On Modal Characterizations and Turning GSOS Rules Into Equations

Maciej Gazda
Vrije Universiteit, Amsterdam, The Netherlands
Jagiellonian University, Kraków, Poland
prostyprex@gmail.com

Master Thesis in Computer Science
written under the supervision of:

Prof. Dr. Wan Fokkink
Dr hab. Wit Foryś
Dr. Femke van Raamsdonk

December 2008

Abstract

In this thesis properties of various process equivalences are analysed, specifically definability with finite HML formulas, soundness of the Approximation Induction Principle and properties of processes that are regular with respect to a given equivalence. Also it is explained how to adapt the existing axiomatisation strategy for bisimulation so that it works for other common equivalences as well. Futhermore, term rewriting properties of the obtained axiomatisations are studied.

KEYWORDS: Approximation Induction Principle, axiomatisation, comparative concurrency semantics, Hennessy-Milner logic, term rewriting

Contents

Chapter 1

Introduction

Issues covered in this thesis are concerned with what is called comparative concurrency semantics. This discipline, among other problems, deals with classification of various notions of equivalence between processes in the concurrency theory. More specifically, I work in the setting of process algebras where processes are expressed as terms over some signature. They are considered modulo certain equivalences which can be defined using modal characterizations. Namely, we define a language of modal formulas and identify processes that satisfy the same formulas from the language. I am mostly interested in deciding equality between two process terms, preferably using the axiomatic approach and determining properties of whole classes of process semantics (definability using modal characterization, compositionality with respect to some basic operators and other "sanity" properties). I will focus on concrete process semantics, which does not take into account the silent step τ. I will also restrict myself to finitely branching processes.

Perhaps the best way to give an overview of the thesis is to tell how it evolved. Initially, I wanted to investigate the possibility of generating complete axiomatisations for some common process equivalences and arbitrary operators defined with structural operational semantics (using the GSOS format). The idea stems from the paper from Aceto, Bloom and Vaandrager [2] where an axiomatisation strategy has been presented for bisimulation equivalence. The problem basically boils down to adapting their strategy into other settings. I soon discovered that this has already been solved in most of the cases, although the different results have not been merged in one paper. Therefore, this part of my work (which is done in Chapter 3) was rather a presentation of what has been achieved so far, relating a few facts from already existing papers and filling some gaps. As a side topic, a question about the Approximation Induction Principle (AIP) emerged. This infinitary conditional equation allows to decide equality between processes whose finite projections are all equal. It seems to be common knowledge that one can apply this rule for any sensible process equivalence provided that we consider only finitely branching processes. I wanted to obtain a clear and fairly general condition guaranteeing that the Approximation Induction Principle would hold.

Another issue was to gain an insight about term rewriting properties of the obtained axiomatisations. This has already been done in case of bisimulation by Bosscher in [12]. His results are valuable also when we consider other equivalences; there is a straightforward extension of his term rewriting strategy so that every term can be rewritten to a normal form. However, I have found that in some cases the obtained term rewriting systems are not confluent, Moreover, there is little or no hope to find such.

During this research I noticed that basic process equivalences have some common properties which have been proved for each equivalence separately and there is a lack of more general statements and proof methods. Since all these equivalences have corresponding modal characterizations with subsets of Hennessy-Milner logic, the natural way of deriving properties of whole classes of process equivalences is to impose restrictions on the modal languages that define them. This approach has resulted in obtaining a sufficient condition for an equivalence to be determined by finite HML formulas, soundness of AIP and also properties of processes that are regular with respect to various equivalences. I think it might be interesting to further apply this reasoning in order to obtain certain congruence results and discuss what are the minimum "sanity" restrictions for a process equivalence (for example congruence w.r.t. basic operations).

The reader is assumed to have some background in process algebra and term rewriting, although I will provide all the necessary definitions concering the former field. As for term rewriting, the reader may consult [4] or simply the Appendix A of [13] for the most basic notions.

1.1 Labelled transition systems and process equivalences

The basic notion that will be used to specify a system is a labelled transition system. This model is more general than finite automata in the sense that it allows an arbitrary set (or even a class [15]) of states. In general, there is no boundary on the cardinality of outgoing transitions from a given state as well. However, I will restrict myself to the finitely branching labelled transition systems with a finite set of transition labels (actions).

Definition (Labelled transition systems)

Assume a set P of processes and a finite set Act of actions. A *labelled transition system* is a pair $(P, \rightarrow)$ where P is a set of processes and $\rightarrow \subseteq P \times Act \times P$ a transition relation. As a standard convention, we will use $p \xrightarrow{a} q$ to denote $(p, a, q) \in \rightarrow$ (*positive literal*) and $p \overset{a}{\not\rightarrow}$ for $\neg\exists q \in P : p \xrightarrow{a} q$ (*negative literal*). We extend the transition relation such that it can be labelled also with traces, namely we define $p \xrightarrow{\epsilon} p$ and $p \xrightarrow{a\sigma} q$ for $\sigma \in Act^*$ iff $\exists r : p \xrightarrow{a} r \wedge r \xrightarrow{\sigma} q$. For a $q \in P$ such that $\exists \sigma : p \xrightarrow{\sigma} q$ we say that q is *reachable* from p. We will also write $I(p) \stackrel{def}{=} \{a \in Act \mid \exists_{q \in P}\ p \xrightarrow{a} q\}$ for the set of *initial actions (initials)* that process p can take in the first step.

An important question is when we should consider two processes undistinguishable. There are many ways to define equivalences between processes. In general, the finer an equivalence is (having more equivalence classes), the more features of a process can be observed. On the other hand, since applied process algebra deals with modelling and verification of complex systems where state spaces tend to grow large, we would require as many identifications as possible in order to make the verification algorithms efficient.

I have mentioned a difference between labelled transiton systems and the earlier "classical" finite automata theory. Another difference is that in our model we usually need finer equivalences which not only take into account execution traces but also either branching structure of a process or at least some additional information about traces (decorated trace semantics like failure or ready trace). Below, I will give a short overview of the most common process equivalences to which I will refer as basic process equivalences throughout the paper; see [15] for more details.

Trace equivalence. One of the earliest notions of equivalence between processes and also the coarsest one (having the least number of equivalence classes). Processes are considered *trace equivalent* if they execute exactly the same traces of actions. Formally, $\sigma \in Act^*$ is a *trace* of a process p if $\exists q : p \xrightarrow{\sigma} q$. The set of all traces of p is denoted $T(p)$ and p and q are *trace equivalent* ($p =_T q$) iff $T(p) = T(q)$.

Completed trace equivalence. In addition to trace equivalence, we distinguish processes according to execution paths that lead to termination. Let $CT(p) = \{\sigma \in Act^* \mid \exists q : p \xrightarrow{\sigma} q \wedge I(q) = \emptyset\}$. Processes p and q are *completed trace equivalent* ($p =_{CT} q$) iff $T(p) = T(q)$ and $CT(p) = CT(q)$.

This equivalence resembles language equivalence on finite automata, if we assume that all states without outgoing transitions are the terminating (or "accepting") states. However we also have to take into account partial traces which do not lead into terminating states, because otherwise we could not distinguish processes with infinite execution paths.

Failures equivalence. Apart from mere trace we also take into account all subsets of actions that cannot be taken after executing a certain trace. The set of *failure pairs* of p is defined as

$$F(p) = \{(\sigma, X) \mid \sigma \in Act^* \wedge X \subseteq Act \wedge \exists q : (p \xrightarrow{\sigma} q \wedge I(q) \cap X = \emptyset)\}$$

Failures equivalence is defined as: $p =_F q$ iff $F(p) = F(q)$.

This equivalence plays a crucial role in a model for Hoare's CSP language which substituted an earlier one based on the trace equivalence.

Readiness. This equivalence is based on a similar idea as failures equivalence, but now we take into account the set of actions that can be taken after executing a certain trace. We define a set of *ready pairs* of p as:

$$R(p) = \{(\sigma, X) \mid \sigma \in Act^* \wedge X \subseteq Act \wedge \exists q : (p \xrightarrow{\sigma} q \wedge I(q) = X\ )\ \}.$$

Processes p and q are *readiness equivalent*, notation $p =_R q$, iff $R(p) = R(q)$.

Observe that the real difference between failures and readiness equivalence is not simply the one that would be indicated by their names. The crucial thing is that in case of failures we allow all the subsets of $Act \setminus I(p)$ and therefore the presence of a failure pair (σ, X) does not imply that there is a state q with

$p \xrightarrow{\sigma} q$ and $I(q) = Act \setminus X$. This is the reason why readiness equivalence is strictly coarser than failures ($=_F \subset =_R$).

Failure trace. This equivalence also distinguishes more processes than failures equivalence. It follows another direction than readiness by taking into account a subset of forbidden actions in all steps of a trace. Therefore, a *failure trace* is an alternating sequence of failure subsets and actions. We define:
$FT(p) = \{X_0 a_1 X_1 ... a_n X_n \mid \exists p_0, ..., p_n : p = p_0 \wedge p_{i-1} \xrightarrow{a_i} p_i \text{ for } i = 1, \ldots, n \wedge \forall_{i \in \{0,\ldots,n\}} I(p_i) \cap X_i = \emptyset \}$.

Processes p and q are *failure trace equivalent* ($p =_{FT} q$) iff $FT(p) = FT(q)$.

Ready trace. A combination of ideas from the previous two equivalences is known as ready trace equivalence. This time actions are interleaved with sets of inital actions of a state:
$RT(p) = \{X_0 a_1 X_1 ... a_n X_n \mid \exists p_0, \ldots, p_n : p = p_0 \wedge p_{i-1} \xrightarrow{a_i} p_i \text{ for } i = 1, \ldots, n \wedge \forall_{i \in \{0,\ldots,n\}} X_i = I(p_i)\}$. Processes p and q are *ready trace equivalent* ($p =_{RT} q$) iff $RT(p) = RT(q)$.

This equivalence has several interesting properties. As well as failure trace it is compositional with respect to the priority operator θ [6]. Moreover, as will be shown in Chapter 4, if we turn its axiomatisation into a term rewriting system, the resulting rulified axiomatisation is terminating and confluent for well-founded terms.

Simulation. Previous equivalences are examples of decorated trace semantics, where we take into account traces of actions, possibly interleaved with information about initial or forbidden action . A different approach is based on the notion of simulation. We say that $S \subseteq P \times P$ is a *simulation relation* iff:

$$pSq \wedge p \xrightarrow{a} p' \Rightarrow \exists q' : q \xrightarrow{a} q' \wedge p' S q'$$

If there is a simulation relation S such that pSq and a simulation relation R with qRp, then p and q are *similar* ($s =_S q$). Simulation relation is finer than trace equivalence and independent of all decorated trace semantics.

Ready simulation. This equivalence is finer than all the aforementioned semantics. This time, we define *ready simulation* relation which is a simulation that satisfies the following additional condition: $pSq \Rightarrow I(p) = I(q)$. Processes p and q are *ready simulation equivalent*, notation $p =_{RS} q$ iff there exists a ready simulation S with pSq and a ready simulation R with qRp.

Bisimulation. The finest of all known "useful" or "sensible" process equivalences. We say that R is a *bisimulation relation* iff:
$pRq \wedge p \xrightarrow{a} p'$ then $\exists q' : q \xrightarrow{a} q' \wedge p' R q'$ and
$pRq \wedge q \xrightarrow{a} q'$ then $\exists p' : p \xrightarrow{a} p' \wedge p' R q'$.

Processes p and q are *bisimilar*, notation $p \leftrightarrow q$, iff there exists a bisimulation relation R such that pRq. In this paper we will consider all bisimilar processes undistinguishable.

Definition (Properties of processes)
A process p is:
- *well-founded* if there is no infinite execution trace starting with p,
- *finitely branching* if for all r reachable from p the set $\{q \mid \exists_{a \in Act}\ r \xrightarrow{a} q\}$ is finite,
- *regular* if p is finitely branching and there are finitely many bisimilar states reachable from p which can be distinguished modulo bisimulation,
- *computable* if p is finitely branching and there exists an algorithm that computes for every state q reachable from p its complete set of outgoing transitions.

Assume from now on that all processes that we consider are finitely branching, unless explicitly stated otherwise.

1.2 Processes as terms and structural operational semantics

We will use the model common for all process algebras where processes are represented as terms over some signature.

Definition (Signature, variables and terms)
Assume a *signature* Σ which is a finite set such that with every $f \in \Sigma$ has been associated a value $ar(f) \in \mathbb{N}$, which is the arity of f. If $ar(f) = 0$ then we will call f a *constant*. Assume further a countably infinite set of variables *Var*. The set $\mathbb{T}(\Sigma)$ of *terms over* Σ is defined inductively:
(1) $x \in \mathbb{T}(\Sigma)$ for every $x \in Var$,
(2) If $f \in \Sigma$ and $t_1, ..., t_{ar(f)}$ are terms then $f(t_1, ..., t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

A term is *closed* if it does not contain any variables. The set of all closed terms over Σ is denoted $\mathcal{T}(\Sigma)$. A term in which all variables belong to a vector of variables $\vec{x}$ is denoted $C[\vec{x}]$.

Definition (Substitution)
A *substitution* is a function $\sigma : Var \to \mathbb{T}(\Sigma)$. If $\sigma(Var) \subseteq \mathcal{T}(\Sigma)$ then σ is called a *closed substitution*. Substitution extends to transitions and any syntactic constructs containing variables in a natural way.

We will work in the setting of labelled transition systems of the form $(\mathbb{T}(\Sigma), \to)$ for some signature Σ of operators on processes. These operators are defined using structural operational semantics with transition system specifications.

