Università degli Studi di Udine
Dipartimento di Matematica e Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Computing in Non Standard Set Theories

Candidate:
Carla Piazza

Supervisor:
Prof. Alberto Policriti

December 20, 2001

to my mother

Acknowledgments

There are so many people to thank, I hope not to forget any of them.

First of all I thank Alberto Policriti. Alberto was a constant guide and he was patient with me during these four years of Ph.D. studies, always showing his confidence in me. Beside the great amount of technical knowledge, he taught me that you need to have both good intuitions, and the determination to work them out.

I owe the beginning of my Ph.D. and my first steps in the scientific research to Gianfranco Rossi, my Master thesis supervisor.

I am grateful to Agostino Dovier, co-author of many important papers, who always encourage me during hard times. Agostino together with Antonella have always been a reference point for any kind of problems, also outside the university life.

Johan van Benthem and Yde Venema helped me to enrich my knowledge during the period I spent working with them at the ILLC in Amsterdam.

Writing a paper with me is not always easy, so I wish to thank two more of my co-authors, Enrico Pontelli and Raffaella Gentilini.

Many people in the Computer Science department of Udine assisted me in different ways, among them in particular Moreno Falaschi, Angelo Montantari, and Giovanna D'Agostino.

I am in debt for the warm hospitality both with the ILLC staff and with the group at the Computer Science department of Valencia.

I will never forget my colleagues and friends in Udine, Luca Di Gaspero, Massimo Franceschet and Alicia Villanueva, Paolo Coppola and Ivan Scagnetto, and all the others, who were close to me and often had to help me in solving the weirdest problems.

Special thanks to my mother: she endlessly pushed me to go forth and she taught me to never fear anything.

Special thanks go to Massimo as well, who helped me in everything, patiently waited for me, and followed me around the world.

A grateful thought goes to Giovanna Dall'Aglio, Patrizia Levati and all my friends in Parma, who saw me coming and going like a meteor in the last four years, and never forgot about me.

To conclude, many thanks to all the people who have been stressed and contributed in writing this thesis, especially this last difficult part.

Udine, December 2001.

Carla Piazza

Abstract

Computable Set Theory investigates the decidability of fragments of well-founded set theories. Phenomena where circularity plays a central role cannot be encoded in well-founded set theories without introducing unnatural extra complexities. The same phenomena can instead be described immediately making use of non-well-founded sets. In this thesis we study decidability and algorithmic problems in the context of non-well-founded set theories.

In the first part of the thesis we mainly concentrate on the decidability of fragments of non-well-founded set theories which are strongly connected with Modal Logic. We prove new decidability results and we provide general strategies which turn out to be useful in studying different fragments.

In the second part of the thesis we deal with the algorithmic problems of determining the maximum bisimulation and simulation relations over a finite non-well-founded structure (set). These problems are well-known in the area of Model Checking, where bisimulations and simulations constitute a possible approach to the so-called *state explosion problem*. We present two algorithms which reduce the time/space complexity of the best algorithms in the literature and we propose an alternative use of symbolic data-structures in the representations of graphs.

Contents

1

Introduction

There is no use trying, said Alice; one can't believe impossible things. I dare say you haven't had much practice, said the Queen. When I was your age, I always did it for half an hour a day. Why, sometimes I've believed as many as six impossible things before breakfast. (L. Carroll)

1.1 Leit-Motiv

The main aim of this thesis is an analysis of decision problems, finite (small) model properties, and algorithms in the context of non standard set theories. This analysis is performed in order to establish generalizations and systematic classifications of set-theoretic decidability results to facilitate their implementations and uses, and to obtain efficient algorithms.

When we talk about *Set Theory* we have in mind a theory which captures the main characteristics of an *intended model of sets*. In the literature the expression *Set Theory* usually refers to the Zermelo-Fraenkel Set Theory, ZF, or to fragments[1] of it, see [Kun80]. In more general terms, we can consider as *set languages* all the first order languages with at least one binary relational symbol $\in$ (the membership predicate), and as *set theories* all the theories on a set language, which define all the other symbols of the language in terms of the membership predicate. These set theories are addressed as *non standard set theories*. There are non standard set theories which are famous, and among these *non-well-founded set theories*. We talk about *non-well-founded set theories* each time we are using a set theory without the Regularity axiom (see Figure 1.1) which avoids the membership relation to be cyclic.

$$\boxed{\forall x\,\exists z\,\forall y\,(y \in x \rightarrow (z \in x \wedge y \notin z))}$$

Figure 1.1: The Regularity Axiom.

[1] Where some of the more powerful axioms, (e.g. Comprehension schema or Infinity axiom) are removed.

A rather natural approach to solve decision problems in set theories could be the one adopted for the Classical Decision Problem for First-Order Logic. For First-Order Logic classifications which discriminate between classes of decidable and undecidable formulae have been proposed, see [BGG97]. However, these classifications are principally based on the quantification prefixes of the formulae. Hence, they turn out to be almost unusable in the context of first order theories (not only set theories).

The *Decision Problem for fragments of Set Theory* has been actively investigated in the last two decades. The original goal was the design and implementation of an inferential engine for theorem proving. [FOS80] can be considered as the first seminal paper in the area of *Computable Set Theory*, while [CFO89] gives a complete account of the state-of-the-art at the end of the eighties and [COP01] presents the most recent results in the field.

These attempts attracted attention from many computer science research areas interested in the definition of programming languages able to provide data-structures powerful enough to simplify the implementation of algorithms and, at the same time, controllable from a computational point of view. In recent years, many proposals in the field of *specification languages* [Spi92, Abr96], *logic-based languages* [JP89, LL91, Ger97, CL96, HL89, Llo99, Smo95, DPPR00], and *database languages* [BNST91, AG91] devoted much more attention to the presence of sets.

For these reasons we can say that the area of Computable Set Theory is placed and establishes a bridge between the area of logic which study expressivity, consistency, constructibility, forcing, large cardinals and determinacy problems on different set theories [Qui67, For95, Lev79, Fri00, Jec01] and the area of computer science which uses propositional calculus and fragments of First-Order Logic in the development of logic and constraint programming languages [SES94, MS99].

The majority of the decidability results obtained in set theories until now, see [CFO89, COP01], make strong use of the Regularity axiom and sometimes of restricted Comprehension schemas. However, well-founded sets do not allow to capture in a natural way the concept of circularity. Phenomena where circularity plays a central role cannot be encoded in well-founded set theories without introducing unnatural extra complexities. The same phenomena can instead be described immediately making use of non-well-founded sets, also known as hypersets, see [BM91]. Let us consider, for example, the notion of graph: if we work in a well-founded set theory, this requires the formalization of the concepts of ordered pair and binary relation, while can be defined immediately (a graph is a non-well-founded set) in a non-well-founded set theory.

The two applications which have mainly motivated our interest in non-well-founded sets are: the translation of the notions of frame validity and derivability of Modal Logic and the notion of *bisimulation* which plays a central role in Model Checking and in many areas of Computer Science.

In [DMP95] a translation method to support derivability in Propositional Modal Logic is defined. In this translation method, called $\Box$-as-$\wp$, sets are used to model frames and the membership relation to render the accessibility relation. Since there are no restriction about the circularity of the accessibility relation, the natural notion of set to be used is the non-well-founded one. It is immediate to observe that in this context an analysis and a classification of decidability results in non-well-founded set theories could lead to a generalization and a systematization of decidability results obtained in different modal theories using ad hoc techniques.

The main disadvantage of Model Checking is the so-called *state explosion* that can occur if the system being verified has many components that can make transitions in parallel. In this case the number of global system states may grow exponentially with the number of processes. The main techniques which are at the moment under study to solve the state explosion problem are: Symbolic Model Checking [McM93], Abstract Model Checking [Dam96], partial order reductions [CGP99], equivalence reductions [LY92, HHK95]. In particular, equivalence relations or preoders on transition systems are introduced with the aim of obtain a smaller structure which satisfies the same properties of the starting one. Among these relations the most famous are the *bisimulation equivalence* [Par81] and the *simulation preorder* [Mil71]. The former guarantees strong preservation of branching-time temporal logics such as CTL and CTL* [CE82]. The latter guarantees weak preservation of the universal fragment of these logics (ACTL and ACTL* [GL94]), see Section 1.2.8. On non-well-founded sets the notion of bisimulation is the operational counterpart of the anti-foundation axiom (AFA): two graphs represent the same set if and only if they are bisimilar. The notion of simulation, to the best of our knowledge, has never been considered in the area of non-well-founded sets.

In particular in this dissertation we focus our attention on the following issues:

1. point out the main differences between well-founded and non-well-founded set theories; while the first are familiar to mathematicians and computer scientists, the second have yet the capability to amaze us with odd phenomena;

2. establish relationships between expressive power and decidability by an analysis of the role played by axioms, operators, and quantifications in different classes of formulae;

3. supply instruments useful both in the development of theorem provers and model checkers; while the material in the first part of this thesis can be useful for theorem provers, in the algorithms presented in the second part we put more attention on model checkers.

It is outside the scope of this thesis to:

1. analyse exhaustively all the existent non-well-founded set theories; whenever it has been possible we have decided to do not assume any form of anti-foundation in order to obtain results useful in as many as possible non-well-founded set theories; when it has been necessary to assume some form of anti-foundation our choice has been to use (AFA) which seems to be the one that has more links with different fields of computer science;

2. give a complete classification of decidability/undecidability results; we have rather turned our attention to make the dependence between decidability results and axioms/operators used stand out; we preferred to direct our efforts towards classes of formulae significant in the context of the $\Box$-as-$\wp$ translation in order to study how much of the *robustly decidability* of Modal Logic move also to set theories;

3. present results optimal from the computational point of view; many of the procedures defined in the part about tableau methods are open to improvements which nevertheless would introduce a lot of technical details and distract from the main motivations behind the decidability results; the encodings presented in the (bi)simulation part allow to

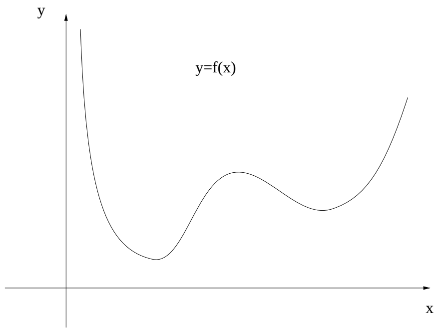
compute bisimulation and simulation with complexities greater than the ones of the algorithms usually in use, but provide new ideas concerning the usage of OBDD's to represent binary relations.

Looking more closely to the subject of this thesis first of all I think that set theories supply us with a framework which is general enough and easy to manipulate. The restriction to well-founded sets is unnatural in many applications: if we want to use membership to model a generic binary relation, we cannot impose to this relation to be acyclic. Just to give an intuitive example let us assume that we want to model the *friendship* relation:

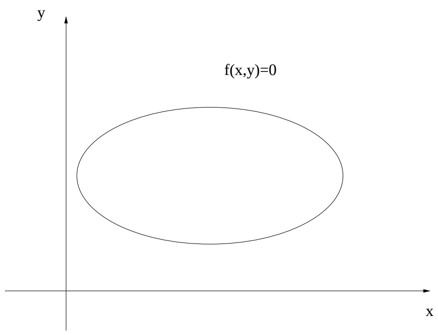
$$m \in n \quad \Leftrightarrow \quad m \text{ is a friend of } n.$$

We cannot impose that if m *is a friend of* n, then n cannot *be a friend of* m. On the other hand, modelling this relation with the membership relation it is immediate to reason about *common friends* (intersection), *false friends* ($m \in n \wedge n \notin m$), and so on. There are plenty of other examples coming from Computer Science. If we want to use the membership relation to model *communications* between processes, we cannot impose a restriction of the form *if A sends a message to B, then B cannot send a message to A.* Hence, we must use non-well-founded sets. When we consider these examples it is important that we keep in mind that we do not want to replace a binary relation with the membership relation $\in$ only to save symbols: this replacement implies the attribution to the binary relation of a semantics which makes us able to reformulate questions in terms of basic set operations and hence to find immediate solutions.

In my opinion the transfer from well-founded sets to non-well-founded sets is similar to the transfer from the study of the graphs of functions of the form $y = f(x)$



to the study of the graphs of equations of the form $f(x, y) = 0$:



the graph of a function $y = f(x)$ cannot pass in a point twice (think to its parametric version w.r.t. to the time t), similarly well-founded sets are represented as acyclic graphs; the graph of an equation $f(x, y) = 0$ can go back to an already visited point, similarly non-well-founded set are represented as cyclic graphs.

But my opinion has not as much credit as the one of Barwise which in his introduction to [Acz88] wrote:

> Just as there used to be complaints about referring to complex numbers as numbers, so there are objections to referring to non-well-founded sets as sets...

On the one hand it is important to understand which notions benefit from a reformulation in set-terms, on the other hand, just because in many cases such reformulations requires the use of non-well-founded set theories, it is necessary to study the properties and the behaviors of non-well-founded set theories. There are some interesting and basic facts about non-well-founded sets that make them so different from well-founded sets. In order to show some of these facts we fix here, once and for all, the notation we will use in the rest of the thesis concerning the representation of non-well-founded sets as graphs.

Definition 1.1 *A graph G is a pair $G = \langle N, \gtrdot \rangle$ in which:*

- *N is the set of nodes[2];*
- *$\gtrdot \subseteq N \times N$ is the set of edges.*

We will use $\lessdot$ to refer to the relation $\gtrdot^{-1}$.

In the usual representation of graph with points and arrows we have

$$n \gtrdot m \quad \Leftrightarrow \quad n \rightarrow m.$$

Each graph $G = \langle N, \gtrdot \rangle$ represents non-well-founded sets in the following sense:

- each node $n \in N$ represents a set;
- the elements of n are all the nodes m which are directly reachable from n, i.e.:

$$m \in n \quad \Leftrightarrow \quad m \lessdot n \quad \Leftrightarrow \quad m \leftarrow n$$

Notice that $\in$ and $\lessdot$ (which are the same) are stylized versions of the arrow $\leftarrow$, similarly to $\int$ which is a stylized (stretched) version of $\sum$.

A detailed explanation on how graphs represent sets and on the relationships between this representation and a representation with systems of equations can be found in [Acz88]. In Figure 1.2 are shown a non-well-founded set (on the left) and a well-founded set (on the right).

In both of them m is an element of n. In particular, the first represents the solution of the system

$$\begin{cases} m = \{n\} \\ n = \{\emptyset\} \cup \{m\} \end{cases}$$

[2] N can be also infinite at this point.

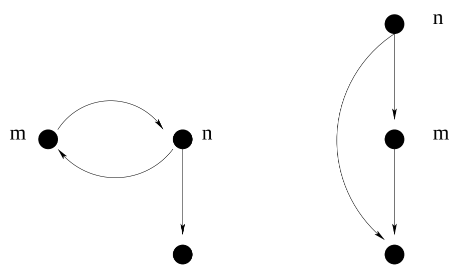


Figure 1.2: Examples of sets.

while the second represents

$$\begin{cases} m = \{\emptyset\} \\ n = \{\emptyset\} \cup \{\{\emptyset\}\} \end{cases}$$

Coming back to the properties which hold on well-founded sets and do not hold on non-well-founded sets, since well-founded set are represented by acyclic graphs, it is quite natural to introduce the notion of rank of a well-founded set, namely the length of the longest path from the node to a leaf. An important property of the notion of rank in the well-founded case is that for each finite rank i there are only a finite number of sets of rank i. In Figure 1.3 are shown all the possible sets of rank 2.

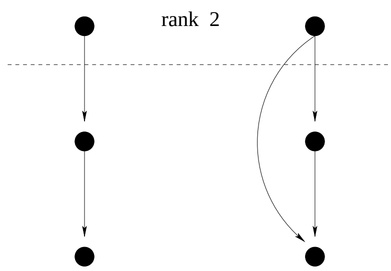


Figure 1.3: Well-founded sets of rank 2.

In order to extend the notion of rank to non-well-founded sets it is reasonable to use the graph of the strongly connected components. All the details about the possible extensions are given in the second part of this thesis (see Chapter 5 and Chapter 7). However, what I want to anticipate here is that there are infinite many non-well-founded sets at each rank. In Figure 1.4 is shown how to build an infinite sequence of sets of rank 1.

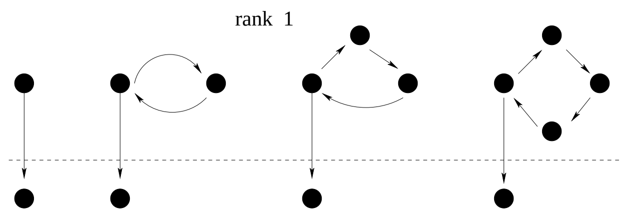


Figure 1.4: Non-well-founded sets of rank 1.

In [COU01] the property that there is a finite number of well-founded sets at each finite rank i is one of the main ingredients in the proof of the small model property of existentially quantified conjunctions of literals based on $=, \in, \cup, \cap, \setminus, \{\}, \wp$ (see Chapter 3). This is no longer true in the case of non-well-founded sets, where the singleton operator together with the powerset operator make true that the formula

$$x = \wp(x)$$

has only solutions which are infinite sets. In Chapter 2 we show under which conditions we are able to prove the small model property for this class of formulae.

The fact that in the well-founded case there are a finite number of sets at each rank is also at the basis of the Ackermann encoding of well-founded sets into natural numbers. As a matter of facts in order to extend the encoding to non-well-founded sets (see Chapter 6) we need to use a double encoding, and the final encoding we obtain is not unique, it depends on the set of sets that we are considering.

Another witness of the differences between well-founded and non-well-founded sets is the formula

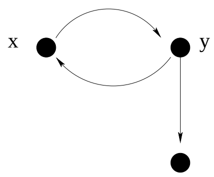
$$\wp(x) \in x.$$

This formula is clearly unsatisfiable if we assume Regularity, since

$$x \in \wp(x) \in x$$

generates a cycle of length 2, while it is satisfiable in non-well-founded set theories. In particular it is satisfiable if we assume (AFA), and its minimum solution (in term of cardinality) is given by the solution of the following system

$$\begin{cases} x = \{y\} \\ y = \{\emptyset, x\} \end{cases}$$



1.2 State-of-the-Art

1.2.1 Computable Set Theory

> *We shall carefully investigate those ways of forming notations and those modes of inference that are fruitful; we shall nurse them, support them, and make them usable, wherever there is the slightest promise of success. No one shall be able to drive us from the paradise that Cantor created for us.* (D. Hilbert, 1925, from [Hil77])

On the one hand, fragments of set theory can be seen as first-order predicate theories, on the other hand they are a natural generalization of the Boole's systematization of Propositional Calculus. This second perspective suggests a different and hopeful approach to the study of the Decision Problem for set theories. The authors of [CFO89] justify the relevance of the *Decision Problem for fragments of Set Theory* in the introduction:

> It is well-known that any mathematical theory can be embedded in set theory. It hence follows that an automated theorem-prover for set theory can be used as an all-purpose theorem prover, or as a useful component of a program that

> checks the correctness of mathematical proofs. Also, various specialized dialects of set theory — often dealing with finite sets — have been designed and successfully exploited for high-level specifications of algorithms, and, more generally, for expressing in an unambiguous form problems that one intends to eventually submit to computers. It is therefore legitimate to expect that the automation of deduction methods for set theory will be very helpful... Decidability results regarding various portions of set theory have been obtained in the last ten years. The research that has led to such results aims at laying the foundation stones upon which a formal set theory, both neat and oriented towards the needs of automated deduction, can develop. ...

Several examples motivating the importance of the study of set theories are presented in Chapter 5 of [COP01].

Different methods have been considered to solve the Decision Problem or to establish the Finite (Small) Model Property of a large number of classes of set formulae. The reason why so many different fragments of set theory and so many different techniques have been taken into consideration is briefly and clearly explained in [COP01]:

> As a matter of fact, the research field named *Computable Set Theory* which is at the core of this book largely consisted in the quest for fragments of the set-theoretical language which are decidable in the sense that, for each sentence within any of them, one can determine automatically whether or not the sentence is valid. In spite of such algorithms often being prohibitively time-consuming, singling them out seemed to be an unavoidable labor before any sensible proof-search method could be implemented for Set Theory. It would have been silly to hope that the full collection of set-theoretic sentences would eventually be brought under the jurisdiction of some mighty decision algorithm; rather, there was hope that the decision algorithms discovered for diverse fragments of set theory could be integrated in some broad spectrum inferential mechanism which, properly driven by human experts, would then offer some flexible support for a good deal of proof-verification work.

The following are the techniques that are generally used in set-theoretical decision problems:

- Model-theoretic techniques:

  > The techniques that it describes combine model-theoretic with syntactic considerations in interesting ways, which, as will be seen, grow increasingly challenging and intricate as successively wider sub-languages of set theory are considered, J. T. Schwartz, foreword to [CFO89]

  In [COP01], Chapter 6, a general technique for turning a model of a formula in a universe that admits individuals into a standard solution is presented.

