + (F + G) = (E + F) + G$

S2 $\Gamma \vdash E + F = F + E$

S3 $\Gamma \vdash E + \mathbf{0} = E$

S4 $\Gamma \vdash E + E = E$

P1 $\Gamma \vdash E|(F|G) = (E|F)|G$

P2 $\Gamma \vdash E|F = F|E$

P3 $\Gamma \vdash E|\mathbf{0} = E$

Assumption introduction and elimination

R1 $\Gamma, \alpha = \beta \vdash \alpha = \beta$

R2 $\dfrac{\Gamma, \alpha = \beta \vdash \mathrm{unf}(\alpha) = \mathrm{unf}(\beta)}{\Gamma \vdash \alpha = \beta}$

Table 6–2: A sequent-based equational theory for $\sim$ on BPP$_\tau$.

Definition 6.13 *A* proof *of $\Gamma \vdash E = F$, which we shall denote by this sequent, consists of a proof tree with root labelled $\Gamma \vdash E = F$, instances of the axioms* E1, S1–S4, P1–P3 *and* R1 *as leaves and where the children of a node are determined by an application of one of the inference rules* E2, E3, C1–C3 *or* R2.

Let us begin by showing soundness of the equational theory thus described.

Theorem 6.14 (Soundness) *If $\Gamma \vdash E = F$ then $\Gamma \models E = F$. In particular, if $\vdash E = F$ then $E \sim F$.*

Proof: Suppose that $\alpha \sim \beta$ for all $(\alpha = \beta) \in \Gamma$, but that $E \not\sim F$. We shall show that no proof exists for $\Gamma \vdash E = F$.

Suppose then that T is such a proof for $\Gamma \vdash E = F$. We can show that there must be a maximal path $\pi = \{\Gamma_i \vdash E_i = F_i\}$ starting from $\Gamma \vdash E = F$ and leading upwards through T such that $E_i \not\sim F_i$ for all i. This is clear by inspection of the axioms and inference rules. We can furthermore choose π so that the sequence $\left\{m_i \;:\; E_i \not\sim_{m_i} F_i \text{ and } E_i \sim_{m_i - 1} F_i\right\}$ is nonincreasing, and strictly decreasing through applications of C1.

The axiom which terminates π, say $\Gamma_l \vdash E_l = F_l$, must be an instance of R1, say $\Gamma', \alpha = \beta \vdash \alpha = \beta$, as otherwise we would have $E_l \sim F_l$.

Since we cannot have $(\alpha = \beta) \in \Gamma$, we must have somewhere in π an application of R2 to eliminate $\alpha = \beta$ from the assumption list. Also, some application of C1 must occur between the axiom and the application of R2, in order for there to be the required guarded expressions on the right of the turnstile at the application of R2. This fact follows from the property that in any sequent $\Gamma \vdash E = F$ of any proof tree, either both E and F are guarded or both E and F are unguarded. Hence the path π is as indicated in Figure 6–3. Now we know that $m_l < m_j$; however, this then implies that $\alpha \not\sim_{m_l} \beta$ and $\alpha \sim_{m_l} \beta$, which gives us our required contradiction. ■

To facilitate the completeness proof, we first introduce the following notation.

Definition 6.15 *For any node* $\mathbf{n}$ *of a tableau,* Recnodes($\mathbf{n}$) *denotes the set of labels of the nodes above* $\mathbf{n}$ *to which the rule* Rec *is applied. In particular,* Recnodes($\mathbf{r}$) $= \emptyset$ *where* $\mathbf{r}$ *is the root of the tableau.*

$$
\begin{array}{lcl}
\text{R1} & \Gamma',\ \alpha=\beta\ \vdash\ \alpha=\beta & (\text{level } l) \\
& \vdots & \\
\text{C1} & \dfrac{\Gamma',\ \alpha=\beta\ \vdash\ \alpha'=\beta'}{\Gamma',\ \alpha=\beta\ \vdash\ \mu\alpha'=\mu\beta'} & \begin{array}{l}(\text{level } k+1)\\(\text{level } k)\end{array} \\
& \vdots & \\
\text{R2} & \dfrac{\Gamma'',\ \alpha=\beta\ \vdash\ \mathrm{unf}(\alpha)=\mathrm{unf}(\beta)}{\Gamma''\ \vdash\ \alpha=\beta} & \begin{array}{l}(\text{level } j+1)\\(\text{level } j)\end{array} \\
& \vdots & \\
& \Gamma\ \vdash\ E=F & (\text{level } 1)
\end{array}
$$

Figure 6–3: The path constructed in the proof of Lemma 6.14.

We are now ready to prove our completeness theorem.

Theorem 6.16 (Completeness) *If $\alpha \sim \beta$ then $\vdash \alpha = \beta$.*

Proof: If $\alpha \sim \beta$, then there exists a finite successful tableau with root labelled $\alpha = \beta$. Let $T(\alpha = \beta)$ be such a tableau. We shall prove that for any node $\mathbf{n} : E = F$ of $T(\alpha = \beta)$ we have $\mathrm{Recnodes}(\mathbf{n}) \vdash E = F$. In particular, for the root $\mathbf{r} : \alpha = \beta$, this reduces to $\vdash \alpha = \beta$, so we shall have our result.

We prove $\mathrm{Recnodes}(\mathbf{n}) \vdash E = F$ by induction on the depth of the subtableau rooted at $\mathbf{n}$. As the tableau is built modulo $\equiv$, i.e. associativity and commutativity of parallel composition and sum, and by removing $\mathbf{0}$-components sitting in parallel or in sum we shall assume that the axioms S1–S3 and P1–P3 are used whenever required to accomplish the proof.

Firstly, if $\mathbf{n} : E = F$ is a terminal node then E and F must be identical expressions, so $\mathrm{Recnodes}(\mathbf{n}) \vdash E = F$ follows from E1.

Hence assume that $\mathbf{n} : E = F$ is not a terminal node. We proceed according to the tableau rule applied to $\mathbf{n}$.

(i) Prefix: Then $E = F$ is of the form $\mu\gamma = \mu\delta$. By the induction hypothesis we have $\mathrm{Recnodes}(\mathbf{n}') \vdash \gamma = \delta$ where $\mathbf{n}'$ is the son of $\mathbf{n}$. By inference rule

C1 we have $\text{Recnodes}(\mathbf{n}') \vdash \mu\gamma = \mu\delta$. As $\text{Recnodes}(\mathbf{n}') = \text{Recnodes}(\mathbf{n})$ the result follows.

(ii) Sum: Then $E = F$ is of the form $\sum_i \mu_i\alpha_i = \sum_j \nu_j\beta_j$. By the induction hypothesis we have for all sons $\mathbf{n}_k : \mu_{i_k}\alpha_{i_k} = \nu_{j_k}\beta_{j_k}$ of node $\mathbf{n}$ the relationship $\text{Recnodes}(\mathbf{n}_k) \vdash \mu_{i_k}\alpha_{i_k} = \nu_{j_k}\beta_{j_k}$. As $\text{Recnodes}(\mathbf{n}_k) = \text{Recnodes}(\mathbf{n})$ for all k we get $\text{Recnodes}(\mathbf{n}) \vdash \mu_{i_k}\alpha_{i_k} = \nu_{j_k}\beta_{j_k}$. By using rules E3, C2, S1, S2 and S4 we have $\text{Recnodes}(\mathbf{n}) \vdash E = F$ as required.

(iii) Rec: Then $E = F$ is of the form $\gamma = \delta$ and $\mathbf{n}' : \text{unf}(\gamma) = \text{unf}(\delta)$ where $\mathbf{n}'$ is the son of $\mathbf{n}$. By induction we have $\text{Recnodes}(\mathbf{n}') \vdash \text{unf}(\gamma) = \text{unf}(\delta)$. As $\text{Recnodes}(\mathbf{n}')$ is equal to $\text{Recnodes}(\mathbf{n})$ together with $\gamma = \delta$, by R2 we have $\text{Recnodes}(\mathbf{n}) \vdash E = F$ as required.

(iv) SubL: Say $E = F$ is of the form $\gamma|\zeta = \eta$ with the corresponding dominated node $\mathbf{n}'$ labelled $\gamma = \delta$ $(\gamma \succ \delta)$ and the son $\mathbf{n}''$ of $\mathbf{n}$ labelled $\delta|\zeta = \eta$. By the induction hypothesis we have $\text{Recnodes}(\mathbf{n}'') \vdash \delta|\zeta = \eta$. As $\text{Recnodes}(\mathbf{n}'')$ is equal to $\text{Recnodes}(\mathbf{n})$ it follows that $\text{Recnodes}(\mathbf{n}) \vdash \delta|\zeta = \eta$. Also, since $(\gamma = \delta) \in \text{Recnodes}(\mathbf{n})$, we have from the rules R1, C3, E1 and E3 that $\text{Recnodes}(\mathbf{n}) \vdash \gamma|\zeta = \eta$ as required. The arguments for the other possible applications of the Sub rules are identical.

This completes the proof. ■

Example 6.17 *Let Δ be the following family in full standard form of degree 2: $\{X_1 \stackrel{\text{def}}{=} a(X_1|X_4), X_2 \stackrel{\text{def}}{=} aX_3, X_3 \stackrel{\text{def}}{=} a(X_3|X_4) + \bar{a}X_2 + \tau(X_2|X_4), X_4 \stackrel{\text{def}}{=} \bar{a}\}$. (This is the family of processes from Example 6.2.) In Figure 6–4 we give a proof for $X_1 = X_2$. In the proof we let Γ_1 denote the set $\{X_1 = X_2\}$ while Γ_2 denote $\Gamma_1 \cup \{X_2|X_4 = X_3\}$.*

Notice that the equational theory for BPP_τ easily transforms to a sound and complete equational theory for BPP; all that is needed (besides replacing $|$ by $\|$) is a simplification of R2, i.e. of $\text{unf}(\alpha)$, since we do not have to cater for synchronisation possibilities in the setting of BPP.

$$
\text{R2}\;\dfrac{\text{C2}\;\dfrac{\overset{\text{E1}}{\Gamma_2 \vdash a(X_3|X_4) + \bar{a}X_2 = a(X_3|X_4) + \bar{a}X_2} \qquad \text{C1}\;\dfrac{\text{E2}\;\dfrac{\overset{\text{R1}}{\Gamma_2 \vdash X_2|X_4 = X_3}}{\Gamma_2 \vdash X_3 = X_2|X_4}}{\Gamma_2 \vdash \tau X_3 = \tau(X_2|X_4)}}{\Gamma_2 \vdash a(X_3|X_4) + \bar{a}X_2 + \tau X_3 = a(X_3|X_4) + \bar{a}X_2 + \tau(X_2|X_4)}}{\Gamma_1 \vdash X_2|X_4 = X_3}
$$

$$
\text{R2}\;\dfrac{\text{C1}\;\dfrac{\text{E2}\;\dfrac{\text{C3}\;\dfrac{\overset{\text{R1}}{\Gamma_1 \vdash X_1 = X_2} \quad \overset{\text{E1}}{\Gamma_1 \vdash X_4 = X_4}}{\Gamma_1 \vdash X_1|X_4 = X_2|X_4} \qquad \Gamma_1 \vdash X_2|X_4 = X_3}{\Gamma_1 \vdash X_1|X_4 = X_3}}{\Gamma_1 \vdash a(X_1|X_4) = aX_3}}{\vdash X_1 = X_2}
$$

Figure 6–4: A proof of $X_1 = X_2$.

The equational theory can be seen as a proper extension of Milner's equational theory of bisimulation equivalence on regular processes [Mil84], i.e. processes for which the transition graphs are finite. However, as our theory is sequent-based, it is different in style to Milner's elegant system which is equation-based and built around explicit fixed point constructors (see Section 2.5.3). It is our hope that Milner's theory for regular processes can be extended to BPP_τ by adding appropriate axioms for parallelism.

6.3 Decidability via Unique Decomposition

We end this chapter by showing decidability of bisimilarity on *normed* BPP_τ using a decision method very different from the tableau method. In obtaining the result we shall see that the property of unique decomposition becomes important, and the discovery of this is due to Hirshfeld. We note that all the arguments carried out here also are valid wrt normed BPP.

We assume a fixed finite family Δ of normed BPP_τ processes in full standard form of degree 2. Again we are interested in deciding whether or not $\alpha \sim \beta$ where $\alpha, \beta \in Var(\Delta)^{\otimes}$ and the method is inspired by Caucal [Cau90] and is also related to the techniques explored in Chapter 4, i.e. we aim to show that the largest bisimulation relation $\sim$ on $Var(\Delta)^{\otimes}$ is generated from a *finite* self-bisimulation. However, here we shall be able to show that the self-bisimulations considered are of a very simple nature: the congruences they generate are decidable and there are only finitely many of them.

In Section 2.5.4 we proved that a self-bisimulation constitute a *witness* for bisimilarity, i.e. any self-bisimulation R on $Var(\Delta)^{\otimes}$. satisfies $R^{\leftrightarrow^*} \subseteq \sim$. We shall restrict attention to those self-bisimulations which are *fundamental* in the following sense.

Definition 6.18 *A binary relation R over $Var(\Delta)^{\otimes}$ is* fundamental *iff*

- *$Dom(R) \subseteq Var(\Delta)$ and $Ran(R) \subseteq (Var(\Delta) \setminus Dom(R))^{\otimes}$,*
- *$\alpha R \beta$ and $\alpha R \gamma$ implies $\beta \equiv \gamma$, and*
- *$\alpha R \beta$ implies $\mathcal{N}(\alpha) = \mathcal{N}(\beta)$.*

From the first and second conditions in the above definition it is clear that a fundamental relation is finite and from the third condition it also follows that there can only be finitely many fundamental relations since there are only finitely many expressions with a given finite norm. Seen as a rewrite system, if R is fundamental then it is also canonical, i.e. confluent and well-founded. Hence its least congruence $R^{\leftrightarrow^*}$ is a *decidable* relation.

Lemma 6.19 *The set of fundamental, self-bisimilar relations for the labelled transition graph over the states $Var(\Delta)^{\otimes}$ can be effectively constructed.*

Proof: Given Δ there are only finitely many fundamental relations. We list these relations systematically so that we recognise when they have all been listed. Now for each relation R in this list, its least congruence $R^{\leftrightarrow^*}$ is decidable. Thus we can in finite time check whether or not R is a self-bisimulation and hence we can also in finite time extract from the list those fundamental relations which are self-bisimilar. ■

It is our aim to show that there exists a fundamental self-bisimulation R over $Var(\Delta)^{\otimes}$ such that $\sim = R^{\leftrightarrow^*}$. Certainly, for any fundamental relation R over $Var(\Delta)^{\otimes}$ we have $R^{\leftrightarrow^*} \subseteq \sim$ provided R is self-bisimilar (Proposition 2.41). But as we shall demonstrate in the next theorem, if R in addition is maximal wrt set-inclusion then also $\sim \subseteq R^{\leftrightarrow^*}$. This result is a direct consequence of the property of unique decomposition for normed processes as described in Chapter 5. To explain the intuition, suppose R is a fundamental relation over $Var(\Delta)^{\otimes}$ and let for $\alpha \in Var(\Delta)^{\otimes}$ the notation $\alpha \downarrow R$ stand for the expression obtained by *rewriting* α according to the rewrite rules of R. Now, if $\alpha \sim \beta$ then all we need to show is that $\alpha \downarrow R \equiv \beta \downarrow R$. But this follows from the unique decomposition result as expressions of $Ran(R)$ can be seen as consisting of *prime* variables since they cannot be expressed (up to bisimilarity) as the parallel composition of other variables when R is assumed to be maximal wrt set-inclusion.

Theorem 6.20 *If R is fundamental, self-bisimilar and maximal wrt set-inclusion then we have $\sim = R^{\leftrightarrow^*}$.*

Proof: We only have to prove that $\sim \subseteq R^{\leftrightarrow^*}$ as the other direction follows from the fact that R is self-bisimilar. Hence suppose $\alpha \sim \beta$. We aim to prove that $\alpha \downarrow R \equiv \beta \downarrow R$ from which it follows that $(\alpha, \beta) \in R^{\leftrightarrow^*}$ as required. The proof is to a large extent similar to the proof of Theorem 5.8 stating a unique decomposition result for normed BPP_τ processes. This is, as we have explained, due to the fact that, when R is maximal, the variables belonging to $Ran(R)$ have the property of primes. We shall prepare the ground here and will then refer to the proof of Theorem 5.8 for the finish.

As in the case of Theorem 5.8 the proof is by *reductio ad absurdum* and proceed by induction on norm assuming for all bisimilar pairs (γ, δ) with less norm that $\gamma \downarrow R \equiv \delta \downarrow R$. Suppose that

$$\begin{aligned} \alpha \downarrow R &\equiv Y_1^{k_1} | Y_2^{k_2} | \cdots | Y_n^{k_n}, \\ \beta \downarrow R &\equiv Y_1^{l_1} | Y_2^{l_2} | \cdots | Y_n^{l_n}. \end{aligned}$$

By contradiction, assume that $\alpha{\downarrow}R \not\equiv \beta{\downarrow}R$. Let m be chosen such that $k_m \neq l_m$, and that $\mathcal{N}(Y_j) > \mathcal{N}(Y_m)$ implies $k_j = l_j$. Thus Y_m is a maximal-sized (wrt norm) "prime" appearing in the expressions $\alpha{\downarrow}R$ and $\beta{\downarrow}R$ in which the exponents differ. Without loss of generality assume that $k_m > l_m$ (otherwise exchange the rôles of $\alpha{\downarrow}R$ and $\beta{\downarrow}R$). The proof now proceeds by cases on the possible forms of $\alpha{\downarrow}R$.

First of all, assume that $\alpha{\downarrow}R$ is a power of a single prime, i.e. $\alpha{\downarrow}R = Y_m^{k_m}$. If $k_m = 1$ then from $k_m > l_m$ we must have $l_m = 0$ and therefore

$$Y_m \quad \sim \quad Y_1^{l_1}|\cdots|Y_{m-1}^{l_{m-1}}|Y_{m+1}^{l_{m+1}}|\cdots|Y_n^{l_n}.$$

But this contradicts the maximality of R (or from a different viewpoint, the primeness of Y_m). Hence assume $k_m > 1$.

Now the proof is similar to the proof of Theorem 5.8, i.e. by finding an appropriate norm-reducing move $\alpha{\downarrow}R \xrightarrow{\mu} \gamma$ one shows that whenever $\beta{\downarrow}R$ matches this move by say $\beta{\downarrow}R \xrightarrow{\mu} \delta$ it is always the case that $\gamma{\downarrow}R \not\equiv \delta{\downarrow}R$. Thus by induction we must have $\gamma \not\sim \delta$ contradicting the fact that $\alpha \sim \beta$. So we stop here as otherwise the proof would be a virtual repetition of that of Theorem 5.8 which we refer to. ■

Using the above theorem we have a procedure for checking bisimilarity on normed BPP_τ: given Δ in full standard form, we construct the finite set of fundamental and self-bisimilar relations over $Var(\Delta)^{\otimes}$ as indicated in the proof of Lemma 6.19. We have for any $\alpha, \beta \in Var(\Delta)^{\otimes}$ that $\alpha \sim \beta$ iff for some relation, say R, among those constructed, it follows that $(\alpha, \beta) \in R^{\leftrightarrow^*}$.

The procedure for checking bisimilarity on normed BPP_τ thus obtained is very indirect compared to the tableau decision method of Section 6.1. However, the complexity of this procedure may be analysed. As the procedure involves an enumeration of all finitely many fundamental relations over $Var(\Delta)^{\otimes}$ together with a search for those being self-bisimilar it must be exponential in the number of variables involved. However we shall not go into further details since our task was to investigate into the possibilities of obtaining decidable theories and *not* to provide (if possible) complexity bounds. We leave it for others to perform a proper complexity analysis of the decision problem.

Chapter 7

Decidability of $\sim_d$ for Basic Parallel Processes

In this chapter we address the questions considered in the previous chapter but now in the context of distributed bisimilarity. Our analytical methods are the same. However, the problems seem to have lesser complexity compared to the previous chapter, i.e. our solutions are of a simpler nature.

First we show that distributed bisimulation equivalence is decidable on the two process classes of BPP and BPP_τ. Again we shall rely on the tableau method in the proof of decidability. However, our tableau system is much simpler compared to that of Chapter 6 due to a cancellation property admitted by distributed bisimilarity. For instance, we shall be able to provide an upper bound on every tableau, something we failed to do for the tableau system of Chapter 6.

From our tableau system we shall extract sequent-based equational theories for the two process classes of BPP and BPP_τ. The theories will to a large extent be similar to the equational theory we presented for bisimilarity in the previous section. The only major differences concern the inclusion of the congruence rule for the combinator of left merge as well as changing the assumption elimination rule which in the case of bisimilarity reflects the *expansion law* for bisimilarity. As this law is not valid in the case of distributed bisimilarity we have to reformulate the assumption elimination rule and shall do so using the left merge operator.

The tableau system offers a very natural procedure for checking distributed bisimilarity; a construction of a tableau for $E = F$ amounts to a constructive search for a distributed bisimulation relation containing (E, F). As in Chapter 6 we shall present an approach very different from the tableau system for deciding distributed bisimilarity on BPP_τ (as well as BPP) by appealing to the property of unique decomposition. However, as the property of unique decomposition is valid for distributed bisimilarity over the *whole* calculus we need not restrict attention to the subclass of normed processes as in the previous chapter.

We conclude the chapter by considering the question of obtaining an equational theory for distributed bisimilarity on a subclass of BPP_τ where general summation is replaced by *guarded summation* (see [Mil89]). We shall see that for such a language the equational theory becomes particularly simple. Our theory is similar in style to Milner's equational theory for bisimilarity on RCCS, i.e. it is equation-based, and the proof of completeness relies on the property of unique decomposition.

As in Chapter 6 we only present our results in the setting of BPP_τ. The corresponding results in the setting of BPP are easily obtained. The changes only regard the replacement of $|$ and $\lfloor$ by $\|$ and $\lfloor\!\lfloor$ respectively as well as the simplification of some of the constructions since wrt BPP we need not take into account processes that synchronise.

7.1 A Tableau Decision Method for $\sim_d$ on BPP_τ

We fix a finite family $\{X_i \stackrel{\mathrm{def}}{=} E_i : i = 1, 2 \ldots n\}$ of BPP_τ equations in guarded standard form of degree 3 (see Definition 2.28). We are interested in deciding for arbitrary $\alpha, \beta \in \mathit{Var}(\Delta)^\otimes$ whether or not $\alpha \sim_d \beta$ is the case, and we shall use a tableau system to decide this question.

A tableau for deciding $\alpha \sim_d \beta$ is a maximal proof tree with root labelled $\alpha = \beta$ and built using the tableau rules of Table 7–1. For the presentation of rule Rec we have introduced the notation $\mathrm{unfd}(\alpha)$ to stand for the *unfolding* of $\alpha \equiv Y_1|Y_2|\cdots|Y_m$ given by

Rec	$$\frac{\alpha = \beta}{\mathrm{unfd}(\alpha) = \mathrm{unfd}(\beta)}$$
Sum	$$\frac{\sum_{i=1}^n \mu_i\alpha_i \lfloor \alpha_i' = \sum_{j=1}^m \nu_j\beta_j \lfloor \beta_j'}{\{\mu_i\alpha_i \lfloor \alpha_i' = \nu_{f(i)}\beta_{f(i)} \lfloor \beta'_{f(i)}\}_{i=1}^n \qquad \{\nu_j\beta_j \lfloor \beta_j' = \mu_{g(j)}\alpha_{g(j)} \lfloor \alpha'_{g(j)}\}_{j=1}^m}$$ where $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$, $g : \{1, \ldots, m\} \to \{1, \ldots, n\}$
Prefix	$$\frac{(\mu\alpha)\lfloor \alpha' = (\mu\beta)\lfloor \beta'}{\alpha = \beta \qquad \alpha' = \beta'}$$

Table 7–1: Rules of the tableau system for $\sim_d$ on BPP_τ.

$$\begin{aligned}\mathrm{unfd}(\alpha) \quad = \quad & \sum \Big\{(\mu\alpha_i)\lfloor(Y_1|\cdots|\beta_i|\cdots|Y_m) : (\mu\alpha_i)\lfloor\beta_i \in Y_i\Big\} \\ & + \sum \Big\{\tau(\alpha_i|\alpha_j)\lfloor(Y_1|\cdots|\beta_i|\cdots|\beta_j|\cdots|Y_m) : \\ & \qquad (a\alpha_i)\lfloor\beta_i \in Y_i,\ (\bar{a}\alpha_j)\lfloor\beta_j \in Y_j,\ i < j\Big\}.\end{aligned}$$

As in Chapter 6 the notation $(\mu\alpha)\lfloor\beta \in Y$ is suppose to mean that $(\mu\alpha)\lfloor\beta$ is a summand of the defining expression for Y. Notice that $\mathrm{unfd}(\alpha)$ represents a substitution replacing each variable of α with its defining expression (as given by Δ) followed by using the *expansion law* for distributed bisimulation equivalence (see [Cas88]). Also notice the importance of the left merge combinator in formulating this principle; the expansion law as we know it for bisimilarity is not valid wrt distributed bisimilarity. However, in the presence of the left merge operator such a principle , i.e. a principle that relates parallelism and nondeterminism, may be formulated for distributed bisimilarity.