Definition (Transition rule / transition system specification (TSS))
A *transition rule* over Σ is an inference rule of the form:

$$\frac{H}{t \xrightarrow{a} t'}$$

where H is a (possibly empty) set of literals called *premises* and the positive literal $t \xrightarrow{a} t'$ is called the *conclusion*. A *transition system specification (TSS)*

is a set of transition rules. A *transition relation* $\rightarrow$ generated by a TSS T contains all rules $t \xrightarrow{a} t'$ such that t and t' are closed terms and there is a rule $\frac{H}{\alpha}$ and a closed substitution σ such that for all positive literals α in $\sigma(H)$, α is generated by T and for all negative literals β in $\sigma(H)$, β is not generated by T.

We will restrict ourselves to an important and widely used class of GSOS rules and TSSs. A transition relation generated by a GSOS system has a nice sanity property: it exists and is always unique, therefore I will not present more advanced tools to obtain a transition, like a well-supported proof.

Definition (GSOS rule / TSS)
A transition rule is in *GSOS format* if it is in the form:

$$\frac{\bigcup_{i=1}^{l}\{x_i \xrightarrow{a_{ij}} y_{ij} | 1 \leq j \leq m_i\} \cup \bigcup_{i=1}^{l}\{x_i \overset{b_{ik}}{\not\rightarrow} | 1 \leq k \leq n_i\}}{f(x_1,...,x_l) \xrightarrow{c} C[\vec{x},\vec{y}]}$$

Where for $X = \{x_i\}$ and $Y = \{y_{ij}\}$ we have $X, Y \subseteq Var$ and $X \cap Y = \emptyset$, and $a_{ij}, b_{ik} \in Act$. If $m_i > 0$ then we say that the rule tests its i-th argument positively. A transition system specification is in GSOS format if all its rules are and there are finitely many of them.

LTSs generated by GSOS systems are also finitely branching and computable. More information about this format can be found in [3] and [11]. GSOS stands for Structural Operational Semantics with Guarded recursion ([11]).

1.3 *FINTREE* and other operators

Now we are ready to define the most basic process operators. The simple algebra *FINTREE* consists of three operators using which we can generate all processes which represented as graphs are finite DAGS. These operators are:

- *action prefix* a for all $a \in Act$, a unary operator which represents execution of a single action followed by the rest of a process; for a process p, ap is a process that first executes a and afterwards proceeds with p. This operator makes the process tree grow one level deeper. Action prefixes are defined with the following rule scheme, one rule for each $a \in Act$:

$$\frac{}{ax \xrightarrow{a} x}$$

- *alternative composition* (choice operator) $+$, a binary operator which represents a choice between two processes and a subsequent execution of one of them. If p and q are processes, then $p + q$ is a process that executes either p or q. Below the alternative composition is defined with two GSOS rules:

$$\frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$$

- a constant 0 which represents a process that cannot execute any action. We will use a as an abbreviation for $a0$ for each $a \in Act$. There are no rules for 0.

We will also use a notation $\Sigma_{i\in I} t_i$ for a finite set of indexes $I = \{i_1, \ldots, i_n\}$ to represent a choice $t_{i_1} + \cdots + t_{i_n}$. For $I = \emptyset$, we define $\Sigma_{i\in\emptyset} = 0$.

Apart from *FINTREE*, I will now define two other important operators. The first one, *parallel composition* $||$, is a crucial operator in process algebra. For two processes p and q, $p||q$ denotes a process that can execute p and q in parallel, which means that it can execute any possible valid interleaving scheme for p and q.

$$\frac{x \xrightarrow{a} x'}{x||y \xrightarrow{a} x'||y} \qquad \frac{y \xrightarrow{a} y'}{x||y \xrightarrow{a} x||y'}$$

Another example of process operators is a family of *one-step encapsulation operators* ∂_B^1 that were used in [2] to obtain a complete axiomatisation for bisimulation equivalence (this issue will be covered in Chapter 3). Process $\partial_B^1(p)$, where $B \subseteq Act$, behaves like p except that in the first step it cannot take any action from the set B. Rules for ∂_B^1 are, for any $a \notin B$:

$$\frac{x \xrightarrow{a} x'}{\partial_B^1(x) \xrightarrow{a} x'}$$

1.4 Axiomatisation of the basic operators

Up to this point we have specified how to define process operators with structural operational semantics and also several notions of equality between processes. The heart of process algebra is another approach: defining operators with axioms and deriving equality between process terms using equational reasoning. Below, I will present axiomatisations of *FINTREE* operators for all basic equivalences that will be dealt with in this paper. Axioms for bisimulation are also valid in other equivalences and are included in their axiomatisation. Proofs of the following theorems can be found in [15] (and [9] for ready simulation).

Theorem 1.1 The following four equations are a sound and complete axiomatisation of *FINTREE* operators for bisimulation equivalence.

(A1)	$(x + y) + z = x + (y + z)$
(A2)	$x + y = y + x$
(A3)	$x + x = x$
(A4)	$x + 0 = x$

□

Theorem 1.2 Sound and complete axiomatisations for *FINTREE* operators with respect to some basic process equivalences are presented below. For each

of the equivalences, its axiomatisation contains axioms A1 - A4 and one or two equations given in the table. Equivalences marked with * contain conditional equations that use an $I()$ operator which, given a process, returns the set of its initials.

trace	$a(x+y) = ax + ay$
completed trace	$a(bx+u) + a(cy+v) = a(bx+cy+u+v)$
failures	$a(bx+u) + a(by+v) = a(bx+by+u) + a(by+v)$ $ax + a(y+z) = ax + a(x+y) + a(y+z)$
readiness	$a(bx+u) + a(by+v) = a(bx+by+u) + a(by+v)$
failure trace*	$I(x) = I(y) \Rightarrow ax + ay = a(x+y)$ $ax + ay = ax + ay + a(x+y)$
ready trace*	$I(x) = I(y) \Rightarrow ax + ay = a(x+y)$
simulation	$a(x+y) = a(x+y) + ay$
ready simulation	$a(x+by+bz) + a(x+by) = a(x+by+bz)$

□

1.5 Modal characterizations of equivalences

An important and very useful tool in the analysis of process equivalences and their connections with TSS formats is the modal characterization of process equivalences. In general, we define a set of modal formulas (language) $\mathcal{L}$ which represents properties of a process we are interested in. For each formula $\varphi \in \mathcal{L}$ we define its semantics with a satisfaction relation $\models$. We define a corresponding process equivalence $\sim_{\mathcal{L}}$ by:

$$p \sim_{\mathcal{L}} q \text{ iff } \{\varphi \in \mathcal{L} \mid p \models \varphi\} = \{\varphi \in \mathcal{L} \mid q \models \varphi\}$$

Among modal characterizations the language and sublanguages of Hennessy-Milner Logic are particularly influential and useful. The original language was established by Hennesy and Milner in [16]. Further in [17] they proved that finitely branching processes are bisimilar if and only if they satisfy the same set of HML formulas. Modal characterizations of other basic equivalences with subsets of Hennesy-Milner logic have been presented in [15].

Definition (HML formulas (potential observations))
The set $\mathbb{O}$ of *finite modal observations (HML formulas)* is defined inductively by:
- $\top \in \mathbb{O}$
- $a\varphi \in \mathbb{O}$ if $\varphi \in \mathbb{O}$ and $a \in Act$
- $\forall_{i \in I} \varphi_i \in \mathbb{O}$ if $\varphi_i \in \mathbb{O}$ where I is a finite set
- $\neg\varphi \in \mathbb{O}$ if $\varphi \in \mathbb{O}$

The set of *infinite HML formulas* $\mathbb{O}^\infty$ contains all the above formulas as well as infinite conjunctions:
- $\mathbb{O} \subset \mathbb{O}^\infty$
- $\forall_{i \in I} \varphi_i \in \mathbb{O}^\infty$ if $\varphi_i \in \mathbb{O}^\infty$ where I is a set

The meaning of HML formulas is given by the following satisfaction relation.

Definition (Satisfaction relation)
Suppose we have a labelled transition system $(P, \rightarrow)$. A *satisfaction relation* $\models \subseteq P \times \mathbb{O}$ is defined inductively by:
- $p \models \top$
- $p \models a\varphi$ if $\exists q : p \xrightarrow{a} q \wedge q \models \varphi$
- $p \models \forall_{i \in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$
- $p \models \neg\varphi$ if $p \not\models \varphi$

This definition is valid for $\mathbb{O}^\infty$ as well.

Each HML formula represents some characteristic of a process behaviour. Finitary HML formulas ($\mathbb{O}$) in particular provide information about process behaviour up to a certain depth. This fact will be formally expressed and proved in Chapter 2. We can define depth of a formula $\varphi \in \mathbb{O}$ in a natural way.

Definition (Depth of a HML formula)
For all $\varphi \in \mathbb{O}$ we define $d(\varphi)$ inductively:
- $d(\top) = 0$
- $d(a\varphi) = d(\varphi) + 1$
- $d(\forall_{i \in I} \varphi_i) = max_{i \in I}\{d(\varphi_i)\}$
- $d(\neg\varphi) = d(\varphi)$

We will work with equivalences induced by sublanguages of $\mathbb{O}$. The following definitions provide a general scheme that will be used throughout the paper.

Definition (N-observations, equivalence defined by a modal language)
For each language $\mathbb{O}_N \subseteq \mathbb{O}$ and process p we define a set of *N-observations* of p:

$$\mathcal{O}_N(p) := \{\varphi \in \mathbb{O}_N \mid p \models \varphi\}.$$

The sets of *infinite N-observations* $\mathcal{O}_N^\infty(p)$ are defined analogously. For each $\mathbb{O}_N \subseteq \mathbb{O}$ a process equivalence $=_N$ is defined as:

$$p =_N q \text{ iff } \mathcal{O}_N(p) = \mathcal{O}_N(q)$$

What remains now is to define the subsets (sublanguages) of $\mathbb{O}$ which would correspond to the common process equivalences. The languages presented below come from [10] and are equal or in some cases slightly modified versions of those from [15].

Definition (Sublanguages of $\mathbb{O}$)
Sublanguages of $\mathbb{O}$ that correspond to basic process equivalences are presented below. We assume that in all the following formulas the sets I and J are finite.

- *trace observations*:
$\mathbb{O}_T \;\; \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_T)$

- *completed trace obervations*:
$\mathbb{O}_{CT}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{CT}) \mid \forall_{a\in Act}\neg a\top$
- *failures observations*:
$\mathbb{O}_{F}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{F}) \mid \forall_{i\in I}\neg a_i\top$
- *readiness observations*:
$\mathbb{O}_{R}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{R}) \mid \forall_{i\in I}\neg a_i\top \wedge \forall_{j\in J} b_j\top$
- *failure trace observations*:
$\mathbb{O}_{FT}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{FT}) \mid \forall_{i\in I}\neg a_i\top \wedge \varphi'(\varphi' \in \mathbb{O}_{FT})$
- *ready trace observations*:
$\mathbb{O}_{RT}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{RT}) \mid \forall_{i\in I}\neg a_i\top \wedge \forall_{j\in J} b_j\top \wedge \varphi'(\varphi' \in \mathbb{O}_{RT})$
- *simulation observations*:
$\mathbb{O}_{S}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{S}) \mid \forall_{i\in I}\varphi_i(\varphi_i \in \mathbb{O}_{S})$
- *ready simulation observations*:
$\mathbb{O}_{RS}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{RS}) \mid \neg a\top \mid \forall_{i\in I}\varphi_i(\varphi_i \in \mathbb{O}_{RS})$
- *bisimulation observations*:
$\mathbb{O}_{B}\ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_{B}) \mid \forall_{i\in I}\varphi_i(\varphi_i \in \mathbb{O}_{B}) \mid \neg\varphi'(\varphi' \in \mathbb{O}_{B})$

The following theorem states that the equivalences induced by the aforementioned subsets of infinite HML formulas coincide with the corresponding process equivalences. The proof can be found in [15].

Theorem 1.3 *For $N \in \{T, CT, F, R, FT, RT, S, RS, B\}$, $p =_N q$ if and only if $\mathcal{O}^{\infty}_{N}(p) = \mathcal{O}^{\infty}_{N}(q)$.*

□

In this paper I will always use the characterizations with finite HML formulas. In order to justify the use of $\mathbb{O}_N$ rather that $\mathbb{O}^{\infty}_{N}$ while reasoning about the corresponding process equivalence $=_N$, I will now prove the counterpart of the above theorem in the setting of finitely branching processes. I could refer to the existing theorems from [15], however van Glabbeek presented there proofs for each equivalence separately. I would like to establish a general condition so that a given equivalence defined with a subset $\mathbb{O}^{\infty}_{N}$ of $\mathbb{O}^{\infty}$ that satisfies this condition is completely determined by its finite formulas $\mathbb{O}_N$. As it turns out, it only suffices to make sure that if we take a $\varphi \in \mathbb{O}^{\infty}_{N}$ and replace an infinite conjunction at some position in φ with a finite one containing an arbitrary finite subset of formulas from the original conjunction, the resulting formula belongs to the language $\mathbb{O}^{\infty}_{N}$ as well.

I will use the following notions (position in a formula etc.) which are similar to those from the term rewriting theory. The definitions below are based on [4] and adapted to modal formulas.