- Unification algorithms: Free unification plays a fundamental role in SLD-resolution. In order to progress from Prolog to a logic programming language with builtin set-handling capabilities an investigation on set unification is a necessary preliminary step. General unification algorithms working under absorption and permutativity laws, capable of handling a signature which contains free functors along with $\emptyset$ and `with`-operator, have been proposed in [DOPR96, Sto99, AD97]. In [ADR99] it is shown how to implement the algorithm in [DOPR96] without explicitly applying substitutions, which leads to an NP implementation. When the signature includes the binary symbol $\cup$ instead of `with`, the unification problem belongs to the family of *ACI1 unification problems*. Various solutions to this problem are discussed in the literature (e.g. cf. [BB88, Büt86, LS76]), which work when no other functor are allowed to occur. Recently, the general problem involving at least one function symbol has also been solved (cf. [DPR98]). Algorithm which deal with non-well-founded sets in a blended context have been presented in [DOP99, ADR99].

- Venn diagrams: In [COU01] and in [COP01] Chapter 9 the notion of Transitive Venn Diagram is used to prove that unquantified set-theoretic formulae with Boolean operators, the singleton former, and the powerset operator have the small model property.

- Tableaux techniques: Syllogistic tableaux were first introduced in [CC95], where tableau calculi were developed for normalized formulae belonging to the extensions of MLSS (multi-level syllogistic with singleton, cf. [CFO89, COP01]) with iterated membership and with the predicate expressing finiteness. Subsequently, an effective saturation strategy for MLSS-tableaux, based on the interleaving of model-checking and linear saturation steps, was proposed in [Can97]. The model-checking steps were completely removed in [CZ00a] and [CZ99]. Finally, [CZ00b] describes a tableau calculus for Set Theory with restricted form of quantifications and uninterpreted function symbols with monotonicity predicates.

- CLP languages: Presenting a logic programming language with primitives to deal with sets as an instance of the Constraint Programming scheme, rather than as an extension of SLD-resolution has the advantage that adding new set-based constraints requires only the development of new decidability/rewriting algorithms for the new operators. Work along this line has led to the realization of $CLP(\mathcal{SET})$, a language with primitives for union, intersection, and set difference (see [DPPR00]). Because of the computational complexity of the decidable cases of the satisfiability problem, set-based constraints of very general form do not allow one to write efficient constraint-based programs. In [Ger97] it has been shown that by interpreting sets over *finite* domains it is possible to obtain an efficient implementation of some constraints over sets useful in scheduling problems. In [DPPR99] a rewriting procedure for the disunification problem in the context of ACI1 equational theories is presented, together with the unification algorithm defined in [DPR98] this can be used to solve constraints on hereditarily finite and hybrid sets.

Model-theoretic techniques are very general and powerful tools to prove decidability results, while the Venn diagrams technique has been defined in Set Theory to manipulate models without losing their properties. The results obtained using model-theoretic techniques and the Venn diagrams technique cannot be easily used in the applications, since the dimensions of the models involved are too large. Unification algorithms, tableau techniques and CLP languages are much addressed to the implementation of theorem provers and constraint solvers, but the classes of formulae they deal with are smaller.

Often in the use of these techniques is not clear which are the axioms of Set Theory really indispensable and which results can be generalized to non standard theories. Some of the decidability/undecidability results are proved using techniques and axiomatizations which are more powerful than the ones which seem to be required. In this context, we find one of our motivations to analyse in more detail the generalization of these decidability/undecidability results. Our analysis would clarify some theoretical questions about set theories as well as be useful in the applications which make use of sets and set constructors, see [Abr96, LL91, Ger97, HL89, Llo99, BNST91, AG91].

1.2.2 The Classical Decision Problem

> *We could consider the fundamental problem of mathematics to be the following. Problem A: What is the necessary and sufficient condition for a theorem to be true in a given theory having only a finite number of hypothesis?* (J. Herbrand, 1930, from [Her71])

In this section we motivate in greater detail why the decision problems for set theories cannot be reduced to a particular case of the Decision Problem for First-Order Logic.

First-Order Logic is one of the most used formalism in mathematics, computer science and philosophy. At the beginning of this century Hilbert was interested in axiomatizing various branches of mathematics by means of finitely many first-order axioms. In principle, such an axiomatization reduces proving a mathematical statement to performing a mechanical derivation in a fixed formal logical system [Hil71]. In this contest Hilbert drew the attention of mathematicians to the classical decision problem. The original *classical decision problem* can be stated in several equivalent ways [BGG97].

- The *Satisfiability Problem* for First-Order Logic: given a first-order formula, decide if it is consistent.
- The *Validity Problem* for First-Order Logic: given a first-order formula, decide if it is valid.
- The *Provability Problem* for a sound and complete formal proof system for First-Order Logic: given a first-order formula, decide if it is provable in the system.

Independently Church in [Chu36] and Turing in [Tur37] proved that there is no decision algorithm for the classical decision problem. The reaction to this discovery was that logicians started to think about the classical decision problem as a classification problem.

- Which fragments are decidable for satisfiability and which are undecidable?
- Which fragments are decidable for finite satisfiability and which are undecidable?
- Which fragments have the finite model property and which contain axioms of infinity?

In [BGG97] a classification of undecidable/decidable classes is given: the formulae are considered in prenex normal form and the discrimination depends both on the quantifier prefix and on the symbols of the first-order language.

When the decision problem is related to a theory and the theory is finitely axiomatizable, it can be stated as an instance of the classical decision problem as follows:

$$\bigwedge_{i=1}^{n}(Ax_i) \models \varphi \quad \Leftrightarrow \quad \models \bigwedge_{i=1}^{n}(Ax_i) \to \varphi$$

where $\bigcup_{i=1}^{n}\{Ax_i\}$ is the axiomatic theory we are considering and φ is the formulae we want to prove.

(N)	$\exists n\, \forall x\, (x \notin n)$
(W)	$\forall y\, v\, \exists w\, \forall x\, (x \in w \leftrightarrow (x \in v \vee x = y))$
(E)	$\forall x\, y\, (\forall z\, (z \in x \leftrightarrow z \in y) \rightarrow x = y)$
(R)	$\forall x\, \exists z\, \forall y\, (y \in x \rightarrow (z \in x \wedge y \notin z))$

Figure 1.5: The Theory $NWER$.

However, also simple theories, for which decidability results are known, are outside the general classification. In [EITM65, FR79, Rab77] decision problems for different mathematical theories formalized in first-order are considered. Let us consider the fragment of ZF shown in Figure 1.5.

Axioms (E) and (R) are respectively the Extensionality and the Regularity axioms, while the axiom (N) state the existence of the empty set, and the axiom (W) is necessary to add elements to a given set. Using the notation of [BGG97] this belongs to the fragment $[\exists\forall^2\exists^2\forall, (0,1), (0)]_=$, and the problem $NWER \models \vec{\exists}\varphi$ can be seen as an instance of the classical decision problem for $[\exists^*\forall^2\exists^2\forall, (0,1), (0)]_=$. This last is a sub-fragment of the Goldfarb class, which is undecidable, [BGG97]. However, it is easy to see that the problem is decidable and has the small model property, [CFO89, Dov96].

For these reasons a correct classification of the decidability and undecidability results in set theories have rather to be based on the type of axioms and operators that are under analysis. In this perspective a review of solvable fragments of Set Theory is presented in Chapter 6 of [COP01].

1.2.3 Non-Well-Founded Sets

> *Let E be a set, E' one of its elements, E'' any element of E', and so on. I call a* descent *the sequence of steps from E to E', E' to E'', etc. ...I say that a set is* ordinary *when it only gives rise to finite descents; I say that it is* extraordinary *when among its descents there are some which are infinite.* (D. Mirimanoff, 1917, from [Mir17])

Let us consider the set language with predicate symbol $\in$, unary functional symbol $\{.\}$, and binary functional symbol $\cup$, and the set theory presented in Figure 1.6, which defines the meaning of the functional symbols.

$$\forall x,\, y\, (x \in \{y\} \leftrightarrow x = y)$$
$$\forall x,\, y,\, z\, (x \in y \cup z \leftrightarrow x = y \cup z)$$

Figure 1.6: A minimal set theory.

This is a very weak fragment of Set Theory. In Set Theory we do not admit that a set has itself as element, hence an equation of the form

$$x = \{x\}$$

has no solution.

The most famous non-well-founded set theory $ZF^- + (\text{AFA})$ allows[3] to find solutions to this kind of equation. The anti-foundation axiom (AFA) entails that each system of equations

[3] ZF^- is the set theory obtained removing the Regularity axiom from ZF.

of the form $x_j = \{y_1, \ldots, y_n\}$, which satisfies some other conditions, has exactly one solution (see [Acz88]). The solution of the equation $x = \{x\}$ is represented in Figure 1.7. In [Acz88] this set is called Ω and an interesting characterization of it is proved: a graph represents Ω if and only if each node has at least one outgoing edge. We will use this characterization in Chapter 5.

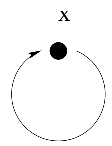


Figure 1.7: The solution of $x = \{x\}$.

1.2.4 Applications of Non-Well-Founded Sets

The relevance of circularity in different research areas is well known: in [BM96] examples coming from computer science, philosophy, and further applications are presented. Examples of notions often used in computer science which can be modelled in a natural way using non-well-founded sets are streams, labelled transition systems, closures, and self-applicative programs. Other theoretical interesting applications can be found in Games Theory, Modal Logic, Feature Logic (see [BM96]), and Shared Information Systems (web-like database [LSY97, Ser99], query languages for semi-structured information [CDQT98]).

A non-well-founded set theory has been introduced for the first time in [FH83], where the axiom X_1, the equivalent of the (AFA) axiom presented in [Acz88], replaces the Regularity axiom. Interest from the computer science community in the notion of non-well-founded sets is stated in [Mil80, Par81], where mathematical theories of concurrent processes are presented. During the preparation of [Acz88], another application of non-well-founded sets was discovered in the area of situation theory and situation semantics and presented in [Bar86]. In [FHL94] a detailed analysis of the expressive power of non-well-founded sets to define the denotational semantics of concurrent languages is carried out: the approach of [Acz88] is generalized making use of hyperuniverses; fixed point theorems are presented and compared with the results obtained in the category of complete metric spaces. At the same time, other applications of the theory of non-well-founded sets were found in the study of intensional semantics for description logics [DMO92, DMO93].

In [DMP95] a translation method to support derivability in Propositional Modal Logic is proposed. The basic idea of the translation is to map modal formulae into set-theoretic terms. The authors prove that the derivability of a modal formula within a normal complete finitely axiomatizable modal logic is equivalent to the derivation of the set-theoretic translation within the weak set theory Ω (see Section 1.2.7 and Chapter 4). The membership relation is used to model the accessibility relation, hence neither the Regularity axiom nor the Extensionality axiom can belong to Ω. Different classes of set formulae emerge from the translation, to capture the concepts of frame validity and modal validity (see Chapters 2, 3, and 4) and among these the easy one is the class of purely existentially quantified formulae. In [BDMP97] a variation of the translation (which is proved to be equivalent to general modal deduction in K) is presented, and the method is generalized to deal with temporal languages. In [BDMP98] all the obtained results are summarized and the connection between Boolean operators of Propositional Logic plus $\Box$-operator and Boolean operators of set theories plus $\wp$-operator is pointed out.

> The main goal of the translation is to extend the celebrated set-theoretic interpretation of the propositional connectives $\wedge$, $\vee$, and $\neg$ to the $\Box$-operator, that is to the (propositional) modal logic language. In other words, we want to extend to the box operator the correspondence that associates set-intersection to conjunction, set-union to disjunction, and complementation to negation. In the Boolean collections used to interpret pure propositional sentences there was no need for a set to be an element of another one; the crucial point of the extension of the interpretation to the modal language is the use of this typical property of the membership relation $\in$. It will be necessary to abandon the 'flat' collection of sets sufficient to analyse true propositional sentences, in favor of more elaborated universes where sets can be elements of other sets (and possibly non-well-founded). From [BDMP98].

In $ZF^- +$ (AFA) the notion of *maximal bisimulation* is the operative counter-part to Extensionality axiom plus (AFA). In other non-well-founded set theories different notions of bisimulation have been introduced. The notion of bisimulation was proposed independently also in Modal Logic and in Process Theory. In [D'A98] the fact that bisimulation can be seen as a bridge between Modal Logic, the non-well-founded set theory $ZF^- +$(AFA) and Process Theory is fruitfully used to transfer results and techniques from one area to the others, with particular reference to interpolation theorems. Actually the main important application of the notion of bisimulation is in Model Checking where it gives a possible solution to the *state explosion problem*. Several existing verification tools make use of bisimulation in order to minimize the state spaces of systems description. The reduction of the number of states is important both in compositional and in non-compositional model checking. Bisimulation serves also as a means of checking equivalence between transition systems. Another equivalence relation important in this context is the *simulation equivalence*, which can be defined and studied as equivalence relation between non-well-founded sets. While bisimulation preserves strongly branching-time temporal logics such as CTL and CTL* [CE82] and, more in general, μ-calculus, simulation only guarantees weak preservation of the universal fragment of these logics (ACTL and ACTL* [GL94]), but it performs a larger reduction of the state spaces.

1.2.5 Results in Computable Non-Well-Founded Set Theories

In [OP95] it is shown that, in spite of its innovative character, $ZF^- +$ (AFA) behaves with respect to the $\exists^*\forall$-sentences over a language with $\in$ as predicate symbol, $\{\cdot \,|\, \cdot\}$[4] as binary set-constructor, and $\emptyset$ as constant symbol, much in the same way as a rather conventional theory of sets. In [DOP99, ADR99] the result is extended to the hybrid case, where non-well-founded sets can be built from atoms different from $\emptyset$, and the results presented are tackled from an algorithmic point of view.

This positive decidability results are not enough to cover all the applications of non-well-founded set theories. It is still an open problem whether the class of purely existential formulae of a language with the $\cup, \setminus$ and $\wp$ operators and the membership predicate is

[4] $z = \{x \,|\, y\}$ has to be interpreted as $z = \{x\} \cup y$.

decidable with respect to weak set theories. The decidability of this class has already been proved in [Can87, CFS85, COU01] assuming Regularity, existence of transitive closure of sets, and Extensionality. In those works it has also been proved that the class has the small model property, which is another missing result in non-well-founded set theories. The techniques used are not generalizable to the non-well-founded case (see Section 1.1).

The first significant result related to the algorithmic solution of the bisimulation problem is in [Hop71], where Hopcroft presents an algorithm for the minimization of the number of states in a given finite-state automaton. The problem is equivalent to that of determining the coarsest partition of a set *stable* with respect to a finite set of functions. A variant of this problem is studied in [PTB85], where it is shown how to solve it in linear time in case of a single function. Finally, in [PT87] Paige and Tarjan solved the problem for the general case (i.e., bisimulation) in which the stability requirement is relative to a relation $\geqslant$ (on a set N) with an algorithm whose complexity is $O(|\geqslant| \log |N|)$.

Several polynomial-time algorithms for computing similarity quotients have been proposed: the ones presented in [Blo89], [CPS93], and [CS92] achieve time complexities of the orders $O(|N|^6|\geqslant|)$, $O(|N|^4|\geqslant|)$, and $O(|\geqslant|^2)$, respectively. The best time complexity for algorithms computing the maximal simulation is the one presented in [HHK95] and [BP95] ($O(|N||\geqslant|)$). The space complexity of these algorithms is limited from below by $O(|N|^2)$. Recently Bustan and Grumberg in [BG00] proposed an algorithm whose space requirements (often more critical, especially in the field of verification) of this algorithm depend on the size of the quotient and are lower than the ones of the previous algorithms. Unfortunately the algorithm has a rather heavy polynomial time complexity.

1.2.6 Which Extension for Propositional Logic?

One main trend since about 1880 has been the interplay of the predicate calculus with set theory. (From [Wan93])

In this section we present in more detail some existing connections between Propositional Logic, Predicate Logic, Modal Logic and Set Theory.

Propositional Logic is decidable. Predicate Logic can be seen as an extension of Propositional Logic: a propositional formula φ can be translated into a monadic predicate formula $\varphi^{\#}$, with one free variable, see Figure 1.8, such that the satisfiability and the validity of φ are respectively equivalent to the satisfiability of $\varphi^{\#}$ and to the unsatisfiability of $(\neg\varphi)^{\#}$.

$$\begin{array}{rcl} p_i^{\#} & = & p_i(x) \\ (\varphi_1 \vee \varphi_2)^{\#} & = & \varphi_1^{\#} \vee \varphi_2^{\#} \\ (\neg\varphi)^{\#} & = & \neg(\varphi^{\#}) \end{array}$$

Figure 1.8: From Propositional to Predicate Logic.

Each model of a monadic first-order logic can be seen as a collection of propositional models. Or, vice-versa, Propositional Logic give us an instrument to focus on an element of a first-order model.

Classical Predicate Logic is undecidable. As discussed in the previous sections, the classification of decidable/undecidable classes for Classical Predicate Logic does not capture useful decidable classes and turns out to be unusable to deal with decision problems in first-order theories.

Classical Modal Logic is another possible extension of Propositional Logic: the modal connective symbol $\Box$ is added to the language and its semantics is given in terms of accessible words. A modal model $\langle W, R, V\rangle$ is composed of a set of words W, together with a binary relation $R \subseteq W^2$, called accessibility relation, and a valuation V, assigning a set of words to each propositional literal (cf. Definition 1.3). A formula φ is true in a word w (see Section 1.2.7) iff:

$$\begin{array}{rcl} w \models p_i & \Leftrightarrow & w \in V(p_i) \\ w \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & w \models \varphi_1 \vee w \models \varphi_2 \\ w \models \neg\varphi & \Leftrightarrow & w \not\models \varphi \\ w \models \Box\varphi & \Leftrightarrow & \forall v\,(wRv \rightarrow v \models \varphi) \end{array}$$

Roughly speaking a modal model is a collection of propositional models, each word is a propositional model, together with a relation establishing that each word can 'know' what is true in a fixed set of *adjacent words.*

Many decidability results in modal logics are known, while other are still under investigation, see [Ben83, HC84, Gol87, Fit93, BRV01].

Modal Logic can be translated into Classical Predicate Logic using a language containing equality, a binary constant R and unary predicate symbols, see [Ben83] as shown from the rules in Figure 1.9.

$$\begin{array}{rcl} p_i^\natural & = & p_i(x) \\ (\varphi_1 \vee \varphi_2)^\natural & = & \varphi_1^\natural \vee \varphi_2^\natural \\ (\neg\varphi)^\natural & = & \neg(\varphi^\natural) \\ (\Box\varphi)^\natural & = & \forall y\,(xRy \rightarrow \varphi(y/x)^\natural) \end{array}$$

Figure 1.9: From Modal to Predicate Logic.

Notice that using this translation to model the notion of modal derivability it is necessary to move from First to Second-Order Logic, using a universal quantification over the predicate symbols p_i.

Different approaches to the translation of modal logics and graded modal logics, into Predicate Logic can be found in [Ohl91, Ohl93, OSH96].

Set theory is a third, different, possible extension of Propositional Logic, in the sense that Propositional Logic can be translated into Set Theory in a intuitive way. The connectives $\vee, \wedge, \neg$ of the propositional case are strongly related to the Boolean $\cup, \cap, \complement$ operators over sets. Each model of Propositional Logic can be seen as a collection of sets, one for each propositional literal, such that each of them is a subset of a fixed singleton set $\{\top\}$. Hence, each propositional formula φ can be translated into the set formula

$$x = \{\emptyset\} \wedge x \subseteq \varphi^\flat,$$

following the rules presented in Figure 1.10.

Using this translation, Set Theory can be seen as a generalization of Propositional Logic. Modal Logic cannot be immediately embedded in Set Theory, using a generalization of this translation, while non-well-founded set theories allow to extend it immediately to Modal

$$
\begin{array}{rcl}
(p_i)^\flat & = & p_i \\
(\varphi_1 \vee \varphi_2)^\flat & = & (\varphi_1)^\flat \cup (\varphi_2)^\flat \\
(\neg\varphi)^\flat & = & x \setminus (\varphi)^\flat
\end{array}
$$

Figure 1.10: From Propositional Logic to set theory.

Logic, see [DMP95]. From this point of view the translation of the $\Box$-connective proposed in [DMP95], naturally involve the use of a non-well-founded membership relation and of the powerset operator. In this sense an analysis and a classification of decidability results in non-well-founded set theories are useful tools to generalize and classify decidability results from Propositional, Modal and Predicate Logics.

1.2.7 The $\Box$-as-$\wp$ Translation

A complete presentation of the results about the $\Box$-as-$\wp$ translation is outside the scope of this section: we intend to recall here the main ideas behind the translation and the principal results. For a complete description the reader can refer to [DMP95] or to Chapter 12 of [COP01]. Moreover we will recall other results about the $\Box$-as-$\wp$ translation in Chapter 2 and in Chapter 3, where we respectively modify the translation and the underlying theory in order to show some new results about the translation.

Definition 1.2 (Modal Formulae) *Let $P = \{p, q, r, \ldots\}$ be an infinite set of propositional literals. A* modal formula γ *is defined as follows:*

$$\gamma \quad := \quad p \mid \gamma_1 \vee \gamma_2 \mid \neg\gamma_1 \mid \Box\gamma_1$$

with $p \in P$.