Compared to the tableau system of Chapter 6 we notice the absent of the two rules SubL and SubR which provided finiteness of tableaux in the case of bisimilarity. As we shall see later, finiteness of tableaux will now be provided by the Prefix rule which has been changed significantly compared to the Prefix rule of Chapter 6; in order to prove a node of the form $(\mu\alpha)\lfloor\alpha' = (\mu\beta)\lfloor\beta'$ we can *split* this into the two subgoals $\alpha = \beta$ and $\alpha' = \beta'$. Note that such a rule is valid for distributed bisimilarity but not for bisimilarity. We call $\alpha = \beta$ the *left consequent* of the Prefix rule and $\alpha' = \beta'$ for the *right consequent*.

We shall assume the same terminology concerning tableaux as in Chapter 6. Furthermore, we shall identify BPP_τ expressions in our tableaux up to the structural congruence $\equiv$, i.e. up to associativity, commutativity and $\mathbf{0}$-absorption of sum and parallel composition.

When building tableaux, the rules are only applied to nodes that are not *terminal*. A terminal node can either be *successful* or *unsuccessful*. A successful terminal node $\mathbf{n}$ is labelled either $\alpha = \alpha$, or $\alpha = \beta$ such that there is another node $\mathbf{n}'$ *above* in the tableau (and an application of Prefix in between) also labelled $\alpha = \beta$. An unsuccessful terminal node is one labelled $(\mu\alpha)\lfloor\alpha' = (\nu\beta)\lfloor\beta'$ such that $\mu \neq \nu$, or $(\mu\alpha)\lfloor\alpha' = \mathbf{0}$, or $\mathbf{0} = (\nu\beta)\lfloor\beta$. A tableau is successful iff all terminal nodes are successful; otherwise it is unsuccessful.

To facilitate the proof of the next lemma stating finiteness of every tableau, we introduce the measure $\mathcal{F}$ on expressions of $Var(\Delta)^\otimes$. We set $\mathcal{F}(\mathbf{0}) = 0$ and for $X \in Var(\Delta)$ we define

$$\mathcal{F}(X) \;=\; 1 + \max\left\{\mathcal{F}(\beta) : (\mu\alpha)\lfloor\beta \in X\right\}$$

where $\mathcal{F}(\alpha) = \sum_{X\in\alpha}\mathcal{F}(X)$. This measure is easily seen to be well-defined, i.e. finite for every $\alpha \in Var(\Delta)^\otimes$, due to guardedness of Δ.

Lemma 7.1 *Every tableau for $\alpha = \beta$ is finite.*

Proof: Let $T(\alpha = \beta)$ be a tableau with root labelled $\alpha = \beta$. It can only be infinite if it has an infinite path as every node has finite branching degree. Hence suppose π is an infinite path starting at the root $\mathbf{r} : \alpha = \beta$.

Notice that, π being infinite, it cannot contain terminal nodes. Now, if π passes through the left consequent of the rule PREFIX infinitely often then π will also contain a terminal node infinitely often. This follows from the fact that the expressions $\mu\gamma$ and $\mu\delta$ of labels of nodes matching the premise of PREFIX along the path will either be subexpressions of Δ or of the form $\tau(\zeta|\eta)$ where $a\zeta$ and $\bar{a}\eta$ are subexpressions of Δ for some $a \in Act$. Hence there are only *finitely many* possibilities for expressions of the form $\mu\gamma$ and $\mu\delta$ and therefore only finitely many possibilities for instances of the left consequent of the rule PREFIX.

Hence from some point along π we know that the path will never go via the left consequent of the PREFIX rule. But this gives us finiteness of π for suppose $\mathbf{n} : \gamma = \delta$ and $\mathbf{n}' : \gamma' = \delta'$ are two nodes along π with $\mathbf{n}$ appearing above $\mathbf{n}'$ and a single application of REC in between. Moreover, assume that $\mathbf{n}' : \gamma' = \delta'$ is the right consequent of an instance of PREFIX. Then we have $\mathcal{F}(\gamma) > \mathcal{F}(\gamma')$ and also $\mathcal{F}(\delta) > \mathcal{F}(\delta')$. Hence π must be finite and we therefore conclude that also $T(\alpha = \beta)$ is finite. ■

Lemma 7.2 (Completeness) *If $\alpha \sim_d \beta$ then there exists a successful tableau with $\alpha = \beta$ as root.*

Proof: Suppose $\alpha \sim_d \beta$. If we can construct a tableau $T(\alpha = \beta)$ for $\alpha = \beta$ with the property that the label of any node of $T(\alpha = \beta)$ relates distributed bisimilar processes then by Lemma 7.1 this construction must terminate and each terminal will be successful. Thus the tableau shall be successful.

Now, $T(\alpha = \beta)$ can be constructed once we verify that each rule of the tableau system is *forward sound* in the sense that if the antecedent relates distributed bisimilar processes then it is possible to find a set of consequents relating distributed bisimilar processes.

It is relatively easily to see that all rules of the tableau system are forward sound in the above sense; REC reflects the expansion law for distributed bisimilarity proven valid in [Cas88], and SUM is straightforward, so we only consider PREFIX. Assume $(\mu\gamma)\lfloor\gamma' \sim_d (\mu\delta)\lfloor\delta'$. We must show that $\gamma \sim_d \delta$ and also $\gamma' \sim_d \delta'$. By the operational semantics we have the move $(\mu\gamma)\lfloor\gamma' \xrightarrow{\mu} (\gamma, \gamma')$ which must be matched by $(\mu\delta)\lfloor\delta'$. But the only possibility for $(\mu\delta)\lfloor\delta'$ to match is by $(\mu\delta)\lfloor\delta' \xrightarrow{\mu} (\delta, \delta')$ and hence $\gamma \sim_d \delta$ and $\gamma' \sim_d \delta'$ as required. ∎

The proof of soundness relies on an alternative characterisation of distributed bisimulation equivalence, viz. as a sequence of approximations.

Definition 7.3 *We define the sequence $\{\sim_d^n\}_{n=0}^{\infty}$ of distributed bisimulation approximations inductively as follows: we have $E \sim_d^0 F$ for all processes E and F, and $E \sim_d^{n+1} F$ iff for all $\mu \in Act$*

- *$E \xrightarrow{\mu} (E_1, E_2)$ implies $F \xrightarrow{\mu} (F_1, F_2)$ such that $E_i \sim_d^n F_i$ for $i = 1, 2$, and*
- *$F \xrightarrow{\mu} (F_1, F_2)$ implies $E \xrightarrow{\mu} (E_1, E_2)$ such that $E_i \sim_d^n F_i$ for $i = 1, 2$.*

Lemma 7.4 *For image-finite, i.e. finitely branching, distributed labelled transition graphs we have*

$$\sim_d \; = \; \bigcap_{n=0}^{\infty} \sim_d^n .$$

Proof: Follows from standard arguments. See for instance [Mil89]. ∎

Lemma 7.5 (Soundness) *If there is a successful tableau for $\alpha = \beta$ then $\alpha \sim_d \beta$*

Proof: Suppose $T(\alpha = \beta)$ is a successful tableau for $\alpha = \beta$ but $\alpha \not\sim_d \beta$. As the distributed labelled transition graphs for α and β are image-finite there is a least $m \in \mathbb{N}$ such that $\alpha \not\sim_d^m \beta$ and $\alpha \sim_d^k \beta$ for all $k < m$.

Notice that all the tableau rules of Table 7–1 are backwards sound wrt $\sim_d^n$. Significantly, in the case of PREFIX we can strengthen this property since $\gamma \sim_d^n \delta$ and $\gamma' \sim_d^n \delta'$ implies $(\mu\gamma)\lfloor\gamma' \sim_d^{n+1} (\mu\delta)\lfloor\delta'$. Hence there is a path π from the root $\mathbf{r} : \alpha = \beta$ to some leaf in the tableau such that every node $\mathbf{n}_i : E_i = F_i$ along π satisfy $E_i \not\sim_d^{m_i} F_i$ for some m_i. For each i chose m_i such that it is the least with this property.

The leaf, say $\mathbf{n}_l$, of π cannot be labelled $\gamma = \gamma$ as $\sim_d$ is reflexive. Thus the leaf must be labelled $\gamma = \delta$ where there is a node $\mathbf{n}_j$ above it also labelled $\gamma = \delta$. Moreover, between these two nodes there must be an instance of rule PREFIX. But then we arrive at a contradiction because the leaf cannot satisfy $\gamma \not\sim_d^{m_l} \delta$ since the node $\mathbf{n}_j : \gamma = \delta$ above it must have the property $\gamma \sim_d^{m_l} \delta$. ∎

Since each of the approximations $\sim_d^n$ clearly is decidable we immediately have semi-decidability of distributed bisimulation inequivalence via the simple procedure that seeks the least n such that $\alpha \not\sim_d^n \beta$ (compare with bisimilarity, Section 4.1). Therefore, in order to show decidability of $\sim_d$ we just need to establish semi-decidability, and this is easily done using the tableau system: start listing tableaux for $\alpha = \beta$ and stop with answer "yes" once a successful tableau has been listed. By soundness and completeness of the tableau system we know that the answer is always correct. Moreover, when $\alpha \sim_d \beta$ is the case we will eventually list a successful tableau.

Theorem 7.6 *Distributed bisimilarity is decidable on* BPP_τ.

Certainly, the above theorem also holds for BPP. The arguments are more or less the same and the only major modification concerns the definition of REC; in the case of BPP we need not take into account the possibility of parallel expressions performing synchronisations. We state the result without further comments.

Corollary 7.7 *Distributed bisimilarity is decidable on* BPP.

Having established the decidability result we could let the case rest. However, we shall now provide an upper bound on every tableau for $\alpha = \beta$ thus guaranteeing that the procedure of listing tableaux eventually stops. Therefore, instead of relying on two semi-decision procedures, we can let our algorithm be based on a single procedure searching for a successful tableau (compare with the decidability of bisimilarity on BPP_τ as established in Chapter 6 using the tableau technique).

The complexity of the tableau system can be measured in terms of the maximal number of tableau rules applied along a path. For our analysis we introduce the *sum degree* of Δ, denoted $sd(\Delta)$, to be the largest number of summands in any equation of Δ.

Suppose $T(\alpha = \beta)$ is a tableau for $\alpha = \beta$. Let π be a path emanating from an instance of the left consequent of PREFIX. Moreover, assume that π never goes via the left consequent of an instance of PREFIX again. We want to provide an upper bound on the length of π. Suppose that $\mathbf{n} : \gamma = \delta$ and $\mathbf{n}' : \gamma' = \delta'$ are two nodes along π with $\mathbf{n}$ appearing above $\mathbf{n}'$ and such that a single application of REC occurs between these two nodes. We then have $\mathcal{F}(\gamma) > \mathcal{F}(\gamma')$ and also $\mathcal{F}(\delta) > \mathcal{F}(\delta')$. So in order to measure the length of π we need to establish the size of $\mathcal{F}(X)$ for any variable $X \in Var(\Delta)$. However, from the definition of $\mathcal{F}$ it is a routine matter to show that

$$\mathcal{F}(X) \;<\; \sum_{k=0}^{n-1} 3^k$$

where n is the number of variables of $Var(\Delta)$. The first node, say $\mathbf{n}_0 : \gamma_0 = \delta_0$, of π satisfies $size(\gamma_0), size(\delta_0) \leq 6$ as $\mathbf{n}_0$ is the left consequent of an instance of PREFIX. Hence, the number of occurrences of REC along π can at most be $6\sum_{k=0}^{n-1} 3^k$. Along π we have for each application of REC at most one application of each of SUM and PREFIX so that gives us a total of $18\sum_{k=0}^{n-1} 3^k$ with which π is bounded.

If $\mathbf{n} : \gamma = \delta$ is the left consequent of an instance of PREFIX then we know that $\mu\gamma$ is a subexpression of Δ for some $\mu \in Act$ or $\gamma \equiv \gamma_1|\gamma_2$ where $a\gamma_1$ and $\bar{a}\gamma_2$ are subexpressions of Δ. Likewise for δ. Hence any path in the tableau $T(\alpha = \beta)$ starting at the root $\mathbf{r} : \alpha = \beta$ can at most pass through the left consequent of PREFIX $\left(n \times sd(\Delta) + (n \times sd(\Delta))^2\right)^2$ times.

Starting at the root $\mathbf{r} : \alpha = \beta$ the only path that never goes via the left consequent of instances of PREFIX is bounded by $3\min\{size(\alpha), size(\beta)\}\sum_{k=0}^{n-1} 3^k$. Thus any path starting at $\mathbf{r} : \alpha = \beta$ is bounded by the constant

$$\left(3\min\{size(\alpha), size(\beta)\} + 18\left(n \times sd(\Delta) + (n \times sd(\Delta))^2\right)^2\right)\sum_{k=0}^{n-1} 3^k$$

and that completes our analysis.

7.2 An Equational Theory for $\sim_d$ on BPP$_\tau$

We now describe a sound and complete sequent-based equational theory for distributed bisimilarity on BPP$_\tau$ processes. The theory is very similar in style to the equational theory of bisimilarity presented in Chapter 6 as it is extracted from the tableau system. We restrict attention to processes in guarded standard form of degree 3. Let Δ be a family of such processes. The theory shall be parameterised by Δ and consists of axioms and inference rules that enable one to derive the root of successful tableaux.

As in Chapter 6 the equational theory is built around sequents of the form $\Gamma \vdash_\Delta E = F$ where Γ is a finite set of *assumptions* of the form $\alpha = \beta$, and E and F are BPP_τ expressions. The semantical interpretation of a sequent $\Gamma \vdash_\Delta E = F$, denoted $\Gamma \models_\Delta E = F$, is as follows: if $\alpha \sim_d \beta$ for all $(\alpha = \beta) \in \Gamma$, then $E \sim_d F$. As in Chapter 6 we shall omit the subscript Δ (of $\vdash$ and $\models$) as well as empty assumption sets. Notice that the relationship $\models E = F$ reduces to $E \sim_d F$.

The axioms and inference rules are presented in Table 7–2. The only major difference compared to the equational theory of bisimilarity presented in Chapter 6 involves the inclusion of the congruence law for left merge (C4). Notice that the assumption elimination rule R2 has been dictated by the rule REC of the tableau system presented in Table 7–1. The rule represents a form of fixed point induction and reflects the *expansion law* for distributed bisimilarity (see [Cas88]).

Castellani presented in [Cas88] a sound and complete equational theory for distributed bisimilarity on the recursion-free fragment of BPP_τ. The theory centred around a number of laws involving the left merge combinator and also the combinator of *communication merge*. This combinator was used to derive the expansion law which was not part of her equational theory. We can dispense with these laws altogether partly because we have included the expansion law (via the assumption elimination rule) and partly because we restrict attention to processes in standard form. We shall see in Appendix A, where we obtain proofs of standard form, that Castellani's axioms involving left merge and also communication merge become important and useful.

Definition 7.8 *A* proof *of $\Gamma \vdash E = F$, which we shall denote by this sequent, consists of a proof tree with root labelled $\Gamma \vdash E = F$, instances of the axioms* E1, S1–S4, P1–P3 *and* R1 *as leaves and where the children of a node are determined by an application of one of the inference rules* E2, E3, C1–C4 *or* R2.

Theorem 7.9 (Soundness) *If $\Gamma \vdash E = F$ then $\Gamma \models E = F$. In particular, if $\vdash E = F$ then $E \sim_d F$.*

Equivalence

E1 $\Gamma \vdash E = E$

E2 $\dfrac{\Gamma \vdash F = E}{\Gamma \vdash E = F}$

E3 $\dfrac{\Gamma \vdash E = F \quad \Gamma \vdash F = G}{\Gamma \vdash E = G}$

Congruence

C1 $\dfrac{\Gamma \vdash E = F}{\Gamma \vdash \mu E = \mu F}$

C2 $\dfrac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 + E_2 = F_1 + F_2}$

C3 $\dfrac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 | E_2 = F_1 | F_2}$

C4 $\dfrac{\Gamma \vdash E_1 = F_1 \quad \Gamma \vdash E_2 = F_2}{\Gamma \vdash E_1 \lfloor E_2 = F_1 \lfloor F_2}$

Axioms

S1 $\Gamma \vdash E + (F + G) = (E + F) + G$

S2 $\Gamma \vdash E + F = F + E$

S3 $\Gamma \vdash E + \mathbf{0} = E$

S4 $\Gamma \vdash E + E = E$

P1 $\Gamma \vdash E|(F|G) = (E|F)|G$

P2 $\Gamma \vdash E|F = F|E$

P3 $\Gamma \vdash E|\mathbf{0} = E$

Assumption introduction and elimination

R1 $\Gamma, \alpha = \beta \vdash \alpha = \beta$

R2 $\dfrac{\Gamma, \alpha = \beta \vdash \mathrm{unfd}(\alpha) = \mathrm{unfd}(\beta)}{\Gamma \vdash \alpha = \beta}$

Table 7–2: A sequent-based equational theory for $\sim_d$ on BPP_τ.

Proof: Suppose that $\alpha \sim_d \beta$ for all $(\alpha = \beta) \in \Gamma$, but that $E \not\sim_d F$. We shall show that no proof exists for $\Gamma \vdash E = F$.

Suppose then that T is such a proof for $\Gamma \vdash E = F$. Since each of the inference rules is forward sound wrt $\sim_d^n$ we can find a maximal path $\pi = \{\Gamma_i \vdash E_i = F_i\}$ starting from the root $\Gamma \vdash E = F$ and leading upwards through T such that $E_i \not\sim_d F_i$ for all i. Significantly, in the case of C1 we have from $E \sim_d^m F$ that $\mu E \sim_d^{m+1} \mu F$ and in the case of C4 we have from $E_1 \sim_d^{m+1} F_1$ and $E_2 \sim_d^m F_2$ that $E_1 \lfloor E_2 \sim_d^{m+1} F_1 \lfloor F_2$. But this implies that we can choose the path π so the the sequence of integers $\left\{m_i : E_i \not\sim_d^{m_i} F_i \text{ and } E_i \sim_d^{m_i - 1} F_i\right\}$ is nonincreasing, and *strictly* decreasing through applications of C1 and also through applications of C4 provided π goes via the right premise of C4.

The axiom which terminates π, say $\Gamma_l \vdash E_l = F_l$, must be an instance of R1, say $\Gamma', \alpha = \beta \vdash \alpha = \beta$, as otherwise we would have $E_l \sim_d F_l$. Since $(\alpha = \beta) \in \Gamma$ cannot be the case, we must have somewhere in π an application of R2 to eliminate $\alpha = \beta$ from the assumption list. If the path π does not go through the right premise of an application of C4 between the axiom $\Gamma_l \vdash E_l = F_l$ and the application of R2 then there must be an application of C1 between the axiom and the application of R2, in order for there to be the required unfolding of the expressions α and β on the right of the turnstile at the application of R2. This important fact follows from the property that in any sequent $\Gamma \vdash E = F$ of any proof tree, either both E and F are guarded or both E and F are unguarded. However, now we have our required contradiction since at the application of R2 we have $\alpha \sim_d^{m_l} \beta$ and at the leaf $\alpha \not\sim_d^{m_l} \beta$. ∎

We remind the reader of the notation Recnodes($\mathbf{n}$) (for any node $\mathbf{n}$ of a tableau) standing for the set of labels of the nodes above $\mathbf{n}$ to which REC is applied. In particular, Recnodes($\mathbf{r}$) $= \emptyset$ where $\mathbf{r}$ is the root of the tableau.

We are now ready to prove our completeness theorem.

Theorem 7.10 (Completeness) *If $\alpha \sim_d \beta$ then $\vdash \alpha = \beta$.*

Proof: If $\alpha \sim_d \beta$, then there exists a finite successful tableau with root labelled $\alpha = \beta$. Let $T(\alpha = \beta)$ be such a tableau. We shall prove that for any node $\mathbf{n} : E = F$ of $T(\alpha = \beta)$ we have Recnodes($\mathbf{n}$) $\vdash E = F$. In particular, for the root $\mathbf{r} : \alpha = \beta$, this reduces to $\vdash \alpha = \beta$, so we shall have our result.

We prove Recnodes($\mathbf{n}$) $\vdash E = F$ by induction on the depth of the subtableau rooted at $\mathbf{n}$. As the tableau is built modulo associativity and commutativity of sum and parallel composition, and by removing $\mathbf{0}$-components sitting in parallel or in sum we shall assume that the axioms S1–S3 and P1–P3 are used whenever required to accomplish the proof.

First of all, if $\mathbf{n} : E = F$ is a terminal node then either E and F must be identical expressions, so Recnodes($\mathbf{n}$) $\vdash E = F$ follows from E1, or $E = F$ is of the form $\gamma = \delta$ where there is a node above $\mathbf{n}$ in the tableau also labelled $\gamma = \delta$. But then $\gamma = \delta \in$ Recnodes($\mathbf{n}$) and therefore our result follows from R1.

Hence assume that $\mathbf{n} : E = F$ is not a terminal node. We proceed according to the tableau rule applied to $\mathbf{n}$.

(i) PREFIX: Then $E = F$ is of the form $(\mu\gamma)\lfloor\gamma' = (\mu\delta)\lfloor\delta'$. By the induction hypothesis we have Recnodes($\mathbf{n}_1$) $\vdash \gamma = \delta$ and also Recnodes($\mathbf{n}_2$) $\vdash \gamma' = \delta'$ where $\mathbf{n}_1$ and $\mathbf{n}_2$ are the two sons of $\mathbf{n}$. By inference rule C1 we have Recnodes($\mathbf{n}_1$) $\vdash \mu\gamma = \mu\delta$. As Recnodes($\mathbf{n}_1$) = Recnodes($\mathbf{n}_2$) = Recnodes($\mathbf{n}$) we conclude from C4 that Recnodes($\mathbf{n}$) $\vdash (\mu\gamma)\lfloor\gamma' = (\mu\delta)\lfloor\delta'$ as required.

(ii) SUM: Then $E = F$ must be of the form $\sum_i(\mu_i\alpha_i)\lfloor\alpha_i' = \sum_j(\nu_j\beta_j)\lfloor\beta_j'$. By the induction hypothesis we have for all sons $\mathbf{n}_k : (\mu_{i_k}\alpha_{i_k})\lfloor\alpha'_{i_k} = (\nu_{j_k}\beta_{j_k})\lfloor\beta'_{j_k}$ of node $\mathbf{n}$ that Recnodes($\mathbf{n}_k$) $\vdash (\mu_{i_k}\alpha_{i_k})\lfloor\alpha'_{i_k} = (\nu_{j_k}\beta_{j_k})\lfloor\beta'_{j_k}$. Since Recnodes($\mathbf{n}_k$) is equal to Recnodes($\mathbf{n}$) we have Recnodes($\mathbf{n}$) $\vdash (\mu_{i_k}\alpha_{i_k})\lfloor\alpha'_{i_k} = (\nu_{j_k}\beta_{j_k})\lfloor\beta'_{j_k}$ for all k. By using E3, C2, S1, S2 and S4 we have Recnodes($\mathbf{n}$) $\vdash E = F$ as required.