Definition (Position in a HML formula, prefix order, subterm, replacement)

1. The set of *positions of a formula* $\varphi \in \mathbb{O}^{\infty}$ is a set $Pos(\varphi)$ of words that may contain elements from Act, $\{\neg\}$ and I for all I such that $\forall_{i\in I}\varphi_i$ is a subformula of φ. $Pos(\varphi)$ is defined inductively as:
- $\varphi = \top \Rightarrow Pos(\varphi) = \{\epsilon\}$

- $\varphi = a\psi \Rightarrow Pos(\varphi) = \{\epsilon\} \cup \{ap | p \in Pos(\psi)\}$
- $\varphi = \forall_{i\in I}\varphi_i \Rightarrow Pos(\varphi) = \{\epsilon\} \cup \bigcup_{i\in I}\{ip | p \in Pos(\varphi_i)\}$
- $\varphi = \neg\psi \Rightarrow Pos(\varphi) = \{\epsilon\} \cup \{\neg p | p \in Pos(\psi)\}$

2. Furthermore we define a *prefix order* on positions: $p \leq p'$ if there exists p'' such that $p = p'p''$.

3. For a formula $\varphi \in \mathbb{O}^\infty$ and position $p \in Pos(\varphi)$ the *subformula of φ at position p*, notation $\varphi|_p$ is defined inductively:

$\varphi|_\epsilon = \varphi \quad (a\psi)|_{ap'} = \psi|_{p'} \quad (\forall_{i\in I}\varphi_i)|_{ip'} = \varphi_i|_{p'} \quad (\neg\varphi)|_{\neg p'} = \varphi|_{p'}$

4. Finally, for formulas $\varphi, \psi \in \mathbb{O}^\infty$ and a position $p \in Pos(\varphi)$ we define the *replacement of the subformula of φ at position p* by ψ, notation $\varphi[\psi]_p$ as:
- $\varphi[\psi]_\epsilon = \psi$
- $(a\varphi)[\psi]_{ap'} = a\varphi[\psi]_{p'}$
- $(\forall_{j\in J}\varphi_j)[\psi]_{ip'} = \forall_{j\in J}\bar{\varphi_i}$ where $\bar{\varphi_i} = \varphi_i[\psi]_{p'}$ and $\bar{\varphi_j} = \varphi_j$ for $j \neq i$
- $(\neg\varphi)[\psi]_{\neg p'} = \neg\varphi[\psi]_{p'}$

Lemma 1.4 *Let $\varphi \in \mathbb{O}^\infty$ be a formula such that $\varphi|_p = \forall_{i\in I}\varphi_i$ for some set of indexes I. Let us define formulas $\varphi_p(J)$ as:*

$$\varphi_p(J) = \varphi[\forall_{i\in J}\varphi_i]_p \text{ for } J \subseteq_{FIN} I$$

1. If p contains an even number of "$\neg$", then

$$q \models \varphi \Leftrightarrow \forall_{J\subseteq_{FIN} I}\ q \models \varphi_p(J)$$

2. If p contains an odd number of "$\neg$", then $q \models \varphi$ implies that

$$\exists_{J_0\subseteq_{FIN} I} : q \models \varphi_p(J_0) \text{ and } \varphi_p(J_0) \text{ implies } \varphi$$

Proof While proving both statements, I will use the following observations. Firstly, we may assume that there are no two subsequent indexes in any position p, because any formula of the form $\forall_{i\in I}\varphi_i$, where some φ_i is again a conjunction, can be replaced by an equivalent "normal form" $\forall_{j\in J}\varphi_j$ where φ_j are of the form $a\varphi'_j$ or $\top$.

Another observation is that if p is a position in a formula φ that does not contain negation symbols, such a formula can be "unfolded" with respect to p. Namely, suppose that p contains actions $a_1, \ldots, a_n$ ($n \in \mathbb{N}$), precisely in this order, possibly interleaved with indexes. There exist formulas $\psi_0, \ldots, \psi_n$ such that $q \models \varphi$ if and only if there exist states $q_0, q_1, \ldots, q_n$ such that $q = q_0$ and $q_k \models \psi_k$ ($k \in \{0, 1, \ldots, n\}$) and $q_n \models \varphi|_p$. The formulas ψ_k for $k \in \{0, 1, \ldots, n\}$ are constructed as follows: if the next symbol after a_k (or the initial symbol for k=0) is a_{k+1} or $k = n$ and a_n is the last symbol in p, then $\psi_k = \top$. Otherwise the symbol after a_k is some index j, so at the position-prefix of p that ends with a_k we have a conjunction of the form $\forall_{i\in I}\varphi_i$. In such case we take $\psi_k = \forall_{i\in I\setminus\{j\}}\varphi_i$. A simple and important fact is that if make a substitution at position p with some formula γ, the values ψ_k in the unfolding of $\varphi[\gamma]_p$ remain unchanged.

1. We will proceed with induction on the number of $\neg$ symbols in p. In the

base case there are none of them, therefore p is a word over $Act \cup I_1 \cup \cdots \cup I_s$ for some sets of indexes $I_1, \ldots, I_s$. Observe that in this case $q \models \varphi$ if and only if there exist processes $q_0, \ldots, q_n$ with $q = q_0$ and $n \geq 0$ and certain formulas depending on φ: $\psi_0, \ldots, \psi_n \in \mathbb{O}^\infty$ (possibly equal to $\top$) such that $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} \ldots \stackrel{a_n}{\to} q_n$ and $q_i \models \psi_n$ for $i = 0, \ldots, n$ and moreover $q_n \models \varphi|_p$, $\varphi|_p = \forall_{i \in I} \varphi_i$. If $q_n \models \forall_{i \in I} \varphi_i$ then it is immediate that for all $J \subseteq_{FIN} I$ $q_n \models \forall_{i \in J} \varphi_i$ and therefore $q \models \varphi_p(J)$. Suppose now that $q \models \varphi_p(J)$ for all $J \subseteq_{FIN} I$. Then for each J there exists a corresponding chain of states, transitions and formulas $q_0^J \stackrel{a_1}{\to} q_1^J \stackrel{a_2}{\to} \cdots \stackrel{a_n}{\to} q_n^J$ with $q = q_0^J$, $q_i^J \models \psi_i$ for $i \leq n$ and $q_n^J \models \forall_{i \in J} \varphi_i$. Since p is finitely branching, the set $\{q_n^J | J \subseteq_{FIN} I\}$ is finite. Let us denote elements of this set with $q^1, \ldots, q^m$. Suppose now, towards contradiction, that none of q^i satisfies $\forall_{i \in I} \varphi_i$. Then for all $j \in \{1, \ldots, m\}$ $q^j \not\models \varphi_{i_j}$ for some $i_j \in I$. But then if we take $J_0 = \{i_1, \ldots, i_m\}$, $q^j \not\models \forall_{i \in J_0} \varphi_i$ for all $j \in \{1, \ldots, m\}$, which is a contradiction. Thus $q^j \models \forall_{i \in I} \varphi_i$ for some j and hence $q \models \varphi$.

Assume now that if $\psi|_p = \forall_{i \in I} \psi_i$ and p contains $2K$ "$\neg$" symbols, then $q \models \psi \Leftrightarrow (\forall J \subseteq_{FIN} I \; q \models \psi_p(J))$. Take φ with $\varphi|_p = \forall_{i \in I} \varphi_i$ where p contains $2K + 2$ "$\neg$" symbols. The case where first the two positions of p are occupied by negations is trivial, let us assume that it is not the case. Observe first that $q \models \varphi$ is equivalent to the statement:
(*) "there exists a chain of transitions and actions $q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} \ldots \stackrel{a_n}{\to} q_n$ and corresponding formulas $\psi_0, \ldots, \psi_n$ with $q_i \models \psi_i$ such that for every chain $q_n \stackrel{a_{n+1}}{\to} q_{n+1} \cdots \stackrel{a_{n+k}}{\to} q_{n+k}$ with $q_i \models \psi_i$ for some φ-dependent ψ_i $q_{n+k} \not\models \bar{\varphi}$ where $\bar{\varphi} = \varphi|_{p'}$ for some p' containing two $\neg$ symbols and ending with a $\neg$ symbol"

Suppose that $q \models \varphi$ and fix such a chain. We have:

$$(**)\; q \models \varphi_p(J) \Leftrightarrow q_{n+k} \models \varphi_{p''}(J)$$

where p'' is a position such that $p = p'p''$ (this is because the chain satisfies all the other necessary subformulas ψ_i). Since $q_{n+k} \models \bar{\varphi}$, from the induction hypothesis we obtain $\forall J \subseteq_{FIN} I \; q_{n+k} \models \bar{\varphi}_{p''}(J)$, and according to (**) we obtain $\forall J \subseteq_{FIN} I \; q \models \varphi_p(J)$. The proof in the other direction uses the fact that q is finitely branching and there are only finite number of possible q_n^J states for each $\varphi_p(J)$. A slightly modified reasoning from the base case proves that in one of these states for each "appropriate" path (with trace $a_1 ... a_{n+k}$ where intermediate states satisfy ψ_i) that leads to q_{n+k} we have $q_{n+k} \models \bar{\varphi}_{p''}(J)$. From the induction hypothesis we obtain $q_{n+k} \models \bar{\varphi}$ and deduce that $p \models \varphi$.

2. Now consider the case when p contains an odd number of "$\neg$" symbols. We will use the first part of the lemma that has already been proven. Let us consider a position $p' < p$ such that the first $\neg$ symbol appears in p just after p' (so $(p'\neg) \leq p$). The fact that $q \models \varphi$ is then equivalent to the existence of a chain of states and transitions $q = q_0 \stackrel{a_1}{\to} q_1 \stackrel{a_2}{\to} \ldots \stackrel{a_n}{\to} q_n$ with $q_i \models \psi_i$ for some ψ_i where $i \leq n$ and $q_n \not\models \bar{\varphi}$ where $\bar{\varphi} = \varphi|_{p'\neg}$. Suppose then that $p \models \varphi$ and fix such a chain $(a_1, q_1), \ldots, (a_n, q_n)$. Observe that for a position p'' such that $p = p'\neg p''$ we have $\bar{\varphi}|_{p''} = \varphi|_p = \forall_{i \in I} \varphi_i$ and p'' contains an even number of $\neg$ symbols. According to the first part of the lemma, $q_n \models \bar{\varphi} \Leftrightarrow \forall J \subseteq_{FIN} I \; q_n \models \bar{\varphi}_{p''}(J)$

which is equivalent to $q_n \not\models \bar{\varphi} \Leftrightarrow \exists J \subseteq_{FIN} I\ q_n \not\models \bar{\varphi}_{p''}(J)$. Thus there exists $J_0 \subseteq_{FIN} I$ such that $q_n \models \bar{\varphi}_{p''}(J_0)$, from which we deduce that $q \models \varphi_p(J_0)$.

Now take any q' such that $q' \models \varphi_p(J_0)$. As before there exists a corresponding chain $q' = q'_0 \stackrel{a_1}{\to} q'_1 \stackrel{a_2}{\to} \ldots \stackrel{a_n}{\to} q'_n$ such that $q'_i \models \psi_i$ and $q'_n \not\models \bar{\varphi}_{p''}(J_0)$ where $\bar{\varphi} = \varphi|_{p'\neg}$. Since $q'_n \not\models \bar{\varphi}_{p''}(J_0)$, $q'_n \not\models \bar{\varphi}$ and thus $q' \models \varphi$.

□

In the following lemma I will use structural induction on the complexity of $\varphi \in \mathbb{O}^\infty$. It is feasible since HML formulas are defined recursively. I will use the term *proper subformula of* $\varphi \in \mathbb{O}^\infty$ to distinguish formulas that are used directly in the definition of φ. So for example if $\varphi = \forall_{n\in\mathbb{N}}\varphi_n$, then φ_i for any $i \in \mathbb{N}$ is a proper subformula of φ whereas $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ is not. In fact, ψ is a proper subformula of φ if $\psi = \varphi|_p$ for some $p \neq \epsilon$.

Lemma 1.5 *Consider a partial order* $\prec \in \mathbb{O}^\infty \times \mathbb{O}^\infty$ *defined as:* $\bar{\varphi} \prec \varphi$ *if either:*
1. $\varphi|_p = \forall_{i\in I}\varphi_i$ *such that* I *is infinite and* p *is a minimal position with this property and* $\bar{\varphi} = \varphi[\forall_{i\in J}\varphi_i]_p$ *where* J *is a finite subset of* I
2. $\exists\psi : \bar{\varphi} \prec \psi \wedge \psi \prec \varphi$ *(transitive closure)*
For such partial order there is no infinite descending chain
$\varphi_1 \succ \varphi_2 \succ \ldots \varphi_n \succ \ldots$.

Proof We will proceed with structural induction on φ and prove that there is no infinite descending chain starting with φ for all $\varphi \in \mathbb{O}^\infty$. Base case ($\varphi = \top$) is immediate. Suppose that for all proper subformulas of φ the lemma holds. Consider the following cases:
- $\varphi = a\varphi'$ or $\varphi = \neg\varphi'$. Observe that if $a\psi_1 \succ \psi$ then $\psi = a\psi_2$ for some ψ_2 and $\psi_1 \succ \psi_2$. A similar fact holds if we replace a with $\neg$. Therefore, if there was an infinite descending chain $a\varphi' = a\varphi'_0 \succ a\varphi'_1 \succ \cdots \succ a\varphi'_n \succ \ldots$ then there would have to be a corresponding chain $\varphi' = \varphi'_0 \succ \varphi'_1 \succ \cdots \succ \varphi'_n \succ \ldots$ which contradicts the induction hypothesis.
- $\varphi = \forall_{i\in I}\varphi_i$. Suppose towards contradiction that there exists infinite descending chain starting with $\varphi = \forall_{i\in I}\varphi_i$. If I is infinite, then for some $J \subseteq_{FIN} I$ $\forall_{i\in J}\varphi_i$ is a successor of φ in such a chain. Thus in any case there is an infinite descending chain containing a finite conjunction of the form $\forall_{i\in J}\varphi_i$ where for all $i \in J$ there is no such chain starting with φ_i. On the other hand, a "lesser" formula (with respect to $\prec$) is obtained by replacing an infinite conjunction with a finite one at some minimal position where it occurs, so in the infinite descending chain starting with $\forall_{i\in J}\varphi_i$ these replacements take place at one of the φ_i formulas and its descendants. So for at least one $i \in J$ there must be an infintite descending chain $\varphi_i = \varphi_i^1 \succ \varphi_i^2 \succ \ldots \varphi_i^n \succ \ldots$, a contradiction with the induction hypothesis.