The connectives $\wedge, \rightarrow$ can be introduced as usual, while $\Diamond\gamma$ stands for $\neg\Box\neg\gamma$.

The semantics of Modal Logic is defined on *Kripke structures.*

Definition 1.3 (Kripke Structures) *A* Kripke frame $\mathcal{F} = \langle W, R\rangle$ *is a pair, where W is a set of* possible worlds *and $R \subseteq W \times W$ is a binary relation on W called* accessibility relation.

A Kripke model $\mathcal{M}$ *is a pair $\mathcal{M} = \langle \mathcal{F}, V\rangle$, where $\mathcal{F}$ is a Kripke frame and $V : P \rightarrow \wp(W)$ is a* valuation *assigning to each propositional literal $p \in P$ a set of worlds $V(p)$.*

In particular the notion of validity of a formula is defined as follows:

Definition 1.4 (Kripke Semantics) *Let $\mathcal{M}$ be a Kripke model and $w \in W$ a world of the model. We inductively define $\mathcal{M}, w \Vdash \gamma$ as*

$$
\begin{array}{lll}
\mathcal{M}, w \Vdash p & \textit{iff} & w \in V(p) \\
\mathcal{M}, w \Vdash \gamma_1 \vee \gamma_2 & \textit{iff} & \mathcal{M}, w \Vdash \gamma_1 \textit{ or } \mathcal{M}, w \Vdash \gamma_2 \\
\mathcal{M}, w \Vdash \neg\gamma_1 & \textit{iff} & \mathcal{M}, w \nVdash \gamma_1 \\
\mathcal{M}, w \Vdash \Box\gamma_1 & \textit{iff} & \forall v \in W(wRv \rightarrow \mathcal{M}, v \Vdash \gamma_1)
\end{array}
$$

We say that $\mathcal{M}$ is a model *of γ ($\mathcal{M} \Vdash \gamma$) if for all $w \in W$ it holds $\mathcal{M}, w \Vdash \gamma$. We say that γ* is valid *on a frame $\mathcal{F}$ ($\mathcal{F} \Vdash \gamma$) if for all V it holds that $\langle \mathcal{F}, V\rangle$ is a model of γ.*

Given a set of modal formulae Λ and a formula γ we say that γ is a frame consequence *of Λ and we write $\Lambda \Vdash \gamma$ if for all the frames $\mathcal{F}$ if $\mathcal{F} \Vdash \Lambda$, then $\mathcal{F} \Vdash \gamma$.*

The minimal modal logic K consists of:

- a set of *propositional axioms* complete for classic propositional logic (see Chapter 2);
- the *modal axiom* $\Box(p \to q) \to (\Box p \to \Box q)$;
- *modus ponens*: infer ψ from φ and $\varphi \to \psi$;
- *substitution*: infer $\psi[p/\varphi]$ (ψ with p replaced by φ) from ψ;
- *necessitation*: infer $\Box\psi$ from ψ.

Given a set of modal formulae Λ we use Λ also to denote the modal logic which is obtained by adding to the minimal modal logic K the formulae in Λ (axioms). The notion of *derivability* of a formula γ in a modal logic Λ is defined in the usual way and it is denoted by

$$\vdash_\Lambda \gamma.$$

A modal logic Λ is said to be *frame complete* (or complete) if and only if for all modal formula γ it holds that

$$\vdash_\Lambda \gamma \quad \text{iff} \quad \Lambda \Vdash \gamma.$$

The key idea of the set-theoretic translation $\Box$-as-$\wp$ is simply to replace the accessibility relation R of Kripke semantics by the membership relation $\in$. This viewpoint entails the following useful facts:

- a frame $\mathcal{F}$ gets identified with its support W, the accessibility relation being implicitly defined as the membership relation on W;
- sets of worlds serve three purposes: some of them are worlds on their own, some constitute the *realm of truth* of a formula, and the largest of them all represents the frame.

These considerations lead to the following translation from a modal formula γ to a set-theoretic formula γ^s

Definition 1.5 ($\Box$-as-$\wp$ Translation) *Let $\gamma(p_1, \ldots, p_n)$ be a modal formula, the set-term $\gamma^*(x, z_1, \ldots, z_n)$ is inductively defined as follows:*

$$\begin{array}{rcl} p_i^* & = & z_i \\ (\gamma_1 \vee \gamma_2)^* & = & \gamma_1^* \cup \gamma_2^* \\ (\neg\gamma)^* & = & x \setminus \gamma^* \\ (\Box\gamma)^* & = & \wp(\gamma^*). \end{array}$$

The set-theoretic translation $\gamma^s(x)$ *of the formula* $\gamma(p_1, \ldots, p_n)$ *is the formula*

$$\forall z_1, \ldots, z_n (x \subseteq \gamma^*(x, z_1, \ldots, z_n)).$$

In order to establish a link between the derivability of a formula γ in modal logic and the first-order derivability of its translation γ^s it is necessary to single out laws on the membership relation. In carrying out this task, the following constraint has to be considered: we cannot impose on $\in$ properties that it cannot share with a generic accessibility relation. In particular $\in$ can neither be acyclic nor extensional. Consider the theory Ω we present here below.

$(\cup)$	$\forall x, y, z\,(x \in y \cup z \leftrightarrow x \in y \vee x \in z)$
$(\setminus)$	$\forall x, y, z\,(x \in y \setminus z \leftrightarrow x \in y \wedge x \notin z)$
$(\subseteq)$	$\forall x, y\,(x \subseteq y \leftrightarrow \forall z\,(z \in x \rightarrow z \in y))$
$(\wp)$	$\forall x, y\,(x \in \wp(y) \leftrightarrow x \subseteq y)$

In [DMP95] the following result has been proved using the theory Ω.

Theorem 1.6 (Correctness and Completeness) *Let Λ be a modal logic with a finite axiomatization α in the propositional literals $q_1, \ldots, q_m$ and let γ be a modal formula in the propositional literals $p_1, \ldots, p_n$, with all the q_i's different from the p_j's. It holds that*

$$\vdash_\Lambda \gamma \quad \Rightarrow \quad \Omega \vdash \forall x(x \subseteq \wp(x) \wedge \alpha^s \rightarrow \gamma^s)).$$

Moreover

$$\Lambda \Vdash \gamma \quad \Leftarrow \quad \Omega \vdash \forall x(x \subseteq \wp(x) \wedge \alpha^s \rightarrow \gamma^s)).$$

1.2.8 Model Checking and State Explosion Problem

Model Checking is a method for formally verifying finite-state concurrent systems. Specifications about the system are expressed as temporal logic formulae, and efficient explicit or symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not.

If we have a finite-state system P and we want to check whether or not it satisfies a specification φ written in temporal logic, we could completely characterize the system P by a temporal logic formula $\psi(P)$ and then check if $\psi(P) \rightarrow \varphi$ is a valid temporal formula [MP81]. Unfortunately, the validity test for temporal logic is EXP time complete [EH85]. Another possible approach could be to have P represented by a Kripke model $\mathcal{M}(P)$: the states in $\mathcal{M}(P)$ represent the possible global states of P, and the edges represent the possible transitions of P. Checking whether P satisfies the specification φ in this approach corresponds to checking if $\mathcal{M}(P)$ is a model of φ [EC82, QS82]. This is what is known as *Model Checking* and it can be done in time linear in the size of $\mathcal{M}(P)$ and φ.

Model checking suffers, however, from the so-called *state explosion problem*. In a concurrent setting, the system under consideration is typically the parallel composition of many modules. As a result, the size of the state space of the system is the product of the sizes of the state spaces of the participating modules. This gives rise to state spaces of exceedingly large sizes, which makes even linear time algorithms impractical. This issue is one of the most important one in the area of computer-aided verification and is the subject of active research. Different techniques have been developed and are still under active investigation to solve this problem: Symbolic Model Checking using OBDD's [McM93], Abstract Model Checking [Dam96], partial order reductions [CGP99], equivalence reductions [LY92, HHK95].

It is common to define equivalence relations or preoders on transition systems in such a way to obtain a smaller structure which satisfies the same properties of the starting one (strong or weak preservation of various logics). Relations of this type that are widely used are the *bisimulation equivalence* [Par81] and the *simulation preorder* [Mil71]. The former guarantees strong preservation of branching-time temporal logics such as CTL and CTL*

[CE82]. The latter guarantees weak preservation of the universal fragment of these logics (ACTL and ACTL* [GL94]).

We have already given the definition of Kripke model in the previous section (cf. Definition 1.3). We give here the definition and the semantic of the logics CTL*, CTL, ACTL*, and ACTL.

Definition 1.7 (CTL* Formulae) *Let P be a set of propositional literals and $p \in P$. We inductively define* state formulae *and* path formulae *of CTL* using the following rules:*

$$\begin{array}{lcll} \textit{state f.} & \varphi & := & p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid A\psi \mid E\psi \\ \textit{path f.} & \psi & := & \varphi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid X\psi \mid \psi_1 U \psi_2 \end{array}$$

A path π in a Kripke frame $\mathcal{F} = \langle W, R \rangle$ is a sequence $w_0, w_1, \ldots$ of worlds (states) of W such that $w_i R w_{i+1}$. Given a path

$$\pi = w_0, w_1, \ldots, w_n, w_{n+1}, \ldots$$

the path π^n is the path obtained from π by forgetting the first n states, i.e.

$$\pi^n = w_n, w_{n+1}, \ldots$$

Definition 1.8 (CTL* Semantics) *Let $\mathcal{M} = \langle \mathcal{F}, V \rangle$ be a Kripke model on the frame $\mathcal{F} = \langle W, R \rangle$. Let $w \in W$ be a state and $\pi : \mathbb{N} \to W$ be a path. Given a state formula φ and a path formula ψ we inductively define the* satisfiability *notion as follows:*

1. $\mathcal{M}, w \Vdash p$ *iff* $w \in V(p)$;
2. $\mathcal{M}, w \Vdash \neg p$ *iff* $\mathcal{M}, w \nVdash p$;
3. $\mathcal{M}, w \Vdash \varphi_1 \vee \varphi_2$ *iff* $\mathcal{M}, w \Vdash \varphi_1$ *or* $\mathcal{M}, w \Vdash \varphi_2$;
4. $\mathcal{M}, w \Vdash \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, w \Vdash \varphi_1$ *and* $\mathcal{M}, w \Vdash \varphi_2$;
5. $\mathcal{M}, w \Vdash A\psi$ *iff for each* $\pi = w_0, w_1, \ldots$ *such that* $w_0 = w$ *it holds* $\mathcal{M}, \pi \Vdash \psi$;
6. $\mathcal{M}, w \Vdash E\psi$ *iff there exists* $\pi = w_0, w_1, \ldots$ *such that* $w_0 = w$ *and* $\mathcal{M}, \pi \Vdash \psi$;
7. $\mathcal{M}, \pi \Vdash \varphi$ *iff* $\mathcal{M}, w_0 \Vdash \varphi$, *where* $\pi = w_0, w_1, \ldots$;
8. $\mathcal{M}, \pi \Vdash \psi_1 \wedge \psi_2$ *iff* $\mathcal{M}, \pi \Vdash \psi_1$ *and* $\mathcal{M}, \pi \Vdash \psi_2$;
9. $\mathcal{M}, \pi \Vdash \psi_1 \vee \psi_2$ *iff* $\mathcal{M}, \pi \Vdash \psi_1$ *or* $\mathcal{M}, \pi \Vdash \psi_2$;
10. $\mathcal{M}, \pi \Vdash X\psi$ *iff* $\mathcal{M}, \pi^1 \Vdash \psi$;
11. $\mathcal{M}, \pi \Vdash \psi_1 U \psi_2$ *iff there exists* $n \in \mathbb{N}$ *such that it holds that* $\mathcal{M}, \pi^n \Vdash \psi_2$ *and for all* $i < n$ $\mathcal{M}, \pi^i \Vdash \psi_1$.

The temporal logic CTL is the fragment of CTL* obtained by considering only the formulae defined by the following grammar:

$$\varphi \;\; := \;\; p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid AX\varphi \mid EX\varphi \mid A(\varphi_1 U \varphi_2) \mid E(\varphi_1 U \varphi_2).$$

The temporal logic ACTL* and ACTL are the universal fragments (only A is allowed) of CTL* and CTL respectively.

We now introduce the notions of *bisimulation* and *simulation* on a Kripke model. In Chapter 5 and in Chapter 8 we will equivalently redefine these notions on graphs and labelled graphs. An example of the quotient of a structure w.r.t. bisimulation and simulation is presented in the introduction to the second part of this thesis. Intuitively bisimulation requires that if w and v are bisimilar then they satisfy the same propositional literals and each step that can be performed from w can be simulated from v and vice-versa, while in the notion of simulation the *vice-versa* condition is dropped.

Definition 1.9 (Bisimulation) *Let* $\mathcal{M} = \langle \mathcal{F}, V \rangle$ *be a Kripke model, with* $\mathcal{F} = \langle W, R \rangle$. *A* bisimulation *is a relation* $B \subseteq W \times W$ *such that:*

(l) $wBv \Rightarrow \forall p \in P(w \in V(p) \leftrightarrow v \in V(p))$;

(f) $wBv \wedge wRw' \Rightarrow \exists v'(w'Bv' \wedge vRv')$;

(b) $wBv \wedge vRv' \Rightarrow \exists w'(w'Bv' \wedge wRw')$.

There exists always a maximum bisimulation relation $\equiv$ which is an equivalence relation.

Theorem 1.10 *Let* $\mathcal{M} = \langle \mathcal{F}, V \rangle$ *be a finite Kripke model. It holds that*

$$\forall w, v \in W(w \equiv v \leftrightarrow \forall \psi \in CTL^*(\mathcal{M}, w \Vdash \psi \leftrightarrow \mathcal{M}, v \Vdash \psi)).$$

Definition 1.11 (Simulation) *Let* $\mathcal{M} = \langle \mathcal{F}, V \rangle$ *be a Kripke model, with* $\mathcal{F} = \langle W, R \rangle$. *A* simulation *is a relation* $S \subseteq W \times W$ *such that:*

(l) $wSv \Rightarrow \forall p \in P(w \in V(p) \leftrightarrow v \in V(p))$;

(f) $wSv \wedge wRw' \Rightarrow \exists v'(w'Sv' \wedge vRv')$.

There exists always a maximum simulation relation $\leq_s$ which is reflexive and transitive. We say that w and v are *similar* $w \equiv_s v$ if $v \leq_s w$ and $w \leq_s v$.

Theorem 1.12 *Let* $\mathcal{M} = \langle \mathcal{F}, V \rangle$ *be a finite Kripke model. It holds that*

$$\forall w, v \in W(w \equiv_s v \leftrightarrow \forall \psi \in ACTL^*(\mathcal{M}, w \Vdash \psi \leftrightarrow \mathcal{M}, v \Vdash \psi)).$$

1.3 Overview of the Thesis

This dissertation is composed by two parts.

In the first part we present some decidability results in the context of non-well-founded set theories. On the one hand the theories we consider are suggested by the $\Box$-as-$\wp$ translation (see Section 1.2.7), and on the other hand they are the most natural ones expressing the meaning of the basic set-operators. We are interested both in proving new decidability results and in providing general techniques easily extendable to other theories. In each chapter of the first part we consider a theory and a class of formulae and we analyse the finite (small) model property and the decidability of the satisfiability (validity) problem.

In Chapter 2 we refer to the theory Ω_P and to the class of purely existentially quantified formulae. We prove the small model property using a generalization of a quotient technique coming from Modal Logic. We define a tableau decision procedure which makes our decidability result more *operative*. We analyse the problems which arise adding to the theory the

Extensionality axiom and we extend our results to this case. Since the theory Ω_P is different from the theory used in the $\Box$-as-$\wp$ translation, we define a variation of the translation which works correctly using the theory Ω_P.

In Chapter 3 we move to the theory Ω plus Extensionality and the anti-foundation axiom (AFA). For the same class of formulae analysed in Chapter 2 we prove some decidability results. In particular, we are able to decide the finite satisfiability over models with atoms, using a tableau technique. A preliminary version of the material in this chapter has been presented at the conference TABLEAUX'00 (see [PP00]).

In Chapter 4 we consider a larger class of formulae, i.e. we allow restricted universal quantifiers. We point out the connections between the class of formulae we consider and the formulae obtained using th $\Box$-as-$\wp$ translation generalizing a result presented in [DMP96]. We define a tableau procedure for the whole class and we establish a termination condition for a subclass. The results in this chapter have been presented at the conference APPIA-GULP-PRODE'01 (see [PP01b]).

In the second part of the thesis we study two algorithmic problems in the context of non-well-founded sets: the computation of the maximum bisimulation relation and of the maximum simulation relation on a finite non-well-founded set. The first of these two problems is equivalent to the computation of equalities among sets when the Regularity axiom is replaced by the anti-foundation axiom (AFA). We prove that the set-theoretical semantics provide us the right instruments to design algorithms which improve in time and space the performances of the algorithms actually in use. Such improvements are particularly important in the area of Model Checking, where bisimulation and simulation are used to reduce the state spaces.

In Chapter 5 we present an algorithm which computes the maximum bisimulation on a given graph (redundant representation of a non-well-founded set) using the notion of rank to introduce an order in the computation. Our algorithm has in the worst case the same complexity of that of Paige and Tarjan (see [PT87]), but it presents two main advantages: it performs the computation on subgraphs, reducing the space complexity; in a large number of cases it works in linear time. Part of the material in this chapter has been presented at the conference CAV'01 (see [DPP01]).

In Chapter 6 we analyse the use of symbolic data-structures (OBDD's) in the representation of graphs. We propose a symbolic representation based on the notion of rank together with a method to determine in one step the quotient w.r.t. the maximum bisimulation and its symbolic representation. The results in this chapter have been presented at the workshop VCL'01 (see [PP01a]).

In Chapter 7 we reduce the problem of computing the maximum simulation relation to an inclusion/membership constraint problem over sets. Also in this case the set-theoretical point of view allows us to use a notion of rank in order to get an incremental computation. Even though the computation at a given rank is not completely independent from the computation at the previous ranks our method naturally suggests a fully incremental determination of the symbolic representation of the quotient w.r.t. the maximum simulation. The results in this chapter will be presented at the workshop SAVE'01.

In Chapter 8 we describe an algebraic rephrasing of the simulation problem in terms of *generalized partition problem*. We exploit such a rephrasing in the definition of an algorithm to compute the maximum simulation which has lower time complexity than the algorithm proposed in [BG00] and the same space complexity.

I

Tableaux Methods

Introduction to Part I

It is theory that decides what can be observed. (A. Einstein)

Traditionally, a logic is presented via the following three components: an intuitive non-formal motivation, a semantic interpretation, a proof theoretical formulation. There are several types of proof theoretical methodologies the most famous of which are Hylbert style, Gentzen style. We can say that tableaux history begins with the introduction of *Gentzen's sequent calculus* in 1935 (see [Sza69]). Before this the norm was to use Hilbert style, or axiomatic, proof procedures. A Hilbert system characterizes a logic as a whole: it is difficult to separate out the role of individual connectives since many of them usually occur in each axiom. The main contribution of Gentzen was the separation, both in classical and in intuitionistic logics, between structural rules and specific rules for each connective and quantifier. But, Gentzen motivation was proof-theoretic, i.e. he was analysing the structure of the proofs and he was not interested in completeness or soundness issues. Beth in 1955, motivated by semantic concerns, introduced for the first time the terminology *semantic tableau* to denote a systematic method to find a counter-example. From the paper of Beth of 1955:

> If such a counter-example is found, then we have a negative answer to our problem. And if it turns out that no suitable counter-example can be found, then we have an affirmative answer. In this case, however, we must be sure that no suitable counter-example whatsoever is available; therefore, we ought not to look for a counter-example in a haphazard manner, but we must rather try to construct one in a systematic way. Now there is indeed a systematic method for constructing a counter-example, if available; it consists in drawing up a *semantic tableau.*

After their introduction tableau systems have been defined for several kinds of logics, such as intuitionistic logics, modal and temporal logics, non-monotonic logics, many-valued logics, while the publication of the first edition of Fitting's book [Fit96] in 1990 diffused the use of tableau methods in automated deduction. Moreover, tableaux have been used to provide alternative proofs of basic theorems about logic: Beth used tableaux to show a subformula principle, Herbrand's Theorem, Gentzen's Cut Elimination Theorem and Craig's Lemma; Smullyan gave proofs of the Compactness Theorem, various interpolation theorems, and the Model Existence Theorem using tableaux in an essential way.

In order to better understand why tableaux look so appealing it is necessary to give some more details. If we want to use a tableau method to prove that a formula φ is a consequence the theory $\mathbb{T}$ we have to systematically look for a refutation of $\neg\varphi$ by applying suitable inference rules. The inference rules closely follow the semantics of logical connectives, quantifiers, and, other symbols of the language. In general, if the theory $\mathbb{T}$ is undecidable, the proof search could not terminate. But in case of termination, if the branches of a tableau for $\neg\varphi$ are all contradictory, the tableau itself is a proof of φ in the theory $\mathbb{T}$. If a tableau has a noncontradictory branch, then the information in such a branch suggest how to build a model for $\mathbb{T} \cup \{\neg\varphi\}$. Hence we can say that a tableau method is both a proof method (syntactic point of view) and a search procedure for models meeting certain conditions (semantic point of view).