(iii) REC: Then $E = F$ is of the form $\gamma = \delta$ and $\mathbf{n}' : \mathrm{unfd}(\gamma) = \mathrm{unfd}(\delta)$ where $\mathbf{n}'$ is the son of $\mathbf{n}$. By induction we have Recnodes($\mathbf{n}'$) $\vdash \mathrm{unfd}(\gamma) = \mathrm{unfd}(\delta)$. As Recnodes($\mathbf{n}'$) is equal to Recnodes($\mathbf{n}$) together with $\gamma = \delta$, by R2 we have Recnodes($\mathbf{n}$) $\vdash E = F$ as required.

This completes the proof. ■

Our equational theory for BPP_τ easily transforms into a sound and complete theory for BPP; we only need to replace $\mid$ and $\lfloor$ by $\|$ and $\lfloor\!\lfloor$ respectively as well as redefining the unfolding operator unfd since wrt BPP we need not take into account the possibility of synchronisations between parallel components.

7.3 Decidability via Unique Decomposition

We now provide a decision procedure very different from the tableau decision method presented in Section 7.1. Our procedure is similar in style to the technique described in Section 6.3 and appeals to the property of unique decomposition. Unlike the case in Section 6.3 the method applies to the whole of our process calculus since the property of unique decomposition is valid for all (not only normed) processes wrt distributed bisimilarity.

We fix a finite family Δ of BPP_τ (or BPP as the arguments of this section apply to both classes) process equations in guarded standard form of degree 3. Again we are interested in deciding whether or not $\alpha \sim_d \beta$ where $\alpha, \beta \in \mathit{Var}(\Delta)^{\otimes}$. We do this by showing that the largest distributed bisimulation equivalence $\sim_d$ on the state space $\mathit{Var}(\Delta)^{\otimes}$ is generated from a finite *self-dbisimulation*, a notion we now introduce.

Definition 7.11 *A binary relation R on $\mathit{Var}(\Delta)^{\otimes}$ is called a* self-dbisimulation *provided $(\alpha, \beta) \in R$ implies for all $\mu \in \mathit{Act}$ that,*

- *if $\alpha \xrightarrow{\mu} (\alpha_1, \alpha_2)$ then $\beta \xrightarrow{\mu} (\beta_1, \beta_2)$ such that $(\alpha_i, \beta_i) \in R^{\leftrightarrow^*}$ for $i = 1, 2$,*
- *if $\beta \xrightarrow{\mu} (\beta_1, \beta_2)$ then $\alpha \xrightarrow{\mu} (\alpha_1, \alpha_2)$ such that $(\alpha_i, \beta_i) \in R^{\leftrightarrow^*}$ for $i = 1, 2$.*

The following two results are easily obtained. The first result states that a self-dbisimulation relation is a witness for distributed bisimilarity.

Lemma 7.12 *If R is a self-dbisimulation relation then $R^{\leftrightarrow^*} \subseteq \sim_d$.*

Corollary 7.13 *$\alpha \sim_d \beta$ if, and only if, there exists a self-dbisimulation R such that $(\alpha, \beta) \in R$.*

We aim to show that there exists a finite self-dbisimulation relation R on $Var(\Delta)^{\otimes}$ generating $\sim_d$, i.e. $\sim_d = R^{\leftrightarrow^*}$. Our proof of this result will proceed by induction on the initial degree of parallelism, $\mathcal{D}$, as defined in Chapter 5. We note that $\mathcal{D}(\alpha)$ is well-defined, i.e. finite for any $\alpha \in Var(\Delta)^{\otimes}$, as Δ is guarded. We restrict attention to self-dbisimulations which are *fundamental* in the following sense.

Definition 7.14 *A binary relation R over $Var(\Delta)^{\otimes}$ is* fundamental *iff*

- $Dom(R) \subseteq Var(\Delta)$ *and* $Ran(R) \subseteq (Var(\Delta) \setminus Dom(R))^{\otimes}$,
- $\alpha R \beta$ *and* $\alpha R \gamma$ *implies* $\beta \equiv \gamma$, *and*
- $\alpha R \beta$ *implies* $\mathcal{D}(\alpha) = \mathcal{D}(\beta)$.

Notice the only difference compared to the notion of a fundamental relation as defined in Chapter 6 (Definition 6.18): here $\alpha R \beta$ implies $\mathcal{D}(\alpha) = \mathcal{D}(\beta)$ whereas in Chapter 6 we required $\mathcal{N}(\alpha) = \mathcal{N}(\beta)$.

From the first and second conditions in the above definition it is clear that a fundamental relation is finite and from the third condition it also follows that there can only be finitely many fundamental relations since there are only finitely many expressions of $Var(\Delta)^{\otimes}$ with a given initial degree of parallelism. Seen as a rewrite system, if R is fundamental then it is also canonical, i.e. confluent and well-founded. Hence its least congruence $R^{\leftrightarrow^*}$ is a *decidable* relation.

It is our aim to show that there exists a fundamental self-dbisimulation relation on $Var(\Delta)^{\otimes}$ such that $\sim_d = R^{\leftrightarrow^*}$. Certainly, for any fundamental relation R over $Var(\Delta)^{\otimes}$ we have $R^{\leftrightarrow^*} \subseteq \sim_d$ provided R is self-dbisimilar (Lemma 7.12). However, as we shall demonstrate in the next theorem, if R in addition is maximal wrt set-inclusion then also $\sim_d \subseteq R^{\leftrightarrow^*}$.

Theorem 7.15 *If R is fundamental, self-dbisimilar and furthermore maximal wrt set-inclusion then we have $\sim_d = R^{\leftrightarrow^*}$.*

Proof: We only have to prove that $\sim_d \subseteq R^{\leftrightarrow^*}$ since the other direction follows from the fact that R is self-dbisimilar (Lemma 7.12). Hence suppose that $\alpha \sim_d \beta$.

We aim to show that $\alpha\downarrow R \equiv \beta\downarrow R$ from which it follows that $(\alpha,\beta) \in R^{\leftrightarrow^*}$ as required.

The proof of $\alpha\downarrow R \equiv \beta\downarrow R$ is to a large extent similar to the proof of Theorem 5.29 stating a unique decomposition result for distributed bisimilarity on BPP_τ processes. This is due to the fact that, when R is maximal, the variables belonging to $Ran(R)$ have the property of primes: they cannot be described as the parallel composition of other variables since this would contradict the maximality of R. Here we shall lift the proof of the ground but will refer to the proof of Theorem 5.29 for the completion.

Suppose that

$$\begin{aligned} \alpha\downarrow R &\equiv X_1|X_2| \cdots |X_k, \\ \beta\downarrow R &\equiv Y_1|\,Y_2| \cdots |Y_m. \end{aligned}$$

We have $\alpha\downarrow R \sim_d \beta\downarrow R$ and may by the cancellation law (Lemma 5.28) assume that $\alpha\downarrow R$ and $\beta\downarrow R$ have no variables in common (otherwise cancel out). As in the case of Theorem 5.29 the proof of $\alpha\downarrow R \equiv \beta\downarrow R$ is by *reductio ad absurdum* and proceeds by induction on the initial degree of parallelism assuming for all distributed bisimilar pairs (γ,δ) with less initial degree of parallelism that $\gamma\downarrow R \equiv \delta\downarrow R$.

We must have $k, m \geq 2$ as otherwise this would contradict the maximality of R. Assume without loss of generality that X_1 is the variable with least initial degree of parallelism, i.e. $\mathcal{D}(X_1) \leq \mathcal{D}(X_i), \mathcal{D}(Y_j)$ for all i, j. Suppose that $X_1 \xrightarrow{\mu} (\alpha_{11}, \alpha_{12})$ with $X_1|X_2|\cdots|X_k \xrightarrow{\mu} (\alpha_{11}, \alpha_{12}|X_2|\cdots|X_k)$. Firstly assume that Y_1 respond to this move by $Y_1 \xrightarrow{\mu} (\beta_{11}, \beta_{12})$ such that $\alpha_{11} \sim_d \beta_{11}$ and

$$\alpha_{12}|X_2|\cdots|X_k \;\sim_d\; \beta_{12}|Y_2|\cdots|Y_m.$$

Now, suppose that $\alpha_{12}\downarrow R \equiv Z_1|Z_2|\cdots|Z_{k_1}$ and $\beta_{12}\downarrow R \equiv U_1|U_2|\cdots|U_{k_2}$. We then have by induction on the initial degree of parallelism that

$$Z_1|Z_2|\cdots|Z_{k_1}|X_2|\cdots|X_k \;\equiv\; U_1|U_2|\cdots|U_{k_2}|Y_2|\cdots|Y_k.$$

Thus Y_2 must appear among $Z_1, Z_2 \ldots Z_{k_1}, X_2 \ldots X_k$. But Y_2 cannot be among the variables $Z_1, Z_2 \ldots Z_{k_1}$ since we have $\mathcal{D}(Z_i) < \mathcal{D}(X_1) \leq \mathcal{D}(Y_2)$ for all i. Hence it must be the case that Y_2 appears among $X_2, X_3 \dots X_k$ contradicting the assumption that $\alpha \downharpoonright R$ and $\beta \downharpoonright R$ have no variables in common.

Now, it could be the case that $\mu = \tau$ and $\beta \downharpoonright R$ respond to the move performed by X_1 via a synchronisation between two variables of $\beta \downharpoonright R$. We stop here since the arguments for this case are exactly as in the proof of Theorem 5.29 which we refer to. ■

Based on the above theorem we have a procedure for checking distributed bisimilarity on BPP_τ. Given Δ start listing possible fundamental relations over $Var(\Delta)^\otimes$ (which is possible since $\mathcal{D}$ is a computable measure); there are only finitely many and we list them systematically so that we can recognise when they have all been listed. Furthermore, for each of them it is decidable whether or not it is a self-dbisimulation relation. We have for any $\alpha, \beta \in Var(\Delta)^\otimes$ that $\alpha \sim_d \beta$ iff among these fundamental self-dbisimulation relations there exists one, say R, such that $(\alpha, \beta) \in R^{\leftrightarrow^*}$.

The procedure also allows for a complexity analysis. As in Section 6.3 the procedure must be exponential in the number of variables since it involves an enumeration of all fundamental relations on $Var(\Delta)^\otimes$ and a search for those being self-dbisimilar.

7.4 An Equational Theory for $\sim_d$ on BPP_g

In this section we deal with the question of finding a sound and complete equational theory for distributed bisimulation equivalence on a subclass of our process calculus BPP_τ where general summation is replaced by *guarded summation* [Mil89]. Our theory is similar in style to Milner's equational theory for bisimilarity on RCCS, a theory we illustrated in Section 2.5.3. Milner's technique centres around a few very elegant laws for recursion which is given by the explicit fixed point constructor *fix*. Thus for the purpose of this section we redefine recursion using explicit fixed point constructors. Our class of *expressions*, which we denote by BPP_g to emphasise the restriction to guarded summation, is given by the following abstract syntax equations.

$$
\begin{array}{rcll}
E & ::= & W & \text{(process variable, } W \in \mathit{Var}) \\
 & | & \sum_{i \in I} \mu_i E_i & \text{(guarded sum, } I \text{ an indexing set, } \mu_i \in \mathit{Act}) \\
 & | & E|E & \text{(parallel composition)} \\
 & | & \mathit{fix}\, W.E & \text{(recursion, } W \in \mathit{Var})
\end{array}
$$

We formally introduced the recursion operator *fix* in Section 2.5.3. In particular, we defined the notions of free and bound variables as well as closed and guarded expressions. We also introduced the notion of substitution; we used $E\{F/W\}$ to denote the expression obtained by substituting F for each free occurrence of W in E, renaming bound variables as necessary. Here we extend the notation for substitution, thus writing $E\{F_1 \ldots F_m/W_1 \ldots W_m\}$ or simply $E\{\tilde{F}/\tilde{W}\}$ for the result of simultaneously substituting F_i $(i = 1, 2 \ldots m)$ for each free occurrence of W_i in E, renaming bound variables as necessary. We define the class of $\mathrm{BPP_g}$ *processes* to be the class of closed and guarded $\mathrm{BPP_g}$ expressions.

The expression $\sum_{i \in I} \mu_i E_i$ stands for guarded summation (see [Mil89]). We always assume that the indexing set I is finite. In case the indexing set is empty we denote the sum by $\mathbf{0}$, i.e. the inactive process; and in case it contains just two elements we use the $+$ notation. By restricting attention to guarded summation we are prohibited from considering processes such as $E|F + G$ expressing a mix between summation and parallelism. However, as noted by Milner [Mil89], in applications such expressions rarely occur; one almost exclusively use summation in the restricted form of guarded summation. We shall see that only allowing guarded summation, our analysis of distributed bisimilarity can be simplified considerably: we shall obtain a very simple sound and complete equational theory *without* recourse to additional operators such as left merge and communication merge.

The operational semantics for processes of $\mathrm{BPP_g}$ is defined via a distributed labelled transition graph. For the combinator of parallel composition we have the same transition rules as in Chapter 2. For the guarded sum combinator and the fixed point operator we have

$$
\forall j \in I : \sum_{i \in I} \mu_i E_i \xrightarrow{\mu_j} (E_j, \mathbf{0}) \qquad\qquad \frac{E\{\mathit{fix}\, W.E/W\} \xrightarrow{\mu} (E_1, E_2)}{\mathit{fix}\, W.E \xrightarrow{\mu} (E_1, E_2)}
$$

Axioms

S1	$E+(F+G)$	$=$	$(E+F)+G$	P1	$E\|(F\|G)$	$=$	$(E\|F)\|G$
S2	$E+F$	$=$	$F+E$	P2	$E\|F$	$=$	$F\|E$
S3	$E+\mathbf{0}$	$=$	E	P3	$E\|\mathbf{0}$	$=$	E
S4	$E+E$	$=$	E				

Recursion

R1 $\mathit{fix}W.E = E\{\mathit{fix}W.E/W\}$

R2 $$\frac{E = F\{E/W\}}{E = \mathit{fix}W.F}\text{(W guarded in F)}$$

Table 7–3: Equational theory $\mathfrak{D}$ for $\sim_d$ on $\mathrm{BPP_g}$.

Distributed bisimilarity is defined over closed and guarded expressions as usual. However, we wish to extend the definition to deal with expressions which are not necessarily closed. Let E and F be guarded expressions with at most $\tilde{W}$ free variables. Then by definition $E \sim_d F$ iff for all $\mathrm{BPP_g}$ processes $\tilde{G}$ we have $E\{\tilde{G}/\tilde{W}\} \sim_d F\{\tilde{G}/\tilde{W}\}$.

The equational theory we now present shall be proven sound and complete for distributed bisimilarity on $\mathrm{BPP_g}$ processes. The theory is denoted $\mathfrak{D}$ and is presented in Table 7–3. We have omitted the usual rules for reflexivity, symmetry and transitivity as well as substitutivity of equality. By $\mathfrak{D} \vdash E = F$ we shall denote the fact that E and F are *provably equal* using the equational theory $\mathfrak{D}$.

The theory consists, in addition to the laws for recursion, of the abelian monoid laws for summation and parallel composition together with idempotency for summation. The two laws for recursion are taken from [Mil84] and they essentially express that a recursive process is equal to its unfolding and that guarded equations have unique solutions (up to $\sim_d$).

Soundness of $\mathfrak{D}$ is not hard. For each axiom it involves the construction of an appropriate distributed bisimulation relation. For R2 all that is needed is uniqueness of solutions to guarded equations (up to distributed bisimilarity). The proof of this can be based on arguments used by Milner [Mil89] in order to show the corresponding property for bisimilarity. We shall state the soundness of $\mathfrak{D}$ without further comments.

Theorem 7.16 (Soundness) *For guarded* $\mathrm{BPP_g}$ *expressions E and F, if we have $\mathfrak{D} \vdash E = F$ then $E \sim_d F$.*

For the proof of completeness we follow Milner [Mil84,Mil89a] closely. We intend to show three main theorems. One theorem will state that every guarded expression of $\mathrm{BPP_g}$ provably satisfies a certain set of equations. A second theorem states that if E and F are processes of $\mathrm{BPP_g}$ such that $E \sim_d F$ and E provably satisfies an equation set while F provably satisfies another equation set then both E and F provably satisfy a single equation set. The last theorem states that if two processes E and F of $\mathrm{BPP_g}$ satisfy the same equation set then they may be proven equal within $\mathfrak{D}$.

In order to achieve this we need to introduce extra definitions and notation. Let $\tilde{X} = \{X_1 \dots X_m\}$ and $\tilde{W} = \{W_1, W_2 \dots\}$ be disjoint sets of variables. Let $\tilde{H} = \{H_1 \dots H_m\}$ be $\mathrm{BPP_g}$ expressions with free variables in $\tilde{X} \cup \tilde{W}$ and consider the set S of *formal equations*

$$S : \tilde{X} = \tilde{H}.$$

We call $\tilde{X}$ the *formal variables* of S, and say that S has *free variables* in $\tilde{W}$. We call the first equation of S, i.e. $X_1 = H_1$, for the *leading equation* of S with X_1 being the *leading variable*. Finally, we say that a free variable W is *guarded* in S if it does not occur as part of the leading equation.

If each equation $X = H$ of a set of formal equations S takes the form

$$H \equiv (\prod_i A_i)|(\prod_k W_k),$$

where each A_i is a guarded sum of the form $\sum_{j\in I_i} \mu_{ij} X_{ij}$, then we say that S has *simple* form. Notice that each of the guarded sums A_i is *prime* (see Chapter 5 for the notion of prime processes) regardless of the substitution performed for the formal variables. This fact is important and shall be used in our later proofs. For a guarded sum A of the above form we shall write $A \xrightarrow{\mu} X$ if μX occurs in A.

We need to introduce one more notion before we can begin showing our results. We say that a BPP_g expression E *provably satisfies* a set of formal equations $S : \tilde{X} = \tilde{H}$ with free variables in $\tilde{W}$ if there are expressions $E_1 \ldots E_m$, where E_1 is E and $fv(\tilde{E}) \subseteq \tilde{W}$, such that

$$\forall i = 1 \ldots m : \mathfrak{D} \vdash E_i = H_i\{\tilde{E}/\tilde{X}\},$$

which we shall abbreviate as $\mathfrak{D} \vdash \tilde{E} = \tilde{H}\{\tilde{E}/\tilde{X}\}$.

We are now ready to prove the first of our main results.

Theorem 7.17 (Equational Characterisation) *Let E be a guarded BPP_g expression (not necessarily closed) and suppose $fv(E) = \tilde{W}$. Then E provably satisfies a set of formal equations $S : \tilde{X} = \tilde{H}$ in simple form with free variables in $\tilde{W}$. Moreover, if $W \in \tilde{W}$ is guarded in E then W is also guarded in S.*

Proof: We proceed by induction on the structure of E.

(i) Suppose E is W. Define S to be the single equation $\{X = W\}$ which is in simple form. Moreover, E provably satisfies S due to reflexivity.

(ii) Suppose E is $\sum_{i\in I} \mu_i E_i$. By induction let $S_j : \tilde{X}_j = \tilde{H}_j$ be a family of formal equation sets in simple form such that E_j provably satisfies S_j for each $j \in I$. Without loss of generality we can assume that S_i and S_j have distinct formal variables for all $i, j \in I$ with $i \neq j$. Now form S from $\bigcup_{j\in I} S_j$ together with a new leading equation

$$X \;=\; \textstyle\sum_{i\in I} \mu_i X_{i1}$$

where X_{i1} is assumed to be the leading variable of S_i for all $i \in I$. Certainly S is in simple form as required. Moreover, it is a routine matter to show that E provably satisfies S.

(iii) Suppose E is $F|G$. By induction let $S_1 : \tilde{X} = \tilde{H}$ and $S_2 : \tilde{Y} = \tilde{J}$ be sets of formal equations in simple form such that F provably satisfies S_1 while G provably satisfies S_2. Without loss of generality we can assume that the formal variables of S_1 and S_2 are distinct. We now form S from $S_1 \cup S_2$ by adding a new leading equation

$$X \quad = \quad H_1|J_1$$

where $X_1 = H_1$ and $Y_1 = J_1$ are the leading equations of S_1 and S_2 respectively. Certainly S is in simple form. Moreover, it is a routine matter to show that E provably satisfies S.

(iv) Suppose E is *fix*$W.F$. By induction let $T : \tilde{Y} = \tilde{J}$ be a set of formal equations in simple form such that F provably satisfies T. As W is guarded in F we also have W guarded in T, i.e. W does not occur as part of the leading equation for T. We form a new set of formal equations $S : \tilde{Y} = \tilde{H}$ from T by transforming each equation $Y = J$ of T into $Y = H$ such that H is $J\{J_1/W\}$ where $Y_1 = J_1$ is assumed to be the leading equation of T. Since W is guarded in T this transformation leaves the leading equation $Y_1 = J_1$ unchanged in S, i.e. $Y_1 = H_1$ is $Y_1 = J_1$. Certainly S is in simple form. We therefore only need to demonstrate that E provably satisfies S. Since F provably satisfies T we have expressions $\tilde{F} = \{F_1 \ldots F_m\}$, where F_1 is F, such that

$$\mathfrak{D} \vdash \tilde{F} = \tilde{J}\{\tilde{F}/\tilde{Y}\}.$$

From $\tilde{F}$ we define the expressions $\tilde{E} = \{E_1 \ldots E_m\}$ by setting E_1 equal to E, and E_i equal to $F_i\{E/W\}$ for $i = 2 \ldots m$. As $\mathfrak{D} \vdash F_i = J_i\{\tilde{F}/\tilde{Y}\}$ we have by substitutivity of equality $\mathfrak{D} \vdash F_i\{E/W\} = J_i\{\tilde{F}/\tilde{Y}\}\{E/W\}$. In case $i = 1$ we have

$$\mathfrak{D} \vdash F\{E/W\} = J_1\{\tilde{F}\{E/W\}/\tilde{Y}\}$$

since J_1 does not contain W. From axiom R1 together with transitivity and substitutivity of equality we now conclude that

$$\mathfrak{D} \vdash E = H_1\{\tilde{E}/\tilde{Y}\}. \tag{1}$$

For $i = 2 \dots m$ we have the relationship $\mathfrak{D} \vdash F_i\{E/W\} = J_i\{\tilde{F}/\tilde{Y}\}\{E/W\}$ which by definition, axiom R1 and the law of substitutivity of equality gives $\mathfrak{D} \vdash E_i = J_i\{E/W\}\{\tilde{E}/\tilde{Y}\}$. From this together with (1) we have by the law of substitutivity of equality and then a rearrange of substitution the relationship $\mathfrak{D} \vdash E_i = J_i\{J_1/W\}\{\tilde{E}/\tilde{Y}\}$, hence $\mathfrak{D} \vdash E_i = H_i\{\tilde{E}/\tilde{Y}\}$ as required.

In each of the above cases it is easy to see that if W is guarded in E then W is also guarded in S. This completes the proof. ■

Let us proceed by stating and prove the second main theorem.

Theorem 7.18 *Let E and F be processes of* $\mathrm{BPP_g}$*, i.e. E and F are closed and guarded expressions. If $E \sim_d F$ then there exists a set of formal equations S in simple form and without free variables such that both E and F provably satisfies S.*

Proof: From Theorem 7.17 we have sets of equations $S_1 : \tilde{X} = \tilde{H}$ and $S_2 : \tilde{Y} = \tilde{J}$ in simple form such that E provably satisfies S_1 while F provably satisfies S_2. Moreover, since E and F are closed expressions, it follows that S_1 and S_2 have no free variables.

As the processes E and F provably satisfy S_1 and S_2 respectively, we can find expressions $\tilde{E} = \{E_1 \dots E_m\}$ with E_1 equal to E, and expressions $\tilde{F} = \{F_1 \dots F_n\}$ with F_1 equal to F, such that

$$\mathfrak{D} \vdash \tilde{E} = \tilde{H}\{\tilde{E}/\tilde{X}\} \quad \text{and} \quad \mathfrak{D} \vdash \tilde{F} = \tilde{J}\{\tilde{F}/\tilde{Y}\}.$$

From $E \sim_d F$ and soundness of $\mathfrak{D}$ we must have for the leading equations $X_1 = H_1$ and $Y_1 = J_1$ of S_1 and S_2 respectively, that

$$H_1\{\tilde{E}/\tilde{X}\} \ \sim_d \ J_1\{\tilde{F}/\tilde{Y}\}.$$

As our equation sets are in simple form without free variables we may write $H_1 = \prod_{i=1}^{n_1} A_{1i}$ and also $J_1 = \prod_{j=1}^{m_1} B_{1j}$ for guarded sums A_{1i} and B_{1j}. Since each of the guarded sums is prime we conclude from the property of unique decomposition (Chapter 5, Theorem 5.29) that $n_1 = m_1$ and there exists a *permutation* σ on $\{1 \ldots n_1\}$ such that for all $i = 1 \ldots n_1$ we have

$$A_{1i}\{\tilde{E}/\tilde{X}\} \ \sim_d \ B_{1\sigma(i)}\{\tilde{F}/\tilde{Y}\}.$$

From this we obtain for each $i = 1 \ldots n_1$ the following relationship between the guarded sums A_{1i} and $B_{1\sigma(i)}$:

(i) whenever $A_{1i} \xrightarrow{\mu} X_j$ then for some k we have $B_{1\sigma(i)} \xrightarrow{\mu} Y_k$ with $E_j \sim_d F_k$,

(ii) whenever $B_{1\sigma(i)} \xrightarrow{\mu} Y_k$ then for some j we have $A_{1i} \xrightarrow{\mu} X_j$ with $E_j \sim_d F_k$.