□

Theorem 1.6 *Suppose we have a sublanguage of all infinite HML formulas* $\mathbb{O}_N^\infty \subseteq \mathbb{O}^\infty$ *satisfying:*

(*) $(\varphi \in \mathbb{O}_N^\infty \wedge (\varphi|_p = \forall_{i\in I}\varphi_i)) \Rightarrow \forall_{J\subseteq_{FIN} I}\ \varphi[\forall_{i\in J}\varphi_i]_p \in \mathbb{O}_N^\infty$
Then for all finitely branching processes the corresponding process equivalence is completely determined by finite HML formulas, namely if q_1 and q_2 are finitely branching then:

$$\mathcal{O}_N(q_1) = \mathcal{O}_N(q_2) \Rightarrow \mathcal{O}_N^\infty(q_1) = \mathcal{O}_N^\infty(q_2)$$

Proof Take two finitely branching processes q_1 and q_2 such that $\mathcal{O}_N(q_1) = \mathcal{O}_N(q_2)$. We proceed with transfinite induction w.r.t. $\prec$. The base case is trivial since the minimal elements are precisely finite HML formulas. Suppose that for all $\psi \prec \varphi$ we have $q_1 \models \psi \Leftrightarrow q_2 \models \psi$. Now let p be a minimal position such that $\varphi|_p = \forall_{i\in I}\varphi_i$ for an infinite set I. There are two possibilities:
1. If p contains an even number of "$\neg$" symbols, then:

$$\begin{array}{ll} q_1 \models \varphi \Leftrightarrow \forall_{J\subseteq_{FIN} I}\ q_1 \models \varphi_p(J) & \text{(Lemma 1.4)} \\ \Leftrightarrow \forall_{J\subseteq_{FIN} I}\ q_2 \models \varphi_p(J) & (\varphi_p(J) \prec \varphi + \text{induction hypothesis}) \\ \Leftrightarrow q_2 \models \varphi & \text{(Lemma 1.4)} \end{array}$$

2. Otherwise, suppose that p contains an odd number of "$\neg$" symbols. I will prove that $q_1 \models \varphi \Rightarrow q_2 \models \varphi$, the proof in other direction is symmetric. According to Lemma 1.4 there exists a witness formula $\varphi_p(J_0)$ for $J_0 \subseteq_{FIN} I$ such that $q_1 \models \varphi_p(J_0)$ and satisfying $\varphi_p(J_0)$ implies satisfying φ. Since $\varphi_p(J_0) \prec \varphi$, from the induction hypothesis we obtain $q_2 \models \varphi_p(J_0)$ and thus $q_2 \models \varphi$.

□

Corollary 1.7 *For $N \in \{T, CT, F, R, FT, RT, S, RS, B\}$ and finitely branching processes p and q, $p =_N q$ if and only if $\mathcal{O}_N(p) = \mathcal{O}_N(q)$.*

Proof Observe first that for $N \in \{T, CT\}$ we have $\mathbb{O}_N^\infty = \mathbb{O}_N$ (in case of completed trace we use the fact that *Act* is finite). For other equivalences it is easy to check that the sets $\mathbb{O}_N^\infty$ where $N \in \{F, R, FT, RT, S, RS, B\}$ meet the requirements of Theorem 1.6 and thus $\mathcal{O}_N(q_1) = \mathcal{O}_N(q_2) \Leftrightarrow \mathcal{O}_N^\infty(q_1) = \mathcal{O}_N^\infty(q_2)$ (the "$\Leftarrow$" implication is obvious since $\mathcal{O}_N(q) \subseteq \mathcal{O}_N^\infty(q)$ for all $q \in P$).

□

In order to show that an arbitrary equivalence $=_N$ definable with infinite modal characterizations is not necessarily determined by finite HML formulas, I will now present an equivalence that does not satisfy the requirements of Theorem 1.3.

Counterexample Let us define a set of infinite modal HML formulas:

$$\mathbb{O}_N^\infty \ \varphi ::= \top \mid a\varphi'(\varphi' \in \mathbb{O}_N^\infty) \mid \ \forall_{\sigma\in S}\sigma\top \text{ where } S \subset Act^* \wedge |S| = \infty$$

Observe that this set violates condition (*) of Thoerem 1.6 . Let us define processes $P = aB + aC$ and $Q = aD$, where $B = aB + b$, $C = aC + c$ and $D = aD + b + c$. It is not difficult to check that P and Q satisfy the same set of finite HML formulas from $\mathbb{O}_N$. However, $P \neq_N Q$ since $a\forall_{n\in\mathbb{N}}(a^n b\top \wedge a^n c\top) \in \mathcal{O}_N^\infty(Q) \setminus \mathcal{O}_N^\infty(P)$.

Chapter 2
Approximation Induction Principle

While considering infinite processes, we encounter the following problem. Suppose we have a sound and complete axiomatisation for a given equivalence $=_N$ that allows us to equate each pair of $=_N$-equivalent finite processes. We are still unable to do the same with terms representing processes that have an infinite execution path. Moreover, for example in case of bisimulation, it is not possible to have a "standard" finite complete axiomatisation in which we could equate all pairs of bisimilar processes over a GSOS-defined signature. We need to employ a more powerful infinitary conditional equation, namely the Approximation Induction Principle (AIP), introduced by Baeten, Bergstra and Klop in [5]. It states that if two processes are equivalent up to any finite depth, then they are equivalent.

2.1 AIP in HML-definable equivalences

I will now give formal definitions concerning AIP. First let us define projection operators π_n, which, given a process, simulate its behaviour up to n steps and then terminate.

Definition (Projection operators)
For each natural number n we define a *projection operator* π_n. The behaviour of an application of the projection operator to a process is given by the following rule scheme:

$$\frac{x \xrightarrow{a} x'}{\pi_{n+1}(x) \xrightarrow{a} \pi_n(x')}$$

For example: $\pi_2(abcd + defg + cd) = ab + de + cd$. For each $n \in \mathbb{N}$ a finite complete axiomatisation for π_n exists ([2],[7]):

$$\begin{array}{rcl} \pi_n(x+y) & = & \pi_n(x) + \pi_n(y) \\ \pi_{n+1}(ax) & = & a\pi_n(x) \\ \pi_n(0) & = & 0 \\ \pi_0(x) & = & 0 \end{array}$$

At this point we should discuss one subtlety. If we used these equations to axiomatize a set of operators containing all the projections, we would obtain an infinite axiomatisation. However, this obstacle can be removed if we mimic each of the projection operators π_n with a binary operation / where the first parameter is the process from which we take the n-th projection and the second one works as an "hourglass", for example $\pi_n(x) = x/b^n$ for some $b \in Act$. The details as well as a finite axiomatisation of the / operator can be found in [2]. Thus we can safely assume that there exists a finite complete axiomatisation of

the projection operators.

With the projection operators defined, we can now phrase the basic version of the Approximation Induction Principle:

Definition (Approximation Induction Principle - AIP)
An *Approximation Induction Principle* is the following assumption:

$$\text{(AIP) If } \pi_n(x) = \pi_n(y) \text{ for all } n \in N, \text{ then } x = y.$$

Now suppose the setting as mentioned in the beginning. We have an axiomatisation for a given equivalence in which AIP holds, which is complete for all terms representing well-founded processes. If we include AIP in this axiomatisation, we will obtain a sound and and complete axiom system. What remains now is to find out for which process equivalences AIP is sound. I will prove that it is the case for all process equivalences that can be defined using modal characterizations within $\mathbb{O}$.

The crucial part of the proof is the following lemma which states that if a finitary modal formula is satisfied by a process p, then it is satisfied by almost all of its projections. The intuition here is simple: each modal formula is satisfied by some finite sub-process of p with depth (maximum execution path) equal to $d(\varphi)$, and from $d(\varphi)$ onward, this sub-process is included in all $\pi_n(p)$ where $n \geq d(\varphi)$.

Lemma 2.1 *For any process p:*

$$\varphi \in \mathcal{O}(p) \Leftrightarrow \forall n \geq d(\varphi)\ \varphi \in \mathcal{O}(\pi_n(p))$$

Proof Assume an arbitrary process p. We will proceed with induction on the size of the formula φ.
"$\Rightarrow$": The base is trivial ($\varphi = \top$). Suppose that for all φ: $|\varphi| \leq k$, $p \models \varphi$ implies that φ is satisfied by all projections $\pi_n(p)$ where $n \geq d(\varphi)$. Let $\psi \in \mathcal{O}(p)$, $|\psi| = k + 1$. There are three possible cases:

- $\psi = a\varphi$
  Then $\exists q$: $p \xrightarrow{a} q \wedge q \models \varphi$ with $\varphi \in \mathcal{O}(q)$. From the induction hypothesis we obtain: $\forall n \geq d(\varphi)\ \varphi \in \mathcal{O}(\pi_n(q))$. Since $\pi_n(p) \xrightarrow{a} \pi_{n-1}(q)$ for $n \geq 1$, we have: $\forall n \geq d(\varphi) + 1\ \pi_n(p) \models a\varphi$, so $a\varphi \in \mathcal{O}(\pi_n(p))$ for $n \geq d(\varphi) + 1 = d(a\varphi)$.

- $\psi = \forall_{i \in I} \varphi_i$
  $p \models \forall_{i \in I} \varphi_i \Leftrightarrow \forall_{i \in I} p \models \varphi_i$
  With the induction hypothesis, the last statement implies:
  $\forall i \in I\ \forall n \geq d(\varphi_i)\ \pi_n(p) \models \varphi_i$
  Therefore:
  $\forall n \geq max_{i \in I}\{d(\varphi_i)\}\ \forall i \in I\ \pi_n(p) \models \varphi_i$
  But $d(\forall_{i \in I} \varphi_i)$ is equal to $max_{i \in I}\{d(\varphi_i)\}$ from the definition. Hence
  $\forall n \geq d(\forall_{i \in I} \varphi_i)\ \pi_n(p) \models \forall_{i \in I} \varphi_i$

- $\psi = \neg\varphi$
  We have to consider all the subcases, depending on φ:
  - $\varphi = \top$: this is impossible (it would mean that $p \not\models \top$ which is never true)
  - $\varphi = a\varphi'$: In this case $p \not\models a\varphi'$ which is equivalent to: $\forall q : p \xrightarrow{a} q \; q \not\models \varphi'$. If there is no a-transition from p, then obviously each projection $\pi_n(p)$ satisfies $\neg a\varphi'$. Otherwise, let $\{q_i\}_{i\in I}$ be the set of all q_i for which $p \xrightarrow{a} q_i$. Now, from the induction hypothesis we know that for each q_i $\forall n \geq d(\neg\varphi')$ $\pi_n(q_i) \models \neg\varphi'$ (observe that $|\neg\varphi'| = k$). Therefore $\forall n \geq d(\neg\varphi') + 1$ $\pi_n(p) \not\models a\varphi'$ and thus $\forall n \geq d(\varphi)$ $\pi_n(p) \models \neg a\varphi'$.
  - $\varphi = \forall_{i\in I}\varphi_i$: This is equivalent to $\exists i_0 \in I : p \not\models \varphi_{i_0} \Leftrightarrow \exists i_0 \in I : p \models \neg\varphi_{i_0}$. Again, $|\neg\varphi_{i_0}| \leq k$, so we can apply induction hypothesis to obtain: $\forall n \geq d(\neg\varphi_{i_0})$: $\pi_n(p) \models \neg\varphi_{i_0} \Rightarrow \pi_n(p) \not\models \forall_{i\in I}\varphi_i \Leftrightarrow \pi_n(p) \models \neg\forall_{i\in I}\varphi_i$ which is the desired statement.
  - $\varphi = \neg\varphi'$: This is immediate (in this case ψ is equivalent to φ').

"$\Leftarrow$": The other direction follows immediately from what we have just proven. Take an arbitrary formula $\psi \in \mathbb{O}$ and a process p such that $\forall n \geq d(\psi)$ $\pi_n(p) \models \psi$. Suppose towards contradiction that $p \not\models \psi$. Then $p \models \neg\psi$ and it was already proven that this implies $\forall n \geq d(\neg\psi)$ $\pi_n(p) \models \neg\psi$. This contradicts our assumptions about ψ. Therefore p must satisfy ψ.

□

Theorem 2.2 *Let $\mathbb{O}_N$ be any subset of the set of all finite observations $\mathbb{O}$ and let $\mathcal{O}_N(p)$ denote the set $\{\varphi \in \mathbb{O}_N \mid p \models \varphi\}$. For a process equivalence $=_N$ defined by:*

$$p =_N q \Leftrightarrow \mathcal{O}_N(p) = \mathcal{O}_N(q)$$

the Approximation Induction Principle is sound, namely:

$$\forall n \in N \; \pi_n(p) =_N \pi_n(q) \Rightarrow p =_N q$$

Proof Suppose that $\forall n \in N$ $\pi_n(p) =_N \pi_n(q)$, which means that $\forall n \in N$ $\mathcal{O}(\pi_n(p)) = \mathcal{O}(\pi_n(q))$. We have to prove that $\mathcal{O}_N(p) = \mathcal{O}_N(q)$. In fact it suffices to prove that $\mathcal{O}_N(p) \subseteq \mathcal{O}_N(q)$, the proof for the other inclusion is symmetric. Take any $\varphi \in \mathcal{O}(p)$. According to the Lemma 2.1, we have $\forall n \geq d(\varphi)$ $\varphi \in \pi_n(p) = \pi_n(q)$. Using the Lemma again we obtain $\varphi \in \mathcal{O}_N(q)$.