Many tableau systems are said to be *analytic*, which means that they only allow the derivation of subformulae of the initial formula or their negation. This property, in particular, makes tableaux very suitable for automation. Moreover, tableaux can be extended by adding new rules, which allows in many cases to obtain generalizations to more expressive theories, and they can be equipped with heuristics and various kinds of controls available to the user, obtaining important optimizations.

The above mentioned advantages of tableau methods suggested to a part of the Computable Set Theory community (see Section 1.2.1) to study tableau calculi for set theories. The first results in this direction were presented in [CC95], where it was considered an extension of MLSS (cf. Chapter 1 Section 1.2) with iterated membership and with the predicate expressing finiteness (cf. also [CF95]). In [Can97] an effective saturation strategy for MLSS-tableaux, based on the interleaving of model-checking and linear saturation steps, was proposed. The model-checking steps have been recently avoided in [CZ00a] and [CZ99]. Finally in [CZ00b] a tableau calculus which integrates first-order and elementary set theory, in the presence of a restricted form of quantification and uninterpreted function symbols with monotonicity predicates has been presented. A detailed presentation of the state-of-the-art about syllogistic tableaux can be found in Chapter 14 of [COP01], while a complete treatment of tableau methods can be found in [DGHJP99].

In this part of the thesis we study *tableau decision procedures* for non-well-founded fragments of set theories. We introduce tableau procedures in order to prove new decidability results about the non-well-founded theories we consider. This means that our focus is more on decidability than on efficiency.

The reasons which lead us to use tableau methods for decidability issues are many. First of all we intended to follow an incremental approach starting from theories with few axioms and syntactically restricted classes of formulae to move to richer theories and less restricted classes of formulae. From this point of view tableau methods are the ones that avoid to *restart each time from the beginning*: new rules are added and the termination conditions are modified when necessary. This is not possible using more *semantically driven* decision methods, as for instance the Venn's diagrams method proposed in [COU01]. A second, but not less important, motivation is given by the naturalness of tableau methods: the rules are immediately suggested by the axioms, especially when we are dealing with a theory in which each symbol is defined by an axiom in terms of the membership predicate; the rules are understandable also if one has never analysed the semantic properties of the models of the theory. On the contrary, the application of the rules leads to the definition of a model of the theory and the formula (in the case there exists such a model) and hence allows to study such semantic properties.

In particular, we start from a minimal set theory and we study the satisfiability and validity problems for a small class of formulae (we allow only existential quantifiers), then we study the same class of formulae in stronger theories. Our third step is to enlarge the class of formulae allowing restricted universal quantifiers.

In Chapter 2 we consider the set theory Ω_P, where we use P to point out that in the language there are only *predicate* symbols, and the class $\mathcal{S}_\exists$ of existentially quantified formulae over the language of this theory. We are able to define a procedure which given a formula $\varphi \in \mathcal{S}_\exists$ decides if there exists a model of Ω_P in which φ is satisfiable. Moreover, we prove that a formula in $\mathcal{S}_\exists$ has a model if and only if it has a finite model of a bounded cardinality (small model property). In the second part of Chapter 2 we analyse the consequences of adding to Ω_P some form of extensionality. We are able to prove the small model property adding a weak form of extensionality with atoms, and we present a tableau decision procedure which works with the classical Extensionality axiom (E). The procedures we present can be used as subroutines to decide the same class of formulae both in Set Theory and in non-well-founded set theories, since we do not assume neither Regularity nor any form of anti-foundation axiom.

In Chapter 3 we study what happens when we add to a the theory similar to Ω_P both the Extensionality axiom and the anti-foundation axiom (AFA). In particular in the first part of the chapter we consider the theory $\Omega^- ENS(\text{AFA})$, where we use Ω^- since the first three axioms of this theory are exactly the first three axioms of Ω, but we do not have the powerset operator. It is immediate to prove the small model property and to give a decision procedure for the satisfiability problem of the class of existentially quantified formulae.

In the second part of Chapter 3 we consider the satisfiability problem with the powerset operator. The theory we refer to in this second part is $\Omega_{P_a} E_a N_a S$, where the first 5 axioms are exactly the axioms of Ω_{P_a} introduced at the end of Chapter 2 (similar to Ω_P, but with *atoms*). In this case the finite model property does not hold any longer, but, thanks to the presence of the atoms, we are able to give a decision procedure to decide the finite satisfiability problem, (atoms avoid infinite descending chains of memberships). This means that our decision procedure returns false when the formula is not satisfiable and when it has only infinite models.

The results in these two chapters allow us to conclude that it is the presence of the powerset operator together with (AFA) that makes it hard to give decidability results: we have a decidability result in the case with powerset and Extensionality, in the case with (AFA) without powerset, but we have only a semi-decision procedure in the case with (AFA), and powerset. In particular the analysis we make in Chapter 3 clarifies some of the difficulties in deciding purely existentially quantified formulae in the case without atoms.

Finally in Chapter 4 we propose a tableau procedure for deciding the satisfiability of set-theoretical formulae with restricted universal quantifiers and the powerset operator over the theory Ω. The theory Ω (already presented in Section 1.2.7) is the 'functional' version of the theory Ω_P, i.e. $\cup, \setminus$, and $\wp$ are function symbols instead of predicate symbols. Our result apply to a rather large class of set theories, since again the procedure we define in this chapter, as the ones in Chapter 2, can be used as a subroutine to decide the same class of formulae both in Set Theory and in non-well-founded set theories.

In all these chapters we describe the connections between the decidability results we present and decidability results in Modal Logic using the $\Box$-as-$\wp$ translation ([DMP95]). In particular in Chapter 4 we motivate the class of formulae we consider in term of the results presented in [DMP96], obtaining that our procedure can be used to uniformly study a large class of modal logics which includes K, T, $S4$, $S5$, $S4.3$.

In a certain sense we can say that the common denominators of these chapters are the non-well-founded context, tableau methods, and the powerset operator. As far as the powerset operator in the well-founded context is concerned, it is worth mentioning that the first decidability results about MLS extended with $\wp$ were obtained by Michael Breban in 1980: he was able to deal with only one occurrence of $\wp$. One year later Alfredo Ferro discovered how to deal with two occurrences of $\wp$. In 1984 J.T. Schwartz found out how to decide MLS extended with an arbitrary number of the unionset operator $\bigcup$. Soon it was understood that a similar technique could be applied to MLS extended with $\wp$ with no restrictions placed on the number of occurrences (see [CFS85, CFS87]). The decidability of MLS extended with both $\wp$ and the singleton operator was first discussed in [Can87], while a systematic effort to combine the treatments of $\wp$ and $\bigcup$ can be found in [CU97]. In [COU01] and in Chapter 9 of [COP01] the decidability result is proved by using *formative processes* and *syllogistic games*.

2

Satisfiability over the Theory Ω_P

In this chapter we consider the theory Ω_P and the class $\mathcal{S}_\exists$ of purely existentially quantified formulae over the language of Ω_P. We are interested in the decidability of the satisfiability problem of formulae in $\mathcal{S}_\exists$ w.r.t. models of Ω_P.

We start with some remarks whose aim is to point out that the peculiarity of the theory Ω_P of being based on predicate symbols, rather than on functional symbols, prunes the construction of models of Ω_P from a number of problems: functional symbols have to be total and functional. The theory Ω_P has also finite models, in particular, each graph is a model of Ω_P, while its functional version Ω has only infinite models. This allows us to concentrate on the interplay among the set-theoretical predicates.

We exploit the fact that each graph is a model of Ω_P to prove the small model property w.r.t. the class $\mathcal{S}_\exists$ a generalization of the quotient technique on which is based the proof of the small model property for the modal logic K (see [Ben83, BRV01]). The idea to generalize this quotient technique has been suggested us by the $\Box$-as-$\wp$ translation (cf. Section 1.2.7). The $\Box$-as-$\wp$ translation is based on the theory Ω, but in Section 2.4 we show how to modify the translation in order to use the theory Ω_P pointing out the connection between the satisfiability problems we are interested in and satisfiability problems in modal logic.

Since in Section 2.3 we prove the small model property for the class $\mathcal{S}_\exists$ w.r.t. the theory Ω_P, we can conclude that the problem under consideration is decidable. In Section 2.5 we prove again the decidability result in a more constructive way by defining a tableau procedure which solves the problem.

In the second part of this chapter we deal with a problem whose nature is more exquisitely set-theoretical: what happens if we add to the set theory Ω_P some form of extensionality? In Section 2.6 we show how to use the quotient technique of Section 2.3 to prove the small model property in presence of a form of extensionality with atoms. In the case we add to Ω_P the *standard* Extensionality axiom it is only thanks to the tableau procedure shown in Section 2.7 that we are able to conclude that the satisfiability problem is still decidable.

2.1 The Language and the Theory

In this chapter we consider the first-order language

$$\mathcal{L} = \langle \mathcal{F}, \mathcal{C}, \mathcal{P}, \mathcal{V} \rangle,$$

with

$$\mathcal{F} = \emptyset, \quad \mathcal{C} = \emptyset, \quad \mathcal{P} = \{\in, \subseteq, =, \cup, \setminus, \wp\},$$

and $\mathcal{V}$ a denumerable set of variables.

The theory on which we focus our attention is the theory Ω_P presented in Figure 2.1.

$(\cup)$	$\forall y,\, z,\, w\, (\cup(y, z, w) \leftrightarrow \forall x\, (x \in w \leftrightarrow x \in y \vee x \in z))$
$(\setminus)$	$\forall y,\, z,\, w\, (\setminus(y, z, w) \leftrightarrow \forall x\, (x \in w \leftrightarrow x \in y \wedge x \notin z))$
$(\subseteq)$	$\forall x,\, y\, (x \subseteq y \leftrightarrow \forall z\, (z \in x \rightarrow z \in y))$
$(\wp)$	$\forall y\, z\, (\wp(y, z) \leftrightarrow \forall x (x \subseteq y \leftrightarrow x \in z))$

Figure 2.1: The axiomatization of Ω_P.

Let $\mathcal{S}_\exists$ be the class of purely existentially quantified formulae over $\mathcal{L}$, i.e. all the formulae in $\mathcal{S}_\exists$ have a prenex form with all the variables existentially quantified.

We study the satisfiability problem of formulae of $\mathcal{S}_\exists$ in models of Ω_P, i.e. if there exists a model of Ω_P in which a formula of $\mathcal{S}_\exists$ is satisfiable. The satisfiability problem we are interested in is equivalent to the satisfiability problem of open formulae (no quantifiers) in models of Ω_P, for this reason in the rest of the chapter we refer to open formulae.

2.2 Basic Considerations about the Models and the Formulae

First of all notice that $\cup, \setminus$, and $\wp$ are predicates and that the theory Ω_P does not ensure neither the existence nor the uniqueness of their results. In the literature it is more frequent to find set theories in which $\cup, \setminus$ and $\wp$ are operators. We decided to have them as predicates in this first chapter about the tableau methods because in this way when we need to build a model of Ω_P we are not forced to:

- put in the model all the results of the unions, set-differences, and powerset operations;
- ensure the uniqueness of the results.

This is not a mere simplification, this is what allows us to focus our attention only in the interplay between the memberships relations and the predicates $\cup$, $\setminus$, and $\wp$.

Example 2.1 *The formula*

$$\cup(x, y, z) \wedge \cup(x, y, w) \wedge z \neq w$$

is satisfiable in a model of Ω_P. The smallest model in which the formula is satisfiable has two elements a and b. They both have no elements. The valuation with $\sigma(x) = \sigma(y) = \sigma(z) = a$ and $\sigma(w) = b$ satisfies the formula.

Hence, the key consequence of having $\cup$, $\setminus$, and $\wp$ as predicates is that it is easy to build models of Ω_P, and in particular

each graph $G = \langle N, \gg \rangle$ can be interpreted as a model of Ω_P.

The interpretation of the membership relation is given by the relation $\lessdot$, i.e.

$$\forall n, m \in N(m \in^{\mathcal{G}} n \leftrightarrow m \lessdot n).$$

The interpretations of the predicate symbols are forced by the axioms in Ω_P. Hence, the interpretation is unique.

This is different from the fact that each graph can be interpreted as a non-well-founded set (cf. Chapter 1): what we have here is that Ω_P is so weak that each graph is a whole model of it.

So, in particular Ω_P have finite models, while this is not true for its functional version Ω. In Chapter 4 we will show how to embed each graph in a (infinite) model of Ω.

Remark 2.2 *The fact that in the language there are not functional symbols implies that the minimum model of Ω_P is the model with only one element:*

$$\begin{array}{rcl} \mathcal{M} & = & \langle M, \in^{\mathcal{M}}, \subseteq^{\mathcal{M}}, \cup^{\mathcal{M}}, \setminus^{\mathcal{M}}, \wp^{\mathcal{M}} \rangle \\ M & = & \{m\} \\ \in^{\mathcal{M}} & = & \emptyset \\ \cup^{\mathcal{M}} & = & \{\langle m, m, m \rangle\} \\ \setminus^{\mathcal{M}} & = & \{\langle m, m, m \rangle\} \\ \wp^{\mathcal{M}} & = & \{\langle m, m \rangle\}. \end{array}$$

For this reason it makes no sense to study the validity problem of purely existentially quantified formulae w.r.t. the theory Ω_P. For instance, the formula

$$\exists x, y(x \neq y)$$

is not valid in Ω_P. In Chapters 3 and 4 we use stronger theories on which it is interesting also to study the validity problem.

It is useful to give a *canonical form* for the formulae in $\mathcal{S}_\exists$ in order to reduce the number of possible literals and of their possible combinations.

Let φ_i be a conjunction of literals of the form:

$$x \in y,\ x \notin y,\ x \neq y,\ \cup(x, y, z), \setminus(x, y, z), \wp(x, y) \tag{2.1}$$

where x, y, z are variables. In this chapter we use φ to refer to a disjunction $\varphi_1 \vee \ldots \vee \varphi_n$ of such conjunctions. The class of all the disjunctions of conjunctions of literals of the form (2.1) will be called $\mathcal{S}$.

It is easy to see that each existentially quantified formula ψ of $\mathcal{L}$ is equivalent to a formula ψ^S in the class $\mathcal{S}$. The procedure REDUCTION1[1] in Figure 2.2 allows, given a formula $\psi \in \mathcal{S}_\exists$ of $\mathcal{L}$, to obtain such an equivalent formula $\psi^S \in \mathcal{S}$.

With $\ell \overset{T1}{\mapsto} p$ we mean that there is a rule in T which rewrites ℓ into p.

Therefore, we can safely move from the satisfiability problem for formulae in $\mathcal{S}_\exists$ to the satisfiability problem for formulae in $\mathcal{S}$.

[1] We call the procedure REDUCTION1 since in Chapter 3 we will present the procedures REDUCTION2 and REDUCTION3.

REDUCTION1(ψ) :
repeat
$\quad\varphi := \psi$;
$\quad$if ℓ is a literal in ψ and $\ell \overset{\mathsf{T1}}{\mapsto} p$ then
$\qquad\psi := \psi$ with ℓ replaced by p
until $(\varphi = \psi)$
$\psi^S :=$ disj_form(ψ).

Figure 2.2: The Procedure REDUCTION1.

$x \subseteq y$	$\mapsto$	$\cup(x,y,y)$
$x \not\subseteq y$	$\mapsto$	$w \in x \wedge w \notin y$
$\neg \cup (x,y,z)$	$\mapsto$	$(w \in x \wedge w \notin z) \vee (w \in y \wedge w \notin z) \vee$ $(w \in z \wedge w \notin x \wedge w \notin y)$
$\neg \setminus (x,y,z)$	$\mapsto$	$(w \in y \wedge w \in z) \vee (w \in z \wedge w \notin x) \vee$ $(w \in x \wedge w \notin y \wedge w \notin z) \vee$
$\neg\wp(x,y)$	$\mapsto$	$(w \subseteq x \wedge w \notin y) \vee (w \not\subseteq x \wedge w \in y)$
$x = y$	$\mapsto$	$\psi[x/y]$

Figure 2.3: The rules T1 for the Procedure REDUCTION1.

Example 2.3 *Consider the formula*

$$\neg\wp(x,y) \wedge \neg \cup (y,y,x).$$

It is equivalent to the formula in the class $\mathcal{S}$

$$\begin{gathered}(\cup(w_1,x,x) \wedge w_1 \notin y \wedge w_2 \in y \wedge w_2 \notin x) \vee \\ (\cup(w_1,x,x) \wedge w_1 \notin y \wedge w_2 \notin y \wedge w_2 \in x) \vee \\ (w_3 \in w_1 \wedge w_3 \notin x \wedge w_1 \in y \wedge w_2 \in y \wedge w_2 \notin x) \vee \\ (w_3 \in w_1 \wedge w_3 \notin x \wedge w_1 \in y \wedge w_2 \notin y \wedge w_2 \in x)\end{gathered}$$

Finally, we do not have the predicate symbol $\cap$ in our language. It is possible to add it to the language and to define it as:

$(\cap)$	$\forall y,z,w(\cap(y,z,w) \leftrightarrow \forall x(x \in w \leftrightarrow x \in y \wedge x \in z))$

It is immediate to extend all the results and the procedures in this Chapter in order to deal also with the predicate $(\cap)$. It is however important to notice that, since we are dealing with a language and a theory with predicates, it is not possible to eliminate $\cap$ using the other predicates, i.e.

$$\cap(y,z,w) \leftrightarrow w \subseteq y \wedge w \subseteq z \wedge \setminus(y,z,w_1) \wedge \setminus(y,w,w_1)$$

is not an equivalent definition for the predicate $\cap$.

2.3 The Small Model Property

We prove here that if a formula φ of $\mathcal{S}$ is satisfiable in a model of Ω_P, then it is satisfiable in a finite model of Ω_P whose size is bounded by a function in the number of variables in φ (small model property).

Let $\mathbb{T}$ be a set theory and $\mathcal{C}$ be a class of open formulae over the language of $\mathbb{T}$. We say that $\mathbb{T}$ has the *small model property* w.r.t. the class $\mathcal{C}$ if for all $\varphi \in \mathcal{C}$ if there exist a model $\mathcal{M}$ of $\mathbb{T}$ and a valuation σ such that $\mathcal{M} \models \varphi[\sigma]$, then there exist a model $\mathcal{N}$ of $\mathbb{T}$ and a valuation ρ such that $\mathcal{N} \models \varphi[\rho]$ and ρ assigns to each variable in φ a finite set whose cardinality is bounded by a computable function of φ.

In the case of the theory Ω_P and the class $\mathcal{S}$ we obtain that the whole $\mathcal{N}$ has a bounded cardinality. For the same class of formulae w.r.t. to a theory with functional symbols and the Regularity axiom (a fragment of ZF) the small model property has been proved in [COU01], but in that case the whole model has to be infinite.

Theorem 2.4 (Small Model Property) *Let φ be a formula in $\mathcal{S}$ and let $|FV(\varphi)| = n$. If φ is satisfiable in a model of Ω_P, then φ is satisfiable in a model of Ω_P of cardinality at most $2^n + n$.*

Proof. Let $\mathcal{N} = \langle N, \in^{\mathcal{N}}, \subseteq^{\mathcal{N}}, \cup^{\mathcal{N}}, \setminus^{\mathcal{N}}, \wp^{\mathcal{N}} \rangle$ be the model of Ω_P in which φ is satisfied by a valuation σ.

For each $m \in N$ let $F(m) = \{x \in FV(\varphi) \mid m \in^{\mathcal{N}} \sigma(x)\}$ and consider the equivalence relation B over N defined as:

$$mBn = \begin{cases} n = m & \text{if } \exists x \in FV(\varphi)(m = \sigma(x)) \\ F(n) = F(m) & \text{otherwise.} \end{cases}$$

Given $m \in N$ let $[m]$ be the equivalence class of m w.r.t. B. Notice that if $m = \sigma(x)$, then $[m] = \{m\}$, otherwise $[m] = \{n \mid F(n) = F(m)\}$.

Consider the model $\mathcal{M} = \langle M, \in^{\mathcal{M}}, \subseteq^{\mathcal{M}}, \cup^{\mathcal{M}}, \setminus^{\mathcal{M}}, \wp^{\mathcal{M}} \rangle$ obtained from $\mathcal{N}$ in the following way:

$$\begin{array}{rcl} M & = & \{[m] \mid m \in N\} \\ [m_1] \in^{\mathcal{M}} [m_2] & \leftrightarrow & \exists m_1' \in [m_1], m_2' \in [m_2](m_1' \in^{\mathcal{N}} m_2') \\ [m_1] \subseteq^{\mathcal{M}} [m_2] & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_1] \to [m] \in^{\mathcal{M}} [m_2]) \\ \cup^{\mathcal{M}}([m_1],[m_2],[m_3]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_3] \leftrightarrow ([m] \in^{\mathcal{M}} [m_1] \vee [m] \in^{\mathcal{M}} [m_2])) \\ \setminus^{\mathcal{M}}([m_1],[m_2],[m_3]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_3] \leftrightarrow ([m] \in^{\mathcal{M}} [m_1] \wedge [m] \notin^{\mathcal{M}} [m_2])) \\ \wp^{\mathcal{M}}([m_1],[m_2]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_2] \leftrightarrow [m] \subseteq^{\mathcal{M}} [m_1]) \end{array}$$

Clearly $\mathcal{M}$ is a finite model of Ω_P. In particular $|M| \leq 2^n + n$, since there is at most one equivalence class for each subset of $FV(\varphi)$ plus one singleton equivalence class for each element of $FV(\varphi)$.