We can generalise the above by a relation $R \subseteq \tilde{X} \times \tilde{Y}$ together with another relation R' between the guarded sums of S_1 and S_2 as follows. Whenever $(X, Y) \in R$ we have for $X = \prod_{i=1}^{n} A_i$ of S_1 and $Y = \prod_{j=1}^{m} B_j$ of S_2 that $n = m$ and there exists a permutation σ on $\{1 \ldots n\}$ such that $(A_i, B_{\sigma(i)}) \in R'$ for all $i = 1 \ldots n$. The relation R' may be explained as follows: if $(A, B) \in R'$ then

(i) if $A \xrightarrow{\mu} X_i$ then for some j we have $B \xrightarrow{\mu} Y_j$ such that $(X_i, Y_j) \in R$, and

(ii) If $B \xrightarrow{\mu} X_j$ then for some i we have $A \xrightarrow{\mu} X_i$ such that $(X_i, Y_j) \in R$.

From the two relations R and R' we now construct a new set of formal equations $S : \tilde{Z} = \tilde{K}$. We set $\tilde{Z} = \{Z_{ij} : (X_i, Y_j) \in R\}$, and $\tilde{K} = \{K_{ij} : (X_i, Y_j) \in R\}$ and define each expression K_{ij} in the following way. From $(X_i, Y_j) \in R$ we get according to the definition of R that

$$X_i = \prod_{k=1}^{n_i} A_{ik} \quad \text{and} \quad Y_j = \prod_{k=1}^{n_i} B_{jk}$$

together with a permutation σ on $\{1 \ldots n_i\}$ such that $(A_{ik}, B_{j\sigma(k)}) \in R'$ for all $k = 1 \ldots n_i$. We now define K_{ij} to be $\prod_{k=1}^{n_i} C_{ijk}$ where C_{ijk} is a guarded sum containing μZ_{fg} whenever $A_{ik} \xrightarrow{\mu} X_f$ and $B_{j\sigma(k)} \xrightarrow{\mu} Y_g$ with $(X_f, Y_g) \in R$. This completes the description of K_{ij}.

Before completing the proof we must demonstrate that both E and F provably satisfy S. We first consider E. Based on the expressions $\tilde{E} = \{E_1 \dots E_m\}$ we define new expressions $\tilde{G}$ with elements G_{ij} equal to E_i if Z_{ij} exists. We claim that

$$\mathfrak{D} \vdash G_{ij} = K_{ij}\{\tilde{G}/\tilde{Z}\}. \tag{2}$$

By definition we have $K_{ij} = \prod_{k=1}^{n_i} C_{ijk}$. Moreover, if C_{ijk} contains μZ_{fg} then A_{ik} contains μX_f and if A_{ik} contains μX_f then there exists g such that C_{ijk} contains μZ_{fg}. Hence we have

$$\forall k = 1 \dots n_i : \mathfrak{D} \vdash A_{ik}\{\tilde{E}/\tilde{X}\} = C_{ijk}\{\tilde{G}/\tilde{Z}\}$$

since $A_{ik}\{\tilde{E}/\tilde{X}\}$ and $C_{ijk}\{\tilde{G}/\tilde{Z}\}$ are identical up to repetition of summands. From the law of substitutivity of equality and a rearrange of substitution we then conclude that $\mathfrak{D} \vdash H_i\{\tilde{E}/\tilde{X}\} = K_{ij}\{\tilde{G}/\tilde{Z}\}$. Since $\mathfrak{D} \vdash E_i = H_i\{\tilde{E}/\tilde{X}\}$ and G_{ij} by definition is identical to E_i we have (2) as required. In a similar fashion it is shown that F provably satisfies S. Hence S is indeed the set of formal equations we were looking for. This completes the proof. ∎

We need one more theorem before having our completeness result; we shall show that every *guarded* set of formal equations (not necessarily in simple form) has a unique solution up to provably equality. The proof of this result is guaranteed by the two rules for recursion (R1 and R2) and the proof is exactly as in [Mil89]. However, we have chosen to give the proof here for sake of self-containment. First we need to define what we mean by a guarded set of formal equations.

Definition 7.19 *A set of formal equations $S : \tilde{X} = \tilde{H}$ (without free variables) is called* guarded *iff every expression H of $\tilde{H}$ is guarded wrt $\tilde{X}$.*

Theorem 7.20 (Unique Solution of Equations) *Suppose $S : \tilde{X} = \tilde{H}$ is a set of formal equations without free variables and assumed to be guarded. Then there exists a process E of* $\mathrm{BPP_g}$ *which provably satisfies S. Moreover, if F is another process of* $\mathrm{BPP_g}$ *which provably satisfies S then $\mathfrak{D} \vdash E = F$.*

Proof: By induction on the number m of equations in $S : \tilde{X} = \tilde{H}$ we find processes $\tilde{E}$ such that $\mathfrak{D} \vdash \tilde{E} = \tilde{H}\{\tilde{E}/\tilde{X}\}$, and show that if there are processes $\tilde{F}$ also satisfying $\mathfrak{D} \vdash \tilde{F} = \tilde{H}\{\tilde{F}/\tilde{X}\}$ then $\mathfrak{D} \vdash E = F$.

If $m = 1$ then S consists of a single equation $X_1 = H_1$ with X_1 guarded in H_1. Let E_1 be $fix X_1.H_1$ and by axiom R1 we have $\mathfrak{D} \vdash fix X_1.H_1 = H_1\{fix X_1.H_1/X_1\}$ as required. If F_1 is a process satisfying $\mathfrak{D} \vdash F_1 = H_1\{F_1/X_1\}$ then by R2, since X_1 is guarded in H_1, we have $\mathfrak{D} \vdash F_1 = H_1$.

Now assume that the result holds for m and let S consists of the $m+1$ equations $\tilde{X} = \tilde{H}, X_{m+1} = H_{m+1}$. We first intend to find processes $\tilde{E}$ together with E_{m+1} such that

$$\begin{aligned} \mathfrak{D} &\vdash \tilde{E} = \tilde{H}\{\tilde{E}/\tilde{X}, E_{m+1}/X_{m+1}\} \\ \mathfrak{D} &\vdash E_{m+1} = H_{m+1}\{\tilde{E}/\tilde{X}, E_{m+1}/X_{m+1}\}. \end{aligned} \tag{3}$$

Define the m expressions $\tilde{J} = \tilde{H}\{fix X_{m+1}.H_{m+1}/X_{m+1}\}$ and consider the guarded set of equations $T : \tilde{X} = \tilde{J}$ of size m. By induction we have m processes $\tilde{E}$ such that

$$\mathfrak{D} \vdash \tilde{E} = \tilde{J}\{\tilde{E}/\tilde{X}\}.$$

Now define the process E_{m+1} to be $(fix X_{m+1}.H_{m+1})\{\tilde{E}/\tilde{X}\}$ and we have the first equation of (3) from the above equation. The second equation of (3) follows from axiom R1.

In order to complete the proof assume we have processes $\tilde{F}, F_{m+1}$ such that

$$\begin{aligned} \mathfrak{D} &\vdash \tilde{F} = \tilde{H}\{\tilde{F}/\tilde{X}, F_{m+1}/X_{m+1}\} \\ \mathfrak{D} &\vdash F_{m+1} = H_{m+1}\{\tilde{F}/\tilde{X}, F_{m+1}/X_{m+1}\}. \end{aligned} \tag{4}$$

From the second equation of (4) we get $\mathfrak{D} \vdash F_{m+1} = (H_{m+1}\{\tilde{F}/\tilde{X}\})\{F_{m+1}/X_{m+1}\}$. Since S is guarded we have by R2 that $\mathfrak{D} \vdash F_{m+1} = fix X_{m+1}.(H_{m+1}\{\tilde{F}/\tilde{X}\})$ which may be rewritten as $\mathfrak{D} \vdash F_{m+1} = (fix X_{m+1}.H_{m+1})\{\tilde{F}/\tilde{X}\}$. This allows us to deduce by the law of substitutivity of equality followed by a reordering of substitutions that

$$\mathfrak{D} \vdash \tilde{F} = \tilde{H}\{fix X_{m+1}.H_{m+1}/X_{m+1}\}\{\tilde{F}/\tilde{X}\}.$$

By induction we now conclude that $\mathfrak{D} \vdash \tilde{E} = \tilde{F}$. Also by the definition of E_{m+1} and the fact that $\mathfrak{D} \vdash F_{m+1} = (\mathit{fix} X_{m+1}.H_{m+1})\{\tilde{F}/\tilde{X}\}$ we conclude by the law of substitutivity of equality and transitivity that $\mathfrak{D} \vdash E_{m+1} = F_{m+1}$. This completes the proof. ■

We now have the result of completeness as a rather simple consequence of our theorems.

Theorem 7.21 (Completeness) *Let E and F be processes of* $\mathrm{BPP_g}$. *If $E \sim_d F$ then $\mathfrak{D} \vdash E = F$.*

Proof: From Theorem 7.18 we can find a set of formal equations S in simple form such that E and F provably satisfy S. Moreover, S does not have any free variables as E and F are processes. Clearly, S is guarded since it is in simple form. We therefore have from Theorem 7.20 that $\mathfrak{D} \vdash E = F$ as required. ■

It is fair to say that distributed bisimilarity is a very strong (maybe the strongest) behavioural equivalence one could reasonably adopt on a language like $\mathrm{BPP_g}$; certainly it would seem odd if any of the axioms of $\mathfrak{D}$ were not valid. Notice that on $\mathrm{BPP_g}$ we need no expansion law to relate parallelism with nondeterminism. In the case of $\mathrm{BPP_\tau}$ we needed such a law and were only able to expressed it via the left merge operator. We can therefore conclude from the results of this section that the complicated relationship between parallelism and nondeterminism which for instance gives rise to infinitely many independent absorption laws (see [Mol89]) solely is due to the possibility of writing expressions like $E|F + G$; in case of guarded summation we only have one kind of absorption, viz. that expression by idempotency of sum.

One final remark before we end this chapter. Castellani studied an equivalence denoted $\sim_{abs}$ in [Cas88] which was characterised exactly by the laws of $\mathfrak{D}$ (without recursion). In trying to give an operational counterpart for $\sim_{abs}$ she came up with $\sim_d$, however soon realised that it was not the right one. We can conclude from the result of this section that restricting attention to a language like $\mathrm{BPP_g}$ with guarded summation, $\sim_d$ is indeed an operational counterpart for $\sim_{abs}$.

Chapter 8

Undecidability of $\sim$ for Labelled Petri Nets

As demonstrated in Chapter 3, labelled Petri nets constitute a natural extension of BPP and BPP_τ regarding expressiveness, i.e. the class of labelled transition graphs generated by BPP and BPP_τ processes is properly included in the class of transition graphs generated by labelled Petri nets (up to bisimilarity).

For some time we were trying to settle the question of deciding bisimilarity between labelled Petri nets (either in the positive or in the negative). Our starting point was the insight gained from the study of the process classes BPP and BPP_τ. However, during a recent visit to University of Edinburgh, Jančar became interested in the problem. Subsequently, he came up with a very delicate construction reducing the Halting problem for two-counter machines to a question of deciding bisimilarity between labelled Petri nets and thus answering the question in the negative (see [Jan93]).

In this chapter we begin by presenting (a modified and simplified version of) Jančar's construction for undecidability of bisimilarity on labelled Petri nets. Next we consider the question of *decomposing* Petri nets as a method of obtaining process representations of labelled Petri nets (up to bisimilarity). The process calculus we shall use for this purpose is denoted SPP (Synchronised Parallel Processes) and consists of BPP extended with a general synchronisation mechanism. Since SPP has the expressive power of labelled Petri nets we must conclude that bisimilarity is undecidable on this class of processes.

8.1 A Reduction from the Halting Problem

We formally introduced the model of labelled Petri nets in Section 2.2.2 and refer the reader to this section for the notation we use in this chapter.

Suppose $NS = (N, l, M_0)$ is a labelled Petri net system. We say that a place p of the underlying net N is *bounded* if there exists a constant $k_p \in I\!N$ such that for all reachable markings M from M_0 we have $M(p) \leq k_p$; otherwise the place is called *unbounded*. We shall call a net system bounded in case all places are bounded. Clearly, for such a net system the corresponding case graph has *finitely* many states. Consequently, bisimilarity is decidable on bounded net systems. Furthermore, bisimilarity between net systems generated from BPP or BPP_τ processes (which includes net systems not necessarily being bounded) is also decidable. An obvious question is whether we can extend these decidability results further. In this section we show that regarding the full model PN such a question has a negative answer. The proof of this result was first obtained by Jančar. We shall give a modified version of Jančar's construction which will suit our later purposes.

In order to show the undecidability result we shall reduce the Halting problem for two-counter machines to a question of bisimilarity between labelled Petri nets (see Section 2.3 for a formal introduction to two-counter machines). The construction we shall perform is to some extent similar to the construction of Section 3.3.2 reducing the Halting problem for two-counter machines to a question of terminal language equivalence between deterministic labelled Petri nets. However, our nets must now be nondeterministic since bisimilarity is decidable on deterministic net systems (see Section 3.3.2).

To any two-counter machine K (with counters `c1` and `c2`) we associate a net N_K and labelling map l_K described as follows. First of all, N_K has (unbounded) places p_{c1} and p_{c2} corresponding to the counters `c1` and `c2`. Furthermore, to each statement $\mathtt{s}_i\!:\!\mathtt{Com}_i$ of K we introduce a place p_i; a token in place p_i shall indicate that the *control* is at statement $\mathtt{s}_i$, i.e. $\mathtt{Com}_i$ is the next command about to be simulated. Moreover, if $\mathtt{Com}_i$ is of type II we include two extra places p_{i1} and p_{i2}. Finally, N_K includes two distinguished places p and p'.

For each type I statement $\mathtt{s}_i$`:cj:=cj+1; goto `$\mathtt{s}_k$ we introduce a transition t_i whose connection to the places are as indicated in Figure 8–1(A). We assume that each such transition is labelled with some $a_i \in Act$. For each type II statement $\mathtt{s}_i$`:if cj=0 then goto `$\mathtt{s}_{k1}$` else cj:=cj-1; goto `$\mathtt{s}_{k2}$ we introduce six transitions $t_i^{nz}, t_i^z, t_i^{zc}, t_{i1}, t_{i2}$ and t_{i3} whose connection to the places are as indicated in Figure 8–1(B). Transition t_i^{nz} is labelled with $b_i \in Act$ while t_i^z and t_i^{zc} both are labelled with $c_i \in Act$ and all of t_{i1}, t_{i2} and t_{i3} are labelled with $d_i \in Act$. Finally, for the halting statement $\mathtt{s}_m$`:halt` we introduce a single transition t_m with label $a_m \in Act$ and connected to the places as indicated in Figure 8–1(C).

A marking for the net N_K will in general have a single token (the *control token*) in p_i indicating that the next statement about to be simulated is $\mathtt{s}_i$`:`$\mathtt{Com}_i$, and also a number of tokens in p_{c1} and p_{c2} corresponding to the values for the counters `c1` and `c2` respectively.

A firing of transition t_i of Figure 8–1(A) correctly simulates the corresponding type I statement by adding a token to the place representing the counter in question while removing the control token from place p_i to p_k. Notice that the net of Figure 8–1(A) is identical to the net for type I statements we constructed in Section 2.3.

A firing of t_i^{nz} of Figure 8–1(B) correctly simulates the non-zero branch of the corresponding type II statement by removing a token from the place representing the counter in question while transferring the control token from place p_i to p_{k2}. For the simulation of the zero branch of a type II statement there are two possibilities; either t_i^{zc} or t_i^z may fire (notice that these two transitions have the same label). Let us first consider a firing of t_i^z. When such a transition fires the control token is transferred to an intermediate place p_{i2} from which only transition t_{i3} can fire. When t_{i3} fires it moves the control token to place p_{k1}, i.e. the next statement to be simulated is $\mathtt{s}_{k1}$. The intermediate place p_{i2} is only included in order to match (wrt bisimilarity) the intermediate place p_{i1} following transition

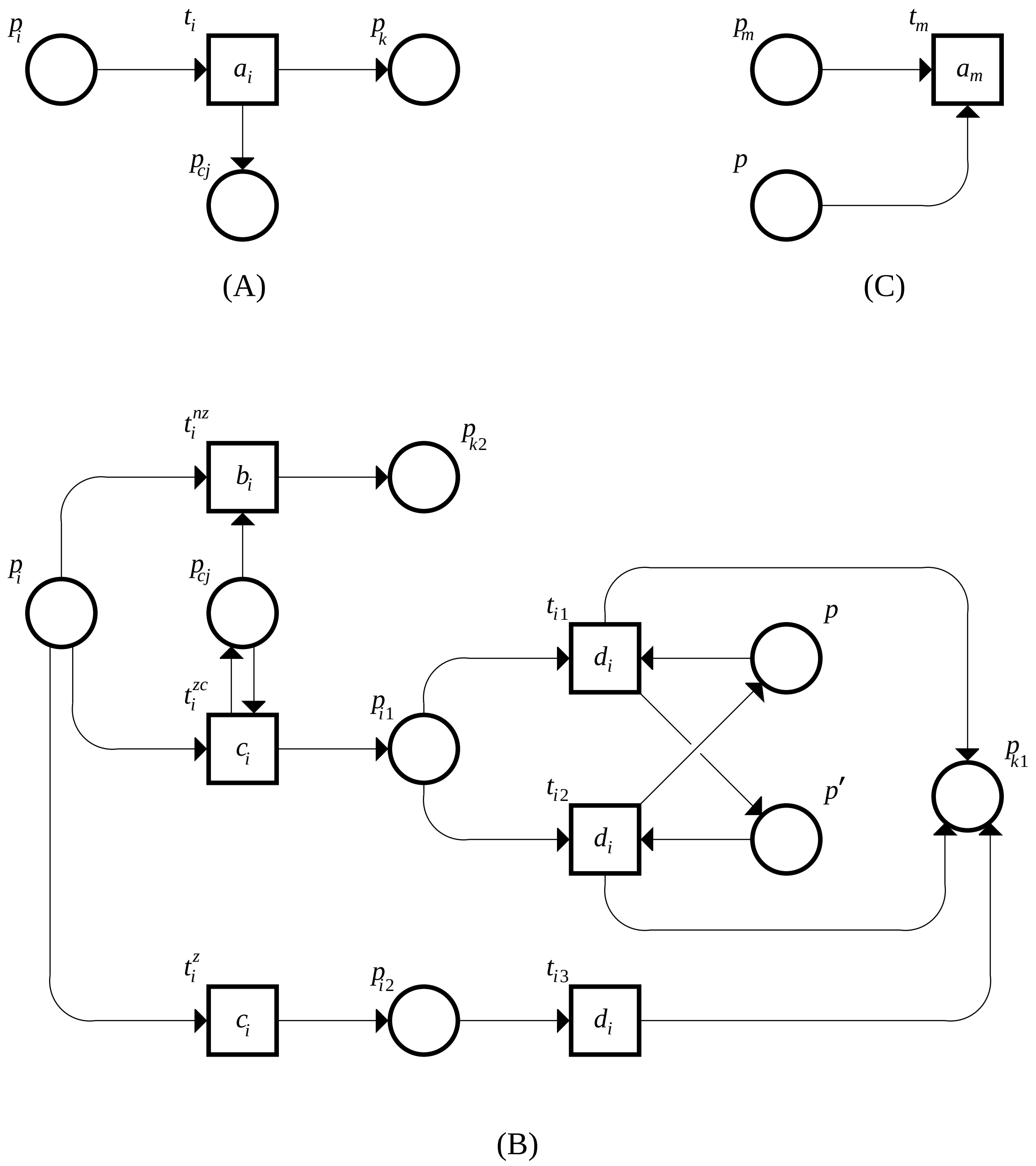

Figure 8–1: The net N_K with labelling map l_K.

t_i^{zc}. Now, transition t_i^z may fire even though the place representing the counter in question contains tokens. Whenever such a situation occurs then the two-counter machine K has been simulated wrongly. In these circumstances we shall say that the net system has *cheated* on the zero branch of a type II command (see also the discussion in Section 3.3.2).

Next let us consider a firing of t_i^{zc} which also is supposed to simulate the zero branch of the corresponding type II statement. Whenever t_i^{zc} does so, it *must* have been a simulation that cheated since the transition is only enabled provided the place representing the corresponding counter is not empty. When t_i^{zc} fires it transfers the control token to an intermediate place p_{i1} from which either t_{i1} or t_{i2} is enabled (but not both due to the fact that any reachable marking will have a single token in p or in p' but not in both). When t_{i1} (or t_{i2}) fires, the effect will be a swapping of the token from p to p' (or from p' to p) while the control token is transferred to place p_{k1}, i.e. the next statement about to be simulated is $\mathtt{s}_{k1}$. In the proof of the following theorem we shall see the important rôle played by this swap construction.

Finally, the subnet of Figure 8–1(C) records the situation where the last (`halt`) statement has been encountered; when transition t_m fires then the control token must be at place p_m. Since p is an input place to transition t_m we shall also be able to detect whether or not a swap has occurred, i.e. whether or not a cheating simulation has taken place.

Theorem 8.1 (Jančar) *Bisimulation equivalence is undecidable on* PN.

Proof: Let K be a two-counter machine using counters `c1` and `c2`. Suppose the counters have been initialised to contain the values $n1$ and $n2$ of $\mathbb{N}$ respectively. We construct two net systems $NS_1 = (N_K, l_K, M_{01})$ and $NS_2 = (N_K, l_K, M_{02})$ as follows. The net N_K and the labelling map l_K are as constructed prior to this theorem. The initial marking M_{01} puts a token (the control token) in p_1 indicating that the first command about to be simulated is $\mathtt{Com}_1$, tokens in p_{c1} and p_{c2} corresponding to the initial values $n1$ and $n2$, and finally puts a token in p while all other places are empty. The initial marking M_{02} is identical to M_{01} except that place p now is empty while p' has a single token. Thus the two net systems NS_1 and NS_2 are identical except for the initial tokens on the places p and p'.

Assume that K halts on the initial values $n1$ and $n2$. We want to show that $NS_1 \sim NS_2$ leads to a contradiction. Hence suppose that $NS_1 \sim NS_2$. Consider the firing sequence $M_{01}[u{>}M_1$ of NS_1 with $M_1(p_m) = 1$ such that u is a *correct* simulation of the machine K, i.e. none of the transitions t_i^{zc} are part of u and whenever t_i^z has fired along u then the place p_{cj} representing the corresponding counter is empty. Hence, for the marking M_1 we also have $M_1(p) = 1$ and therefore we can extend the firing sequence u to $M_{01}[u{>}M_1[t_m{>}M_1'$. Since $NS_1 \sim NS_2$ we may find a firing sequence v of NS_2 with $l_K(ut_m) = l_K(v)$. For such a sequence we must have $v = wt_m$ for some w and we may write $M_{02}[w{>}M_2[t_m{>}M_2'$. However, for t_m to be enabled at M_2 there must have been somewhere along w an occurrence of t_i^{zc} (in order to activate the swap construction that moves the token from p' to p) which is impossible since the place p_{cj} representing the corresponding counter has been empty whenever t_i^z occurred along u (and also w). We now have our required contradiction and must therefore conclude that $NS_1 \not\sim NS_2$.