□

The assumption made about the mapping $\mathcal{O}_N$ could be further relaxed. For a given process p it can return any set of formulas satisfied by p, provided that ($\varphi \in \mathcal{O}_N(p)$ and $\pi_n(p) \models \varphi$) $\Rightarrow$ $\varphi \in \mathcal{O}_N(\pi_n(p))$ for any projection of p. An important consequence of Theorem 2.2 is given below:

Corollary 2.3 *AIP is sound with respect to all the basic process equivalences on finitely branching processes, namely $T, CT, F, R, FT, RT, S, RS, B$.*

□

2.2 Constructive AIP and N-regular processes

AIP in its original version can serve us only as an elegant complementary rule allowing to obtain a complete axiom system. It is still a purely theoretical concept. For given arbitrary processes p and q, we cannot effectively compute whether they are equivalent or not, because we would have to check the equivalence of all their countably infinite projections.

To overcome this obstacle, the notion of constructive AIP (AIP^c) was introduced by Mauw in [18] and investigated further by Barros and Hou in [8]. The constructive version of AIP allows us to decide bisimilarity between two regular processes by checking only one pair of their projections.

We shall observe first that a regular process can be expressed with linear process equations. In fact, a process is regular if and only if it is a solution of a set of linear equations (linear recursive specification) [13].

Definition (Linear recursive specification)
A set of recursive process equations E is a *linear recursive specification* if its equations are of the form:

$$X_i = a_{i,1}X_1 + \cdots + a_{i,k}X_k + b_{i,1} + \cdots + b_{i,l_i}$$

with $i \in \{1, \ldots, k\}$ and $a_{i,1}, \ldots, a_{i,k}, b_{i,1}, \ldots, b_{i,l_i} \in A \cup \{0\}$.

A solution of a recursive equation $X_i = a_{i,1}X_1 + \cdots + a_{i,k}X_k + b_{i,1} + \cdots + b_{i,l_i}$ is unique modulo bisimulation [13] and denoted with $< X_i|E >$ where E is the linear recursive specification to which the given equality belongs.

Theorem 2.4 *Process p is regular if and only if it can be represented by a linear recursive specification. In other words, there exists a linear recursive specification E:*

$$X_i = a_{i,1}X_1 + \cdots + a_{i,k}X_k + b_{i,1} + \cdots + b_{i,l_i}$$

such that $p \underline{\leftrightarrow} < X_i|E >$ for some i.

□

Now suppose we have two processes p and q which we know are regular. Ideally, we would like to have the corresponding linear recursive specifications given for each of them such that $p =< X_i|E_1 >$ and $q =< Y_j|E_2 >$. If this is the case, we can further assume that there is only one set of equations E and $p =< X_i|E >$ and $q =< Y_j|E >$ for some i and j. Barros and Hou [8] have proved that in this case we only need to check the $(n-1)$-th projection where n is the number of equations in E:

Theorem 2.5 (AIP^c). *Let E be a linear recursive specification with n variables and let X_p and X_q be two recursion variables in E, then:*

$$\pi_{n-1}(< X_p|E >) = \pi_{n-1}(< X_q|E >) \;\Rightarrow\; < X_p|E >=< X_q|E >$$

□

With AIP^c we obtain a useful tool that allows us to actually compute whether two processes are bisimilar, provided that we are given the appropriate linear recursive specification. Barros and Hou also presented a way to obtain such a specification for all terms over ACP with linear recursion.

I will show that AIP^c also holds in case of slightly modified linear recursive specifications, where equations are given with respect to an arbitrary finite observations-based process equivalence. Firstly, I will define N-regular processes, which generalize the class of regular processes.

Definition (N-regular process)
For an equivalence on processes $=_N$ based on finite observations $\mathbb{O}_N$, a finitely branching process p is *N-regular* if the set of its reachable states which are distinguishable modulo $=_N$ is finite.

We will also restrict ourselves to the class of equivalences satisfying certain sanity conditions. I will refer to them as regular equivalences. Note that the conditions below have been established so that the subsequent lemma could hold and perhaps it would be more appropriate to call an equivalence regular if it is compositional w.r.t. projections and Lemma 2.6 holds for this equivalence.

Definition (Regular equivalence)
Suppose that $\mathbb{O}_N \subseteq \mathbb{O}$ induces an equivalence $=_N$. The equivalence $=_N$ is *regular* if:
1. $(\varphi \in \mathbb{O}_N \wedge (\varphi|_p = a\psi)) \Rightarrow \varphi[\psi]_p \in \mathbb{O}_N$
2. $(\varphi \in \mathbb{O}_N \wedge (\varphi|_p = \forall_{i\in I}\psi_i)) \Rightarrow \varphi[\psi_i]_p \in \mathbb{O}_N$
3. $(\varphi \in \mathbb{O}_N \wedge (\varphi|_p = \neg\psi)) \Rightarrow \varphi[\psi]_p \in \mathbb{O}_N$
4. $=_N$ is a congruence w.r.t. all projection operators.

Lemma 2.6 *Let $=_N$ be a regular equivalence. If a process p is N-regular, then $p =_N p'$ where p' is a regular process.*

Proof Suppose p is N-regular and let Q be the set of all states reachable from p. Now we create a new labelled transition system with a set of states $Q' = \{[q]_{=_N} \mid q \in Q\}$ and transition defined with: $[q]_{=_N} \xrightarrow{a} [s]_{=_N}$ whenever $q \xrightarrow{a} s$ (so we include all the so-called "may transitions"). Then $[p]_{=_N} \in Q'$ is obviously regular. The fact that $\mathcal{O}_N(p) = \mathcal{O}_N([p]_{=_N})$ can be proved with induction on the size of a modal formula. I will use $[p]$ as an abbreviation of $[p]_{=_N}$. Base case ($\varphi = \top$) is obvious. Suppose that for all $\varphi \in \mathbb{O}_N$ such that $|\varphi| \leq k$ we have $\varphi \in \mathcal{O}_N(p) \Leftrightarrow \varphi \in \mathcal{O}_N([p])$. Take ψ such that $|\psi| = k+1$. We have to consider the following cases:

- $\psi = a\varphi$
  "$\Rightarrow$": We have: $a\varphi \in \mathcal{O}_N(p)$. Then there exists a process q: $p \xrightarrow{a} q$ and

$q \models \varphi$. By definition of the transition relation on Q', there is a transition $[p] \xrightarrow{a} [q]$. From the induction hypothesis $[q] \models \varphi$, so $[p] \models a\varphi$.
"$\Leftarrow$": Now $a\varphi \in \mathcal{O}_N([p])$. Then there exists q such that $[p] \xrightarrow{a} [q]$ and $[q] \models \varphi$, so from the induction hypothesis $q \models \varphi$. From $[p] \xrightarrow{a} [q]$ it follows further that there exists some p' such that $p' =_N p$ and a transition $p' \xrightarrow{a} q$. For this p' we have $p' \models a\varphi$ and since $p' =_N p$, we obtain $a\varphi \in \mathcal{O}_N(p)$.

- $\psi = \forall_{i\in I}\varphi_i$
We have: $\forall_{i\in I}\varphi_i \in \mathcal{O}_N(p) \Leftrightarrow p \models \forall_{i\in I}\varphi_i \Leftrightarrow \forall_{i\in I} p \models \varphi_i$
$\Leftrightarrow \forall_{i\in I}[p] \models \varphi_i$ (from the induction hypothesis) $\Leftrightarrow [p] \models \forall_{i\in I}\varphi_i$

- $\psi = \neg\varphi$
We shall consider three subcases, since $\varphi = \top$ is not possible:
- $\psi = \neg a\varphi'$:
"$\Rightarrow$": We assume that there is no q such that $p \xrightarrow{a} q$ and $q \models \varphi$. We have to prove that there is also no q' such that $[p] \xrightarrow{a} [q']$ and $[q'] \models \varphi'$. Suppose towards a contradiction that there exists such q'. By the induction hypothesis $q' \models \varphi'$. Then there must be a process p' with $p' =_N p$ and $p' \xrightarrow{a} q'$. This implies $p' \models a\varphi'$ but since $p' =_N p$ we would also have $p \models a\varphi'$, a contradiction.
"$\Leftarrow$": Now assume $[p] \models \neg a\varphi'$. Then there is no q such that $[p] \xrightarrow{a} [q]$ and $[q] \models \varphi'$. Since from the induction hypothesis $([q] \models \varphi') \Leftrightarrow (q \models \varphi')$, there is also no q with $p' \xrightarrow{a} q$ for any p' such that $p' =_N p$, so for p in particular. Thus $p \not\models a\varphi'$, so $p \models \neg a\varphi'$.
- $\psi = \neg\forall_{i\in I}\varphi'_i$:
For each i $|\neg\varphi_i| \leq k$ and from the induction hypothesis we have $p \models \neg\varphi_i \Leftrightarrow [p] \models \neg\varphi_i$. Thus $p \models \neg\forall_{i\in I}\varphi'_i \Leftrightarrow \exists_{i\in I} p \not\models \varphi'_i \Leftrightarrow \exists_{i\in I}[p] \not\models \varphi'_i \Leftrightarrow [p] \models \neg\forall_{i\in I}\varphi'_i$.
- $\psi = \neg\neg\varphi'$: here $\psi \equiv \varphi'$ with $|\varphi'| = k - 1$ and we obtain
$(p \models \psi = \varphi') \Leftrightarrow ([p] \models \psi = \varphi')$ immediately from the induction hypothesis.

□

For equivalences based on arbitrary subsets of modal observations the converse does not hold, a process which is N-equivalent to a regular process is not necessarily an N-regular process itself. Take, for instance, a labelled transition system with only one action a and readiness as a process equivalence. Process A defined as follows:

$$A = a.A + A_1$$
$$A_n = a.A_{n+1} + a^n$$

is readiness equivalent to a process B with $B = a.B + a$. However, the first process is not readiness - regular.

Theorem 2.7 *Let $=_N$ be a regular equivalence. Consider a set of equations E:*

$$X_i =_N a_{i,1}X_1 + \cdots + a_{i,k}X_k + b_{i,1} + \cdots + b_{i,l_i}$$

with $i \in \{1, \ldots, k\}$ and $a_{i,1}, \ldots, a_{i,k}, b_{i,1}, \ldots, b_{i,l_i} \in A \cup \{0\}$.

Let $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ be two solutions for E, namely:

$$\forall_{i\in\{1,\ldots,k\}}\ p_i =_N a_1p_1 + \cdots + a_kp_k + b_1 + \cdots + b_l \text{ and}$$
$$\forall_{i\in\{1,\ldots,k\}}\ q_i =_N a_1q_1 + \cdots + a_kq_k + b_1 + \cdots + b_l$$

Then $\forall_{i\in\{1,\ldots,k\}}\ p_i =_N q_i$.

Proof Take an arbitrary i. I will prove with induction on n that $\forall_{n\in\mathbb{N}}\ \pi_n(p_i) = \pi_n(q_i)$. Base case ($n = 0$) is trivial. Now suppose that $\forall_{k\leq n}\ \pi_k(p_i) = \pi_k(q_i)$. We have:

$\pi_{n+1}(p_i) =_N \pi_{n+1}(a_{i,1}p_1 + \cdots + a_{i,k}p_k + b_{i,1} + \cdots + b_{i,l_i})$ (congruence w.r.t. π_{n+1})
$=_B a_{i,1}\pi_n(p_1) + \cdots + a_{i,k}\pi_n(p_k) + b_{i,1} + \cdots + b_{i,l_i}$ (property of projection)
$=_N a_{i,1}\pi_n(q_1) + \cdots + a_{i,k}\pi_n(q_k) + b_{i,1} + \cdots + b_{i,l_i}$ (induction hypothesis)
$=_B \pi_{n+1}(a_{i,1}q_1 + \cdots + a_{i,k}q_k + b_{i,1} + \cdots + b_{i,l_i})$ (property of projection)
$=_N \pi_{n+1}(q_i)$ (congruence w.r.t. π_{n+1})

Having proved equality of all projections, now we can apply the AIP theorem for all finite observations-based equivalences, which yields: $p_i =_N q_i$.

□

Theorem 2.8 *AIP^c is sound for any finite HML observations-based regular process equivalence, namely, if we have a linear recursive specification consisting of k equations modulo $=_N$, and compare two processes from the vector of solutions, we only need to check $(k-1)$-th projection to decide if they are $=_N$-equivalent.*

Proof The same reasoning as in the proof of AIP^c in [8], with the exception that we would use $=_N$ instead of $=$ which is there implicitly assumed to be bisimulation equivalence. In fact, it can be any of the equivalences based on finite observations provided that it is compositional w.r.t. the projections.

□

Corollary 2.9 *Suppose that $=_N$ is a regular equivalence. For every pair of N-regular processes p and q, there exists $n \in \mathbb{N}$ such that $\pi_n(p) =_N \pi_n(q) \Leftrightarrow p =_N q$ where $=_N$ is an arbitrary equivalence based on finite observations.*

Proof Since p and q are N-regular, there exist regular processes p' and q' such that $p =_N p'$ and $q =_N q'$. Let $E_{p'}$ and $E_{q'}$ be linear recursive specifications such that p' and q' respectively are in their vectors of solutions. Suppose without loss of generality that sets of variables in $E_{p'}$ and $E_{q'}$ are disjoint. Let

E be a linear recursive specification with equations from $E_{p'} \cup E_{q'}$ where $=$ is replaced with $=_N$. From generalized AIP^c for arbitrary equivalences it follows that $\pi_{|E|-1}(p) =_N \pi_{|E|-1}(q) \Leftrightarrow p =_N q$.

□

2.3 Which equivalences are useful?

So far I have presented several theorems concerning classes of process equivalences satisfying certain conditions. These conditions have mostly been expressed in terms of modal characterizations (Theorem 1.3, theorems for N-regular processes). In the definition of a regular equivalence there is also a requirement that an equivalence should be compositional w.r.t. projection. In this section I would like to discuss the "sanity" properties that we usually have to impose on equivalences that we analyse and the possibility of expressing those conditions in terms of modal characterizations with HML formulas. Below, four sanity conditions are given that are used throughout this paper:

1. Equivalence is definable with finite HML formulas.
2. Compositionality w.r.t. *FINTREE*.
3. Compositionality w.r.t. the projection operators.
4. For each N-regular process p, $p =_N q$ where q is a regular process.