Consider the valuation θ over $\mathcal{M}$ defined as

$$\forall x \in FV(\varphi)(\theta(x) = [\sigma(x)] = \{\sigma(x)\}).$$

We want to prove that θ satisfies φ in $\mathcal{M}$.
If in φ there is a literal of the form $x \in y$, then $\sigma(x) \in^{\mathcal{N}} \sigma(y)$, hence $\theta(x) \in^{\mathcal{M}} \theta(y)$.
Similarly we obtain the thesis for the literals of the form $x \notin y$ and $x \neq y$.
If in φ there is a literal of the form $\cup(x, y, z)$, then: if $[m] \in^{\mathcal{M}} \theta(z)$, then there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(z)$, hence $m' \in^{\mathcal{N}} \sigma(x)$ or $m' \in^{\mathcal{N}} \sigma(y)$, from which $[m] \in^{\mathcal{M}} \theta(x)$ or $[m] \in^{\mathcal{M}} \theta(y)$; if $[m] \in^{\mathcal{M}} \theta(x)$ (or $\theta(y)$) we similarly obtain that $[m] \in \theta(z)$.
If in φ there is a literal of the form $\setminus(x, y, z)$, then: if $[m] \in^{\mathcal{M}} \theta(z)$, then there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(z)$, from which $m' \in^{\mathcal{N}} \sigma(x)$ and $m' \notin^{\mathcal{N}} \sigma(y)$, using the definition of B we obtain that for all $m' \in [m]$ it holds that $m' \notin^{\mathcal{N}} \sigma(y)$, hence $[m] \in^{\mathcal{M}} \theta(x)$ and $[m'] \notin^{\mathcal{M}} \theta(y)$; if $[m] \in^{\mathcal{M}} \theta(x)$ and $[m] \notin^{\mathcal{M}} \theta(y)$, we similarly obtain $[m] \in^{\mathcal{M}} \theta(z)$.
Assume that in φ there is a literal of the form $\wp(x, y)$. If $[m] \in^{\mathcal{M}} \theta(y)$, then we have to prove that $[m] \subseteq^{\mathcal{M}} \theta(x)$, i.e. we have to prove that if $[n] \in^{\mathcal{M}} [m]$, then $[n] \in^{\mathcal{M}} \theta(x)$. Hence our

hypothesis is that there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(y)$ and there exist $n'' \in [n]$ and $m'' \in [m]$ such that $n'' \in^{\mathcal{N}} m''$. If we prove that $n'' \in^{\mathcal{N}} \sigma(x)$ we have the thesis. From the fact that $m', m'' \in [m]$ and $m' \in^{\mathcal{N}} \sigma(y)$ we obtain that $m'' \in^{\mathcal{N}} \sigma(y)$, hence since σ satisfies $\wp(x, y)$, from $n'' \in^{\mathcal{N}} m''$ we obtain $n'' \in^{\mathcal{N}} \sigma(x)$. If $[m] \subseteq^{\mathcal{M}} \theta(x)$, then we have to prove that $[m] \in^{\mathcal{M}} \theta(y)$, i.e. we have to prove that there exists $m' \in [m]$ such that $m' \in \sigma(y)$. We prove that $m \subseteq^{\mathcal{N}} \sigma(x)$. If $n \in^{\mathcal{N}} m$, then $[n] \in^{\mathcal{M}} [m]$, hence $[n] \in^{\mathcal{M}} \theta(x)$. This means that there exists $n' \in [n]$ such that $n' \in^{\mathcal{N}} \sigma(x)$. This last implies that for all $n' \in [n]$ we have $n' \in^{\mathcal{N}} \sigma(x)$, hence $n \in^{\mathcal{N}} \sigma(x)$. So we have proved that $m \subseteq^{\mathcal{N}} \sigma(x)$, hence, since σ satisfies $\wp(x, y)$, we have that $m \in^{\mathcal{N}} \sigma(y)$, from which $[m] \in^{\mathcal{M}} \theta(y)$. □

Example 2.5 *Consider the formula* φ

$$\cup(x, y, y).$$

A possible infinite model of Ω_P *can be obtained considering integer numbers as sets and using* $\leq$ *to model the membership relation. In particular it is*

$$\mathcal{N} = \langle N, \in^{\mathcal{N}}, \subseteq^{\mathcal{N}}, \cup^{\mathcal{N}}, \setminus^{\mathcal{N}}, \wp^{\mathcal{N}} \rangle,$$

where

$$N = \mathbb{Z}$$
$$n \in^{\mathcal{N}} m \leftrightarrow n \leq m$$

As a consequence using the axioms in Ω_P *we obtain that the interpretation of the other symbols is*

$$\cup^{\mathcal{N}} = \{(n, m, r) \mid r = max(n, m)\}$$
$$\setminus^{\mathcal{N}} = \emptyset$$
$$\wp^{\mathcal{N}} = \{(n, n) \mid n \in N\}$$

In this model a valuation σ *which satisfies* φ *is:*

$$\sigma(x) = -1, \sigma(y) = 2.$$

In Figure 2.4 we describe pictorially the finite model construction of the proof of Theorem 2.4 relative to this case.

2.4 The Theory Ω_P and Modal Logic

The □-as-$\wp$ translation, briefly introduced in Section 1.2.7 and presented in details in [DMP95] and in a Chapter 12 of [COP01], refers to the theory Ω which is stronger than Ω_P.

The translation maps decision problems coming from Modal Logic into set-theoretical decision problems. In this section we give a version of the □-as-$\wp$ translation adapted to Ω_P. In this way on the one hand we give another motivation for the decision problem we are interested in, and on the other hand we justify the fact that some of the techniques we use in this chapter are generalization of standard techniques of Modal Logic (for instance the quotient technique used in the previous section).

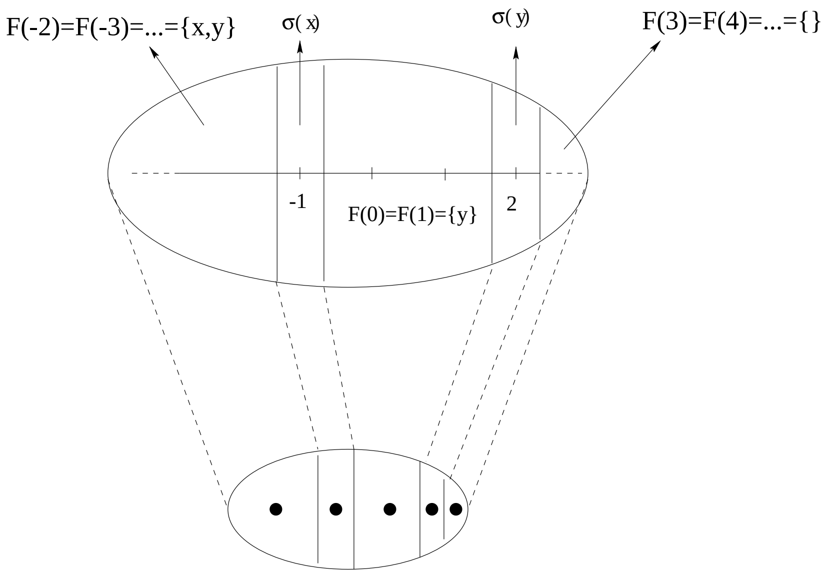


Figure 2.4: Model Reduction.

Definition 2.6 *Given a modal formula γ Consider the set-theoretic formula γ^t inductively defined as follows:*

$$\begin{array}{rcl} p^t & = & \top \\ (\gamma)^t = (\alpha \vee \beta)^t & = & \cup(x_\alpha, x_\beta, x_\gamma) \wedge (\alpha)^t \wedge (\beta)^t \\ (\gamma)^t = \neg(\alpha)^t & = & \backslash(x, x_\alpha, x_\gamma) \wedge (\alpha)^t \\ (\gamma)^t = \Box(\alpha)^t & = & \wp(x_\alpha, x_\gamma) \wedge (\alpha)^t \end{array}$$

The set-theoretic translation $\gamma^p(x)$ of the formula γ is the formula

$$\forall \vec{x}_\alpha(\gamma^t \rightarrow x \subseteq x_\gamma),$$

where $\vec{x}_\alpha$ is the set of variables associated to the subformulae of γ.

Notice that in γ^t the variables of the form x_p, with p predicate, never occur as last element of a predicate $\cup, \backslash, \wp$, i.e. they are completely free, while all the other variables are constrained by a predicate.

Example 2.7 *Consider the formula γ*

$$\Box p \rightarrow \neg\Box\Box p.$$

We have that γ^t is the formula

$$\begin{array}{c} \wp(x_p, x_{\Box p}) \wedge \backslash(x, x_{\Box p}, x_{\neg\Box p}) \wedge \\ \wp(x_{\Box p}, x_{\Box\Box p}) \wedge \backslash(x, x_{\Box\Box p}, x_{\neg\Box\Box p}) \wedge \\ \cup(x_{\neg\Box p}, x_{\neg\Box\Box p}, x_\gamma). \end{array}$$

and γ^p is the formula

$$\forall x_p, x_{\Box p}, \ldots, x_\gamma(\gamma^t \rightarrow x \subseteq x_\gamma).$$

The translation we obtain by applying the $\Box$-as-$\wp$ translation presented in [DMP95] is the formula γ^*

$$\forall x_p(x \subseteq (x \setminus \wp(x_p)) \cup (x \setminus \wp(\wp(x_p)))).$$

In practice the formula γ^t is a flattening *(see [COP01]) of γ^*.*

The formula $\gamma^p(x)$ translates in the language of Ω_P the modal formula γ. In such a translation the variable x *represent* the set of worlds W of a frame $\mathcal{F}$. The main idea behind the translation is to map the accessibility relation R into the membership relation. This implies in particular that W have to be a transitive set (see [COP01]), hence we have to use the formula $Tr(x)$ to impose to x to be transitive.

Definition 2.8 *Let $Tr(x)$ be the formula*

$$\forall y(\wp(x,y) \rightarrow x \subseteq y).$$

Before proving the main theorem of this section we need to establish a link between frames and models of Ω_P.

A frame $\mathcal{F} = \langle W, R\rangle$ is nothing but a graph whose nodes are the worlds in W and whose edges are the pairs in R. We present a general method to map a graph $G = \langle N, \gtrdot\rangle$ into a model G_{ω_P} of Ω_P. The main features of the model G_{ω_P} are that given two elements m_1, m_2 of the model:

- we do not add out-going edges from the nodes in G, i.e. G is a complete subgraph of G_{ω_P};
- there always exists at least one element m_3 in the model such that $\cup(m_1, m_2, m_3)$;
- there always exists at least one element m_3 in the model such that $\setminus(m_1, m_2, m_3)$;
- there always exists at least one element m_3 in the model such that $\wp(m_1, m_3)$.

We do not give the proofs here because we will see them in Chapter 4. As a matter of fact this method gives us not only a model of Ω_P, but also a model of Ω.

First, we introduce some notations. Let $G = \langle N, \gtrdot\rangle$, let $v, u \in N$ be two nodes, and let $V \subseteq N$ be a set of nodes:

- $\mathsf{in}(G, v, u, \cup)$ iff there exists a node w in N such that
  $\forall x \in N((x \lessdot v \vee x \lessdot u) \leftrightarrow x \lessdot w)$;
- $\mathsf{in}(G, v, u, \setminus)$ iff there exists a node w in N such that
  $\forall x \in N((x \lessdot v \wedge \neg x \lessdot u) \leftrightarrow x \lessdot w)$;
- $\mathsf{sb}(G, V, v)$ iff $\forall x \in N(x \in V \rightarrow x \lessdot v)$;
- $\mathsf{in}(G, V, v, \subseteq)$ iff there exists a node w in N such that
  $\forall x \in N((x \in V \wedge x \lessdot v) \leftrightarrow x \lessdot w)$;
- $\mathsf{csb}(G, V, v)$ iff $\neg\mathsf{sb}(G, V, v) \vee \mathsf{in}(G, V, v, \subseteq)$;
- $\mathsf{in}(G, v, \wp)$ iff there exists a node w in N such that
  $\forall x \in N(\forall y \in N(y \lessdot x \rightarrow y \lessdot v) \leftrightarrow x \lessdot w)$.

Intuitively $\mathsf{in}(G, v, u, \cup)$ means that there is a node w in G which can be used has $\cup(v, u, w)$ (i.e., it satisfies the axiom $(\cup)$ w.r.t. v and u). Since $x \in v$ is interpreted on the graph as $\langle v, x\rangle$, $\mathsf{sb}(G, V, v)$ means that V is a subset of v, while $\mathsf{in}(G, V, v, \subseteq)$ expresses the fact that there is a node in G which can represent the subset $V \cap v$ of v. It follows that $\neg\ \mathsf{csb}(G, V, v)$ is equivalent to the fact that V is a subset of v and there is no node in G which represents this subset (csb is an anonymous for *complete subset*, i.e. there are all the subsets).

Starting from $G = G_0$ we introduce a sequence of graphs G_n and then we show that the graph obtained as the union of all the graphs in the sequence is the model of Ω_P we are looking for.

If we have defined G_n, with $n = 4m$, we inductively define

$$G_{n+i} = \langle N_{n+i}, \gtrdot_{n+i} \rangle,$$

for $i = 1, 2, 3, 4$, as follows:

$$\begin{array}{rcl} N_{n+1} & = & N_n \cup \{\langle v, u, \cup \rangle \mid v, u \in N_n, \neg \mathsf{in}(G_n, v, u, \cup)\} \\ \gtrdot_{n+1} & = & \gtrdot_n \cup \{x \lessdot_{n+1} \langle v, u, \cup \rangle \mid \langle v, u, \cup \rangle \notin N_n, x \lessdot_n v \text{ or } x \lessdot_n u\} \end{array}$$

$$\begin{array}{rcl} N_{n+2} & = & N_{n+1} \cup \{\langle v, u, \setminus \rangle \mid v, u \in N_{n+1}, \neg \mathsf{in}(G_{n+1}, v, u, \setminus)\} \\ \gtrdot_{n+2} & = & \gtrdot_{n+1} \cup \{x \lessdot_{n+2} \langle v, u, \setminus \rangle \mid \langle v, u, \setminus \rangle \notin N_{n+1}, x \lessdot_{n+1} v, \neg x \lessdot_{n+1} u\} \end{array}$$

$$\begin{array}{rcl} N_{n+3} & = & N_{n+2} \cup \{\langle v, V, \subseteq \rangle \mid \{v\}, V \subseteq N_{n+2}, \neg \mathsf{csb}(G_{n+2}, V, v)\} \\ \gtrdot_{n+3} & = & \gtrdot_{n+2} \cup \{x \lessdot_{n+3} \langle v, V, \subseteq \rangle \mid \langle v, V, \subseteq \rangle \notin N_{n+2}, x \in V\} \end{array}$$

$$\begin{array}{rcl} N_{n+4} & = & N_{n+3} \cup \{\langle v, \wp \rangle \mid v \in N_{n+3}, \neg \mathsf{in}(G_{n+3}, v, \wp)\} \\ \gtrdot_{n+4} & = & \gtrdot_{n+3} \cup \{x \lessdot_{n+4} \langle v, \wp \rangle \mid \langle v, \wp \rangle \notin N_{n+3},\ \mathsf{sb}(G_{n+3}, \{y \mid y \lessdot_{n+3} x\}, v)\} \end{array}$$

Let

$$G_{\omega_P} \quad = \quad \langle N_{\omega_P}, \gtrdot_{\omega_P} \rangle, \tag{2.2}$$

with $N_{\omega_P} = \bigcup_{n \in \mathbb{N}} N_n$ and $\gtrdot_{\omega_P} = \bigcup_{n \in \mathbb{N}} \gtrdot_n$.

From G_{ω_P} we obtain an interpretation of $\mathcal{L}$ defining $\forall v_1, v_2, \in N_{\omega_P}$:

- $u \in v$ if and only if $u \lessdot_{\omega_P} v$;
- $\cup(v, u, w) \leftrightarrow \forall x(x \in w \leftrightarrow x \in v \vee x \in u)$;
- $\setminus(v, u, w) \leftrightarrow \forall x(x \in w \leftrightarrow x \in v \wedge x \notin u)$;
- $u \subseteq v \leftrightarrow \forall x(x \in u \rightarrow x \in v))$;
- $\wp(v, w) \leftrightarrow \forall x(x \in w \leftrightarrow x \subseteq v)$.

We use G_{ω_P} to refer to this interpretation of $\mathcal{L}$. It is immediate to prove that G_{ω_P} is a model of Ω_P and that G is a complete subgraph in G_{ω_P}. All the details are given in Chapter 4: this construction generates models for the functional version Ω of Ω_P, and, hence, since each model of Ω is a model of Ω_P, it generates models of Ω_P. The models we obtain using this construction are particularly rich: we do not only put all the results of the set operations $\cup, \setminus, \wp$ (which are not necessary when we deal with Ω_P), but we also add all the subsets of the elements of the models (which are not necessary even when we deal with Ω). The fact that the models we obtain using this construction are so rich facilitate the proof of the fact that we always obtain a model of Ω. So, we have that $\forall u, v \in N_{\omega_P}$

$$\begin{array}{c} \exists w(\cup(u, v, w)) \\ \exists w(\setminus(u, v, w)) \\ \exists w(\wp(u, w)). \end{array}$$

Notice that also if here G_{ω_P} is not introduced to prove the consistency of Ω_P, which follows immediately from the fact that each graph can be interpreted as a model of Ω_P, the construction of G_{ω_P} has some similarity with the construction proposed by Henkin to build the canonical structure for a theory (the method is used to prove Gödel's Completeness Theorem, see [Sho67]): the main difference is that in Henkin's method it is necessary to start by putting in the model the constants of the language, while here we start from the elements of the graph G.

Definition 2.9 *Let $\mathcal{F} = \langle W, R\rangle$ be a frame and $\mathcal{F}_{\omega_P}$ be the model of Ω_P defined as described above.*

Given a valuation V on $\mathcal{F}$ we define the valuation σ_V on $\mathcal{F}_{\omega_P}$ as

$$
\begin{array}{lcll}
\sigma_V(x) & = & W, & \\
\sigma_V(x_p) & = & \{w \in W \mid \mathcal{F}, V, w \Vdash p\} & \textit{where } p \textit{ is a propositional literal,} \\
\sigma_V(x_{\gamma_1 \vee \gamma_2}) & = & m & \textit{with } \cup(\sigma_V(x_{\gamma_1}), \sigma_V(x_{\gamma_2}), m), \\
\sigma_V(x_{\neg\gamma_1}) & = & m & \textit{with } \setminus(\sigma_V(x), \sigma_V(x_{\gamma_1}), m), \\
\sigma_V(x_{\Box\gamma_1}) & = & m & \textit{with } \wp(\sigma_V(x_{\gamma_1}), m).
\end{array}
$$

Given a valuation σ on $\mathcal{F}_{\omega_P}$ such that $\sigma(x) = W$, we define the valuation V_σ on $\mathcal{F}$ as

$$V_\sigma(p) = \{w \in W \mid w \in_F \sigma(x_p)\}.$$

The following lemmas establish a link between the frame $\mathcal{F}$ and the model $\mathcal{F}_{\omega_P}$ using the valuations σ_V and V_σ.

We consider the language $\mathcal{L}$ enriched with a constant symbol w_c for each element $w \in W$.

Lemma 2.10 *Let V be a valuation over $\mathcal{F}$ and $w \in W$ be a world.*

$$\mathcal{F}_{\omega_P} \models (\gamma^t \rightarrow w_c \in x_\gamma)[\sigma_V] \quad \Leftrightarrow \quad \langle \mathcal{F}, V\rangle, w \Vdash \gamma.$$

Proof. First of all it is immediate to prove by induction on γ and using the definition of σ_V, that for all γ it holds that

$$\mathcal{F}_{\omega_P} \models \gamma^t[\sigma_V].$$

Hence, we prove, by induction on γ, that

$$\mathcal{F}_{\omega_P} \models w_c \in x_\gamma[\sigma_V] \Leftrightarrow \langle \mathcal{F}, V\rangle, w \Vdash \gamma.$$

If $\gamma = p$, then

$$w_c \in \sigma_V(x_p) \Leftrightarrow w \in V(p).$$

If $\gamma = \gamma_1 \vee \gamma_2$, then from $w_c \in \sigma_V(x_\gamma)$ we obtain $w_c \in \sigma_V(x_{\gamma_1})$ or $w_c \in \sigma_V(x_{\gamma_2})$, and hence by inductive hypothesis $\langle \mathcal{F}, V\rangle, w \Vdash \gamma_1$ or $\langle \mathcal{F}, V\rangle, w \Vdash \gamma_2$. The other direction is similar.