Next assume that K does not halts on the initial values $n1$ and $n2$. We want to construct a bisimulation relation containing the pair (M_{01}, M_{02}) and thus showing that $NS_1 \sim NS_2$. To accomplish this we introduce the following notation. A marking M_p^q for arbitrary places p and q indicates that $M_p^q(p) = 1$ while $M_p^q(q) = 0$. When we write (M_p^q, M_q^p) then we understand the markings M_p^q and M_q^p to be *identical* except on the places p and q. We extend this notation to M_{pq}^{rs} defined in the obvious way.

Consider the relation B_1 on $R(M_{01}) \times R(M_{02})$ given by

$$B_1 \;\; = \;\; \Big\{(M_p^{p'}, M_{p'}^{p}) : M_p^{p'} \text{ and } M_{p'}^{p} \text{ have been reached without cheating}\Big\}.$$

Also consider the relation B_2 on $R(M_{01}) \times R(M_{02})$ given by

$$B_2 \;\; = \;\; \Big\{(M_{pp_{i2}}^{p'p_{i1}}, M_{p'p_{i1}}^{pp_{i2}})\Big\} \cup \Big\{(M_{pp_{i1}}^{p'p_{i2}}, M_{p'p_{i2}}^{pp_{i1}})\Big\}.$$

We shall now show that $B = B_1 \cup B_2 \cup \sim$ where $\sim$ is the largest bisimulation on $R(M_{01}) \times R(M_{02})$ is a bisimulation relation. Since $(M_{01}, M_{02}) \in B_1$ we shall have our result.

First consider a pair $(M_p^{p'}, M_{p'}^{p})$ of B_1. Suppose $M_p^{p'}[t{>}M'$ of NS_1 such that t does not give rise to a simulation that cheats. Since $M_p^{p'}$ has been reached without cheating t cannot be t_{i1} or t_{i2}. But then M' is of the form $\tilde{M}_p^{p'}$. Again since $M_p^{p'}$ has been reached without cheating t cannot be t_m. But then we have $M_{p'}^{p}[t{>}\tilde{M}_{p'}^{p}$ of NS_2 as well. By assumption t does not give rise to a simulation that cheats so we have $(\tilde{M}_p^{p'}, \tilde{M}_{p'}^{p}) \in B_1$ as required.

Suppose $M_p^{p'}[t_i^z{>}M_{pp_{i2}}^{p'p_{i1}}$ of NS_1 and that the firing of t_i^z gives rise to a simulation that cheats. For the net system NS_2 we then chose $M_{p'}^{p}[t_i^{zc}{>}M_{p'p_{i1}}^{pp_{i2}}$ where the transitions t_i^z and t_i^{zc} have the same label. Furthermore $(M_{pp_{i2}}^{p'p_{i1}}, M_{p'p_{i1}}^{pp_{i2}}) \in B_2$ as required. Finally, suppose $M_p^{p'}[t_i^{zc}{>}M_{pp_{i1}}^{p'p_{i2}}$ of NS_1. We then chose $M_{p'}^{p}[t_i^z{>}M_{p'p_{i2}}^{pp_{i1}}$ of NS_2 where t_i^z and t_i^{zc} have the same label. Furthermore, $(M_{pp_{i1}}^{p'p_{i2}}, M_{p'p_{i2}}^{pp_{i1}}) \in B_2$ as required.

Next consider a pair $(M_{pp_{i2}}^{p'p_{i1}}, M_{p'p_{i1}}^{pp_{i2}}) \in B_2$. The only transition of NS_1 enabled at $M_{pp_{i2}}^{p'p_{i1}}$ is t_{i3}, and the only transition of NS_2 enabled at $M_{p'p_{i1}}^{pp_{i2}}$ is t_{i2}. Since t_{i2} and t_{i3} have the same label and the firing of these two transitions will result in *identical* markings we get our result from reflexivity of $\sim$. Similar arguments apply to pairs $(M_{pp_{i1}}^{p'p_{i2}}, M_{p'p_{i2}}^{pp_{i1}}) \in B_2$.

By symmetry we now conclude that B is a bisimulation relation and therefore it follows that $NS_1 \sim NS_2$. Hence $NS_1 \not\sim NS_2$ if, and only if, the two-counter machine K halts on the initial values $n1$ and $n2$. We therefore conclude that bisimilarity is undecidable for PN. ∎

8.2 Decomposition of Labelled Petri Nets

A particular technique for the analysis of Petri nets involves *decomposition* of a net into subnets in order to study every component separately (see e.g. [Ber86, Hac74,Maz87]). Basically, two approaches have been considered; either a net is decomposed *place-disjointly* while transitions may be shared between the individual components, or vice versa, a net is decomposed *transition-disjointly* while places are shared. Generally, it is a good discipline to avoid a mix of these two kinds of decomposition since it is semantically very difficult to handle (see [Ber86] for a discussion on this subject).

We shall decompose net systems place-disjointly as a step towards obtaining a transformation from Petri nets into an appropriate extension of BPP. That is, we aim at decomposing any net system NS place-disjointly into simpler subnets $NS_1, NS_2 \ldots NS_k$ such that each of the individual components, seen as a labelled transition graph, is representable (up to bisimilarity) as a process. Subsequently, we shall *compose* each of the processes for the subnets thus obtaining a process representation for the labelled transition graph generated by the entire net system.

Relationships between the model of Petri nets and process algebras have been explored by other researchers; some examples are [Tau89,Nie87,BRS85,GM84, Gol88]. Often Petri nets serve as a semantical foundation for process algebras thereby obtaining so-called "true concurrency" interpretations of such algebras (see e.g. [Nie87,Gol88]). It seems that the relationship in the other direction has been less investigated. The only example we have found regards [BRS85] where Boudol, Roucairol and de Simone show that, as transition graphs, Petri nets may be expressed by processes of a variant of SCCS [Mil83]. However, we believe that the construction of [BRS85] is too primitive; essentially every place and every transition of a net become a process and therefore the structure or components of the net is not inherent in the process representation.

The process algebra we use for our transformation is based on BPP. However, we must extend the expressive power of BPP since PN is a proper extension of this calculus regarding expressiveness. We shall add a general synchronisation mechanism (somewhat similar to ACP synchronisation [BW90] and also related so the so-called synchronisation algebras of [Win84]) to BPP and thus call the resulting process algebra SPP (Synchronised Parallel Processes).

8.2.1 Synchronised Parallel Processes

For the definition of SPP we presuppose for each $n \geq 1$ a countably infinite set Λ_n of actions such that every two sets Λ_i and Λ_j are disjoint (for $i \neq j$). We also assume associated to each set Λ_n a *partial synchronisation function* $f_n : \Lambda_n^n \to \Lambda_1$

assumed to be commutative, i.e. $f_n(a_1, a_2 \dots a_n) = f_n(a_{i_1}, a_{i_2} \dots a_{i_n})$ for any permutation $a_{i_1}, a_{i_2} \dots a_{i_n}$ of $a_1, a_2 \dots a_n$. We shall assume that f_1 is the total identity function on Λ_1. We call actions of Λ_1 *observable* while actions of Λ_n (for $n > 1$) are called *unobservable*, and finally by Λ we denote the union of Λ_n for all $n \geq 1$.

An action of Λ_n (sometimes said to be of *type* Λ_n) is an n*-ary synchronisation action*. That is, n actions of Λ_n are required in order for a process to produce an observable action which is determined by the synchronisation function f_n. If for $a_1, a_2 \dots a_n \in \Lambda_n$ the application $f_n(a_1, a_2 \dots a_n)$ is not defined then we say that $a_1, a_2 \dots a_n$ do not synchronise. On the other hand, if $f_n(a_1, a_2 \dots a_n) = a$ then $a_1, a_2 \dots a_n$ do synchronise and the result of the synchronisation is the observable action $a \in \Lambda_1$. We let $a, b, c \dots$ range over all actions of Λ as the type of an action will be determined from the context in which it appears.

To minimise the introduction of new syntax we have chosen to let the syntax for SPP processes be based on that for BPP. Thus for instance the parallel operator is denoted $\|$. However, actions are now drawn from Λ. We point out that Λ_1 shall be the *only* observable actions from any process of SPP, i.e. transition graphs determined by SPP processes will be labelled only by actions of Λ_1.

Of course the operational semantics for SPP differs considerably from that of BPP since we now have to cater for all the unobservable synchronisation actions. Any finite family Δ of guarded SPP process equations determines a labelled transition graph where states are processes of Δ, actions are drawn from Λ_1 and the transition relations $\{\xrightarrow{a}\}_{a \in \Lambda_1}$ are given as the least relations satisfying the rules of Table 8–1. The definition of $\xrightarrow{a}$ for observable actions is obtained via an auxiliary transition relation $\overset{b}{\dashrightarrow}$ for unobservable actions. In Table 8–1 we use $\rightsquigarrow$ as a common appellation for the transition relations $\longrightarrow$ and $\dashrightarrow$.

Observe that SPP essentially is BPP together with all the unobservable actions. The ability to use such actions in order to *force* synchronisation between processes will provide the extra descriptive power to match (at least) labelled Petri net systems; in Section 8.2.2 we shall decompose and subsequently transform net systems into processes of SPP while preserving bisimilarity between the associated transition graphs.

$$aE \xrightarrow{a} E \;\text{ if } a \in \Lambda_1 \qquad \frac{E \overset{a}{\rightsquigarrow} E'}{E + F \overset{a}{\rightsquigarrow} E'} \qquad \frac{F \overset{a}{\rightsquigarrow} F'}{E + F \overset{a}{\rightsquigarrow} F'}$$

$$aE \overset{a}{\dashrightarrow} E \;\text{ if } a \notin \Lambda_1 \qquad \frac{E \overset{a}{\rightsquigarrow} E'}{E \| F \overset{a}{\rightsquigarrow} E' \| F} \qquad \frac{F \overset{a}{\rightsquigarrow} F'}{E \| F \overset{a}{\rightsquigarrow} E \| F'}$$

$$\frac{E \overset{a}{\rightsquigarrow} E'}{X \overset{a}{\rightsquigarrow} E'}(X \overset{\text{def}}{=} E \in \Delta) \qquad \frac{E_1 \overset{a_1}{\dashrightarrow} E_1' \quad \cdots \quad E_n \overset{a_n}{\dashrightarrow} E_n'}{E_1 \| \cdots \| E_n \xrightarrow{a} E_1' \| \cdots \| E_n'}\Big(f_n(a_1 \ldots a_n) = a\Big)$$

Table 8–1: Transition rules for SPP.

In Chapter 2 we remarked that bisimilarity is substitutive under parallelism (or equivalently, a congruence wrt parallelism) on the two process classes BPP and BPP_τ. The congruence property was further emphasised in Chapter 6 where the success of the tableau decision method relied on the two tableau rules SubL and SubR to be valid, which is the case when bisimilarity is a congruence wrt parallel composition. Moreover, in Section 6.1.1 we demonstrated that the largest bisimulation relation on $Var(\Delta)^{\otimes}$, where Δ is a finite family of process equations in full standard form, is finitely generated *whenever* bisimilarity is a congruence wrt the parallelism used in the description of Δ (whether full merge, CCS parallelism, the CSP related parallel operator $\|_A$ or otherwise). However, for our calculus SPP bisimilarity is no longer a congruence wrt parallel composition as illustrated by the following example.

Example 8.2 *Let a and b be actions of type Λ_2 and suppose $f_2(a, a) = c$ for some $c \in \Lambda_1$ while f_2 is undefined elsewhere. We then have $a \sim \mathbf{0} \sim b$ but $a \| a \not\sim b \| a$.*

8.2.2 The Decomposition

In order for our decomposition to proceed properly we shall assume for the rest of this chapter that all nets considered satisfy $F(p, t) \leq 1$ for all transitions t and all places p. We note that this is not a major restriction; all nets considered hitherto satisfy this property and in particular, bisimilarity is still undecidable on this restricted class of net systems (which we still denote by PN).

We need to introduce extra definitions and notions before being able formally to state the decomposition of net systems that we are interested in.

Definition 8.3 *Let $N = (S, T; F)$ and $N_1 = (S_1, T_1; F_1)$ be two nets. We say that N_1 is a* subnet *of N iff $S_1 \subseteq S$, $T_1 \subseteq T$ and $F_1 = F \cap \Big((S_1 \times T_1) \cup (T_1 \times S_1)\Big)$.*

The type of components that we intend to decompose a net system into are the so-called *synchronisation free* nets defined as follows.

Definition 8.4 *A net $N = (S, T; F)$ is called* synchronisation free *iff for every transition $t \in T$ we have $0 \leq \sum_{p \in S} {}^{\bullet}t(p) \leq 1$. A synchronisation free net system is any $NS = (N, l, M_0)$ such that N is a synchronisation free net.*

Note that for a family Δ of BPP equations in full standard form, the associated net system NS_Δ as constructed in Section 3.1 is synchronisation free in the above sense. As we shall see shortly, to any synchronisation free net system there also correspond a family of BPP equations (or equivalently, SPP equations without using unobservable actions) in full standard form.

Definition 8.5 *Let $N = (S, T; F)$ and $N_1 = (S_1, T_1; F_1)$ be two nets. We say that N_1 is a* synchronisation free component *(or simply an* SF *component) of N if N_1 is a subnet of N which is synchronisation free and furthermore satisfies $T_1 = \{t \in T : \exists p \in S_1 \text{ such that } t \in {}^{\bullet}p \cup p^{\bullet}\}$, i.e. T_1 is generated from S_1 via the flow relation F.*

We are now ready to define the notion of decomposition that we intend to exploit.

Definition 8.6 *If $N_1, N_2 \ldots N_k$ are place disjoint* SF *components of a net N such that*

$$N \;=\; N_1 \cup N_2 \cup \cdots \cup N_k,$$

where $\cup$ *refers to union of places, transitions and flow relations respectively, then we say that* N *is* SF decomposed *into* $N_1, N_2 \dots N_k$. *A net system* $NS = (N, l, M_0)$ *is* SF decomposed *into* $NS_1, NS_2 \dots NS_k$ *if* $NS_i = (N_i, l_i, M_{0i})$ *for* $i = 1, 2 \dots k$ *such that* $N_1, N_2 \dots N_k$ *is an* SF *decomposition of* N, l_i *is* l *restricted to* T_i *and finally* M_{0i} *is* M_0 *restricted to* S_i.

We aim, given an SF decomposition $NS_1, NS_2 \dots NS_k$ of a labelled Petri net system $NS = (N, l, M_0)$, to construct process expressions E_i of BPP satisfying for $i = 1 \dots k$ that $E_i \sim M_{0i}$ where M_{0i} is the initial state of the case graph for component NS_i. Subsequently we aim to transform each of E_i into appropriate SPP processes (by replacing action prefixes) such that

$$E_1 \| E_2 \| \cdots \| E_k \quad \sim \quad M_0, \tag{$\star$}$$

where M_0 is viewed as the initial state of the case graph for NS. We want this transformation to be *independent* of the actual decomposition considered, i.e. ($\star$) should hold for *any* SF decomposition of NS. First we note that there *always* exist SF decompositions of any net system.

Definition 8.7 *Let* $NS = (N, l, M_0)$ *be a net system. For each place* p *of* N *define* $NS_p = (N_p, l_p, M_{0p})$ *to be the* atomic *net system given by*

- $N_p = \Big(\{p\}, {}^\bullet p \cup p^\bullet; F \cap ({}^\bullet p \times \{p\} \cup \{p\} \times p^\bullet)\Big)$,
- l_p *is* l *restricted to* ${}^\bullet p \cup p^\bullet$, *and*
- M_{0p} *is* M_0 *restricted to the singleton* $\{p\}$.

Clearly $\left\{NS_p\right\}_{p \in P}$ is an SF decomposition of NS (assuming the net does not have any isolated transitions, i.e. transitions t for which ${}^\bullet t = t^\bullet = \emptyset$) thus showing that a particular decomposition could involve only atomic net systems. We shall call such a decomposition *atomic* and it represents a particular decomposition whose components cannot be decomposed further. Notice, that an atomic net is an SF component of a simple kind; every transition has at most one input and also at most one output place. Such nets are commonly known as S-nets (see [RT86]).

8.2.3 The Transformation from PN to SPP

Let us now describe the transformation from PN to SPP. First of all, we shall construct to every synchronisation free net system a corresponding (wrt bisimilarity) family of BPP equations (in full standard form). To facilitate this we introduce the notion of a trigger.

Definition 8.8 *Let $N = (S, T; F)$ be a net. A transition $t \in T$ satisfying ${}^{\bullet}t = \emptyset$ is called a* trigger *transition.*

Let $NS = (N, l, M_0)$ with $N = (S, T; F)$ be a synchronisation free net system. Suppose the places are $S = \{p_1, p_2 \dots p_n\}$ and that $\{t_1, t_2 \dots t_m\} \subseteq T$ are the trigger transitions of N. We build a family Δ_{NS} of BPP process equations by setting $Var(\Delta_{NS}) = \{X_1, X_2 \dots X_n, Y_1, \dots Y_m\}$ and construct the defining expressions for the variables as follows. First of all, if $t \in T$ with $t^{\bullet}(p_i) = k_i$ for $i = 1 \dots n$, then we let A_t denote the parallel expression $X_1^{k_1} \| X_2^{k_2} \| \cdots \| X_n^{k_n}$. Now, for $i = 1, 2 \dots n$, if $p_i^{\bullet} = \{t_{i1}, t_{i2} \dots t_{im_i}\}$ we set

$$X_i \stackrel{\text{def}}{=} \sum_{j=1}^{m_i} l(t_{ij}) A_{t_{ij}}.$$

Finally, for each Y_i we set $Y_i \stackrel{\text{def}}{=} l(t_i)(Y_i \| A_{t_i})$ and this completes the description of the family Δ_{NS}. Notice that Δ_{NS} is in full standard form (see Definition 2.30, page 46) with degree $d = \max\{size(t^{\bullet}) : t \in T\}$ if there is no trigger transitions; otherwise Δ_{NS} has degree $\max\Big\{d, 1 + \max\{size(t^{\bullet}) : t \text{ is a trigger transition}\}\Big\}$.

For every Δ_{NS} as constructed above we let $trig(\Delta_{NS})$ denote the parallel expression $Y_1 \| Y_2 \| \cdots \| Y_m$, i.e. the composition of all the variables originating from trigger transitions of NS. Furthermore, to every marking M of NS with $M(p_i) = k_i$ for $i = 1, 2 \dots n$ we let E_M denote the process $X_1^{k_1} \| X_2^{k_2} \| \cdots \| X_n^{k_n}$.

Lemma 8.9 *Let $NS = (N, l, M_0)$ be a synchronisation free net system (an* SF *component) and suppose Δ_{NS} is the associated family of* BPP *equations. We then have $M_0 \sim E_{M_0} \| trig(\Delta_{NS})$.*

Proof: We show that the relation B given by

$$B \quad = \quad \Big\{ \Big(M, E_M \| trig(\Delta_{NS}) \Big) : M \text{ is a marking of } N \Big\}$$

is a bisimulation relation. Since $\Big(M_0, E_{M_0} \| trig(\Delta_{NS}) \Big) \in B$ we shall have our result.

Suppose $\Big(M, E_M \| trig(\Delta_{NS}) \Big) \in B$ and consider $M \xrightarrow{a} M'$ for some marking M'. Hence there exists a transition t of N with label a such that $M[t{>}M'$ where the marking M' is given by $M' = M - {}^\bullet t + t^\bullet$. Assume that $t^\bullet(p_i) = k_i$ for $i = 1, 2 \ldots n$, where n is the number of places of NS. First of all, if t is a trigger transition, i.e. ${}^\bullet t = \emptyset$, then $trig(\Delta_{NS})$ contains a variable Y_t with defining equation

$$Y_t \stackrel{\text{def}}{=} l(t)(Y_t \| X_1^{k_1} \| \cdots \| X_n^{k_n}) \quad \in \quad \Delta_{NS}.$$

But then $E_M \| trig(\Delta_{NS}) \xrightarrow{a} E_{M'} \| trig(\Delta_{NS})$ as required. Now suppose that t is not a trigger transition. Thus, as NS is synchronisation free, we have ${}^\bullet t = \{p_j\}$ for some place p_j. Since transition t is enabled at M we have $M(p_j) > 0$. Hence E_M contains the variable X_j whose defining equation has a summand of the form $l(t)(X_1^{k_1} \| \cdots \| X_n^{k_n})$. Thus $E_M \| trig(\Delta_{NS}) \xrightarrow{a} E_{M'} \| trig(\Delta_{NS})$ as required.

Now suppose $E_M \| trig(\Delta_{NS})$ performs a move. From the construction of Δ_{NS} every reachable state of $E_M \| trig(\Delta_{NS})$ will be of the form $E_{M'} \| trig(\Delta_{NS})$ for some marking M' of N. Suppose then that $E_M \| trig(\Delta_{NS}) \xrightarrow{a} E_{M'} \| trig(\Delta_{NS})$. The performance of this move must be caused by some transition t with $l(t) = a$. Assume that $t^\bullet(p_i) = k_i$ for $i = 1, 2 \ldots n$. First of all, suppose that the move originates from a variable Y_j of the expression $trig(\Delta_{NS})$. Hence Y_j has the defining equation

$$Y_j \stackrel{\text{def}}{=} l(t)(Y_j \| X_1^{k_1} \| \cdots \| X_n^{k_n}) \quad \in \quad \Delta_{NS},$$

and M' is $M + t^\bullet$. But since t is a trigger transition we also have $M[t{>}M'$ and therefore $M \xrightarrow{a} M'$ as required. Finally, suppose that the move originates from a variable X_j of E_M. Thus X_j has a summand of the form $l(t)(X_1^{k_1} \| X_2^{k_2} \| \cdots \| X_n^{k_n})$. By the construction of Δ_{NS} and the fact that NS is synchronisation free we know that ${}^\bullet t = \{p_j\}$, thus $M' = M - {}^\bullet t + t^\bullet$. Since X_j is part of E_M we have $M(p_j) > 0$

and therefore $M[t{>}M'$ from which $M \xrightarrow{a} M'$ follows as required. This completes the proof. ■

Notice that by the above theorem and the fact that the net systems constructed from BPP processes (see Chapter 3, Section 3.1) are synchronisation free, we have characterised a subclass of PN, viz. the synchronisation free net systems, corresponding to BPP processes (up to bisimilarity). From the above theorem together with the result of Chapter 6 we may furthermore conclude that bisimilarity is decidable for synchronisation free net systems.

Let $NS = (N, l, M_0)$ be an arbitrary net system. Suppose NS is SF decomposed into $NS_i = (N_i, l_i, M_{0i})$ for $i = 1, 2 \dots k$. To each SF component NS_i we have associated a family Δ_{NS_i} of BPP process equations together with the processes $E_{M_{0i}}$ and $trig(\Delta_{NS_i})$ satisfying Lemma 8.9. We now transform each Δ_{NS_i} (and therefore also the processes $E_{M_{0i}}$ and $trig(\Delta_{NS_i})$) into appropriate SPP processes by replacing action prefixes such that by combining all the (new) processes $E_{M_{0i}}$ and $trig(\Delta_{NS_i})$ in parallel we get the behaviour of the original net system up to bisimilarity, i.e.

$$M_0 \;\sim\; \prod_{i=1}^{k} \Big(E_{M_{0i}} \| trig(\Delta_{NS_i})\Big).$$

In order to do this we associate to every transition t of the net N a number $c(t)$ indicating that t is shared between exactly $c(t)$ components of the decomposition of NS. Notice that $c(t)$ is easily computed and that $1 \leq c(t) \leq k$ for every transition t of N.

Now let $t \in T$ be a transition of $N = (S, T; F)$ and suppose $c(t) = m$ such that t is shared between the components $NS_{i1}, NS_{i2} \dots NS_{im}$. For $j = 1, 2 \dots m$ we replace every prefix $l(t)$ of any equation of $\Delta_{NS_{ij}}$ by the m-ary synchronisation action a_{ij} while at the same time defining the synchronisation function f_m to be $f_m(a_{i1}, a_{i2} \dots a_{im}) = l(t)$. We perform this construction for every transition $t \in T$ thereby completely describing the families Δ_{NS_i} (for $i = 1, 2 \dots k$) that we wish to obtain. Finally, we close all the synchronisation functions that are used under commutativity and assume that they are undefined elsewhere. Notice that we at most use synchronisation actions from $\Lambda_1, \Lambda_2 \dots \Lambda_k$.