I will now discuss sufficient conditions such that four of the above properties are met:

1. A fairly general sufficient condition has been established in Theorem 1.3.

2. In [15] it has been proved that all basic process equivalences are compositional w.r.t. *FINTREE* operators. It might be interesting to give a more general condition involving properties of a modal characterization.

3. The folllowing lemma states that if two distinguishable processes of depth at most K are guaranteed to differ on a formula of depth less or equal K then the equivalence is a congruence w.r.t. projection operators.

Lemma 2.10 *Suppose $=_N$ is a HML-based process equivalence and let us denote the maximum length of an execution path of a finite process p with $depth(p)$. If for all finite processes $p \neq_N q$ implies that there is a HML formula φ such that*
(1) $\varphi \in (\mathcal{O}_N(p) \setminus \mathcal{O}_N(q)) \cup (\mathcal{O}_N(q) \setminus \mathcal{O}_N(p))$ and
(2) $d(\varphi) \leq max\{depth(p), depth(q)\}$,
then $=_N$ is compositional w.r.t. the projection operators.

Proof Suppose that $\pi_n(s) \neq \pi_n(t)$ for some $n \in \mathbb{N}$. Then there exists $\varphi \in (\mathcal{O}_N(\pi_n(p)) \setminus \mathcal{O}_N(\pi_n(q))) \cup (\mathcal{O}_N(\pi_n(q)) \setminus \mathcal{O}_N(\pi_n(p)))$ with depth less than or equal to n. Without loss of generality suppose that $\varphi \in \mathcal{O}_N(\pi_n(p)) \setminus \mathcal{O}_N(\pi_n(q))$. But since $depth(\varphi) \leq n$, by applying Lemma 2.1 we obtain $\varphi \in \mathcal{O}_N(p) \setminus \mathcal{O}_N(q)$.

□

Let me remark at this point that compositionality w.r.t. *FINTREE* operators does not necessarily imply compositionality for the projection operators. For example, for $Act = \{a, b\}$ and the set of modal formulas:

$$\mathbb{O}_N = \{\top, \neg a\top \wedge \neg b\top, a \wedge \neg a^n\top \ (n \geq 2)\}$$

the corresponding process equivalence $=_N$ is a congruence w.r.t. *FINTREE* operators. However, it is not compositional for any projection operator since $b =_N a^\infty$ whereas $a \wedge \neg a^{n+1}\top \in \mathcal{O}_N(\pi_n(a^\infty)) \setminus \mathcal{O}_N(\pi_n(b))$.

The Corollary below could be proven from the Lemma 2.10 or simply by observing that projection operators are defined with rules satisfying certan congruence formats given in Chapter 3 (except for completed trace).

Corollary 2.11 *For $N \in \{T, CT, F, R, FT, RT, S, RS, B\}$, $=_N$ is compositional w.r.t. the projection operators.*

□

As for more general and elegant conditions, I will now present a hypothesis that I suspect is true. It would be a nice challenge to complement what has been proven in this thesis with a proof of the following statement:

Conjecture *An HML-definable equivalence $=_N$ is a congruence w.r.t. all the projection operators if the following properties holds for $\mathbb{O}_N$:*
$(\varphi \in \mathbb{O}_N \wedge (\varphi|_p = \forall_{i\in I}\varphi_i)) \Rightarrow \forall_{J\subseteq_{FIN} I}\ \varphi[\forall_{i\in J}\varphi_i]_p \in \mathbb{O}_N$
$(\varphi \in \mathbb{O}_N \wedge (\varphi|_p = \neg\psi)) \Rightarrow \varphi[\psi]_p \in \mathbb{O}_N$

4. The conditions in the definition of a regular equivalence are sufficient, but on the other hand too restrictive. In particular, failures and failure trace equivalences do not satisfy them. A generalization is needed in this case as well.

Chapter 3

Axiomatisation strategy for basic process equivalences

In this chapter I would like to present a problem, as well as its solution, that has been an inspiration for my research. Suppose we have a process equivalence $=_N$ and an arbitrary GSOS system G, possibly satisfying some additional restrictions. We would like to have an algorithm that would generate a sound and complete axiomatisation for all operators in Σ_G modulo $=_N$.

Input: GSOS system G
Output: A sound and (ground) complete axiomatisation for Σ_G modulo $=_N$, that is, an axiomatisation $\mathcal{T}_G$ that satisfies, for all closed terms s,t:

$$\mathcal{T}_G \models s = t \;\Leftrightarrow\; s =_N t$$

3.1 Axiomatisation strategy for bisimulation equivalence

The central algorithm (strategy) discussed here is the one presented in [2] for bisimulation semantics. It has two variants, the basic and alternative strategy. Strategies for other equivalences that I will discuss later are slight variations of it with a few axioms added, depending on the equivalence. I will present briefly the main idea of the algorithm, as much as it is necessary for general understanding and the forthcoming proofs.

First, let us recall the basic definitions concerning TSS formats used in the axiomatisation strategy. It is not necessary to digest the exact definitions of smooth, distinctive and discarding operations, the important thing is how an arbitrary operation that does not satisfy these additional restrictions is expressed with the "good" operations in the axiomatisation.

Definition 2. (Smooth GSOS rule / operation)
A GSOS transition rule is *smooth* if it is in the form:

$$\frac{\{x_i \xrightarrow{a_i} y_i | i \in I\} \cup \{x_i \overset{b_{ij}}{\not\rightarrow} | i \in K, 1 \leq j \leq n_i\}}{f(x_1,\ldots,x_l) \xrightarrow{c} C[\vec{x},\vec{y}]}$$

An operation is smooth if all its rules in the defining TSS are.

Definition 3. (Distinctive operation)
An operation f from a GSOS system is *distinctive* if:
- it is smooth,
- for each argument i, either all rules for i test i positively or none of them does,
- for each pair of different rules for f there is an argument for which both rules

have a positive antecedent, but with a different action.

The alternative strategy uses the notion of discarding operation in order to introduce an additional peeling law.

Definition 4. (Discarding rule / operation)
A GSOS rule is *discarding* if:
- it is smooth,
- for no argument i that is tested negatively, x_i occurs in the target.
An operation is discarding if all of its rules in the defining TSS are.

Before I will proceed with the description of the axiomatisation, there is one important property of an axiomatisation that has to be defined. It is crucial in the completeness proof as well as in analysis of term rewriting properties of the rulified axiomatisations.

Definition (Head normalizing property)
A term $t \in \mathcal{T}(\Sigma)$ is in *head normal form* if it is of the form $\Sigma_{i \in I} a_i t_i$. An axiomatisation $\mathcal{T}$ over Σ is *head normalizing* for t if there exists a Σ-term t' in head normal form such that $\mathcal{T} \vdash t = t'$.

The strategy takes as input an arbitrary TSS G in GSOS format and produces an axiomatisation $\mathcal{T}$. It proceeds in the following steps:

1. If G does not contain all *FINTREE* operations, then they are added to G. Axioms A1 - A4 are included in $\mathcal{T}$.
2. For each non-smooth operation f (if such exists) a new smooth operation f^c with higher arity is introduced so that the following axiom is sound:

$$f(x_1, ..., x_{ar(f)}) = f^c(x'_1, ..., x'_{ar(f^c)}) \text{ with } \forall i \exists j \mid x'_i = x_j$$

This equation (copying axiom) is included in the axiomatisation.
2a. In the alternative strategy the same kind of axiom is added for each operation f that is not both smooth and discarding, where f^c is a smooth discarding operation.
3. For each smooth (smooth and discarding in the alternative strategy) but not distinctive operation f add distinctive (and discarding - alternative strategy) operations $f_1, \ldots, f_n$ such that for each $\vec{x}$:

$$f(\vec{x}) = \sum_{i=1}^{n} f_i(\vec{x})$$

Functions f_i are obtained by simply partitioning the set of transition rules of f. Each subset from the partition gives rise to a new distinctive function. Thus, f is equal to the choice between f_i. The above equation (distinctifying axiom) is added to the axiomatisation.
4. Once all the fresh auxiliary operators with the corresponding axioms have been added to G and the output axiomatisation, axioms for distinctive operations are added according to the rules specified in [2]. These include:

- distributivity laws of the form:
$f(x_1, \ldots, x_i + y_i, \ldots, x_{ar(f)}) = f(x_1, \ldots, x_i, \ldots, x_{ar(f)}) + f(x_1, \ldots, y_i, \ldots, x_{ar(f)})$
- action laws (recall section 1.3 and the ∂_B^1 operator):
$f(P_1, \ldots, P_{ar(f)}) = a.C[x_1, \ldots, x_{ar(f)}]$, where $P_i \in \{\partial_{B_i}^1(x_i), a_i.x_i, x_i, 0\}$ for $\emptyset \subset B_i \subset Act$
- inaction laws:
$f(P_1, \ldots, P_{ar(f)}) = 0$, where $P_i \in \{a_i.x_i, b_i.x_i + y_i, x_i, 0\}$.
4a. The alternative strategy introduces the same distributivity and inaction axioms as above. The action axioms are of the same form except that one-step encapsulaton is not used, so $P_i \in \{a_i.x_i, x_i, 0\}$. Furthermore, an additional peeling law is introduced which takes the form:
$f(P_1, \ldots, b_i.x_i + y_i, \ldots, P_{ar(f)}) = f(P_1, \ldots, y_i, \ldots, P_{ar(f)})$, where $P_j \in \{a_j.x_j, x_j\}$

3.2 Completeness and possible extensions

Once an axiomatisation strategy for bisimulation equivalence has been defined, a natural question is whether we can use this result to obtain complete axiomatisations of other basic process equivalences. Our suspicions are justified since bisimulation is the finest of all basic process equivalences, and equations derived from its axiomatisation are valid in other equivalences. In particular, head normalization is preserved. Moreover, we have complete axiomatisations of *FINTREE* operators at our disposal. The following theorem indicates further conditions that have to be met in order to obtain a complete axiomatisation of arbitrary operations.

Theorem 3.1 *Suppose we have a head normalizing axiomatization $\mathcal{T}$ over a set of operators $\Sigma \cup FINTREE \cup \{\pi_n\}_{n\in\mathbb{N}}$ which is sound for an equivalence $=_N$. Moreover, $\mathcal{T}$ contains a sound and complete axiomatisation of FINTREE for $=_N$. If*
1) $=_N$ is a congruence for all operators in $\Sigma \cup FINTREE \cup \{\pi_n\}_{n\in\mathbb{N}}$, and
2) AIP is sound for $=_N$
then $\mathcal{T}$ + AIP is complete for $=_N$.

Proof Let $s, t \in \mathcal{T}(\Sigma)$ be closed terms such that $s =_N t$. Since $=_N$ is compositional for each projection π_n, we have $\pi_n(s) =_N \pi_n(t)$ for each $n \in \mathbb{N}$. Take any $n \in \mathbb{N}$. $\mathcal{T}$ is head normalizing for all terms, therefore it is easy to notice that $\mathcal{T} \models \pi_n(s) = s_n$ and $\mathcal{T} \models \pi_n(t) = t_n$ where $s_n, t_n \in FINTREE$. From the transitivity of $=_N$ and soundness of $\mathcal{T}$ for $=_N$ we obtain $s_n =_N t_n$. Since $\mathcal{T}$ contains a complete axiomatisation of *FINTREE* for $=_N$, $\mathcal{T} \models s_n = t_n$. Hence $\mathcal{T} \models \pi_n(s) = s_n = t_n = \pi_n(t)$. We have proved that for an arbitrary $n \in \mathbb{N}$, $\mathcal{T} \models \pi_n(s) = \pi_n(t)$. Thus $\mathcal{T} + AIP \models s = t$.

□

Together with the head normalizing property of the presented axiomatisation and the result from Chapter 2,hich yields soundness of AIP for all basic process equivalencess, the facts presented above imply that in order to obtain a sound

and complete axiomatisation for other basic process we only need now to have a TSS format that would ensure that the equivalence in question is a congruence for all the generated operators.

3.3 Congruence formats

In the remainder of this chapter I will give a short introduction of TSS formats for basic process equivalences such that operations generated by TSSs in these formats respect a given equivalence. The following overview is based on [3] and [10]. The congruence formats presented there are more general (ntyft/ntyxt or panth), for our purposes it is enough to consider GSOS and its subformats (we can always take intersection of GSOS and the most general congruence format for a given equivalence).

GSOS is a congruence format for bisimulation equivalence and ready simulation. Its positive variant (without negative premises) generates operations that respect trace and simulation equivalences. Therefore, GSOS (or positive GSOS) can be used as an input format while generating axiomatisation for the aforementioned equivalences. Before descibing more cumbersome formats for decorated trace semantics, I would like to mention completed trace equivalence, for which no general congurence format is known and moreover there is almost no hope to find one. The following example comes from [3].

A unary encapsulation operator ∂_B with $B \subseteq Act$ for a given process t behaves like t except that it cannot perform any action $b \in B$. It can be defined with the following rule scheme:

$$\frac{x \xrightarrow{a} y \ \ (a \notin B)}{\partial_B(x) \xrightarrow{a} y}$$

Now consider two process terms $a(b+c)$ and $ab+ac$. Clearly they are completed trace equivalent. However, since $a \in CT(\partial_{\{c\}}(ab+ac)) \setminus CT(\partial_{\{c\}}(a(b+c)))$ we have $\partial_{\{c\}}(a(b+c)) \neq_{CT} \partial_{\{c\}}(ab+ac)$. Thus $=_{CT}$ is not compositional w.r.t. $\partial_{\{c\}}$, in fact this is true if we can take any nontrivial subset B of Act as a set of forbidden actions for ∂_B. Observe that the predicate in the premise is a simplified notation denoting a set of rules, each one for different $a \in Act \setminus B$. Perhaps in an attempt to specify a congruence format for completed trace equivalence, we would have to use variables ranging over actions rather than actions themselves. However, it might be possible as well that no sensible general congruence format exists for this equivalence.