If $\gamma = \neg\gamma_1$, then from $w_c \in \sigma_V(x_\gamma)$ we obtain $w_c \notin \sigma_V(x_{\gamma_1})$, and hence, by inductive hypothesis $\langle \mathcal{F}, V\rangle, w \not\Vdash \gamma_1$. The other direction is similar.

If $\gamma = \Box\gamma_1$, then from $w_c \in \sigma_V(x_\gamma)$ we obtain $w_c \subseteq \sigma_V(x_{\gamma_1})$, and hence, by inductive hypothesis $\langle \mathcal{F}, V\rangle, v \Vdash \gamma_1$, for all v such that wRv, i.e. the thesis. If $\langle \mathcal{F}, V\rangle, w \Vdash \gamma$, then for all v such that $v \in w$ we have $\langle \mathcal{F}, V\rangle, v \Vdash \gamma_1$, hence $w_c \subseteq \sigma_V(x_{\gamma_1})$, i.e. $w_c \in \sigma_V(x_\gamma)$. □

Lemma 2.11 *Let σ be a valuation over $\mathcal{F}_{\omega_P}$ such that $\sigma(x) = W$.*

$$\mathcal{F}_{\omega_P} \models (\gamma^t \wedge w_c \notin x_\gamma)[\sigma] \quad \Leftrightarrow \quad \langle \mathcal{F}, V_\sigma\rangle, w \not\Vdash \gamma.$$

Proof. By induction on γ. □

Lemma 2.12 *If there exists σ such that $\sigma(x) = W$ and $\mathcal{F}_{\omega_P} \not\models (\gamma^t \rightarrow x \subseteq x_\gamma)[\sigma]$, then $\langle \mathcal{F}, V_\sigma\rangle \not\Vdash \gamma$.*

Proof. It is a conseguence of Lemma 2.11. □

We are now ready to prove the theorem which establish the link between Modal Logic and Ω_P.

Theorem 2.13 (Correctness and Completeness) *Given a complete logic Λ obtained by adding the axiom Ax to the modal logic K, it holds that*

$$\vdash_\Lambda \gamma \quad \Leftrightarrow \quad \Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \gamma^p(x)).$$

Proof. ($\Rightarrow$) We prove that if $\vdash_\Lambda \gamma$, then $\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \gamma^p(x))$ by induction on the length of the derivation.

First of all, we consider the case of tautologies, and show that the thesis is true for the axiom schemata:

$$\begin{array}{rcl} Ax_1 & = & \alpha \rightarrow (\beta \rightarrow \alpha) \\ Ax_2 & = & (\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma)) \\ Ax_3 & = & (\neg\beta \rightarrow \neg\alpha) \rightarrow ((\neg\beta \rightarrow \alpha) \rightarrow \beta). \end{array}$$

We have that Ax_1 is equivalent to

$$\neg\alpha \vee (\neg\beta \vee \alpha).$$

We obtain that

$$\begin{array}{rcl} Ax_1^t & = & \backslash(x, x_\alpha, x_{\neg\alpha}) \wedge \alpha^t \wedge \backslash(x, x_\beta, x_{\neg\beta}) \wedge \beta^t \wedge \\ & & \cup(x_{\neg\beta}, x_\alpha, x_{\neg\beta\vee\alpha}) \wedge \cup(x_{\neg\alpha}, x_{\neg\beta\vee\alpha}, x_{Ax1}). \end{array}$$

We must prove that

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \forall\vec{x}(Ax_1^t \rightarrow x \subseteq x_{Ax1})).$$

Let us assume by contradiction that there exists a model $\mathcal{M}$ of Ω_P in which the formula is not valid. This means that there exists m in $\mathcal{M}$ such that $Tr(m)$, $Ax^p(m)$, and there exist $\vec{m}$ in $\mathcal{M}$ such that $Ax_1^t(\vec{m})$ and $m \not\subseteq m_{Ax1}$. Let n be an element of m which is not an element of m_{Ax1}. Using $Ax_1^t(\vec{m})$ and the axioms in Ω_P we obtain that n is not an element of $m_{\neg\alpha}$ and it is not an element of $m_{\neg\beta\vee\alpha}$, from which we obtain that n is not an element of m_α, hence, since n is an element of m we obtain that m is an element of $m_{\neg\alpha}$ which is a contradiction.

The axiom Ax_2 is equivalent to

$$(\neg\alpha \vee (\neg\beta \vee \gamma)) \vee (\neg(\neg\alpha \vee \beta) \vee (\neg\alpha \vee \gamma)).$$

We have that

$$\begin{array}{rcl} Ax_2^t & = & \backslash(x, x_\alpha, x_{\neg\alpha}) \wedge \alpha^t \wedge \\ & & \backslash(x, x_\beta, x_{\neg\beta}) \wedge \beta^t \wedge \\ & & \cup(x_{\neg\beta}, x_\gamma, x_{\neg\beta\vee\gamma}) \wedge \gamma^t \wedge \\ & & \cup(x_{\neg\alpha}, x_{\neg\beta\vee\gamma}, x_{\neg\alpha\vee(\neg\beta\vee\gamma)}) \wedge \\ & & \backslash(x, x_{\neg\alpha\vee(\neg\beta\vee\gamma)}, x_{\neg(\neg\alpha\vee(\neg\beta\vee\gamma))}) \wedge \\ & & \cup(x_{\neg\alpha}, x_\beta, x_{\neg\alpha\vee\beta}) \wedge \\ & & \backslash(x, x_{\neg\alpha\vee\beta}, x_{\neg(\neg\alpha\vee\beta)} \wedge \\ & & \cup(x_{\neg\alpha}, x_\gamma, x_{\neg\alpha\vee\gamma}) \wedge \\ & & \cup(x_{\neg(\neg\alpha\vee\beta)}, x_{\neg\alpha\vee\gamma}, x_{\neg(\neg\alpha\vee\beta)\vee(\neg\alpha\vee\gamma)}) \wedge \\ & & \cup(x_{\neg(\neg\alpha\vee(\neg\beta\vee\gamma))}, x_{\neg(\neg\alpha\vee\beta)\vee(\neg\alpha\vee\gamma)}, x_{Ax2}). \end{array}$$

We must prove that

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \forall\vec{x}(Ax_2^t \rightarrow x \subseteq x_{Ax2})).$$

Let us assume by contradiction that there is a model $\mathcal{M}$ of Ω_P in which the formula is not valid. This means that there exists m in $\mathcal{M}$ such that $Tr(m)$, $Ax^p(m)$, and there exist $\vec{m}$ in $\mathcal{M}$ such that $Ax_2^t(\vec{m})$ and $m \not\subseteq m_{Ax2}$. Let n be an element of m which is not an element of m_{Ax2}. Using Ax_2^t and the axioms in Ω_P we obtain that n is not an element of $m_{\neg(\neg\alpha\vee(\neg\beta\vee\gamma))}$, and it is not an element of $m_{\neg(\neg\alpha\vee\beta)\vee(\neg\alpha\vee\gamma)}$. Hence we obtain that n is an element of $m_{\neg\alpha\vee(\neg\beta\vee\gamma)}$, it is not an element of $m_{\neg(\neg\alpha\vee\beta)}$ (2), and it is not an element of $m_{\neg\alpha\vee\gamma}$. From this last we obtain that n is not an element of $m_{\neg\alpha}$, hence in the first it must be the case that n is an element of $m_{\neg\beta\vee\gamma}$. But from the last assertion of the previous sentence we also obtain that n is not an element of m_γ, hence it must be an element of $m_{\neg\beta}$, which implies that n is not an element of m_β. From (2) we obtain that n is an element of $m_{\neg\alpha\vee\beta}$, and since we already have that n is not an element of $m_{\neg\alpha}$ it must be that n is an element of β which give us a contradiction.

Axiom Ax_3 is equivalent to

$$\neg(\neg\neg\beta \vee \neg\alpha) \vee (\neg(\neg\neg\beta \vee \alpha) \vee \beta).$$

Hence we obtain that

$$\begin{array}{rcl} Ax_3^t & = & \backslash(x, x_\beta, x_{\neg\neg\beta}) \wedge \beta^t \wedge \backslash(x, x_{\neg\beta}, x_{\neg\neg\beta}) \wedge \\ & & \backslash(x, x_\alpha, x_{\neg\alpha}) \wedge \alpha^t \wedge \\ & & \cup(x_{\neg\neg\beta}, x_{\neg\alpha}, x_{\neg\neg\beta\vee\neg\alpha}) \wedge \\ & & \backslash(x, x_{\neg\neg\beta\vee\neg\alpha}, x_{\neg(\neg\neg\beta\vee\neg\alpha)} \wedge \\ & & \cup(x_{\neg\neg\beta}, x_\alpha, x_{\neg\neg\beta\vee\alpha}) \wedge \\ & & \backslash(x, x_{\neg\neg\beta\vee\alpha}, x_{\neg(\neg\neg\beta\vee\alpha)} \wedge \\ & & \cup(x_{\neg(\neg\neg\beta\vee\alpha), x_\beta, x_{\neg(\neg\neg\beta\vee\alpha)\vee\beta}} \wedge \\ & & \cup(x_{\neg(\neg\neg\beta\vee\neg\alpha)}, x_{\neg(\neg\neg\beta\vee\alpha)\vee\beta}, x_{Ax3}). \end{array}$$

We must prove that

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \to \forall\vec{x}(Ax_3^t \to x \subseteq x_{Ax3})).$$

Let us assume by contradiction that there is a model $\mathcal{M}$ of Ω_P in which the formula is not valid. This means that there exists m in $\mathcal{M}$ such that $Tr(m)$, $Ax^p(m)$, and there exist $\vec{m}$ in $\mathcal{M}$ such that $Ax_3^t(\vec{m})$ and $m \not\subseteq m_{Ax3}$. Let n be an element of m which is not an element of m_{Ax3}. We obtain that n is an element of $m_{\neg\neg\beta\vee\neg\alpha}$, it is an element of $m_{\neg\neg\beta\vee\alpha}$, and it is not an element of m_β. But from the first and the second we obtain that n is an element of $m_{\neg\neg\beta}$, and hence it is an element of m_β, which is a contradiction.

Now we prove the closure under *modus ponens*, i.e.: if $\vdash_\Lambda \alpha$ and $\vdash_\Lambda (\alpha \to \beta)$, then $\vdash_\Lambda \beta$. We have to prove that if

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \to \forall\vec{x}(\alpha^t \to x \subseteq x_\alpha)) \qquad (1)$$

and

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \to \forall\vec{x}((\alpha \to \beta)^t \to x \subseteq x_{\alpha\to\beta})), \qquad (2)$$

then

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \to \forall\vec{x}(\beta^t \to x \subseteq x_\beta)).$$

We recall that

$$(\alpha \to \beta)^t = \alpha^t \wedge \beta^t \wedge \backslash(x, x_\alpha, x_{\neg\alpha}) \wedge \cup(x_{\neg\alpha}, x_\beta, x_{\alpha\to\beta}).$$

Let us assume by contradiction that there exists a model $\mathcal{M}$ of Ω_P such that $Tr(m)$, $Ax^p(m)$ and $m \not\subseteq m_\beta$. Let n be an element of m which is not an element of m_β. From (2) we obtain that n must be an element of $m_{\neg\alpha}$, i.e. n is not an element of m_α, and this is in contradiction with (1).

We prove the closure under *necessitation*, i.e.: if $\vdash_\Lambda \alpha$, then $\vdash_\Lambda \Box\alpha$. We have to prove that if

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \forall \vec{x}(\alpha^t \rightarrow x \subseteq x_\alpha)),$$

then

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \forall \vec{x}((\alpha^t \wedge \wp(x_\alpha, x_{\Box\alpha}) \rightarrow x \subseteq x_{\Box\alpha})).$$

This is equivalent to prove that all the elements y's of x are subsets of x_α. If y is an element of x and t is an element of y, then, from $Tr(x)$, we have that t is an element of x, hence from the hypothesis we obtain that y is an element of x_α.

We consider the K axiom schema:

$$\Box(\alpha \rightarrow \beta) \rightarrow (\Box\alpha \rightarrow \Box\beta).$$

We have to show that:

$$\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \forall \vec{x}(K^t \rightarrow x \subseteq x_k)),$$

where

$$\begin{aligned} K^t \quad = \quad & \backslash(x, x_\alpha, x_{\neg\alpha}) \wedge \alpha^t \wedge \cup(x_{\neg\alpha}, x_\beta, x_{\neg\alpha\vee\beta}) \wedge \beta^t \wedge \\ & \wp(x_{\neg\alpha\vee\beta}, x_{\Box(\neg\alpha\vee\beta)}) \wedge \backslash(x, x_{\Box(\neg\alpha\vee\beta)}, x_{\neg(\Box(\neg\alpha\vee\beta))}) \wedge \\ & \wp(x_\alpha, x_{\Box\alpha}) \wedge \backslash(x, x_{\Box\alpha}, x_{\neg\Box\alpha}) \wedge \\ & \wp(x_\beta, x_{\Box\beta}) \wedge \cup(x_{\neg\Box\alpha}, x_{\Box\beta}, x_{\neg\Box\alpha\vee\Box\beta}) \wedge \\ & \cup(x_{\neg(\Box(\neg\alpha\vee\beta))}, x_{\neg\Box\alpha\vee\Box\beta}, x_k). \end{aligned}$$

Let us assume by contradiction that there exists a model $\mathcal{M}$ of Ω_P in which the formula is not valid. This means that in $\mathcal{M}$ there exist m and $\vec{m}$ such that $Tr(m)$, $Ax^p(m)$, $K^t(m, \vec{m})$, and $m \not\subseteq m_k$. Let n be an element of m which is not an element of m_k. We obtain that n is not an element of $m_{\neg(\Box(\neg\alpha\vee\beta))}$ and it is not an element of $m_{\neg\Box\alpha\vee\Box\beta}$. Hence n is an element of $m_{\Box(\neg\alpha\vee\beta)}$, it is not an element of $m_{\neg\Box\alpha}$, and it is not an element of $m_{\Box\beta}$. Hence we obtain that $n \subseteq m_{\neg\alpha\vee\beta}$, $n \subseteq m_\alpha$, and there exists an element n' of n which is not an element of β. Since n' is an element of n, it must be an element of $m_{\neg\alpha\vee\beta}$, hence two cases are possible: n' is an element of $m_{\neg\alpha}$ or n' is an element of m_β. In the first case we obtain a contradiction, since we obtain that n' is not an element of m_α, while $n \subseteq m_\alpha$. In the second case we also obtain a contradiction, since n' is not an element of β by hypothesis.

Finally we have to prove the closure for the axiom schema Ax. Let us assume that $p_1, \ldots, p_n$ are the propositional literals in Ax. We have to prove that

$$\begin{aligned} \Omega_P \quad \vdash \quad & \forall x(Tr(x) \wedge Ax(p_1, \ldots, p_n)^p(x) \rightarrow \\ & \qquad \forall \vec{x}(Ax(\alpha_1, \ldots, \alpha_n)^t(x, \vec{x}) \rightarrow x \subseteq x_{Ax(\alpha_1, \ldots, \alpha_n)}), \end{aligned}$$

where

$$Ax(p_1, \ldots, p_n)^t(x) = \forall \vec{x}(Ax(p_1, \ldots, p_n)^t(x, \vec{x}) \rightarrow x \subseteq x_{Ax(p_1, \ldots, p_n)}).$$

It is immediate to prove that

$$Ax(\alpha_1, \ldots, \alpha_n)^t(x, \vec{x}) = Ax(p_1, \ldots, p_n)^t(x, \vec{x})[x_i/x_{\alpha_i}]_{i=1}^n \wedge \bigwedge_{i=1}^n \alpha_i^t.$$

Hence from the hypothesis we obtain that

$$x \subseteq x_{Ax(\alpha_1,\dots,\alpha_n)}.$$

($\Leftarrow$) As far as the opposite direction is concerned we have to prove that if $\Omega_P \vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \gamma^p(x))$, then $\vdash_\Lambda \gamma$. We prove that if $\not\vdash_\Lambda \gamma$, then $\Omega_P \not\vdash \forall x(Tr(x) \wedge Ax^p(x) \rightarrow \gamma^p(x))$. From $\not\vdash_\Lambda \gamma$, since Λ is a complete modal logic we obtain that there exists a frame $\mathcal{F}$ such that $\mathcal{F} \Vdash \Lambda$ and there exists a valuation V and a world w such that $\mathcal{F}, V, w \not\Vdash \gamma$.

It is immediate to prove that in $\mathcal{F}_{\omega_P}$ it holds $Tr(W)$.

From Lemma 2.12 and from $\mathcal{F} \Vdash \Lambda$ we obtain that

$$\mathcal{F}_{\omega_P} \models \forall x(Ax^p(W)).$$

From Lemma 2.10 and from $\mathcal{F}, V, w \not\Vdash \gamma$ we obtain that

$$\mathcal{F}_{\omega_P} \models \neg\gamma^p(W).$$

□

The set-theoretical formula involved in Theorem 2.13 has also universal quantifiers, while the formulae in the class $\mathcal{S}$ are existentially quantified. The following corollary clarify which (decidable) problem of Modal Logic is generalized by our satisfiability problem.

Corollary 2.14 *A modal formula γ is satisfiable if and only if*

$$Tr(x) \wedge \gamma^t \wedge \backslash(x, x_\gamma, x_{\neg\gamma}) \wedge x \not\subseteq x_{\neg\gamma}$$

is satisfiable in a model of Ω_P.

Proof. If γ is satisfiable if and only if $\neg\gamma$ is not a theorem of K, hence, from Theorem 2.13, if and only if $\Omega_P \not\vdash \forall x(Tr(x) \rightarrow \forall\vec{x}((\gamma^t \wedge \backslash(x, x_\gamma, x_{\neg\gamma}) \rightarrow x \subseteq x_{\neg\gamma})$. This means that there exists a model of Ω_P in which $Tr(x) \wedge \gamma^t \wedge \backslash(x, x_\gamma, x_{\neg\gamma}) \wedge x \not\subseteq x_{\neg\gamma}$ is satisfiable. □

Since the formula in Corollary 2.14 is a formula in $\mathcal{S}_\exists$, we have that the small model property we have proved can be seen as a generalization of the small model property for the modal logic K. In particular if we apply the technique presented in the proof of Theorem 2.4 to a formula of the form $Tr(x) \wedge \gamma^t \wedge \backslash(x, x_\gamma, x_{\neg\gamma}) \wedge x \not\subseteq x_{\neg\gamma}$ we obtain that the quotient of the frame is exactly the one usually presented for the modal logic K: we have to put the elements of the set-theoretical model which are representation of the variables in the formula into singleton classes, but these elements of the model are not elements of the frame, they are the elements added to represent the interpretation of the propositional literals. The fact that we have used a theory with predicates plays two roles in the generalization of the quotient technique: on the one hand it avoids to have to add elements to obtain a model of Ω_P, on the other hand it avoids that by adding new elements we have to change the interpretation of $\wp$.

2.5 A Tableaux Decision Procedure

From Theorem 2.4 we immediately obtain a decision procedure: in order to decide if a formula φ is satisfiable in a model of Ω_P we could check exhaustively in all the models of Ω_P

of size at most $2^{|FV(\varphi)|} + |FV(\varphi)|$, which, up to isomorphisms, are a finite number. However, this is not an efficient way to check satisfiability.

A way to improve our search for a model is to guide the construction of a model through a tableau method. We present in this section a tableau procedure and we prove its correctness and completeness.

The formulae in $\mathcal{S}$ are disjunctions of conjunctions of literals. Clearly a formula in $\mathcal{S}$ of the form $\varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_k$, where each φ_i is a conjunction of literals is satisfiable if and only if there exists an index j, with $1 \leq j \leq k$, such that φ_j is satisfiable. For this reason, without lost of generality, we can restrict ourselves to decide the satisfiability of formulae of $\mathcal{S}$ which in particular are conjunctions of literals.

The procedure Satisfiable takes in input a formula φ which is a conjunction of literals of $\mathcal{S}$ and non-deterministically decides if there exists a model of Ω_P in which φ is satisfiable.

First the procedure Decide non-deterministically decides all the membership relations between the variables in the formula. This is necessary since we need to explicit all the literals of the form $z \notin x$ whenever we have also a literal of the form $\wp(y, x)$: in this case we have to put in z an element which is not an element of y.

As a second step we enter in a repeat-loop in which we repeatedly apply the rules in Test, Add, and Decide.

The rules in Test allow us to propagate the membership relations using the definitions of the predicates and to check that there are no inconsistency between these propagations and the choices we perform with the Decide-rule. Let $\frac{l}{r}$ be one of the rules in Test. Applying once this rule to a formula φ in the class $\mathcal{S}$ consists in:

1. check if there exists a subformula ℓ in φ and a substitution σ such that $l[\sigma] = \ell$;
2. check if $r[\sigma]$ is already a subformula of φ and, if this is not the case, map φ into $\varphi \wedge r[\sigma]$.