Theorem 8.10 *Let $NS = (N, l, M_0)$ be a labelled Petri net system and suppose it is* SF *decomposed into the components $NS_1, NS_2 \dots NS_k$. To every* SF *component $NS_i = (N_i, l_i, M_{0i})$ we assume associated the family Δ_{NS_i} of guarded* SPP *process equations together with the processes $E_{M_{0i}}$ and $trig(\Delta_{NS_i})$. We have*

$$M_0 \;\sim\; \prod_{i=1}^{k} \Big(E_{M_{0i}} \| trig(\Delta_{NS_i})\Big).$$

Proof: Any marking M for the underlying net N of NS may be divided into markings $M_1, M_2 \dots M_k$ such that M_i is a marking for N_i $(i = 1, 2 \dots k)$. Whenever we write M_i in the following it is understood that M_i is the marking for component NS_i as given by marking M for NS.

Consider the relation B defined by

$$B \;=\; \Big\{\Big(M, \prod_{i=1}^{k}(E_{M_i} \| trig(\Delta_{NS_i}))\Big) : M \text{ is a marking for } N\Big\}.$$

We show that B is a bisimulation relation. Since $\Big(M_0, \prod_{i=1}^{k}(E_{M_{0i}} \| trig(\Delta_{NS_i}))\Big) \in B$ we shall have our result.

Suppose $\Big(M, \prod_{i=1}^{k}(E_{M_i} \| trig(\Delta_{NS_i}))\Big) \in B$ and let $M \xrightarrow{a} M'$ for some marking M' of N. Hence there exists a transition t of N with label a such that $M[t{>}M'$. For sake of clarity we shall assume that $c(t) = 2$ and that t is shared between the two components NS_1 and NS_2. The arguments for the general case are easily derived. The divisions $M_1, M_2 \dots M_k$ and $M'_1, M'_2 \dots M'_k$ of M and M' respectively therefore satisfy $M_i = M'_i$ for $i > 2$. Now from $M[t{>}M'$ we have $M_1[t{>}M'_1$ of NS_1 and similarly $M_2[t{>}M'_2$ of NS_2. Using Lemma 8.9 we may deduce that

$$E_{M_i} \| trig(\Delta_{NS_i}) \overset{a_i}{\dashrightarrow} E_{M'_i} \| trig(\Delta_{NS_i})$$

for $i = 1, 2$ and therefore (since $f_2(a_1, a_2) = a$)

$$\prod_{i=1}^{k} \Big(E_{M_i} \| trig(\Delta_{NS_i})\Big) \xrightarrow{a} \prod_{i=1}^{k} \Big(E_{M'_i} \| trig(\Delta_{NS_i})\Big)$$

as required.

Now suppose that $\prod_{i=1}^{k}(E_{M_i} \| trig(\Delta_{NS_i}))$ performs an a-move for some observable action $a \in \Lambda_1$. From the construction of the families Δ_{NS_i} such a move must be based on a transition t of N with label a. Suppose again for clarity that $c(t) = 2$ and that t is shared between the components NS_1 and NS_2. The arguments for the general case are easily reproduced. We have $E_{M_i} \| trig(\Delta_{NS_i}) \overset{a_i}{\dashrightarrow} E_{M_i'} \| trig(\Delta_{NS_i})$ where $a_i \in \Lambda_2$ (for $i = 1, 2$) and $f_2(a_1, a_2) = a$. Once again, relying on Lemma 8.9, we may deduce that $M_i[t{>}M_i'$ for $i = 1, 2$ and this gives us $M \xrightarrow{a} M'$ for the marking M' given by $M_1', M_2', M_3 \dots M_k$. Since the move by $\prod_{i=1}^{k}(E_{M_i} \| trig(\Delta_{NS_i}))$ is of the form

$$\prod_{i=1}^{k} \left(E_{M_i} \| trig(\Delta_{NS_i})\right) \xrightarrow{a} E_{M_1'} \| trig(\Delta_{NS_1}) \| E_{M_2'} \| trig(\Delta_{NS_2}) \prod_{i=3}^{k} \left(E_{M_i} \| trig(\Delta_{NS_i})\right)$$

we have the required result. This completes the proof. ■

Example 8.11 *The net system of Figure 8–2 illustrates two processes E and F competing on the semaphore S in order to prevent the processes from entering their critical regions (symbolised by the places p_{12} and p_{22} respectively) at the same time. If the net is* SF *decomposed into the components for E, F and S as illustrated by the dashed lines of Figure 8–2 then we obtain the following process representation for the behaviour of the net system.*

$$\begin{array}{llllllllllll}
\Delta_E: & X_{11} & \stackrel{\text{def}}{=} & b_{11}X_{12} & \Delta_S: & X_1 & \stackrel{\text{def}}{=} & b_{12} + b_{22} & \Delta_F: & X_{21} & \stackrel{\text{def}}{=} & b_{21}X_{22} \\
& X_{12} & \stackrel{\text{def}}{=} & c_{11}X_{13} & & Y_{12} & \stackrel{\text{def}}{=} & c_{12}(Y_{12}\|X_1) & & X_{22} & \stackrel{\text{def}}{=} & c_{21}X_{23} \\
& X_{13} & \stackrel{\text{def}}{=} & a_1X_{11} & & Y_{22} & \stackrel{\text{def}}{=} & c_{22}(Y_{22}\|X_1) & & X_{23} & \stackrel{\text{def}}{=} & a_2X_{21}
\end{array}$$

We have actions a_i, b_i, c_i for $i = 1, 2$ of type Λ_1 while the rest of the actions are of type Λ_2. The synchronisation function f_2 is defined by $f_2(b_{11}, b_{12}) = b_1$, $f_2(b_{21}, b_{22}) = b_2$, $f_2(c_{11}, c_{12}) = c_1$ and $f_2(c_{21}, c_{22}) = c_2$, and assumed to be closed under commutativity and undefined elsewhere. The process representing the behaviour of the net system is now given by $X_{11}\|X_1\|X_{21}\|Y_{12}\|Y_{22}$.

Notice that instead of decomposing the net system of Figure 8–2 into the three components represented by E, F and S we could have decomposed it into its atomic nets. This would give seven components and a completely different process description of the behaviour of the net system. However, by Theorem 8.10 it follows that no matter which decomposition we choose we will always obtain a correct (wrt bisimilarity) process representation of the behaviour of the net system.

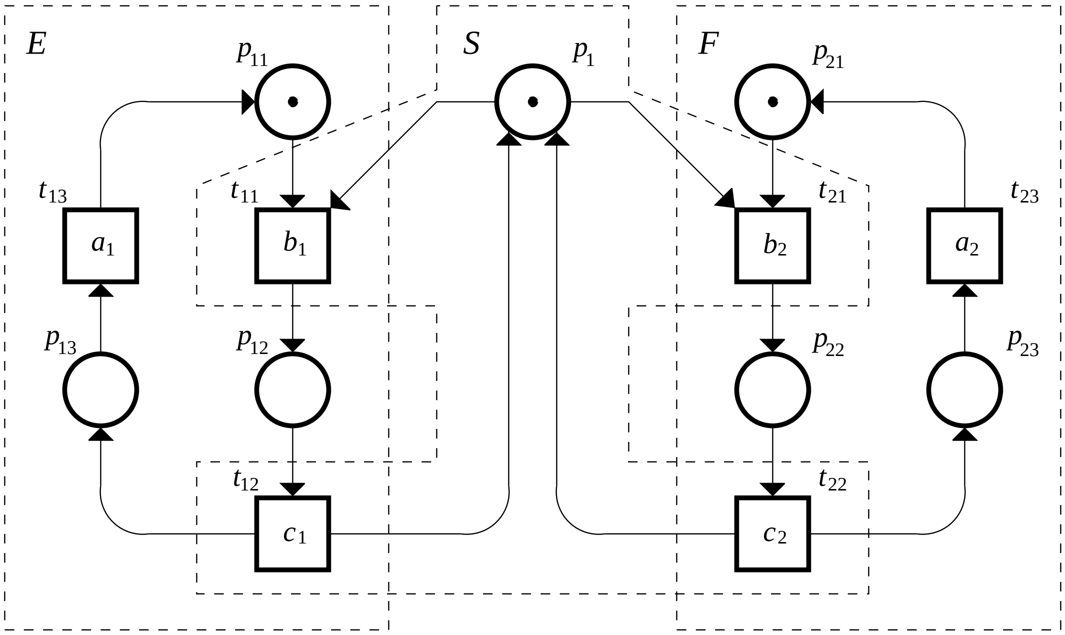

Figure 8–2: Processes E and F coordinating mutual exclusion.

From Theorem 8.10 and Theorem 8.1 we conclude that bisimilarity is undecidable on SPP. We can strengthen this even further since the construction of the net system NS_K used in the proof of Theorem 8.1 has the property that every transition has at most two input places. Thus a particular decomposition of NS_K and subsequent transformation into SPP will use only the action sets Λ_1 and Λ_2 symbolising either *atomic* actions (via Λ_1) or *binary* synchronisation (via Λ_2). Thus bisimilarity is undecidable on the fragment of SPP that uses only unary and binary synchronisation actions.

The interest in this fragment of SPP is due to the fact that it constitutes a natural extension of BPP_τ regarding expressiveness. In BPP_τ we have binary synchronisation as expressed via complementary actions but cannot *force* it to occur. This is illustrated by the fact that the process $aE|\bar{a}F$ of BPP_τ *may* synchronise thus performing a τ-action and become the process $E|F$. However $aE|\bar{a}F$ may also perform the a-move resulting in $E|\bar{a}F$, or the $\bar{a}$-move resulting in $aE|F$. A natural extension to BPP_τ would then be to force synchronisation between processes, and therefore resulting in a calculus similar to SPP (restricted to unary and binary synchronisation actions). For such a calculus we must conclude that bisimilarity is undecidable.

8.2.4 On Defining Distributed Bisimilarity for SPP

In Chapter 7 we demonstrated decidability of distributed bisimilarity on the process classes of BPP and BPP_τ. A natural next step would be to ask whether distributed bisimilarity is decidable on the extended calculus SPP. However, we encounter a serious problem here since it is not clear how to define distributed labelled transition graphs (as required for the definition of $\sim_d$) regarding SPP processes. In transition steps we should *not* separate processes into local and concurrent residuals since this would lead to unwanted deadlock situations due to the enforced synchronisation between parallel processes. If our definition of $\sim_d$ were based on an operational semantics using local and concurrent residuals then it would become *incomparable* with $\sim$ (and not included as it ought to be).

Example 8.12 *Let $a, b \in \Lambda_2$ and $a', b', c, d \in \Lambda_1$. Suppose that $f_2(a, a) = a'$ and $f_2(b, b) = b'$ while f_2 is undefined elsewhere. We clearly have $a \| ab \| bc \not\sim a \| ab \| bd$ since, after the synchronisation on a and then b, the first process can only do a c-action while the second process only can do a d-action. However, we have $a \| ab \| bc \sim_d a \| ab \| bd$ since after the synchronisation on a, the first process yields the compound residual (b, bc) while the second process yields (b, bd). We clearly have $b \sim_d b$ and $bc \sim_d bd$ as all processes now are deadlocked (wrt the transition relation $\longrightarrow$).*

We note that Castellani encountered a similar problem when considering the question of giving a distributed operational semantics to the combinator of restriction (see [Cas88]). An inference rule like

$$\frac{E \xrightarrow{\mu} (E_1, E_2)}{E \setminus L \xrightarrow{\mu} (E_1 \setminus L, E_2 \setminus L)} (\mu \notin L \cup \bar{L})$$

is obviously not the right one since it prohibits any further communication between E_1 and E_2 on actions from L or $\bar{L}$.

If an equivalence like $\sim_d$ is to be reasonably defined on SPP we have to abandon the idea of letting transitions evolve into a compound residual consisting of a local and a concurrent residual. Instead the operational semantics could be based on transitions of the form $E \xrightarrow{\mu} (E_1, E_2, E_3)$ where E_1 and E_2 are the local and concurrent residuals (as before) but E_3 now the *global* residual, i.e. E_3 is the result of E performing the action μ as we know it from the standard operational semantics. An equivalence $\sim_d'$ based on this operational interpretation of SPP processes would certainly be included in $\sim$. Another possibility would be to take the approach of [BCHK92] where a distributed operational semantics is defined for CCS using explicit location names. We shall leave it for future work to define an appropriate notion of distributed bisimilarity on SPP. However, notice that once we have such a definition then we would also get an equivalence on labelled Petri net systems, viz. that induced by the transformation of net systems into SPP processes.

Chapter 9

Conclusion

Searching for decidable theories is an important task; it may provide us with a boundary between decidable and undecidable theories and it may offer a deeper understanding of the subject being explored. In this thesis we have investigated the question of obtaining decidable theories for behavioural equivalences on various models of computation encompassing systems with infinitely many states.

In a broader perspective our work should be seen as part of (and is driven by) the search for more and more expressive models of computation while maintaining decidable theories for behavioural equivalences. This research should be viewed in its widest possible interpretation. For instance, what do we mean by a model of computation, how is expressiveness understood and which are the behavioural equivalences that we consider?

The ultimate question is whether there exists a model of computation being Turing powerful (as for instance measured in terms of the possibility to generate all recursive enumerable languages) for which there exists a (nontrivial) decidable theory.

In this thesis we have considered various models of computation within the tradition of process algebras and Petri nets. We have compared these models from the viewpoint of formal language theory (see Figure 3–2 for a comparison of the various models). The behavioural equivalences we have considered are the (widely accepted) equivalences of language equivalence and bisimulation equivalence together with the less known distributed bisimulation equivalence as defined by Castellani in [Cas88].

In the next section we summarise the work achieved and in the final section of this chapter we outline directions for further work.

9.1 Summary of the Main Results

In Chapter 3 we investigated the class L(BPP) from the viewpoint of formal language theory. In particular, we showed that L(BPP) is incomparable with the class of context-free languages (or equivalently the class of languages generated by $\mathrm{BPA}_{\mathbf{0}}$) and contained in a class of languages generated by labelled Petri nets as well as being included in the class of context-sensitive languages. We furthermore investigated decidability questions for language equivalence on classes related to L(BPP) and L(PN). In particular, we showed that language equivalence is undecidable on L(BPP∩RL), i.e. the class of languages obtained by the intersection of languages from L(BPP) with regular languages[1]. We also showed that language equivalence is undecidable on $\mathrm{L}(\mathrm{PN}_{\mathrm{d}})$, i.e. the class of terminal languages generated by deterministic labelled Petri nets.

In Chapter 4 we showed that bisimilarity is decidable on $\mathrm{BPA}_{\mathbf{0}}$ thus extending the result of [BBK87] beyond the normed case. Our technique for establishing this result relies on showing that the largest bisimulation relation on $Var(\Delta)^*$, where Δ is a finite family of $\mathrm{BPA}_{\mathbf{0}}$ equations in GNF, is generated from a finite self-bisimulation. Decidability questions for behavioural equivalences on $\mathrm{BPA}_{\mathbf{0}}$ have been investigated by others. It is well-known that language equivalence is undecidable on $\mathrm{BPA}_{\mathbf{0}}$. Furthermore, in [HT90,GH91] it is demonstrated that *all* the equivalences except bisimilarity in the linear time–branching time spectrum of [Gla90] are undecidable for $\mathrm{BPA}_{\mathbf{0}}$.

In Chapter 5 we proved several results on the decomposition of processes into the parallel composition of other (simpler) processes. In particular, we showed

[1] Hirshfeld's very recent result on the undecidability of language equivalence on L(BPP) [Hir93] renders our result somewhat superfluously.

unique decomposition results for bisimilarity on normed BPP and BPP_τ as well as unique decomposition results for distributed bisimilarity on all of BPP and BPP_τ. We furthermore proved a theorem of degeneration for distributed bisimilarity, i.e. a theorem stating that every prime component is contained in the process being decomposed (into primes), and remarked that such a result does not hold for bisimilarity.

As mentioned, decidability of behavioural equivalences on $\mathrm{BPA}_\mathbf{0}$ has been investigated by others. Less investigated has been the decidability of behavioural equivalences on models of computation having parallelism as its main constructor. We believe to have initiated the work in this area by showing decidability of bisimilarity on the two process classes of BPP and BPP_τ (Chapter 6). Our proof of decidability relies on the tableau decision method. However, we also demonstrate the decidability result via a finite representability property as in the case of $\mathrm{BPA}_\mathbf{0}$. By these results we have obtained a delicate line between decidable and undecidable theories for bisimilarity: by extending BPP_τ with the operator of restriction we obtain a calculus with full Turing power and know that bisimilarity is undecidable by a reduction from the Halting problem.

In Chapter 6 we furthermore found sufficient conditions on the parallel operator for which the largest bisimulation relation on $Var(\Delta)^\otimes$, where Δ is a finite family of process equations in full standard form, is semi-decidable. These conditions are that bisimilarity should satisfy the laws of associativity, commutativity and be a congruence wrt the parallel operator. If the family Δ in addition generates an image-finite transition graph then we have decidability of bisimilarity on $Var(\Delta)^\otimes$. This is for instance the case with the CSP related parallel operator of $\|_A$.

In Chapter 6 we finally presented a sound and complete sequent-based equational theory for bisimilarity on the process class BPP_τ (as well as BPP). As this calculus clearly contains the regular processes our result may be seen as a proper extension of Milner's equational theory for regular processes.

In Chapter 7 we considered distributed bisimulation equivalence. We showed this equivalence to be decidable on the two process classes of BPP and BPP_τ. We obtained these results via the use of the tableau technique as in the case of bisimilarity. However, the tableau system for distributed bisimilarity is simpler to analyse due to a strong cancellation property admitted by this equivalence. For instance, we were able to provide upper bounds on the size of tableaux, something we failed to do in the case of our tableau system for bisimilarity. We furthermore presented a proof of decidability of distributed bisimilarity on BPP_τ (as well as BPP) by a delicate use of the unique decomposition property which we proved (in Chapter 5) to be valid for distributed bisimilarity on the process classes of BPP and BPP_τ.

In Chapter 7 we furthermore presented a sound and complete sequent-based equational theory for distributed bisimilarity on BPP_τ (as well as BPP). This result may be seen as an extension of Castellani's equational theory for distributed bisimilarity on the recursion-free fragment of BPP_τ. Finally, we investigated the question of obtaining a sound and complete equation-based theory for distributed bisimilarity on a fragment of BPP_τ where general summation is replaced by guarded summation. In this case the equational theory become particularly simple due to the unique decomposition property admitted by distributed bisimilarity on the process class BPP_τ.

In Chapter 8 we presented a modified and simplified version of Jančar's reduction showing bisimilarity to be undecidable on the class of labelled Petri nets. We furthermore presented a transformation from PN into SPP through a delicate decomposition of labelled Petri nets. Using this transformation together with Jančar's result we may further narrow the gap between decidable and undecidable theories for bisimilarity. It is not necessary to add restriction to BPP_τ in order to obtain a calculus with full Turing power and thereby undecidability of bisimilarity; a sufficient condition is a notion of *forced* binary synchronisation added on top of BPP.

9.2 Further Work

In the rest of this chapter we shall describe open questions related to this thesis as well as outline further work.

9.2.1 Other Equivalences

We have only considered decidability on BPP and BPP_τ of a few of the many equivalences that have been suggested in the area of process algebra. It would be nice to have a picture as complete as that for $\mathrm{BPA}_\mathbf{0}$ wrt the linear time–branching time spectrum of [Gla90] where bisimilarity is decidable while all the other equivalences are undecidable. Hirshfeld's construction for the undecidability of language equivalence on L(BPP) also applies to show that trace equivalence and simulation equivalence are undecidable on L(BPP). Indeed, by relying on these undecidability results it might be possible to show many of the other equivalences within the linear time–branching time spectrum to be undecidable for BPP as well (this is the case for $\mathrm{BPA}_\mathbf{0}$ where the well-known undecidability of language equivalence is reduced to a number of the other equivalences [GH91]).

Of particular interest on BPP_τ are the weak versions of bisimilarity, viz. *weak bisimulation equivalence*, denoted $\approx$, and *observational congruence*, denoted $=$ (see [Mil89]). Since both of these equivalences are congruences wrt the parallel combinator of CCS a possible technique for the decidability of $\approx$ and $=$ on BPP_τ would be to show a finite representability result along the lines of the method described in Section 6.1.1. In fact, using this technique, it is possible to show that both of $\approx$ and $=$ can be generated from a finite binary relation on $Var(\Delta)^{\otimes}$ where Δ is a finite family of BPP_τ process equations in full standard form. From these results it is furthermore possible to show that $\approx$ and $=$ are semi-decidable on BPP_τ. However, we do not know whether we also have semi-decidability of $\not\approx$ and $\neq$ as in the case of bisimilarity.

9.2.2 Infinite Transition Graphs

We have mainly investigated decidability questions of behavioural equivalences on the classes of $\mathrm{BPA}_\mathbf{0}$, BPP and BPP_τ with $\mathrm{BPA}_\mathbf{0}$ being incomparable to BPP (as well as BPP_τ) from the point of view of language equivalence. A natural question would be whether we can obtain decidability of bisimilarity on a calculus which consists of $\mathrm{BPA}_\mathbf{0}$ together with BPP (such a calculus would to a large extent correspond to the Dutch calculus PA [BW90]). However, we have failed to find an appropriate notion of normal form for such a calculus. The notion of normal form is important for the success of our decidability techniques; in the case of BPA_0 the notion of normal form (GNF) enables us to restrict attention to the state space $Var(\Delta)^*$ and in the case of BPP (as well as BPP_τ) the notion of normal form (full standard form) enables us to restrict attention to the state space $Var(\Delta)^\otimes$. A related question of interest would be whether a BPA_0 process $\alpha \in Var(\Delta)^*$ can be shown equal (wrt bisimilarity or any other equivalence) to a BPP process $\beta \in Var(\Delta)^\otimes$.

Regarding BPA_0 a natural extension consists of the class of transition graphs generated by push-down automata. From the viewpoint of language equivalence such graphs are just as expressive as BPA_0. However, from the viewpoint of bisimulation equivalence transition graphs generated by push-down automata are more expressive than those of BPA_0 (see [Cau90a]). A natural question is whether the decidability result of bisimilarity on BPA_0 extends to the class of push-down automata. Regarding this question it would be worthwhile to explore whether the notions of decomposition and finite representability considered for BPA_0 are extendable to the class of push-down automata. The work of [MS85] establishing an elegant characterisation of transition graphs generated by push-down automata might be useful regarding the search for an appropriate finite representability for such transition graphs.

Regarding BPP and BPP_τ a more natural extension would be to add a notion of *forced* synchronisation. However, by relying on Jančar's result for undecidability of bisimulation equivalence on labelled Petri nets we must conclude that bisimilarity is undecidable on such a calculus. In Section 6.1.1 we mentioned other (stronger) notions of parallelism for which it is possible to obtain decidability of bisimilarity. For instance, bisimulation equivalence is decidable on $Var(\Delta)^\otimes$ where Δ is a finite family of process equations in full standard form and based on the CSP related operator of $\|_A$. We also demonstrated in Section 6.1.1 that such an operator is more expressive compared to the full merge of BPP or the parallel composition of BPP_τ.

However, it still remains to find natural extensions to BPP and BPP_τ (besides the inclusion of sequential composition) for which we should start searching for decidable theories for bisimilarity.

As illustrated in Chapter 8 there is a problem with defining distributed bisimilarity on models of computation which are stronger than BPP and BPP_τ. Of particular interest would be the possibility to define an appropriate notion of distributed bisimilarity on the calculus SPP since this would give us a notion of distributed bisimilarity on the model of labelled Petri nets via our transformation from PN to SPP.

On defining distributed bisimulation equivalence we have to abandon the idea of transition steps developing into a local and a concurrent residual as otherwise the corresponding equivalence $\sim_d$ is incomparable with $\sim$ (see Example 8.12). As mentioned in Chapter 8 a possibility would be to let a transition step result in a local, concurrent and global residual (where the global residual should be the standard result as we know it from the ordinary operational semantics). The corresponding equivalence is certainly included in bisimilarity. It would then be interesting to study properties of the induced equivalence on labelled Petri nets; for instance, how does it relate to other notions of equivalence on Petri nets that have been suggested in the literature, and can it be decided?

9.2.3 Equational Theories

In this thesis we have illustrated two very different equational theories for the behavioural equivalences of bisimilarity and distributed bisimilarity. Milner's equational theory for regular processes (which we demonstrated in Chapter 2) centres around elegant laws for recursion which is given by explicit fixed point operators. In contrast to this theory is the sequent-based equational theory relying on a corresponding tableau system.