Bloom, Fokkink and van Glabbeek have obtained congruence formats for ready trace, failure trace, readiness and failures in [10]. It is another example of a result obtained with the analysis of modal characterization of processes with HML formulas. I will now present the congruence formats for decorated trace equivalences adapted into the setting of GSOS format.

Definifion (Propagation, polling)
An occurrence of a variable in a GSOS rule is *propagated* if the occurrence is

either in the target or in the left-land side of a positive premise whose right-hand side occurs in the target. An occurrence of a variable in a GSOS rule is *polled* if the variable occurs on the left-hand side of a positive premise and is not propagated (does not occur in the target).

The congruence formats use the notion of a floating variable, which may represent a running process. To this end we need to introduce a predicate Λ on arguments of function symbols. The interested reader is referred to [10] for details underlying these concepts.

Definition (Liquid and frozen arguments / floating variables)
Assume a predicate Λ on arguments of function symbols. If $\Lambda(f, i)$, then we say that an argument i of f is *liquid* (w.r.t Λ), otherwise it is *frozen*. A variable in a GSOS rule is Λ-*floating* if either it occurs as a right-hand side of a positive premise or it occurs in the source at a Λ-liquid position.

Definition (Decorated trace safe GSOS rules and decorated trace formats)
Assume a predicate Λ on arguments of function symbols. A GSOS rule is:
- Λ-*ready trace safe* if each Λ-floating variable is propagated at most once, and at a liquid position,
- Λ-*readiness safe* it is Λ-ready trace safe and each Λ-floating variable is not both propagated and polled,
- Λ-*failure trace safe* if it is Λ-readiness safe and each Λ-floating variable is polled at most once, at a liquid position in a positive premise.

A TSS is in *ready trace, readiness or failure trace format* if all its rules are Λ-ready trace safe, Λ-readiness safe or Λ-failure trace safe respectively for some predicate Λ.

Theorem 3.2 *If a TSS in ready trace/readiness preorder, then the operations that it defines respect ready race/readiness congruence. If a TSS is in failure trace format, then its operations respect failure trace and failures equivalence.*

□

In order to adapt the decorated trace formats in our axiomatisation strategy, we only need to make sure that the operations introduced by copying and distinctifying axioms are definable with decorated trace rules, provided that the whole input TSS is in one of the above formats.

Lemma 3.3 *Let $P = (\Sigma, R)$ be a TSS in GSOS format and also in N format for $N \in \{$ ready simulation, ready trace, readiness, failure trace $\}$. Then the TSS $P' = (\Sigma', R')$ with $\Sigma \subset \Sigma'$ and $R \subset R'$ which contains extra operations introduced by both strategies from [2] is also in GSOS N format.*

Proof. P' is obviously in GSOS format. Let $f' \in \Sigma' \setminus \Sigma$ be an operation introduced by one of the strategies. Then f' is either used as an auxiliary operator for a distinctifying or a copying axiom.

In the first case, the rules for f' are a subset of rules for some $g \in \Sigma$ which are given in N format. Therefore, rules that define f' are also in N format.
In the latter case, f' is obtained from some $f \in \Sigma$ by introducing extra variables so that $f(x_1, ..., x_m) = f'(x'_1, ..., x'_n)$ where $n > m$ and for each i there exists j: $x'_i = x_j$. The rules for f' are obtained by replacing occurrences of the x - variables by the corresponding x' - variables. For $i \neq j$, x_i and x_j are replaced by disjoint sets of corresponding x'_{k_i}, x'_{k_j}.

Consider a rule from R' defining f':

$$\frac{\bigcup_{i \in I}\{x'_i \xrightarrow{a_i} y_i\} \cup \bigcup_{i \in K}\{x'_i \overset{b_{i_j}}{\not\rightarrow}\}}{f'(x'_1, ..., x'_n) \xrightarrow{a} C[x', y]} \tag{1}$$

In both strategies, there is a surjective mapping $\sigma : \{1, ..., n\} \to \{1, ..., m\}$ and a corresponding rule from R of the form:

$$\frac{\bigcup_{i \in \sigma(I)}\{(x_{\sigma(i)}) \xrightarrow{a_i} y_i\} \cup \bigcup_{i \in \sigma(K)}\{x_{\sigma(i)} \overset{b_{i_j}}{\not\rightarrow}\}}{f(x_1, ..., x_m) \xrightarrow{a} C[\sigma(x'), y]} \tag{2}$$

Where $\sigma(x')$ denotes the vector $x_{\sigma(1)}, ..., x_{\sigma(n)}$.

Let us also define $\Lambda(f', i) \Leftrightarrow \Lambda(f, \sigma(i))$, so that an argument of the function f is liquid if and only if the corresponding argument in the f' function is also liquid.

Following are the proofs that the first rule preserves the syntactic restrictions of the second rule:

1) Ready trace:
Suppose rule 2 is in ready trace format. We have to prove that each Λ-floating variable has at most one propagated occurrence at a Λ-liquid position.
Let z be a Λ-floating variable. There are two possible cases. First, z can occur at the right-hand side of a positive premise, so it is one of y_i for some i. The target in rule 1 does not change occurrences of y - variables as compared to the rule 2, in which it occurred at most once. In the second case $z = x_i$ for some i and it occurs exactly once in the source, at a Λ-liquid position. Therefore $\Lambda(f, x_{\sigma(i)}) = \Lambda(f', i) = 1$ and $x_{\sigma(i)}$ has at most one propagated occurrence in rule 2. We can view the target of rule 2 as the same open term as the target of rule 1 with variables x_i substituted with $x_{\sigma(i)}$. So if $x_{\sigma(i)}$ has at most one occurrence in rule 2, then x_i must have at most one occurrence in rule 1 as well.

2) Readiness:
Ready trace format has already been proven. We have to show that no Λ-floating variable has both propagated and polled occurrences.
If z is a Λ-floating variable that has a polled occurrence, then $z = x_i$ for some i

Process equivalence	**Input SOS format**
trace	positive failure trace GSOS
completed trace	-
simulation	positive GSOS
failures	failure trace GSOS
readiness	readiness GSOS
failure trace	failure trace GSOS
ready trace	ready trace GSOS
ready simulation	GSOS
bisimulation	GSOS

Figure 1: Formats of TSSs for the axiomatisation strategies

(this is because the rule doesn't contain any lookahead). Since $x_{\sigma(i)}$ is also Λ-floating in rule 2, it doesn't have a propagated occurrence there. Thus, neither has x_i.

3) Failure trace:
If rule 2 is in failure trace format, then rule 1 is in readiness format. What remains to prove is that each Λ-floating variable has at most one polled occurrence, which must be at a Λ-liquid position in a positive premise. Actually, the second fact is immediate since the rule is in GSOS format, so the left-hand sides of premises are single variables.
As before, we derive the desired property from the fact that for a Λ-floating variable x_i, $x_{\sigma(i)}$ is also Λ-floating and therefore has at most one polled occurrence in rule 2. Since x_i cannot have more occurrences than $x_{\sigma(i)}$, we have proven that rule 1 is in failure trace format.

Corollary 3.4 *For $P = (\Sigma, R)$ a TSS in GSOS N format for some $N \in \{$ ready simulation, ready trace, readiness, failure trace $\}$, there exists an algorithm (strategy) to produce a sound and complete axiomatisation. It is a variation of the axiomatisation strategy from [2], which adds to the set of produced axioms a finite number of specific axioms for each equivalence, according to Theorem 1.2.*

□

Chapter 4

Term rewriting properties of the generated axiomatisations

Term rewriting properties of axiomatisations generated by the alternative strategy from [2] were studied by D.J.B. Bosscher in [12]. He provided a rulified axiomatisation which is a term rewriting system based on the generated axioms, and also gave a rewriting strategy such that every well-founded term (representing a well-founded process) is rewritten to a normal form which is unique modulo associativity and commutativity of $+$, for all bisimilar process terms. In this chapter I will try to use this result to obtain term rewriting systems for other equivalences, normalising and possibly confluent.

4.1 TRS and normalizing strategy for bisimulation

First I will present a summary of the term rewriting system and arewriting strategy for it which yields normalisation. These have been presented in Bosscher's paper [12].

Definition (Rulified axiomatisation)
Suppose we have an axiomatisation $\mathcal{T}$ generated by the alternative strategy for some GSOS system G. A *rulified axiomatisation* $\rightarrow$ consists of the following rewrite rules:

(1) $x + x \rightarrow x$
(2) $x + 0 \rightarrow x$
(3) $f(x_1, \ldots, x_{ar(f)}) \rightarrow f^c(x'_1, \ldots, x'_{ar(f^c)})$ for each copying axiom in $\mathcal{T}$
(4) $f(x_1, \ldots, x_{ar(f)}) \rightarrow \Sigma_{i=1}^{p} f_i(x_1, \ldots, x_{ar(f)})$ for each distinctifying axiom in $\mathcal{T}$
(5) $f(P_1, \ldots, P_{ar(f)}) \rightarrow a.C[x_1, \ldots, x_{ar(f)}]$ where $P_i \in \{a_i.x_i, x_i, 0\}$ for each action axiom in $\mathcal{T}$
(6) $f(x_1, \ldots, x_i + y_i, \ldots, x_{ar(f)}) \rightarrow$
$f(x_1, \ldots, x_i \ldots, x_{ar(f)}) + f(x_1, \ldots, y_i, \ldots, x_{ar(f)})$
for each distributivity axiom in $\mathcal{T}$
(7) $f(P_1, \ldots, P_{ar(f)}) \rightarrow 0$ where $P_i \in \{a_i.x_i, b_i.x_i + y_i, x_i, 0\}$ for each inaction axiom in $\mathcal{T}$
(8) $f(P_1, \ldots, b_i.x_i + y_i, \ldots, P_{ar(f)}) \rightarrow f(P_1, \ldots, y_i, \ldots, P_{ar(f)})$ where $P_j \in \{a_j.x_j, x_j\}$ for each peeling axiom in $\mathcal{T}$
(9) $f(P_1, \ldots, b_i.x_i, \ldots, P_{ar(f)}) \rightarrow 0$ whenever there exists a rule $f(P_1, \ldots, b_i.x_i + y_i, \ldots, P_{ar(f)}) \rightarrow 0$
(10) $f(P_1, \ldots, b_i.x_i, \ldots, P_{ar(f)}) \rightarrow f(P_1, \ldots, 0, \ldots, P_{ar(f)})$ whenever there exists a rule
$f(P_1, \ldots, b_i.x_i + y_i, \ldots, P_{ar(f)}) \rightarrow f(P_1, \ldots, y_i, \ldots, P_{ar(f)})$

The above rewrite rules are mostly directed versions of equations from $\mathcal{T}$. There are some exceptions, though. Firstly, we do not include the rules for commutativity and associativity, because the resulting TRS would not be terminating. Thus the rewriting is done for equivalence classes modulo these two laws for $+$. Furthermore, rules (9) and (10) are added to obtain a confluent TRS in situations when we would need a rule $x \to x + 0$ to be able to use the inaction axiom. For a rulified axiomatisation obtained in this way Bosscher proposed a rewriting strategy that works as follows:

1. Contract all non-action redexes, repeat this step until no more non-action redexes are left.
2. Contract the outermost redex surrounded by the least number of action prefixing operations.
3. Repeat the whole procedure from point 1.

Furthermore, he has proved that this strategy is head normalizing and rewrites each pair of bisimilar well-founded terms to a unique normal form which consist only of *FINTREE* operators.

4.2 TRSs for other equivalences and their properties

A simple variation of the original strategy for bisimulation provides us with a weakly normalising TRS for the other equivalences. Namely, after the strategy given above terminates, producing a normal form which consists only of *FINTREE* operations, we introduce a subsequent step depending on the chosen equivalence $=_N$:

4. Contract the redex according to the rules based on additional one or two axioms (apart from E_A), which are unique for every equivalence $=_N$. The two rules for *FINTREE* ($x + x \to x, x + 0 \to x$) may also be used. Repeat until no more redexes are left.

Definition (Rulified axiomatisations for basic process equivalences)
For an equivalence $=_N$ where $N \in \{T, CT, F, R, FT, RT, S, RS\}$, its *rulified axiomatisation* $\to_N$ consists of rules (1) - (10) of $\to$ with an addition of the following one or two rules according to Figure 2.

I will include completed trace equivalence in the considerations, even though no axiomatisation strategy has been defined for this equivalence. Since it is a congruence with respect to *FINTREE* and has a complete axiomatisation for these basic operators, we can analyse term rewriting properties of the rulified axioms for *FINTREE* only.

Having defined the rulified axiomatisations, we would like to know whether the resulting TRSs will be confluent and terminating. While it is easy to observe that termination holds for all equivalences, some of the obained TRSs are not

Process equivalence	Additional rewrite rules
ready simulation	$a(x+by+bz)+a(x+by) \rightarrow_S a(x+by+bz)$
ready trace	$I(x)=I(y) \Rightarrow ax+ay \rightarrow_{RT} a(x+y)$
failure trace	$I(x)=I(y) \Rightarrow ax+ay \rightarrow_{FT} a(x+y)$, $ax+ay+a(x+y) \rightarrow_{FT} ax+ay$
readiness	$a(bx+by+u)+a(by+v) \rightarrow_R a(bx+u)+a(by+v)$
failures	$a(bx+by+u)+a(by+v) \rightarrow_F a(bx+u)+a(by+v)$, $ax+a(x+y)+a(y+z) \rightarrow_F ax+a(y+z)$
completed trace	$a(bx+u)+a(cy+v) \rightarrow_{CT} a(bx+cy+u+v)$
simulation	$a(x+y)+ay \rightarrow_S a(x+y)$
trace	$ax+ay \rightarrow_T a(x+y)$

Figure 2: Additional rewrite rules for rulified axiomatisations

confluent.