Whenever we are in presence of two literals of the form $\wp(y, x) \wedge z \notin x$ we have to ensure that in z there is an element which is not an element of y: the rule in Add is devoted to this purpose. Notice that the rule is not applied to the new variables just added in order to avoid to enter into an infinite loop.

Finally, the rule in Decide non-deterministically decides which are the variables of the initial formula to which the new variables have to belong.

Similarly to what happened in the proof of Theorem 2.4, if, for instance, during the last iteration of the repeat-loop we added only a new variable m and there was a variable n previously added which belongs to the same set of variables to which m belongs, then there is no reason to consider m different from n.

All the non-deterministic branches of Satisfiable terminate after at most $4^{|FV(\varphi)|} + 1$ iterations, since at most $4^{|FV(\varphi)|}$ different pairs can be put into the set B. In particular B and $NewB$ are used to force termination in the following way: whenever the situation concerning the membership relations of the variables introduced during the last iteration is similar to situations occurred during previous iteration we can stop the procedure, since in this case we are able to produce an infinite branch and to generate a solution for this infinite branch.

If, at the end of all the non-deterministic branches, we have $\perp$ in φ, then φ is not satisfiable. Otherwise there exists a model of Ω_P in which φ is satisfiable.

Lemma 2.15 (Termination) *Let φ be a conjunction of literals in $\mathcal{S}$.*
All the non-deterministic computations of Satisfiable*(φ) terminate after at most $4^{|FV(\varphi)|+1}$ iterations of the repeat-loop.*

```
Satisfiable(φ) :
S := FV(φ);
for all n ∈ S do φ := Decide(φ, n);
B := ∅;
Alt := ⊥;
repeat
        ψ := φ;
        φ := Test(φ);
        φ := Add(φ, ψ);
        N := FV(φ) \ FV(ψ);
        for all n ∈ N do φ := Decide(φ, n);
        NewB := {⟨V_n, V_m⟩ | n ∈ N, (n ∈ m) ∈ φ};
        if NewB ⊆ B then Alt := ⊤ else B := B ∪ NewB;
until Alt or ⊥ ∈ φ.
```

Figure 2.5: The Procedure Satisfiable.

```
Decide(φ, n) :
let V_n ⊆ S;
φ := φ ∧ ⋀_{y∈V_n} n ∈ y ∧ ⋀_{y∈S\V_n} n ∉ y;
return φ.
```

Figure 2.6: The Procedure Decide.

Proof. We have that $|S| = |FV(\varphi)|$. Hence $|B| \leq 4^{|S|}$, and when $|B| = 4^{|S|}$ we are sure that Alt is $\top$ at the next iteration. $\square$

Lemma 2.16 (Completeness) *If φ is a conjunction of literals in $\mathcal{S}$ and it is satisfiable in a model of Ω_P, there exists a non-deterministic computation of* Satisfiable*(φ) which terminates without $\perp$ in φ.*

Proof. Let $\mathcal{M}$ be a model of Ω_P and σ a valuation which satisfies φ in $\mathcal{M}$. We use $\mathcal{M}$ and σ in order to obtain a non-deterministic computation of Satisfiable(φ) which terminates without $\perp$ in φ.

For each variable n in S we choose $V_n = \{x \in S \,|\, \sigma(n) \in \sigma(x)\}$. It is not possible that we obtain $\perp$ during the first execution of Test, since σ satisfies φ, and hence the membership relations between the variables in S are coherent with the definition of the predicates. Clearly during the execution of Test if we have to choose $x \in y \vee x \in z$ we use σ to decide. Then we execute Add, and when we execute Decide we use σ in order to perform our choices: if m is a new variable which has been added using the rule in Add, then there are two variables z and x in S, such that $\wp(y, x)$, $z \notin x$, $m \in z$ and $m \notin y$ are all literals in φ; hence we also have that $\wp^{\mathcal{M}}(\sigma(y), \sigma(x))$ and $\sigma(z) \notin \sigma(x)$, hence there exists an element $m' \in \sigma(z)$ which is not in $\sigma(y)$, so we put $V_m = \{n \in S \,|\, m' \in \sigma(n)\}$. Since σ satisfies φ it is not possible that during the next iteration of the repeat-loop we obtain $\perp$ in φ. In general we can inductively prove that going on in this way there is not an iteration of the repeat-loop in which we add $\perp$ to φ, i.e. our thesis. $\square$

$\mathsf{Test}(\varphi)$:
repeat
  $\tau := \top$;
  for all ℓ subformulae of φ do
    if there is $i \in$ Test-Rules and σ such that $\ell = l_i[\sigma]$ then
      if $r_i[\sigma]$ is not subformula of φ then
        $\varphi := \varphi \wedge r_i[\sigma]$;
        $\tau := \bot$;
until τ;
return φ.

Figure 2.7: The Procedure Test.

$$\frac{\cup(y,z,w) \quad x \in y}{x \in w}\ (1_\cup) \qquad \frac{\cup(y,z,w) \quad x \in w}{x \in y \mid x \in z}\ (2_\cup)$$

$$\frac{\cup(y,z,w) \quad x \notin y}{x \notin z \wedge x \notin w \mid x \in z \wedge x \in w}\ (3_\cup) \qquad \frac{\cup(y,z,w) \quad x \notin w}{x \notin y \wedge x \notin z}\ (4_\cup)$$

$$\frac{\backslash(y,z,w) \quad x \in y}{x \notin z \wedge x \in w \mid x \in z \wedge x \notin w}\ (1_\backslash) \qquad \frac{\backslash(y,z,w) \quad x \in z}{x \notin w}\ (2_\backslash)$$

$$\frac{\backslash(y,z,w) \quad x \in w}{x \notin z \wedge x \in y}\ (3_\backslash) \qquad \frac{\backslash(y,z,w) \quad x \notin y}{x \notin w}\ (4_\backslash)$$

$$\frac{\backslash(y,z,w) \quad x \notin w}{x \notin y \mid x \in z}\ (5_\backslash) \qquad \frac{\wp(y,x) \quad z \in x \quad w \in z}{w \in y}\ (1_\wp)$$

$$\frac{\wp(y,x) \quad z \notin y \quad w \in x}{z \notin w}\ (2_\wp) \qquad \frac{x \in y \quad x \notin y}{\bot}\ (\bot)$$

Figure 2.8: The Test-Rules for the Procedure Test.

Lemma 2.17 (Correctness) *Let φ be a conjunction of literals in $\mathcal{S}$. If there is a non-deterministic computation of* Satisfiable *which terminates without $\bot$ in φ, then there exists a model of Ω_P in which φ is satisfiable.*

Proof. We use the formula φ' which is the output of the computation on φ in order to define a model of φ.

We have to distinguish two cases:

- if during the last iteration of the repeat-loop it was $NewB = \emptyset$, then we put $F = FV(\varphi')$;

- otherwise let N be the set of new variables in the last iteration and put $F = FV(\varphi') \backslash N$.

```
Add(φ, ψ):
repeat
    τ := ⊤;
    for all ℓ subformulae of φ do
        if ∃i ∈ Add-Rule and σ such that ℓ = l_i[σ] with z ∈ FV(ψ) then
            if r_i[σ] is not subformula of φ for any value of m then
                φ := φ ∧ r_i[σ];
                τ := ⊥;
until τ.
```

Figure 2.9: The Procedure Add.

$$\frac{\wp(y,x) \quad z \notin x}{m \in z \wedge m \notin y}\ (\wp - add)$$

Figure 2.10: The Add-Rule for the Procedure Add.

Consider the model $\mathcal{M}$ of Ω_P defined as follows:

$$\begin{array}{rcl} M & = & \{m \mid m \in F\} \\ m_1 \in^{\mathcal{M}} m_2 & \leftrightarrow & \begin{cases} m_1 \in m_2 \text{ is in } \varphi' \\ m' \in m_2 \text{ is in } \varphi', m' \notin F, V_{m'} = V_{m_2} \end{cases} \\ m_1 \subseteq^{\mathcal{M}} m_2 & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_1 \rightarrow n \in^{\mathcal{M}} m_2) \\ \cup^{\mathcal{M}}(m_1, m_2, m_3) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \in^{\mathcal{M}} m_1 \vee n \in^{\mathcal{M}} m_2) \\ \setminus^{\mathcal{M}}(m_1, m_2, m_3) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \in^{\mathcal{M}} m_1 \wedge n \notin^{\mathcal{M}} m_2) \\ \wp^{\mathcal{M}}(m_1, m_2) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \subseteq^{\mathcal{M}} m_1) \end{array}$$

Clearly $\mathcal{M}$ is a model of Ω_P.

Consider the valuation σ over $\mathcal{M}$ such that $\sigma(x) = x$ for all $x \in FV(\varphi)$. We want to prove that σ satisfies φ. Notice that the second case in the definition of $\in^{\mathcal{M}}$ introduces no new elements for the variables in $FV(\varphi)$, while it introduces new elements for the new variables whenever they had an element added during the last iteration.

If $x \in y$ is a literal in φ, then we immediately have the thesis. Similarly it is immediate to see that the literals of the form $x \notin y$ and $x \neq y$ are satisfied.

If $\cup(x, y, z)$ is a literal in φ, then: if $m \in \sigma(x)$ (or $\sigma(y)$), then m is a variable in φ' such that $m \in x$ is a literal in φ', moreover m has not been added during the last iteration of the repeat loop, since we use F and not $FV(\varphi')$ in order to define M, hence Test ensures that also $m \in z$ is a literal in φ', from which we have $m \in \sigma(z)$; if $m \in \sigma(z)$, then we obtain the thesis using a similar argument.

If $\setminus(x, y, z)$ is a literal in φ, then: if $m \in \sigma(x)$ and $m \notin \sigma(y)$, then, since m is a variable in φ' which has not been added during the last iteration, Test ensures that also $m \in z$ is a literal in φ'; if $m \in \sigma(z)$ the thesis is obtained similarly.

Consider the case in which $\wp(y, x)$ is a literal in φ. If $m \in \sigma(x)$, then we have to prove that all the elements of m are elements of $\sigma(y)$. An element of m in $\mathcal{M}$ is a variable n of φ' which has not been added during the last step. Two cases are possible: it can be that the literal $n \in m$ is in φ', hence, since in φ' we have $m \in x$, $n \in m$, and $\wp(y, x)$ before the last iteration, the rules in Test ensure that in φ' we also have the literal $n \in y$, from which $n \in \sigma(y)$; or it can be the case that there was a variable n' introduced during the last iteration such that $n' \in m$ and n is a variable introduced before the last step and such

that $V_{n'} = V_n$. We have to prove that $y \in V_n$, or, equivalently, that $y \in V_{n'}$. We know that $x \in V_m$ (since $m \in x$ is a literal in φ'), moreover we know that there is a pair $\langle n_1, m_1 \rangle$ of variables introduced before the last iteration such that $\langle V_{n'}, V_m \rangle = \langle V_{n_1}, V_{m_1} \rangle$. Hence $m_1 \in x$ and $n_1 \in m_1$ are in φ' before the last iteration, from which the rules in Test ensure that there must be $n_1 \in y$ in φ', i.e. $y \in V_{n_1} = V_{n'} = V_n$, which is our thesis.
If m is such that $m \subseteq^{\mathcal{M}} y$, then we have to prove that $m \in x$ is a literal in φ'. Assume by contradiction that $m \notin x$ is a literal in φ'. Two cases are possible. In the first case a variable n has been introduced before the last iteration such that $n \in m$ and $n \notin y$ are in φ', and this is in contradiction with the fact that $m \subseteq^{\mathcal{M}} y$. In the second case a variable n' has been introduced during the last iteration such that $n' \in m$ and $n' \notin y$, in this case there exists a pair $\langle n_1, m_1 \rangle$ such that $\langle V_{n_1}, V_{m_1} \rangle = \langle V_{n'}, V_m \rangle$, hence we have $n_1 \notin^{\mathcal{M}} y$ and $n_1 \in^{\mathcal{M}} m$, which is in contradiction with the fact that $m \subseteq^{\mathcal{M}} y$. □

Example 2.18 *Consider the formula*

$$\wp(x, y) \wedge y \in x \wedge w \notin y.$$

We have that $S = \{x, y, w\}$. *We can assume that* Decide(φ, x) *generates:*

$$x \notin x \wedge x \in y \wedge x \notin w.$$

Similarly we can assume that Decide(φ, y) *and* Decide(φ, w) *generate:*

$$y \in x \wedge y \notin y \wedge y \notin w \wedge w \notin x \wedge w \notin y \wedge w \notin x.$$

At its first iteration Test *leaves the formula unalterate, since no rule is applicable. Then* Add *adds the following literals:*

$$m_y \in y \wedge m_y \notin x \wedge m_w \in w \wedge m_w \notin x.$$

We can assume that at this point Decide *generates:*

$$m_y \notin x \wedge m_y \in y \wedge m_y \notin w \wedge m_w \notin x \wedge m_w \in y \wedge m_w \in w.$$

During the second iteration Test *generates the negative literals:*

$$w \notin m_y \wedge m_y \notin m_y \wedge m_w \notin m_y \wedge \ldots.$$

Now Add *does not add new literals, and hence* Satisfiable *terminates. In particular we have obtained the model of the formula presented in Figure 2.11.*

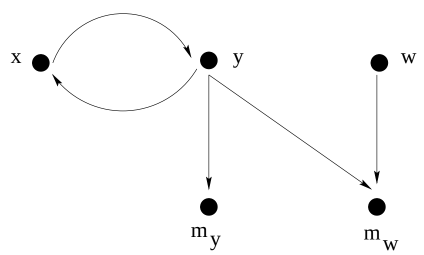


Figure 2.11: Example.

2.6 Extensionality

The Extensionality Axiom in its strongest form states that whenever two sets have the same elements they are equal.

$$\boxed{(E) \quad \forall x, y(x = y \leftrightarrow \forall z(z \in x \leftrightarrow z \in y))}$$

Consequences of the adoption of the axiom (E) are that all the elements in the model are of sort *set*, they are differentiated by their elements, it cannot be the case that there are two different empty sets, and hence, the empty set, when it exists in the model, constitutes a sort of starting point from which other sets are built by adding elements. An alternative possibility to the use of the axiom (E) is the introductions of atoms (also known as *urelements*) and of a form of extensionality which does not apply to atoms. The atoms are elements of the model which do not have any element.

$$\boxed{(at_a) \quad \forall x(\mathsf{at}(x) \leftrightarrow \forall y(y \notin x))}$$

They are used as starting points in the construction of sets: the elements of a set can be both atoms and other sets. The extensionality principle, (E_a), is applied only to sets, i.e. to the objects which have at least one element)

$$\boxed{(E_a) \quad \forall x, y(\exists z(z \in x) \rightarrow (x = y \leftrightarrow \forall z(z \in x \leftrightarrow z \in y)))}$$

On the one hand it has been proved that *every part of mathematics that can be formalized within set theory with atoms can also be formalized within a theory without atoms.* On the other hand the use of atoms solves in a more natural way many problems. In the theory ZF the axiom (E) is assumed and the existence of atoms is excluded. In [Bar75] the theory KPU with atoms is introduced and used. In the opinion of Barwise

> Set theory, as formalized in ZF, provides an extremely powerful and elegant way to organize existing mathematics. It is not without its drawbacks, nevertheless. While it is too weak to decide some questions (like the continuum hypothesis) which seem meaningful (even important), it is in some ways too strong. Some examples:
>
> 1. The most obvious advantage of the axiomatic method is lost since ZF has so few recognizable models in which to interpret its theorems.
> 2. Important distinctions on the nature of the sets asserted to exist are completely lost.
> 3. The principle of parsimony, of established value throughout the mathematical ages, is violated at every turn.
> 4. Large parts of mathematical practice are distorted by the demand that all mathematical objects have to be realized as sets.
>
> ...

In [BM96] together with the assumption of the existence of atoms and of the axiom (E_a) the axiom of Plenitude is introduced in order to ensure that *there are always fresh atoms around.* Moreover, the axiom (E_a) has been used in different consistency proofs, for instance in [Jen69] Jensen showed that the set theory NFU, the theory obtained from NF (see [Qui67]) by replacing (E) with (E_a), is consistent.

In this second part of the chapter we try to investigate what happens if we add to our theory Ω_P the axiom (E_a) or the axiom (E).

If we add the axiom (E) we are no longer able to prove the small model property: it is necessary to add new elements to the quotient in order to have a model of (E), and then new elements to these new elements, and so on, with the risk to enter in an infinite loop.

Instead, we can easily prove the small model property if we consider (E_a). When we add the axiom (E_a) we have to modify the axiom which define the powerset operator in the following way:

$$(\wp_a) \quad \forall y\, z\, (\wp(y,z) \leftrightarrow \forall x \ ((x \subseteq y \wedge \neg \mathsf{at}(x) \to x \in z) \wedge (x \in z \to x \subseteq y))$$

It can be the case that in the model there are infinitely many atoms, and we do not want that all the atoms are elements of all the powersets. What we think it is reasonable to require is that a powerset of y contains all the subsets of y which are not atoms plus an arbitrary number of atoms.

Let Ω_{Pa} be the theory obtained from Ω_P by adding (at_a) and by replacing $(\wp)$ with $(\wp_a)$, and let $\Omega_{Pa}E_a$ be the theory obtained from Ω_{Pa} by adding (E_a).

Theorem 2.19 (Small Model Property) *Let φ be a formula in $\mathcal{S}$ with $|FV(\varphi)| = n$. If φ is satisfiable in a model of $\Omega_{Pa}E_a$, then φ is satisfiable in a model of $\Omega_{Pa}E_a$ of cardinality at most $2 * [2^n + n]$.*

Proof. Let $\mathcal{N} = \langle N, \in^{\mathcal{N}}, \subseteq^{\mathcal{N}}, \cup^{\mathcal{N}}, \setminus^{\mathcal{N}}, \wp^{\mathcal{N}} \rangle$ be the model of Ω_P in which φ is satisfied by a valuation σ. For each $n \in N$ let $F(m)$ and the relation B be defined as in the proof of Theorem 2.4. Notice that $F([m])$ is correctly defined, since it does not depend on the element we choose in $[m]$. In the quotient N/B we define, for each equivalence class a privileged connected class in the following way: for all $[m] \in N/B$ $p([m])$ is a class $[m_1]$ such that there exist $m' \in [m]$ and $m'_1 \in [m_1]$ such that $m'_1 \in^{\mathcal{N}} m'$. Notice that $p([m])$ is only one class and not all the possible.

Consider the model $\mathcal{M} = \langle M, \in^{\mathcal{M}}, \subseteq^{\mathcal{M}}, \cup^{\mathcal{M}}, \setminus^{\mathcal{M}}, \wp^{\mathcal{M}} \rangle$ obtained from $\mathcal{N}$ in the following way:

$$\begin{array}{rcl} M & = & \{[m] \mid m \in N\} \uplus \{k_{[m]} \mid m \in N\} \\ [m_1] \in^{\mathcal{M}} [m_2] & \leftrightarrow & \exists m'_1 \in [m_1], m'_2 \in [m_2] (m'_1 \in^{\mathcal{N}} m'_2) \\ k_{[m]} \in^{\mathcal{M}} [n] & \leftrightarrow & [m] = [n] \vee n \in F(p([m])) \\ [m_1] \subseteq^{\mathcal{M}} [m_2] & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_1] \to [m] \in^{\mathcal{M}} [m_2]) \\ \cup^{\mathcal{M}}([m_1],[m_2],[m_3]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_3] \leftrightarrow ([m] \in^{\mathcal{M}} [m_1] \vee [m] \in^{\mathcal{M}} [m_2])) \\ \setminus^{\mathcal{M}}([m_1],[m_2],[m_3]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_3] \leftrightarrow ([m] \in^{\mathcal{M}} [m_1] \wedge [m] \notin^{\mathcal{M}} [m_2])) \\ \wp^{\mathcal{M}}([m_1],[m_2]) & \leftrightarrow & \forall [m] \in M([m] \in^{\mathcal{M}} [m_2] \leftrightarrow [m] \subseteq^{\mathcal{M}} [m_1]) \end{array}$$

where $\uplus$ is the disjoint union operator. The difference here w.r.t. the model presented in the proof of Theorem 2.4 is that now each class $[m]$ has a new element k_m. The element k_m ensures that (E_a) holds in $\mathcal{M}$. We have to put k_m also as element of some of the classes of the form $[x]$, with $x \in FV(\varphi)$, in order to satisfy the literals with $\cup, \setminus$ and $\wp$.

Clearly $\mathcal{M}$ is a finite model of Ω_{Pa}. In particular $|M| \leq 2 * (2^n + n)$.