In [HS91] a tableau method is presented for deciding bisimilarity on *normed* context-free processes. Furthermore, a sound and complete sequent-based equational theory is extracted from the tableau system. Since our proof of decidability of bisimulation equivalence on all context-free processes is not based on the tableau technique it is unclear how our result should support a sound and complete equational theory.

It therefore remains open how to extend the equational theory of [HS91] to the class of all context-free processes. A possible solution would be to present our decidability result along the lines of [HS91], i.e. by utilising the tableau method. However, the tableau system of [HS91] relies on the cancellation law (see Proposition 2.44) which is valid in the normed case but not in the presence of unnormed processes. Therefore it is not clear how the tableau system of [HS91] extends to the class of all context-free processes, and further work is needed.

We have been able to extract a sound and complete sequent-based equational theory from our tableau system for deciding bisimilarity on BPP_τ (as well as BPP). As mentioned, our equational theory may be seen as a proper extension of Milner's result for regular processes. However, we would prefer our theory to be in the style of Milner's, i.e. equation-based, but fail to see how Milner's technique for proving completeness can be extended to the process classes of BPP and BPP_τ. Maybe our unique decomposition properties for bisimilarity on normed BPP and BPP_τ can be of help. We used the corresponding unique decomposition property for distributed bisimilarity on BPP_τ in order to obtain an equational theory in the style of Milner's on the subclass of BPP_τ where general summation is replaced by guarded summation.

We have also presented a sequent-based equational theory for distributed bisimilarity on BPP_τ (as well as BPP). Again an interesting question is whether it is possible to obtain an equation-based theory in the style of Milner's. For the class of BPP (with recursion given by explicit fixed point constructors) we believe it it possible to obtain such an equational theory. The theory consists of Castellani's axioms for distributed bisimilarity on the recursion-free fragment of BPP together with Milner's laws for recursion. The proof of completeness follows Milner [Mil89a] closely (we presented this technique in Chapter 7). However, we have not been able to extend the proof of completeness to the class of BPP_τ. The problem concerns the notion of formal equation sets (a kind of normal form) which is central to the proof of completeness (see Section 7.4). Wrt BPP we can show that every process provably satisfies an appropriate formal equation set (the result corresponding to Theorem 7.17) but have not been able to obtain the same result for BPP_τ.

9.2.4 Complexity Bounds

Having established our decidability results the question remains whether the decision procedures have realistic complexity bounds. However, often our decidability results consist of two semi-decision procedures and therefore it is not obvious how to establish the complexity of solving the problems. For instance, wrt bisimulation equivalence on $\mathrm{BPA}_\mathbf{0}$ we have semi-decidability of $\not\sim$ based on a search for $\not\sim_n$ together with semi-decidability of $\sim$ based on a search for a finite self-bisimulation containing the pair of processes that we are comparing.

For the subclass of normed context-free processes there exist various algorithms for the decidability of bisimilarity, see [BBK87,Cau90,HS91,Gro91]. In [HT92] the complexity of these various algorithms are analysed. Also in [HT92] Huynh and Tian improve on the algorithm of [Gro91] and show that the decision problem is in $\mathbf{\Sigma_2^p}$, i.e. the second-order level of the polynomial-time hierarchy.

Wrt deciding bisimilarity on BPP_τ (as well as BPP) we have also not been able to estimate the complexity of solving the problem. A natural question is whether we can provide an upper bound on the size of tableaux. Related to this question is the question, given an infinite sequence of vectors $\tilde{v}_1, \tilde{v}_2 \ldots$ over $\mathbb{N}^n$, which is the least j (relative to the initial vector $\tilde{v}_1$) such that $\tilde{v}_i \leq \tilde{v}_j$ for some $i < j$ (this puzzle is related to the question as to when a Sub rule in a tableau is applicable). Such a problem seems to be non-trivial and we have no definite answers yet.

In Chapter 6 we described a decision method for bisimilarity on normed BPP_τ which is very different from the tableau technique; it relies on rewrite techniques and is similar in style to Caucal's procedure for deciding bisimilarity on normed BPA (see [Cau90]). This method allows for a complexity analysis; as mentioned in Chapter 6 it must be exponential in the number of variables involved since it relies on an enumeration of all fundamental relations over $Var(\Delta)^{\otimes}$ where Δ is the BPP_τ family we consider. In obtaining a more precise estimate it would be worthwhile to consider the work of [HT92] establishing a complexity analysis in the case of normed context-free processes.

In case of distributed bisimilarity our tableau system for decidability on BPP_τ (as well as BPP) is much simpler compared to our tableau system for bisimilarity due to a strong cancellation law admitted by this equivalence. This allowed us to give an upper bound on every tableau (which is exponential in terms of the number of variables involved). Hence the tableau decision procedure of Chapter 7 is at least exponential (in the number of variables involved).

Bibliography

[AK77] T. Araki and T. Kasami. *Some Decision Problems Related to the Reachability Problem for Petri Nets.* TCS 3, pp 85–104, 1977.

[BBK87] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. *Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages.* In Proceedings of PARLE 87, J.W. de Bakker, A.J. Nijman, P.C. Treleaven (eds), Lecture Notes in Computer Science 259, pp 93–114. Springer-Verlag, 1987.

[BBK87a] J.C.M. Baeten, J.A. Bergstra and J.W. Klop. *Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages.* Technical Report CS-R8632, CWI, September 1987.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18, 1990

[Bak72] H. Baker. *Petri Nets and Languages.* CSG, Memo 68, Project MAC, MIT, 1972.

[Ber86] G. Berthelot. *Transformations and Decompositions of Nets.* Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 (Part I), Edited by W. Brauer, W. Reisig and G. Rozenberg, LNCS 254, 1986.

[Bet59] E.W. Beth. *The Foundations of Mathematics.* North-Holland Publishing Co., 1959.

[BRS85] G. Boudol, G. Roucairol and R. de Simone. *Petri Nets and Algebraic Calculi of Processes.* LNCS 222, Advances in Petri Nets 1985, Edited by G. Rozenberg, Springer-Verlag, 1985

[BCHK92] G. Boudol, I. Castellani, M. Hennessy and A. Kiehn. *A Theory of Processes with Localities.* In proceedings of CONCUR 92, LNCS 630, pp 108–123, Springer-Verlag, 1992.

[BS90] J. Bradfield and C. Stirling. *Verifying Temporal Properties of Processes.* In proceedings of CONCUR 90, LNCS 458, pp 115–125, Springer-Verlag, 1990.

[Buc62] J.R. Büchi. *On the Decision Method in Restricted Second-Order Arithmetic.* Logic, Methodologi and Philosophy of Science Proc., Intern Congr, E. Nagel et al (Eds.), Stanford Uni. Press pp 1–11, 1962.

[BB92] O. Burkart and B. Steffen. *Model Checking for Context-Free Processes.* In Proceedings of CONCUR 92, W.R. Cleaveland (ed), Lecture Notes in Computer Science 630, pp 123–137, Springer-Verlag, 1992.

[Cau86] D. Caucal. *Décidabilité de l'egalité des langages algébriques infinitaires simples.* In Proceedings of STACS 86, LNCS 210, pp 37–48. Springer-Verlag, 1986.

[Cau88] D. Caucal. *Graphes Canoniques des Graphes Algébriques.* Rapport de Recherches 872, INRIA, Juillet 1988.

[Cau90] D. Caucal. *Graphes Canoniques des Graphes Algébriques.* Informatique Théorique et Applications (RAIRO) 24(4), pp 339–352, 1990.

[Cau90a] D. Caucal. *On the Regular Structure of Prefix Rewriting.* In Proceedings of CAAP90, LNCS 431, Springer-Verlag, 1990.

[CH87] I. Castellani and M. Hennessy. *Distributed Bisimulations.* Research Report 5/87, University of Sussex, July 1987.

[Cas88] I. Castellani. *Bisimulations for Concurrency.* PhD thesis, CST-51-88, University of Edinburgh, 1988.

[Chr92] S. Christensen. *Distributed Bisimilarity is Decidable for a Class of Infinite-State Systems.* In Proceedings of CONCUR 92, W.R. Cleaveland (ed), Lecture Notes in Computer Science 630, pp 148–161, Springer-Verlag, 1992.

[CHS92] S. Christensen, H. Hüttel and C. Stirling. *Bisimulation Equivalence is Decidable for all Context-Free Processes.* In Proceedings of CONCUR 92, W.R. Cleaveland (ed), Lecture Notes in Computer Science 630, pp 138–147, Springer-Verlag, 1992.

[CHM93] S. Christensen, Y. Hirshfeld and F. Moller. *Decomposability, Decidability and Axiomatisability for Bisimulation Equivalence on Basic Parallel Processes.* In Proceedings of LICS93. IEEE Computer Society Press, 1993.

[CHM93a] S. Christensen, Y. Hirshfeld and F. Moller. *Bisimulation is Decidable for Basic Parallel Processes.* To appear in Proceedings of CONCUR 93, Springer-Verlag.

[Cou83] B. Courcelle. *An Axiomatic Approach to the Korenjak-Hopcroft Algorithms.* Mathematical Systems Theory 16, pp 191–231, 1983.

[Gis81] J. Gischer. *Shuffle Languages, Petri Nets, and Context-Sensitive Grammars.* Communications of the ACM, Vol. 24, Number 9, pp 597–605, September 1981.

[Gla90] R.J. van Glabbeek. *The Linear Time–Branching Time Spectrum.* In Proceedings of CONCUR 90, J. Baeten, J.W. Klop (eds), Lecture Notes in Computer Science 458, pp 278–297, Springer-Verlag, 1990.

[Gro91] J.F. Groote. *A Short Proof of The Decidability of Bisimulation for Normed BPA-Processes.* Technical Report, CWI, 1991.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel 1983.

[Gra79] J. Grabowski. *The Unsolvability of some Petri Nets Language Problems.* Information Processing Letters, Volume 9, number 2, pp 60–63, 1979.

[Gen35] G. Gentzen. *Untersuchungen über das logischen Schliessen.* Mathematishe Zeitschrift, Vol 39, pp 176–210, 1935.

[GM84] U. Goltz and A. Mycroft. *On the relationship of CCS and Petri nets.* J. Paredaens (Ed.): ICALP 84, Springer LNCS 172, pp 196–208, 1984.

[Gol88] U. Goltz. *On Representing CCS Programs by Finite Petri Nets.* M.P. Chytil et al (Eds.): MFCS, Springer LNCS 324, pp 339–350, 1988.

[Gri68] T.Y. Griffiths. *The Unsolvability of the Equivalence Problem for Λ-Free Nondeterministic Generalised Automata.* J.ACM 15, pp 409–413, 1968.

[GH91] J.F. Groote and H. Hüttel. *Undecidable Equivalences for Basic Process Algebra.* Research report ECS-LFCS-91-169, University of Edinburgh, August 1991.

[GM92] J.F. Groote and F. Moller. *Verification of Parallel Systems via Decomposition.* In Proceedings of CONCUR 92, W.R. Cleaveland (ed), Lecture Notes in Computer Science 630, pp 62–76, Springer-Verlag, 1992.

[Hac74] M. Hack. *Extended State-Machine Allocatable Nets, an Extension of Free Choice Petri Nets Results.* Cambridge, Massachussets, MIT Project MAC, CSG-Memo 78-1, 1974.

[Hac75] M. Hack. *Petri Net Languages.* CSG, Memo 124, Project MAC, MIT, 1975.

[Hac75a] M. Hack. *Decidability Questions For Petri Nets.* Ph.D. Thesis, CSG, Memo 161, Project MAC, MIT, December, 1975.

[Hir93] Y. Hirshfeld. *Finitely Generated Processes, Petri Nets and the Equivalence Problem.* Unpublished notes, University of Edinburgh, May 1993.

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

[HK66] J. Hopcroft and A. Korenjak. *Simple Deterministic Languages.* 7th IEEE Annual Symp. on Switching and Automata Theory, Barkeley, California, pp 36–46, 1966.

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

[HC68] G.E. Huges and M.J. Cresswell. *An Introduction to Modal Logic.* Methuen and Co. Ltd., 1968.

[HS91] H. Hüttel and C. Stirling. *Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes.* In Proceedings of LICS 91, pp 376–386, IEEE Computer Society Press, 1991.

[Hüt91] H. Hüttel. *Decidability, Behavioural Equivalences and Infinite Transition Graphs.* Ph.D thesis, University of Edinburgh, December 1991.

[HT92] D.T. Huynh and L. Tian. *Deciding Bisimilarity of Normed Context-Free Processes is in* $\mathbf{\Sigma_2^p}$. The University of Texas at Dallas, Richardson, Texas 75083-0688, Technical Report UTDCS-1-92, 1992.

[HT90] D.T. Huynh and L. Tian. *On Deciding Readiness and Failures Equivalences for Processes.* Research report UTDCS-31-90, University of Texas at Dallas, September 1990.

[Iba77] O.H. Ibarra. *The Unsolvability of the Equivalence Problem for ϵ-free NGSM's with unary input (output) alphabet and Applications to some Grammar and Graph Problems.* Proc. 18th Annual IEEE Symposium on the Foundation of Computer Science, pp 74–81, 1977.

[Jan93] P. Jančar. *Decidability Questions for Bisimilarity of Petri Nets and Some Related Problems.* University of Edinburgh, LFCS report Series, ECS-LFCS-93-261, 1993.

[Jan86] M. Jantzen. *Language Theory of Petri Nets.* Petri Nets: Central Models and their Applications, LNCS 254, pp 397–412, 1986.

[Jan88] M. Jantzen. *Confluent String Rewriting.* EATCS Monographs on Theoretical Computer Science, Vol. 14, W. Brauer, G. Rozenberg and A. Salomaa (Eds.), 1988.

[Kel76] R. Keller. *Formal Verification of Parallel Programs.* Comm. ACM 19, no. 7, pp 561–572, 1976.

[Kos82] S.R. Kosaraju. *Decidability of Reachability in Vector Addition Systems.* Proc. 14th annual ACM STOC, pp 267–281, 1982.

[Koz91] D. Kozen. *A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events.* In Proceedings of LICS 91, pp 214–225, IEEE Computer Society Press, 1991.

[Mad92] E. Madelaine. *Verification tools from the CONCUR project.* Bulletin of the European Association of Theoretical Computer Science, 47, pp 110–126, June 1992.

[May81] E. Mayr. *An Algorithm for the General Petri Net Reachability Problem.* Proc. 13th annual ACM STOC, pp 238–246, 1981.

[May84] E. Mayr. *An Algorithm for the General Petri Net Reachability Problem.* SIAM, J.Comp. Vol. 13, No 3, pp 441–460, 1984.

[Maz87] A. Mazurkiewicz. *Trace Theory.* W. Brauer et all (Eds.). Petri Nets: Applications and relationships to other models of concurrency. Springer LNCS 255, pp 279–324, 1987.

[Mil80] R. Milner *A Calculus of Communicating Systems.* LNCS 92, Springer-Verlag, 1980.

[Mil83] R. Milner. *Calculi for Synchrony and Asynchrony.* TCS 25, pp 267–310, 1983.

[Mil84] R. Milner. *A Complete Inference System for a Class of Regular Behaviours.* Journal of Computer and System Sciences 28, pp 439–466, 1984.

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

[Mil89a] R. Milner. *A Complete Axiomatisation for Observational Congruence of Finite-State Behaviours.* Journal of Information and Computation, Vol. 81, No. 2, May 1989.

[MM90] R. Milner and F. Moller. *Unique Decomposition of Processes.* Bulletin of the European Association for Theoretical Computer Science 41, pp 226–232, 1990.

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

[Mol89] F. Moller. *Axioms for Concurrency.* PhD thesis CST-59-89. University of Edinburgh, 1989.

[Mul84] H. Muller. *The Reachability Problem for VAS.* Adv. in Petri Nets, LNCS 188, pp 376–391, 1984.

[MS85] D.E. Muller and P.E. Schupp. *The theory of Ends, Pushdown Automata, and Second-Order Logic.* TCS 37, pp 51–75, 1985.

[Nie87] M. Nielsen. *CCS-and its relationship to Net Theory.* W. Brauer et all (Eds.). Petri Nets: Applications and relationships to other models of concurrency. Springer LNCS 255 pp 393–415, 1987.

[PP85] P. Parigot and E. Pelz. *A Logical Approach of Petri Net Languages.* TCS 39, pp 155–169, 1985.

[Par81] D.M.R. Park. *Concurrency and Automata on Infinite Sequences.* Proceedings of the 5th G.I. Conference, LNCS 104, Springer-Verlag, 1981.

[Pet74] J. Peterson. *Modeling of Parallel Systems.* Tech. Report STAN-CS-74-410, Computer Science Department, Standford University, 1974.

[Plo81] G. Plotkin. *A Structural Approach to Operational Semantics.* Daimi FN-19, Computer Science Department, Aarhus University, 1981.

[Pra65] D. Prawitz. *Natural Deduction.* Almqvist and Wiksell. Stockholm, 1965.

[Pre29] M. Presburger. *Über die Vollständigkeit eines gewissen System der Arithmetic ganzer Zahlen in welchem die Addition als winzige Operation hervortritt.* Comptes Rendus du I^{er} Congrès des Mathématiciens des Pays Slaves, Warszawa, pp 92–101, 1929.

[Rab69] M. Rabin. *Decidability of Second-Order Theories and Automata on Infinite Trees.* Trans. Am. Math. Soc. **141**, pp 1–35, 1969.

[Rab92] A. Rabinovich. *A Complete Axiomatisation for Trace Congruence of Finite State Behaviours.* Draft, Department of Computer Science, The University of Texas at El Paso, 1992.

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

[RT86] G. Rozenberg and P.S Thiagarajan. *Petri Nets: Basic Notions, Structures and Behaviour.* J.W. de Bakker, W.-P. de Roever, G. Rozenberg (Eds.): Current Trends in Concurrency, LNCS 224, pp 585–667, 1986.

[Sal66] A. Salomaa. *Two Complete Axiom Systems for the Algebra of Regular Events.* Journal of the Association of Computing Machinery 13, pp 158–169, 1966.

[Smu68] R.M. Smullyan. *First-Order Logic.* Springer-Verlag, Berlin, 1968.

[Sta78] P.H. Starke. *Free Petri Net Languages.* Proc. 7th Symp. MFCS 78, Winkowski (Ed.), LNCS 64, pp 506–515, 1978.

[SW89] C. Stirling and D. Walker. *Local Model Checking in the Modal Mu-Calculus.* In LNCS 351, pp 369–383, Springer-Verlag, 1989.

[Tau89] D. Taubner. *Finite Representations of CCS and CSP programs by Automata and Petri Nets.* G. Goos and J. Hartmanis (Eds.), Springer LNCS 369, 1989.

[Tho90] W. Thomas. *Automata on Infinite Objects.* In Van Leeuwen (Eds): Handbook of Theoretical Computer Science, pp 133–191, North-Holland, 1990.

[Win84] G. Winskel. *Synchronization Trees.* TCS Vol. 34, pp 33–82, 1984.

Appendix A

Proofs of Standard Forms

This appendix is devoted to a proof of Proposition 2.29 and Proposition 2.31 of Chapter 2. That is, we shall prove that any finite family Δ of guarded BPP_τ process equations can be transformed *effectively* into guarded standard form of degree 3 while preserving distributed bisimilarity (Proposition 2.29), and we shall furthermore prove that Δ can be *effectively* transformed into full standard form of degree 2 while preserving bisimilarity (Proposition 2.31).

First of all, we present a definition of *normal form* concerning processes of BPP_τ and show that every such process can be *effectively* transformed into normal form while preserving distributed bisimilarity. Thereafter we shall use this normal form in our proof of obtaining standard form.

For the definition of normal form we introduce another operator for forming processes, viz. *communication merge* (see [BW90]). We have mentioned this operator before; notably, in Chapter 7 we remarked that Castellani's equational theory for the recursion-free fragment of BPP_τ includes such an operator (see [Cas88]).

We denote the communication merge operator by $|_c$. It is an operator very similar to the parallel combinator: given processes E and F then $E|_cF$ is a process that behaves like $E|F$ however under the constraint that the first action observed must be a synchronisation between E and F. We assume that BPP_τ is extended with this operator and likewise assume the operational semantics as given by distributed labelled transition graphs to be extended with the following inference rule.

$$\frac{E \xrightarrow{a} (E_1, E_2) \quad F \xrightarrow{\bar{a}} (F_1, F_2)}{E|_c F \xrightarrow{\tau} (E_1|F_1, E_2|F_2)}$$

We note that the communication merge operator only serve auxiliary purposes and that it is not part of the notion of standard form and full standard form.

Definition A.1 *A* BPP$_\tau$ *expression E is in* normal form (nf) *iff* $E = \sum_{i=1}^{n} E_i \lfloor F_i$ *such that F_i is in normal form and E_i takes one of the following three forms:*

- *a variable X_i,*
- *an expression $X_i|_c G_i$ (or $G_i|_c X_i$) where G_i is either a variable Y_i or a prefix $\mu_i H_i$ with H_i being in normal form, or*
- *a prefix $\mu_i G_i$ with G_i being in normal form.*

Variables occurring under the first two forms are called front variables. *A* BPP$_\tau$ *expression E is in* guarded normal form (gnf) *iff* $E = \sum_{i=1}^{n} (\mu_i E_i) \lfloor F_i$ *such that E_i is in normal form and F_i is in guarded normal form.*

We recognise the empty sum as $\mathbf{0}$. We furthermore ignore the ordering of expressions in sums, hence defining the notion of normal form modulo commutativity and associativity of summation. Notice that an expression which is in gnf also is in nf.

The first result of this appendix shall state that any process of BPP$_\tau$ can be effectively transformed into an equivalent process (wrt distributed bisimulation equivalence) in normal form. Thus we must provide a system to *rewrite* any expression into normal form. This is the purpose of the axiom system $\mathfrak{C}$ presented in Table A–1.

The axiom system $\mathfrak{C}$ or more correctly the equational theory $\mathfrak{C}$ consists of the axioms of Table A–1 together with laws for equational reasoning, i.e. the laws of reflexivity, symmetry and transitivity in addition to the laws of congruence wrt all the process constructions. If E and F are process expressions then by $\mathfrak{C} \vdash E = F$

S1	$E + (F + G)$	$=$	$(E + F) + G$
S2	$E + F$	$=$	$F + E$
S3	$E + \mathbf{0}$	$=$	E
S4	$E + E$	$=$	E
LM1	$E\vert F$	$=$	$E\lfloor F + F\lfloor E + E\vert_c F$
LM2	$(E + F)\lfloor G$	$=$	$E\lfloor G + F\lfloor G$
LM3	$(E\lfloor F)\lfloor G$	$=$	$E\lfloor(F\vert G)$
LM4	E	$=$	$E\lfloor \mathbf{0}$
LM5	$\mathbf{0}\lfloor E$	$=$	$\mathbf{0}$
CM1	$E\vert_c F$	$=$	$F\vert_c E$
CM2	$E\vert_c \mathbf{0}$	$=$	$\mathbf{0}$
CM3	$(E + F)\vert_c G$	$=$	$E\vert_c G + F\vert_c G$
CM4	$E\vert_c F\vert_c G$	$=$	$\mathbf{0}$
CM5	$(E\lfloor F)\vert_c G$	$=$	$(E\vert_c G)\lfloor F$
CM6	$\mu E\vert_c \nu F$	$=$	$\begin{cases} \mathbf{0} & \text{if } \mu \neq \bar{\nu},\ \mu = \tau, \text{ or } \nu = \tau \\ \tau(E\vert F) & \text{otherwise} \end{cases}$

Table A–1: Equational theory $\mathfrak{C}$.

or equivalently $E =_{\mathfrak{C}} F$ we shall denote the fact that E and F are *provably equal* using the equational theory $\mathfrak{C}$.

It is a routine matter to verify that all the axioms of Table A–1 are sound wrt distributed bisimilarity. Hence, if $E =_{\mathfrak{C}} F$ for BPP_τ processes E and F then we also have $E \sim_d F$. The equational theory is almost identical to Castellani's theory for distributed bisimulation equivalence on the recursion-free fragment of BPP_τ (see [Cas88]). Notably, we have included axiom CM4 which is not part of Castellani's equational theory. However, it is included only due to convenience, since expressions of the form $E|_c F|_c G$ could be dealt with using structural induction (and relying on the other axioms). This is possible since we are only interested in *processes* of BPP_τ, i.e. all variables have defining equations.