Before I will present the main result of this chapter, I need to make certain definitions clear. I will call well-founded any term that represents a well-founded process, and a well-founded TSS will stand for a TSS that generates only well-founded processes [1] (observe that such processes are bisimilar to process terms consisting of *FINTREE* operators only). I will denote equality modulo associativity and commutativity of + with $=_{AC}$.

Theorem 4.1 *For each equivalence* $=_N$ *where* $N \in \{T, S, CT, F, R, FT, RT, RS\}$, $\rightarrow_N$ *is terminating for* $t \in FINTREE$.

Proof We can observe that each of the additional rewrite rules decreases the number of summands at some level, possibly increasing their number at an outer (higher) level (by level I mean the number of nested action prefixing operations). However, the number of levels remains constant. Therefore, the summands disappear or are "pushed" to the outer term. This can't take forever since the number of levels is constant.

□

Corollary 4.2 *For each equivalence* $=_N$ *where* $N \in \{T, S, CT, F, R, FT, RT, RS\}$, $\rightarrow_N$ *is normalising for a well-founded term* t.

□

Theorem 4.3 *The rulified axiomatisations of well-founded GSOS systems are confluent for trace, completed trace, ready trace and bisimulation equivalence. In other cases, namely simulation, failures, readiness, failure trace and ready simulation we obtain a non-confluent term rewriting system.*

[1] In [12] there are notions of syntactic and semantic well-foundedness. In this paper I do not deal at all with the former and well-founded always means semantically well-founded.

Proof *Trace.* Any strategy leads to construction of a "trace-normal" term which is 0 or

$$\sum_{i \in I} a_i.t_i$$

where $i \neq j \Rightarrow a_i \neq a_j$ and all t_i are in trace-normal form. The set of all such terms is isomorphic to the set of all prefix trees.

Now, as I have already proven normalisation for well-founded terms, it suffices to show that any two normal forms s and t representing trace equivalent terms are equal modulo commutativity and associativity of $+$. I will proceed with induction on the size of a normal form. The base case when $|s| = |t| = 0$ is trivial . For larger sizes, we can't yet assume that s and t have the same size, so let the induction hypothesis be that if $s =_T t$ and $max\{|s|, |t|\} \leq K$ then $s =_{AC} t$. Take s and t such that $max\{|s|, |t|\} = K + 1$. We have: $t = \sum_{i \in I} a_i.t_i$ and $s = \sum_{j \in J} a_j.s_j$ for some finite sets of indexes I and J.

Now suppose, on the contrary, that $s \neq_{AC} t$. Since s and t satisfy the same set of trace formulas and for each $a \in I(s) = I(t)$, $a_i = a$ and $a_j = a$ for exactly one i and j respectively. So if s and t are different modulo AC, then the difference must be on some of the terms t_i and s_j, namely there exist $i \in I, j \in J$ such that $a_i = a_j = a$ and $t_i \neq_{AC} s_j$ and as a consequence of the induction hypothesis $t_i \neq_T s_j$. On the other hand, we have $\mathcal{O}_T(s) = \mathcal{O}_T(t)$, so in particular (*) $\{a\varphi \mid a\varphi \in \mathcal{O}_T(s)\} = \{a\varphi \mid a\varphi \in \mathcal{O}_T(t)\}$.
But since $\{a\varphi \mid a\varphi \in \mathcal{O}_T(s)\} = \{a\varphi \mid \varphi \in \mathcal{O}_T(s_i)\}$ and $\{a\varphi \mid a\varphi \in \mathcal{O}_T(t)\} = \{a\varphi \mid \varphi \in \mathcal{O}_T(t_j)\}$, we obtain $\{a\varphi \mid a\varphi \in \mathcal{O}_T(s)\} \neq \{a\varphi \mid a\varphi \in \mathcal{O}_T(t)\}$, contradiction to (*).

Completed trace. In this case normal forms are 0 and terms of the form:

$$\sum_{i \in I} a_i.t_i$$

where $i \neq j \Rightarrow$ ($a_i \neq a_j$ or ($t_i = 0$ and $t_j \neq 0$) or ($t_i \neq 0$ and $t_j = 0$)) and all t_i are in completed trace-normal form.
The proof can be done in the same way as in the case of trace equivalence. The only difference is that a normal form can now contain two summands with the same prefix, $a.0$ and $a.t$ where $t \neq 0$. We proceed with induction on the size of formula as before and again deduce that for two completed trace equivalent terms difference is only possible on the deeper level but there we have completed trace equivalent terms and the induction hypothesis yields equality modulo AC.

Ready trace. Normal forms are 0 and terms of the form:

$$\sum_{i \in I} a_i.t_i$$

where $i \neq j \Rightarrow$ ($a_i \neq a_j$ or $I(t_i) \neq I(t_j)$)
This is a generalization of the previous two normal forms. Again we use induction on the complexity of a term. Observe first that for each well-founded term in normal form t, action prefix a and set of initials X, if there is a summand of t of the form $a.t_i$ with $I(t_i) = X$ then it is unique. As a consequence, for any

other summand $a.t_j$ with $i \neq j$, $RT(t_i) \cap RT(t_j) = \{\top\}$. Therefore in any term s such that $s =_{RS} t$ for each summand $a_i.t_i$ of t there must be a unique corresponding summand $a.s_j$ of s such that $RT(t_i) = RT(s_j)$. We have $t_i =_{AC} s_j$ from the induction hypothesis. Therefore for all normal forms s ready trace equivalent to t we obtain $s =_{AC} t$.

Bisimulation. Confluence of the rulified axiomatisation $\rightarrow_B$ has already been proved in [12].

Now I will prove that for other equivalences confluence does not hold.

Simulation: Terms

$$a(b(c+d)) + bc$$
$$a(b(c+d))$$

are both normal forms representing similar terms, thus $\rightarrow_S$ is not confluent.

We could try to overcome this obstacle by involving sequential composition instead of action prefixing in the rewrite rules. This would allow us to compare sequences of any arbitrary length. For example, for the two above terms a rule $x;(y+z)+x;y \rightarrow_S x;(y+z)$ would work (provided that we would define sensible rules for ;). However, we still face a problem with the choice operator. There can be an arbitrary number of alternating + and action prefixing/sequential composition operations in a path to a leaf of a term (in one of the execution paths of a process). Consider two similar process terms:

$$b + a\underbrace{(b + a(b + a(\ldots(b + a)\ldots)))}_{(n-1)\ brackets}$$
$$b + a\underbrace{(b + a(b + a(\ldots(b + a)\ldots)))}_{(n-1)\ brackets} + a^n$$

I will present a proof sketch and argue that there exists no term rewriting system involving action prefixing and/or sequential composition, choice operator and 0 that would rewrite each of the above terms to a unique form for each $n \in \mathbb{N}$. In a finite set of rules there is a boundary on the number of nested choice operators in a term (I will call this value a depth), say K. Now suppose that we have two terms as above with $n = K+1$. Suppose further that they are rewritten to a unique form $\sum_{i=1}^{p} t_i$ with $p \geq 1$. We can distinguish two cases; either there is only one summand t_i of depth $K+1$ or there are at least two of them. In the first case the second term would be rewritten to a form with only one summand of depth $K+1$, so the second summand a^{K+1} would disappear at some point of the reduction. Let $l \rightarrow_S r$ be the rule applying which we can rewrite the term $\sum_{i=1}^{p} t'_i$ with two or more summands of depth $K+1$ to a term with only one summand of depth $K+1$ and other possible summands with lower depth. It is rather easy to see that it cannot be applied at position other than ϵ (root position); this is because no term is similar to a term of a lower depth. Consider the case when $l \rightarrow_S r$ is applied at the root position and let t'_{i_0} be a

term of depth $K+1$ that would disappear by applying $l \to_S r$. Take a leaf of t'_{i_0} with depth $K+1$ an add a term $a(a+b)$ at the end of it. Call the resulting term t'' and consider a term obtained from $\sum_{i=1}^{p} t'_i$ by replacing t'_{i_0} with t''. Its depth is $K+2$ and thus is obviously not similar to our redex $\sum_{i=1}^{p} t'_i$. However, this term would be rewritten with $l \to_S r$ to the same term as the result of applying $l \to_S r$ to $\sum_{i=1}^{p} t'_i$. The same argument can be used in the second case where the unique normal form would have two or more summands of depth $K+1$.

Ready simulation: The counterexample is a slightly modified version of the one that I used in case of simulation. The two following terms are ready simulation equivalent.

$$\underbrace{ab + a(ab + a(ab + a(\dots(ab + aa)\dots)))}_{(n-1)\ brackets}$$
$$\underbrace{ab + a(ab + a(ab + a(\dots(ab + aa)\dots)))}_{(n-1)\ brackets} + a^n$$

The reasoning follows the same scheme as in the previous case. In general, two terms $t+s$ and t are equivalent because $\mathcal{O}_N(s) \subseteq \mathcal{O}_N(t)$ (s is a "redundant" summand). However, we cannot reduce $t+s$ to t without the knowledge about at least $K+1$ levels of s because for any position p of t with length less than K, there exists no q such that $t|_p =_N s|_q$.

For the remaining equivalences, I will only provide counterexamples for the rulified axiomatisations defined in this paper, without a general proof that there cannot exist a confluent TRS.

Readiness and failures: Terms

$$a(bc+d) + a(b+e)$$
$$a(b+d) + a(bc+e)$$

are readiness and failure equivalent and both are normal forms for $\to_R$ and $\to_F$. We can consider a more general form of these terms:

$$t_{n+1} = a(bs_n + d) + a(b+e)$$
$$s_{n+1} = a(b+d) + a(bt_n + e)$$
$$t_0 = t'_0 = c$$

Failure trace: Below we have a term in a normal form. The underlined subterm has all its failure trace modal formulas contained in the other two summands:

$$a(a^n + a) + ab + \underline{a(a^n + b)} \text{ for } n \geq 1$$

Therefore, the above term is failure trace equivalent to the following:

$$a(a^n + a) + ab$$

□

Results of this chapter are summarized in Figure 3.

Process equivalence	**Term rewriting system**
trace	confluent
completed trace	confluent
simulation	not confluent / confluent TRS does not exist
failures	not confluent / no confluent TRS known
readiness	not confluent / no confluent TRS known
failure trace	not confluent / no confluent TRS known
ready trace	confluent
ready simulation	not confluent / confluent TRS does not exist
bisimulation	confluent

Figure 3: Term rewriting properties of the rulified axiomatisations

Bibliography

[1] L. Aceto, *Deriving Complete Inference Systems for a Class of GSOS Languages Generating Regular Behaviours.* In (B. Jonsson, J. Parrow eds) Proceedings CONCUR 94, Uppsala, Sweden, Vol. 836 of Lecture Notes in Computer Science, Springer-Verlag, pp. 449-464 (1994)

[2] L. Aceto, B. Bloom and F. Vaandrager *Turning SOS Rules into Equations.* Information and Computation, 111, pp. 1–52 (1994)

[3] L.Aceto, W. Fokkink, C. Verhoef, *Structural Operational Semantics.* In (J.A. Bergstra, A. Ponse and S.A. Smolka, eds) Handbook of Process Algebra, pp. 197-292, Elsevier (February 2001)

[4] F. Baader and T. Nipkov, *Term Rewriting and All That.* Cambridge University Press (1998)

[5] J.C.M. Baeten, J.A. Bergstra and J.W. Klop, *On the consistency of Koomen's fair abstraction rule.* Theoretical Computer Science 51(1/2), pp. 129-176 (1987)

[6] J.C.M. Baeten, J.A. Bergstra and J.W. Klop, *Ready-trace semantics for concrete process algebra with the priority operator.* Computer Journal 30(6),pp. 48-506 (1987)

[7] J.C.M. Baeten and W.P. Weijland, *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press (1990)

[8] A. Barros and T. Hou, *A Constructive Version of AIP Revisited.* Electronic report PRG0802, Programming Research Group, University of Amsterdam (January 2008)

[9] S. Blom, W. Fokkink, S. Nain, *On the axiomatizability of Ready Traces, Ready Simulation and Failure Traces.* ICALP'03, Eindhoven, Lecture Notes in Computer Science 2719, pp. 109-118, Springer (June 2003)

[10] B. Bloom, W.J. Fokkink and R.J. van Glabbeek, *Precongruence formats for decorated trace semantics.* ACM Transactions on Computational Logic 5(1):26-78 (January 2004)

[11] B. Bloom, S. Istrail and A.R. Meyer, *Bisimulation can't be traced.* Journal of the ACM 42(1), pp.232-268 (1995)

[12] D.J.B. Bosscher, *Term Rewriting Properties of SOS Axiomatisations.* Lecture Notes In Computer Science; Vol. 789 (1994)

[13] W. Fokkink, *Introduction to Process Algebra.* Texts in Theoretical Computer Science, An EATCS Series. Springer (January 2000)

[14] R.J. van Glabbeek, *Bounded Nondeterminism and the Approximation Induction Principle in Process Algebra.* STACS 1987: 336-347

[15] R.J. van Glabbeek, *The Linear Time - Branching Time Spectrum I.* In (J.A. Bergstra, A. Ponse and S.A. Smolka, eds) Handbook of Process Algebra, pp. 3-99, Elsevier (February 2001).

[16] M. Hennessy and R. Milner, *On observing nondeterminism and concurrency.* In (J.W. de Bakker and J. van Leeuwen, eds) Proceedings 7th ICALP, Noorwijkerhout, LNCS 85, Springer, pp. 299-309 (1980)

[17] M. Hennessy and R. Milner, *Algebraic laws for nondeterminism and concurrency.* Journal of the ACM 32(1), pp. 137-161 (1985)

[18] S. Mauw, *A constructive version of the approximation induction principle.* Proc. SION Conf. CSN 87, pp. 235-252 (1987)