In our proofs we shall mainly use the axioms of Table A–1 as rewrite rules in the direction from left to right. In order for our inductive arguments to proceed correctly we need a nontrivial complexity measure on the structure of BPP_τ expressions.

Definition A.2 *Let $\mathcal{A}$ be a measure on BPP_τ expressions defined by*

- $\mathcal{A}(\mathbf{0}) = \mathcal{A}(X) = 1$,
- $\mathcal{A}(\mu E) = 1 + \mathcal{A}(E)$,
- $\mathcal{A}(E + F) = \mathcal{A}(E|_c F) = \mathcal{A}(E) + \mathcal{A}(F)$, *and*
- $\mathcal{A}(E|F) = \mathcal{A}(E \lfloor F) = 1 + \mathcal{A}(E) + \mathcal{A}(F)$.

Furthermore, based on $\mathcal{A}$, we define the measure $\mathcal{B}$ by

$$\mathcal{B}(E) \quad = \quad \begin{cases} \mathcal{A}(F) & \text{if } E = F \lfloor G \\ \mathcal{A}(E) & \text{otherwise.} \end{cases}$$

Finally, let the complexity measure $\mathcal{E}$, that we wish to base our inductive arguments on, be

$$\mathcal{E}(E) \quad = \quad \big(\mathcal{A}(E), \mathcal{B}(E)\big),$$

and let $<$ be the lexicographical ordering on $I\!N \times I\!N$.

$\mathcal{E}(E\lfloor F)$	$<$	$\mathcal{E}(E\vert F)$
$\mathcal{E}(F\lfloor E)$	$<$	$\mathcal{E}(E\vert F)$
$\mathcal{E}(E\vert_c F)$	$<$	$\mathcal{E}(E\vert F)$
$\mathcal{E}(E\lfloor(F\vert G))$	$<$	$\mathcal{E}((E\vert F)\lfloor G)$
$\mathcal{E}(F\lfloor(E\vert G))$	$<$	$\mathcal{E}((E\vert F)\lfloor G)$
$\mathcal{E}((E\vert_c F)\lfloor G)$	$<$	$\mathcal{E}((E\vert F)\lfloor G)$
$\mathcal{E}(E\vert F)$	$<$	$\mathcal{E}(\mu E\vert_c \nu F)$
$\mathcal{E}(E\lfloor(F\vert G))$	$<$	$\mathcal{E}((E\lfloor F)\lfloor G)$
$\mathcal{E}((E\vert_c G)\lfloor F)$	$<$	$\mathcal{E}((E\vert F)\vert_c G)$
$\mathcal{E}((F\vert_c G)\lfloor E)$	$<$	$\mathcal{E}((E\vert F)\vert_c G)$
$\mathcal{E}((E\vert_c G)\lfloor F)$	$<$	$\mathcal{E}((E\lfloor F)\vert_c G)$
$\mathcal{E}((E\vert_c G)\lfloor(F\vert H))$	$<$	$\mathcal{E}(((E\vert F)\vert_c G)\lfloor H)$
$\mathcal{E}((F\vert_c G)\lfloor(E\vert H))$	$<$	$\mathcal{E}(((E\vert F)\vert_c G)\lfloor H)$
$\mathcal{E}((E\vert_c G)\lfloor(F\vert H))$	$<$	$\mathcal{E}(((E\lfloor F)\vert_c G)\lfloor H)$

Table A–2: Requirements on the complexity measure $\mathcal{E}$.

We require of $\mathcal{E}$ that it satisfies standard inequalities on the substructure of expressions, i.e. inequalities like $\mathcal{E}(E) < \mathcal{E}(E|F)$. But in addition we require that $\mathcal{E}$ satisfies the requirements of Table A–2. We shall also be using that from $\mathcal{A}(E) < \mathcal{A}(F)$ we have $\mathcal{E}(E\lfloor G) < \mathcal{E}(F\lfloor G)$ as well as $\mathcal{E}(E|_c G) < \mathcal{E}(F|_c G)$. By inspection all these requirements on $\mathcal{E}$ are easily seen to be satisfied.

Lemma A.3 *Let E be a process of* BPP_τ. *Then we can construct another process F of* BPP_τ *in* nf *such that $E =_{\mathfrak{E}} F$.*

Proof: We proceed by case analysis on the structure of E using induction on $\mathcal{E}(E)$. We shall only demonstrate the most involved case of $E = E_1\lfloor E_2$. We proceed by case analysis on the structure of E_1.

(i) $E_1 = \mathbf{0}$. By axiom LM5 we have $E =_{\mathfrak{c}} \mathbf{0}$. Hence define F to be $\mathbf{0}$ and we clearly have $E =_{\mathfrak{c}} F$.

(ii) $E_1 = X$. By induction let F_2 be a process in nf such that $E_2 =_{\mathfrak{c}} F_2$ and define F to be $X \lfloor F_2$. By equational reasoning we have $E =_{\mathfrak{c}} F$ as required.

(iii) $E_1 = \mu E_{11}$. By induction let F_{11}, F_2 be processes in nf such that $E_{11} =_{\mathfrak{c}} F_{11}$ and $E_2 =_{\mathfrak{c}} F_2$. Define F to be $\mu F_{11} \lfloor F_2$. By equational reasoning we have $E =_{\mathfrak{c}} F$ as required.

(iv) $E_1 = E_{11} + E_{12}$. By axiom LM2 we have $E =_{\mathfrak{c}} E_{11} \lfloor E_2 + E_{12} \lfloor E_2$. From the induction hypothesis we have processes F_1, F_2 in nf such that $E_{11} \lfloor E_2 =_{\mathfrak{c}} F_1$ and $E_{12} \lfloor E_2 =_{\mathfrak{c}} F_2$. Then define F to be $F_1 + F_2$. By equational reasoning we have $E =_{\mathfrak{c}} F$ as required.

(v) $E_1 = E_{11}|E_{12}$. We have $E =_{\mathfrak{c}} E_{11} \lfloor (E_{12}|E_2) + E_{12} \lfloor (E_{11}|E_2) + (E_{11}|_c E_{12})|E_2$ from axiom LM1,LM2 and LM3. By induction let F_1, F_2 and F_3 be processes in nf such that $E_{11} \lfloor (E_{12}|E_2) =_{\mathfrak{c}} F_1$, $E_{12} \lfloor (E_{11}|E_2) =_{\mathfrak{c}} F_2$ and finally $(E_{11}|_c E_{12})|E_2 =_{\mathfrak{c}} F_3$. Then define F to be $F_1 + F_2 + F_3$ and $E =_{\mathfrak{c}} F$ follows from equational reasoning.

(vi) $E_1 = E_{11} \lfloor E_{12}$. From axiom LM3 we have $E =_{\mathfrak{c}} E_{11} \lfloor (E_{12}|E_2)$. By induction let F be a process in nf such that $E_{11} \lfloor (E_{12}|E_2) =_{\mathfrak{c}} F$. Then we also have $E =_{\mathfrak{c}} F$ as required.

(vii) $E_1 = E_{11}|_c E_{12}$. From the following Lemma A.4 we obtain a process F in nf such that $E =_{\mathfrak{c}} F$ as required.

We note that in each of the above cases the process F constructed is in nf (modulo axiom S3). This completes the proof. ■

Lemma A.4 *Let E be a* BPP_τ *of the form $(E_1|_c E_2) \lfloor E_3$ for processes E_1, E_2 and E_3. Then we can construct a* BPP_τ *process F in* nf *such that $E =_{\mathfrak{c}} F$.*

Proof: We proceed by case analysis on the structure of E_1 and E_2 using induction on $\mathcal{E}(E)$.

(i) $E_1 = \mathbf{0}$. From axiom CM1, CM2 and LM5 we have $E =_{\mathfrak{c}} \mathbf{0}$. Hence define F to be $\mathbf{0}$ and $E =_{\mathfrak{c}} F$ clearly follows.

(ii) $E_1 = E_{11} + E_{12}$. We have $E =_{\mathfrak{c}} (E_{11}|_c E_2)\lfloor E_3 + (E_{12}|_c E_2)\lfloor E_3$ from axiom CM3 and LM2. From the induction hypothesis we find processes F_1 and F_2 in nf such that $(E_{11}|_c E_2)\lfloor E_3 =_{\mathfrak{c}} F_1$ and $(E_{12}|_c E_2)\lfloor E_3 =_{\mathfrak{c}} F_2$. Then define F to be $F_1 + F_2$ and $E =_{\mathfrak{c}} F$ follows from equational reasoning.

(iii) $E_1 = E_{11}|E_{12}$. We get $E =_{\mathfrak{c}} (E_{11}|_c E_2)\lfloor(E_{12}|E_3) + (E_{12}|_c E_2)\lfloor(E_{11}|E_3)$ from axiom LM1, LM3, CM3, CM4 and CM5. By induction let F_1, F_2 be processes in nf with $(E_{11}|_c E_2)\lfloor(E_{12}|E_3) =_{\mathfrak{c}} F_1$ and $(E_{12}|_c E_2)\lfloor(E_{11}|E_3) =_{\mathfrak{c}} F_2$. Now define F to be $F_1 + F_2$ and we have $E =_{\mathfrak{c}} F$ by equational reasoning.

(iv) $E_1 = E_{11}\lfloor E_{12}$. We have $E =_{\mathfrak{c}} (E_{11}|_c E_2)\lfloor(E_{12}|E_3)$ from axiom CM5 and LM3. From the induction hypothesis let F be a process in nf such that $(E_{11}|_c E_2)\lfloor(E_{12}|E_3) =_{\mathfrak{c}} F$. Clearly, we also have $E =_{\mathfrak{c}} F$.

(v) $E_1 = E_{11}|_c E_{12}$. From axiom CM4 and LM5 we have $E =_{\mathfrak{c}} \mathbf{0}$. Let F be $\mathbf{0}$ and we have $E =_{\mathfrak{c}} F$ as required.

(vi) $E_1 = X$. We proceed by case analysis on the structure of E_2. Because of axiom CM1 and the analysis above we only need to consider the following two cases.

   (a) $E_2 = Y$. By induction let F_3 be a process in nf such that $E_3 =_{\mathfrak{c}} F_3$ and define F to be $(X|_c Y)\lfloor F_3$. We then have $E =_{\mathfrak{c}} F$ as required.

   (b) $E_2 = \nu E_{21}$. By induction let F_{21} and F_3 be processes in nf such that $E_{21} =_{\mathfrak{c}} F_{21}$ and $E_3 =_{\mathfrak{c}} F_3$. Then define F to be $(X|_c \nu F_{21})\lfloor F_3$. By equational reasoning we then have $E =_{\mathfrak{c}} F$ as required.

(vii) $E_1 = \mu E_{11}$. We proceed by case analysis on the structure of E_2. Because of axiom CM1 and the analysis above we only need to consider the case of $E_2 = \nu E_{21}$. Firstly, if $\mu \neq \bar{\nu}$, or $\mu = \tau$, or $\nu = \tau$, then from axiom CM6 together with LM5 we have $E =_{\mathfrak{c}} \mathbf{0}$. Thus let F be $\mathbf{0}$ and we have

$E =_{\mathfrak{C}} F$ as required. Finally, if $\mu = \bar{\nu}$ and $\mu, \nu \neq \tau$ then from CM6 we have $E =_{\mathfrak{C}} \tau(E_{11}|E_{21})\lfloor E_3$. By induction let F_1 and F_2 be processes in nf such that $E_{11}|E_{21} =_{\mathfrak{C}} F_1$ and $E_3 =_{\mathfrak{C}} F_2$. Define F to be $(\tau F_1)\lfloor F_2$ and we have $E =_{\mathfrak{C}} F$ by equational reasoning.

We note that in each of the above cases the process F constructed is in nf (modulo axiom S3). This completes the proof. ■

Corollary A.5 *Suppose E is a guarded* BPP_τ *process. Then we can construct another process F of* BPP_τ *in* gnf *such that $E =_{\mathfrak{C}} F$.*

Proof: Follows readily from Lemma A.3 ■

Next we shall use the notion of normal form as a step towards obtaining *guarded standard form* for finite families of guarded BPP_τ process equations wrt distributed bisimilarity. For a definition of guarded standard form we refer the reader to Definition 2.28 (see page 45).

Proposition A.6 *Given any finite family of guarded* BPP_τ *equations Δ we can* effectively *construct another finite family of* BPP_τ *equations Δ' in guarded standard form of degree 3 in which $\Delta \sim_d \Delta'$.*

Proof: Let $\Delta = \{X_i \stackrel{\mathrm{def}}{=} E_i : 1 = 1, 2 \ldots n\}$ be a finite family of guarded BPP_τ process equations. By Corollary A.5 we can assume that each process E_i is in gnf.

We shall define a sequence $\Delta_0, \Delta_1 \ldots \Delta_k$ of finite families of BPP_τ process equations beginning with Δ, ending with Δ', and constructed in such a way that distributed bisimilarity is preserved along the way, i.e. for $i = 1, 2 \ldots k$ we have $\Delta_{i-1} \sim_d \Delta_i$. Furthermore, each Δ_i is divided disjointly into Δ_i^+ and Δ_i^- where

(i) Δ_i^+ contains equations in guarded standard form of degree 3, and

(ii) Δ_i^- contains defining equations $X \stackrel{\mathrm{def}}{=} E$ with E being in nf and a *proper* subexpression of some equation of Δ_{i-1}^- (this will assure us that the sequence of transformations $\Delta_0, \Delta_1 \ldots$ eventually comes to an end).

Initially, we define Δ_0^+ to be $\emptyset$ (empty) while Δ_0^- is defined to be all of Δ thus indicating that none of the equations are yet in standard form and that all equations still have to be transformed. We first describe Δ_1 in terms of Δ_1^+ and Δ_1^-; later we shall give the general definition of Δ_i based on that of Δ_{i-1}.

For each defining equation $X \stackrel{\text{def}}{=} E$ of Δ_0^- we form a new defining equation for X in standard form of degree 3 and add it to Δ_1^+. Along the way we shall also describe the defining equations which form Δ_1^-. So initially both Δ_1^+ and Δ_1^- are defined to be $\emptyset$. As E is in gnf we know that each summand of E takes the form $(\mu E_1)\lfloor E_2$ where E_1 is in nf while E_2 is in gnf. We now replace the summand $(\mu E_1)\lfloor E_2$ of E by $(\mu X_{E_1})\lfloor X_{E_2}$ for *new* variables X_{E_1} and X_{E_2} while at the same time adding the two defining equations $X_{E_1} \stackrel{\text{def}}{=} E_1$ and $X_{E_2} \stackrel{\text{def}}{=} E_2$ to Δ_1^-. Performing this transformation for every summand of E we will eventually end up with a defining equation for X which is in standard form of degree 3. We now add this equation to Δ_1^+. Finally, we carry out this transformation for every equation of Δ_0^- thus defining Δ_1^+ and Δ_1^- which are also seen to satisfy the requirements of (i) and (ii) above. Furthermore, $\Delta_0 \sim_d \Delta_1$ since all we have been doing is introducing new variables that rename expressions.

We shall next give the general definition of Δ_i in terms of Δ_{i-1}. By construction of Δ_{i-1} we know that it consists of Δ_{i-1}^+ and Δ_{i-1}^- both satisfying the above clauses (i) and (ii). Initially, let Δ_i^+ be given by Δ_{i-1}^+ and Δ_i^- by $\emptyset$. As in the construction of Δ_1 from Δ_0 we shall transform each equation of Δ_{i-1}^- into an equation which can be added to Δ_i^+ while at the same time possibly introducing new variables which become the defining variables of equations of Δ_i^-. However, now we must take greater care since the structure of equations of Δ_{i-1}^- can be more complex than those of Δ_0^-.

Let $X \stackrel{\text{def}}{=} E$ be an equation of Δ_{i-1}^-. We know that the defining expression E is in nf so it contains summands of the following three forms: either a summand is $(\mu E_1)\lfloor E_2$ with E_1 and E_2 being in nf, or $(X\lfloor_c F_1)\lfloor F_2$ with F_2 being in nf and F_1 being either a variable Y or a prefix νF_{11} where F_{11} is in nf, or finally $Z\lfloor G$ with G being in nf. For a summand of the first type, i.e. $(\mu E_1)\lfloor E_2$, we replace it by $(\mu X_{E_1})\lfloor X_{E_2}$ while introducing new variables X_{E_1} and X_{E_2}, and adding $X_{E_1} \stackrel{\text{def}}{=} E_1$

and $X_{E_2} \stackrel{\mathrm{def}}{=} E_2$ to Δ_i^- as described in the construction of Δ_1 from Δ_0.

For a summand of the second type, i.e. $(X\lfloor_c F_1)\lfloor F_2$, we proceed as follows. First of all, assume that F_1 is a variable Y (the case of F_1 being a prefix follows from similar but simpler arguments). As $(X\lfloor_c Y)\lfloor F_2$ is a subexpression of equations of Δ we know that X and Y belong to $Var(\Delta)$ and must have defining equations of the following form (these equations were constructed as part of Δ_1^+)

$$X \stackrel{\mathrm{def}}{=} \sum_i (\mu_i X_i)\lfloor X_i',$$

$$Y \stackrel{\mathrm{def}}{=} \sum_j (\nu_j Y_j)\lfloor Y_j'.$$

For X and Y in $(X\lfloor_c Y)\lfloor F_2$ we substitute their defining expressions. By using axiom CM1,CM3,CM5,CM6 together with LM2 and LM3 of Table A–1 we can transform the resulting expression into

$$\sum_{\mu_i=\bar{\nu}_j\neq\tau} \tau(X_i|Y_j)\lfloor(X_i'|Y_j'|X_{F_2}),$$

where X_{F_2} is a new variable and $X_{F_2} \stackrel{\mathrm{def}}{=} F_2$ is added to Δ_i^-. We now replace the summand $(X\lfloor_c Y)\lfloor F_2$ by the above expression.

Finally, for a summand of the form $Z\lfloor G$ we again have $Z \in Var(\Delta)$ so its defining equation must be part of Δ_1^+. Hence

$$Z \stackrel{\mathrm{def}}{=} \sum_i (\mu_i X_i)\lfloor Y_i.$$

For Z in the summand $Z\lfloor G$ we now substitute the above defining expression. By using axiom LM2 and LM3 of Table A–1 we can transform the resulting expression into

$$\sum_i (\mu_i X_i)\lfloor(Y_i|X_G),$$

where X_G is a new variable with defining equation $X_G \stackrel{\mathrm{def}}{=} G$ which is added to Δ_i^-. We now replace the summand $Z\lfloor G$ by the above expression.

Performing this transformation for every summand of the defining expression E of $X \stackrel{\mathrm{def}}{=} E$ we see that eventually X will have a defining equation in standard form of degree 3 which we then add to Δ_i^+. We carry out this transformation for each defining equation of Δ_{i-1}^{-} in order to form Δ_i^{+} which satisfies the requirements of (i). We also see that each new equation of Δ_i^{-} satisfies the requirements of (ii). Finally, as our transformation from Δ_{i-1} to Δ_i only involves the introduction of new variables that rename expressions or involves equational reasoning using the axiom system $\mathfrak{C}$ we clearly have the relationship $\Delta_{i-1} \sim_d \Delta_i$ as required. This completes the proof. ∎

In [Cas88] it is remarked that particularly one axiom emphasises the difference between distributed bisimilarity and bisimilarity, viz. the axiom $(\mu E)\lfloor F = \mu(E|F)$ which is valid wrt bisimilarity but *not* wrt distributed bisimilarity. Using this axiom together with Proposition A.6 we immediately see that any finite family Δ of guarded BPP$_\tau$ equations can be effectively presented in full standard form of degree 6 while preserving bisimilarity (see Definition 2.30, page 46 for the notion of full standard form). However, we can do better than this.

Proposition A.7 *Given any finite family of guarded* BPP$_\tau$ *equations* Δ *in full standard form of degree* d *for some* $d > 2$ *we can* effectively *construct another finite family of* BPP$_\tau$ *equations* Δ' *in full standard form of degree* 2 *in which* $\Delta \sim \Delta'$.

Proof: Let Δ be a finite family of process equations in full standard form of degree d for some $d > 2$. Via an intermediate family Δ'' which will also be in full standard form of degree d we shall transform Δ into Δ' with $\Delta \sim \Delta'$ such that Δ' is in full standard form of degree $\lceil \frac{d}{2} \rceil$ if d is odd and $\frac{d}{2} + 1$ if d if even.

First we describe the transformation from Δ to Δ''. For each pair of variables X and Y of $Var(\Delta)$ we introduce a new so-called pairing variable U_{XY}. We now replace every occurrence of $X|Y$ in each equation of Δ by U_{XY} thus obtaining equations in full standard form of degree $\lceil \frac{d}{2} \rceil$. These equations are now added to Δ''. However, in order to describe Δ'' completely we also need to give defining equations for the pairing variables.

For each new variable U_{XY} we introduce the equation $U_{XY} = X|Y$. Suppose X and Y have been assigned the expressions $\sum_i \mu_i \alpha_i$ and $\sum_j \nu_j \beta_j$ relative to Δ''

and hence with $size(\alpha_i), size(\beta_j) \leq \lceil \frac{d}{2} \rceil$. We now substitute these expressions for X and Y in the equation for U_{XY} thus obtaining

$$\begin{aligned} U_{XY} &= X|Y \\ &= X\lfloor Y + Y\lfloor X + X|_c Y \\ &= \sum_i \mu_i(\alpha_i|Y) + \sum_j \nu_j(\beta_j|X) + \sum_{\mu_i = \bar{\nu}_j \neq \tau} \tau(\alpha_i|\beta_j) \end{aligned} \qquad (\star)$$

The above equation is in full standard form. But the degree is not as small as we would like it to be and we possibly have to transform the last expression of $(\star)$ even further before obtaining the defining expression for U_{XY}.

Let us first determine the size of $\alpha_i|\beta_j$, i.e. expressions that belong to the third group of summands of $(\star)$. If d is even then each of α_i and β_j can have size at most $\frac{d}{2}$ and hence $\alpha_i|\beta_j$ is of size at most d. However, if d is odd then the size of $\alpha_i|\beta_j$ may be $d+1$. But in this case $\alpha_i|\beta_j$ will contain two variables of $Var(\Delta)$ and therefore by commutativity and associativity of parallel composition these two variables can be substituted by the appropriate pairing variable. Thus in any case we can make the size not bigger than d. Notice that the expressions α_i and β_j that form the third group of summands of $(\star)$ are all prefixed by actions *different* from τ in the equations for X and Y.

Let us now determine the size of $\alpha_i|Y$ (or $\beta_j|X$) thus getting a measure on the size of the first and second group of summands of $(\star)$. If d is even then the size of $\alpha_i|Y$ cannot exceed $\frac{d}{2}+1$. If d is odd then the expression $\alpha_i|Y$ may contain two variables of $Var(\Delta)$ hence by commutativity and associativity of parallel composition these two variables can be replaced by the appropriate pairing variable. We therefore obtain an expression of size not bigger that $\lceil \frac{d}{2} \rceil$. So in any case the first (and second) group of summands of $(\star)$ can be made into full standard form with a degree strictly less than d.

Carrying out the above we obtain a defining equation for each pairing variable which we now add to Δ''. We see that Δ'' is in full standard form of degree d and that $\Delta \sim \Delta''$. Furthermore, in every equation of Δ'' any summand of the form $a\alpha$ for some atomic action a, i.e. a summand being prefixed by an action different from τ, will satisfy $size(\alpha) \leq \lceil \frac{d}{2} \rceil$ if d is odd and $size(\alpha) \leq \frac{d}{2}+1$ if d is even.

Starting with Δ'' we now repeat the whole transformation once more thus obtaining a family Δ' of process equations in which the equations of Δ'' have been transformed into full standard form of degree $\lceil \frac{d}{2} \rceil$. However, the pairing variables introduced in the transformation from Δ'' to Δ' will be assigned expressions in full standard form of degree only $\lceil \frac{d}{2} \rceil$ in case d is odd and of degree only $\frac{d}{2} + 1$ in case d is even. This completes the proof as we now have obtained a family Δ' in full standard form with a degree strictly less that d and such that $\Delta \sim \Delta'$. $\blacksquare$