We have to prove that $\mathcal{M}$ is a model of (E_a). We have to consider only non-empty elements of $\mathcal{M}$. If $[m] \neq [n]$ with m and n not of the form $\sigma(x)$, then $k_{[m]} \in^{\mathcal{M}} [m]$ and $k_{[m]} \notin^{\mathcal{M}} [n]$. If $[m] \neq [n]$ with only m of the form $\sigma(x)$, then $k_{[m]} \in^{\mathcal{M}} [m]$ and $k_{[m]} \notin^{\mathcal{M}} [n]$ (this is not necessarily true for $k_{[n]}$). If $[m] \neq [n]$ with $m = \sigma(x)$ and $n = \sigma(y)$, then, since

$\mathcal{N}$ satisfy (E_a), there exists s such that $s \in^{\mathcal{N}} \sigma(x)$ and $s \notin^{\mathcal{N}} \sigma(y)$ (or vice-versa), hence $[s] \in^{\mathcal{M}} [m]$ and $[s] \notin^{\mathcal{M}} [n]$.

As in the proof of Theorem 2.4 we consider the valuation θ over $\mathcal{M}$ defined as

$$\forall x \in FV(\varphi)(\theta(x) = [\sigma(x)] = \{\sigma(x)\}).$$

We want to prove that θ satisfies φ in $\mathcal{M}$.
The proof proceeds as the one of Theorem 2.4.

If in φ there is a literal of the form $x \in y$, then $\sigma(x) \in^{\mathcal{N}} \sigma(y)$, hence $\theta(x) \in^{\mathcal{M}} \theta(y)$.
Similarly we obtain the thesis for the literals of the form $x \notin y$ and $x \neq y$.
Let us assume that in φ there is a literal of the form $\cup(x, y, z)$. If $[m] \in^{\mathcal{M}} \theta(z)$, then there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(z)$, hence $m' \in^{\mathcal{N}} \sigma(x)$ or $m' \in^{\mathcal{N}} \sigma(y)$, from which $[m] \in^{\mathcal{M}} \theta(x)$ or $[m] \in^{\mathcal{M}} \theta(y)$. If $k_{[m]} \in^{\mathcal{M}} \theta(z)$, then let $[m_1] = p([m])$, it holds that $z \in F(m_1)$, hence it must also be $x \in F(m_1)$ or $y \in F(m_1)$, from which $k_[m] \in^{\mathcal{M}} \theta(x)$ or $k_[m] \in^{\mathcal{M}} \theta(y)$. We similarly obtain the other inclusion.
Assume that in φ there is a literal of the form $\backslash(x, y, z)$. If $[m] \in^{\mathcal{M}} \theta(z)$, then there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(z)$, from which $m' \in^{\mathcal{N}} \sigma(x)$ and $m' \notin^{\mathcal{N}} \sigma(y)$, using the definition of B we obtain that for all $m' \in [m]$ it holds that $m' \notin^{\mathcal{N}} \sigma(y)$, hence $[m] \in^{\mathcal{M}} \theta(x)$ and $[m'] \notin^{\mathcal{M}} \theta(y)$. We similarly obtain the thesis for all the other cases.
Assume that φ there is a literal of the form $\wp(x, y)$. If $[m] \in^{\mathcal{M}} \theta(y)$, then we have to prove that $[m] \subseteq^{\mathcal{M}} \theta(x)$, i.e. we have to prove that if $[n] \in^{\mathcal{M}} [m]$, then $[n] \in^{\mathcal{M}} \theta(x)$ and if $k_{[n]} \in^{\mathcal{M}} [m]$, then $k_{[n]} \in^{\mathcal{M}} \theta(x)$. Hence our first hypothesis is that there exists $m' \in [m]$ such that $m' \in^{\mathcal{N}} \sigma(y)$ and there exist $n'' \in [n]$ and $m'' \in [m]$ such that $n'' \in^{\mathcal{N}} m''$. If we prove that $n'' \in^{\mathcal{N}} \sigma(x)$ we have the thesis. From the fact that $m', m'' \in [m]$ and $m' \in^{\mathcal{N}} \sigma(y)$ we obtain that $m'' \in^{\mathcal{N}} \sigma(y)$, hence since σ satisfies $\wp(x, y)$, from $n'' \in^{\mathcal{N}} m''$ we obtain $n'' \in^{\mathcal{N}} \sigma(x)$. Our second hypothesis is that $k_{[n]} \in^{\mathcal{M}} [m]$. Two cases are possible: $[n] = [m]$ or $[m] = [\sigma(z)]$ and $z \in F(p([n]))$. In the first case $k_[m]$ is an element of all the classes of which $p([m])$ is an element, hence from what we have just proved, we have $k_[m] \in^{\mathcal{M}} \theta(x)$. In the second case from what we proved before it holds that $\sigma(z) \subseteq^{\mathcal{M}} \sigma(x)$, hence, since $z \in F(p([n]))$ we obtain $x \in F(p([n]))$, from which the thesis.

If $[m] \subseteq^{\mathcal{M}} \theta(x)$ and $[m]$ is not empty, then we have to prove that $[m] \in^{\mathcal{M}} \theta(y)$, i.e. we have to prove that there exists $m' \in [m]$ such that $m' \in \sigma(y)$. We prove that $m \subseteq^{\mathcal{N}} \sigma(x)$. If $n \in^{\mathcal{N}} m$, then $[n] \in^{\mathcal{M}} [m]$, hence $[n] \in^{\mathcal{M}} \theta(x)$. This means that there exists $n' \in [n]$ such that $n' \in^{\mathcal{N}} \sigma(x)$. This last implies that for all $n' \in [n]$ we have $n' \in^{\mathcal{N}} \sigma(x)$, hence $n \in^{\mathcal{N}} \sigma(x)$. So we have proved that $m \subseteq^{\mathcal{N}} \sigma(x)$, hence since σ satisfies $\wp(x, y)$ we have that $m \in^{\mathcal{N}} \sigma(y)$, from which $[m] \in^{\mathcal{M}} \theta(y)$. □

The fact that we are not able to prove the small model property for the class $\mathcal{S}$ over the theory $\Omega_P E$ does not mean that we are not able to decide the satisfiability problem. Our believe is that the small model property holds also for $\Omega_P E$, but more complicate techniques seem to be necessary to prove it. A motivation in favor of our believe is that the formula presented in [PP91], which is the simplest known formula which express the infinite in a theory without the Regularity axiom, when translated in our language involves one universal quantification.

In the next section we propose a tableau decision procedure to decide the satisfiability problem of the formulae in $\mathcal{S}$ over the theory $\Omega_P E$.

$$\frac{\wp(y,x) \quad z \notin x}{w \in z \wedge w \notin y | m \in z \wedge m \notin y} \ (\wp - add)$$

$$\frac{x \neq y}{w \in x \wedge w \notin y | w \notin x \wedge w \in y | m \in x \wedge m \notin y | m \notin x \wedge m \in y} \ (\equiv -add)$$

with $x, y, z \in FV(\psi)$, $w \in FV(\varphi)$, m new variable.

Figure 2.12: The new Add.

2.7 A Tableaux Procedure with Extensionality

We describe here how to modify the procedure presented in Section 2.5 in order to deal with the Extensionality Axiom.

The Extensionality Axiom allows us to eliminate all the literals of the form $x \neq y$ by replacing them with

$$\exists w((w \in x \wedge w \notin y) \vee (w \notin x \wedge w \in y)).$$

Hence we use $\mathcal{S}'$ to refer to the class disjunctions of conjunctions of literals of the form

$$x \in y, x \notin y, \cup(x,y,z), \setminus(x,y,z), \wp(x,y).$$

We give a non-deterministic tableau procedure in order to decide the satisfiability of formulae of this class in models of $\Omega_P E$. In order to obtain such a procedure from the procedure presented in Section 2.5 it is necessary to modify the Add rules as described in Figure 2.12. The intuition behind the first modification is that here we cannot introduce a new element m of z which is not an element of y if such a witness is already in the formula. The second modification is that we need to ensure that each variable represents a different set: if there are two distinct variables representing the same set we identify them. Moreover it is necessary to modify the termination condition defining $NewB$ as:

$$\begin{array}{rcl} NewB & := & \{\langle V_n, V_m, 1\rangle \mid n \in N, (n \in m) \in \varphi\} \cup \\ & & \{\langle V_n, V_m, 2\rangle \mid n, m \in FV(\varphi), n \not\equiv m\}. \end{array}$$

This is because now the situation from which we are able to describe an infinite computation is more complicate (see the proof of Lemma 2.21).

Let us call the new decision procedure E-Satisfiable. From the fact that Satisfiable always terminate we immediately have that E-Satisfiable always terminate.

Lemma 2.20 (Completeness) *Let φ be a conjunction of literals in $\mathcal{S}'$. If φ is satisfiable in a model of $\Omega_P E$, then there exists a non-deterministic computation of* Satisfiable*(φ) which terminates without $\perp$ in φ.*

Proof. The proof is similar to the proof of Lemma 2.16: we use the model in order to guide the choices we have to perform in the non-deterministic computation. □

Lemma 2.21 (Correctness) *Let φ be a conjunction of literals in $\mathcal{S}'$. If there exists a non-deterministic computation of φ which terminates without $\perp$ in φ, then there exists a model of $\Omega_P E$ in which φ is satisfiable.*

Proof. Let φ' be the output formula of our computation.

First we prove that if the non-deterministic computation terminates with $\varphi' = \psi$, i.e. no new literals are added during the last iteration, then we can easily build a model of $\Omega_P E$ in which φ is satisfiable. Let $F = FV(\varphi')$ and let $\mathcal{M}$ be defined as follows

$$\begin{array}{rcl} M & = & \{m \mid m \in F\} \\ m_1 \in^{\mathcal{M}} m_2 & \leftrightarrow & m_1 \in m_2 \text{ is in } \varphi' \\ m_1 \subseteq^{\mathcal{M}} m_2 & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_1 \rightarrow n \in^{\mathcal{M}} m_2) \\ \cup^{\mathcal{M}}(m_1, m_2, m_3) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \in^{\mathcal{M}} m_1 \vee n \in^{\mathcal{M}} m_2) \\ \setminus^{\mathcal{M}}(m_1, m_2, m_3) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \in^{\mathcal{M}} m_1 \wedge n \notin^{\mathcal{M}} m_2) \\ \wp^{\mathcal{M}}(m_1, m_2) & \leftrightarrow & \forall n(n \in^{\mathcal{M}} m_3 \leftrightarrow n \subseteq^{\mathcal{M}} m_1) \end{array}$$

We have to prove that $\mathcal{M}$ is a model of (E). Let us assume by contradiction that there exist m_1, m_2 in M such that $m_1 \not\equiv m_2$, and $\forall m \in M(m \in^{\mathcal{M}} m_1 \leftrightarrow m \in^{\mathcal{M}} m_2)$. This is in contradiction with the fact that the second rule in Add implies that there is at least a variable which belongs to m_1 and not to m_2 or viceversa.

Consider the valuation

$$\theta : FV(\varphi) \rightarrow M$$
$$\theta(x) = x.$$

It is immediate to prove that θ satisfies φ.

Now we prove that in case the non-deterministic computation terminates with $\varphi' \neq \psi$ it is possible to eliminate the termination condition on the variable Alt and generate an infinite computation in which we never add $\perp$ to φ.

We are in the case in which the computation terminates with $Alt = \top$ and $\perp$ is not in φ. We prove that we are able to extend the computation with a further iteration without introducing $\perp$ in φ and at the end of this further iteration it still holds $NewB \subseteq B$. From this it follows that we are able to generate an infinite computation.

Let $s_1, \ldots, s_m$ be the new variables introduced during the last step. Notice that it cannot be the case that there exists a variable w such that $w \in s_i$. Consider $s \in \{s_1, \ldots, s_m\}$ and let $m_1, \ldots, m_k$ be all the variables such that $s \in m_i$. Since $NewB \subseteq B$ it holds that there exists $\langle t_1, r_1\rangle, \ldots, \langle t_k, r_k\rangle$ which have been introduced before the last iteration and such that $V_s = V_{t_i}$ and $V_{m_i} = V_{r_i}$. Assume by contradiction that we are able to generate $s \in y \wedge s \notin y$ using the rules in Test. Notice that using the rules in Test we are able to generate three kind of literals:

- $s \in y$ with $y \in S$;
- $s \notin y$ with $y \in S$;
- $s \notin y$ with $y \notin S$.

We distinguish two cases: $y \in S$ and $y \notin S$. If $y \in S$, and $s \notin y$ have been generated by Decide, then all the literals $t_i \notin y$ have been generated by Decide. Moreover since $V_s = V_{t_i}$ we are able to generate also $t_i \in y$ by applying the same rule of Test which generate $s \in y$. This is in contradiction with the fact that $\perp$ was not in φ before the last iteration. Similarly we obtain a contradiction in the case in which $y \in S$ and $s \in y$ have been generated by Decide. If $y \notin S$, then it must be the case that $s \in y$ have been generated by Add. There exists $\langle t_i, r_i\rangle$ such that $V_{r_i} = V_y$ and $t_i \in r_i$. We have obtained $s \notin y$ by applying a rule of the form

$$\wp(x_1, x_2) \wedge s \notin x_1 \wedge y \in x_2,$$

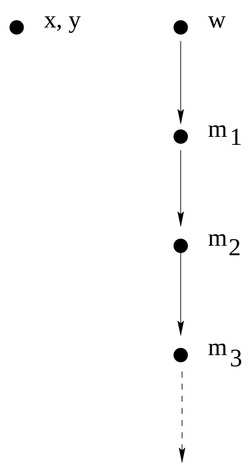


Figure 2.13: An infinite model example.

hence, from the fact that $V_{t_i} = V_s$ and $V_{r_i} = V_y$, by applying the same rule we would have generated $t_i \notin r_i$ which is a contradiction. We have obtained a contradiction in all the possible cases, hence it is not possible that we generate $s \in y$ and $s \notin y$.

Now we have to prove that we are able to apply the rules in Add and Decide obtaining that it still holds $NewB \subseteq B$.

If we have to apply $\wp(y, x) \wedge s \notin y$, then we choose one of the t_i's and we perform the same choices that we had done for $\wp(y, x) \wedge t_i \notin y$. If we have to apply $s \not\equiv y$, then let $\langle t, v\rangle$ be such that $\langle V_s, V_y, 2\rangle = \langle V_t, V_v, 2\rangle$ we perform the same choices that we had done for $t \not\equiv v$. It is immediate that $NewB \subseteq B$, and hence we are able to obtain an infinite computation. Moreover the model we can define from this infinite computation as done in the case of the finite computation is a model of $\Omega_P E$ and of φ. $\square$

Example 2.22 *Consider the formula*

$$\wp(x, y) \wedge w \notin y.$$

It is clearly satisfiable. Since it has three variables all the non-deterministic computations of Satisfiable *terminate after at most*

$$2 * (2^3 * 2^3) = 128 \textit{ iterations.}$$

Without control on the termination a possible infinite computation on this formula goes as follows:

$$\begin{array}{ll} m_1 \in w \wedge m_1 \notin x & \textit{from } w \notin y \\ m_2 \in m_1 \wedge m_2 \notin y & \textit{from } m_1 \not\equiv y \\ m_3 \in m_2 \wedge m_3 \notin x & \textit{from } m_2 \notin y \\ m_4 \in m_3 \wedge m_4 \notin y & \textit{from } m_3 \not\equiv y \\ \ldots & \end{array}$$

In the description we put only the membership relations necessary to understand why the computation does not ends, and we assume that all the missing ones are negations of memberships. The infinite model described from this computation is presented in Figure 2.13.

2.8 Concluding Remarks

The procedures we presented in this chapter allow us to decide the satisfiability of purely existentially quantified formulae over the weak theories Ω_P and $\Omega_P E$.

In particular, in the case of the theory Ω_P the absence of Extensionality, Regularity, any form of anti-foundation and the fact that $\cup, \setminus$, and $\wp$ are predicates gives us a large number of models, or in other words few tableau rules and few new variables added by the rules.

However, we showed that using a variant of the $\Box$-as-$\wp$ translation the theory Ω_P and the satisfiability problem we study turn out to be useful in the context of modal logics.

We think that what is important in the material presented in this chapter are the decidability results as well as the various general techniques introduced to construct and to quotient models of Ω_P. In particular notice that both the models used in Section 2.4 and the models used to prove the correctness of the tableau procedures are syntactical models similar to the canonical models proposed by Henkin to prove the Gödel's Completeness Theorem (see [Sho67]).

3

Extensionality, and (AFA)

In this chapter we enrich the theory used in the previous one with the anti-foundation axiom (AFA). We introduced the axiom (AFA) in Sections 1.2.3, 1.2.4 and 1.2.5 by giving some historical information, mentioning the applications and the already known results. Here we consider decidability issues again on the class of purely existentially quantified formulae. In the chapters in the second part of this thesis we address algorithmic problems related to (AFA), namely the computation of the maximum bisimulation.

We think it is important to premiss that we do not consider problems concerning the independence of the axioms we use. We present (AFA) following the same approach adopted by Aczel in his book ([Acz88]). In such an approach the power of (AFA) and the fact that that the other axioms are or not consequences of it depend on which are the instruments that we assume we can use to manipulate graphs. All the axioms in $\Omega^{-}ENS$ are consequences of (AFA) if we assume that we are able to perform a sufficient number of operations on graphs (add and remove nodes and edges following particular rules).

The axioms of our theory are sufficient to design a tableau procedure which does not use (AFA). Hence the tableau procedure we present can be used also when (AFA) is not assumed. The axiom (AFA) and also the axioms (N) and (S) are necessary in the construction of the solution that we present in order to prove the correctness of the tableau procedure.

In particular, in order to make clear which are the difficulties of the decision problem we consider, the material of this chapter is organized into two parts. In the first part we consider a language without the powerset operator, with functional symbols for the operators $\cup$ and $\setminus$, and with the Extensionality axiom (E) and the anti-foundation axiom (AFA). We define a tableau procedure to decide the validity of purely existentially quantified formulae. The insertion of the powerset operator in such a context would determine non-terminating computations due to the generation of both infinite descending chains and infinite ascending chains of membership relations. For this reason in the second part of the chapter we move to a theory with atoms (see Section 2.6). This implies that we have to use (E_a) instead of (E) and $(\mathrm{AFA})_a$ instead of (AFA), moreover we have to replace the functional symbols with predicate symbols. In this case we are able to define a tableau procedure which decides the finite satisfiability of purely existentially quantified formulae. Indeed the introduction of atoms allows us to avoid the generation of infinite descending chain of membership.

3.1 The Language and the Theory in the Basic Case

The language $\mathcal{L}$ we use in the first part of this chapter is the same as the language used in the previous chapter, but without the powerset operator and with $\cup$ and $\setminus$ that are binary function symbols instead of ternary predicates.

In order to introduce the anti-foundation axiom (AFA) (cf. [Acz88, FH83]), we give the definition of *decoration*.

Definition 3.1 (Decoration) *Given a graph $\mathcal{G} = \langle G, \gtrdot \rangle$ and an interpretation $\mathcal{M} = \langle M, (.)^{\mathcal{M}} \rangle$ of $\mathcal{L}$, a* decoration *of $\mathcal{G}$ over $\mathcal{M}$ is a function $d : G \to M$ such that:*

$$\forall n \in G\, \forall y \in M (y \in^{\mathcal{M}} d(n) \leftrightarrow \exists m \in G (y = d(m) \wedge n \gtrdot m)).$$

The anti-foundation axiom (AFA) states:

Every graph has a unique decoration.

We use the notation $\Omega^- ENS$ to refer to the first-order theory whose axiomatization is presented in Figure 3.1, and $\Omega^- ENS$(AFA) to refer to $\Omega^- ENS$ together with the anti-foundation axiom (AFA). Notice that (N) states the existence of the empty set, and (S) the definability of the *singleton* operator. The axioms (N) and (S) are not used in the decision procedure, but in the construction of a solution. We use Ω^- to refer to the axioms $(\cup), (\setminus)$, and $(\subseteq)$ because this set of axioms belongs also to the theory Ω we will use in Chapter 4. The only difference between Ω and Ω^- is the lack of the axiom $(\wp)$.

$(\cup)$	$\forall x, y, z\, (x \in y \cup z \leftrightarrow x \in y \vee x \in z)$
$(\setminus)$	$\forall x, y, z\, (x \in y \setminus z \leftrightarrow x \in y \wedge x \notin z)$
$(\subseteq)$	$\forall x, y\, (x \subseteq y \leftrightarrow \forall z\, (z \in x \to z \in y))$
(E)	$\forall x, y\, (x = y \leftrightarrow \forall z\, (z \in x \leftrightarrow z \in y))$
(N)	$\exists y\, \forall x\, (x \notin y)$
(S)	$\forall x\, \exists y\, \forall z\, (z \in y \leftrightarrow z = x)$

Figure 3.1: The axiomatization of $\Omega^- ENS$.

We study the satisfiability and the validity problem of purely existentially quantified formulae of $\mathcal{L}$ (the class $\mathcal{S}_\exists$).

3.2 The Formulae in the Basic Case

Let φ_i be a conjunction of literals of the form:

$$x \in y,\ x \notin y,\ x \cup y = z, x \setminus y = z \tag{3.1}$$

where x, y, z are variables. The formula φ is a disjunction $\varphi_1 \vee \ldots \vee \varphi_n$ of such conjunctions. The class of all the disjunctions of conjunctions of literals of the form (3.1) will be called $\mathcal{S}$. This is exactly the same class considered in Chapter 2, but w.r.t. the new language.

REDUCTION2(ψ) :
repeat
$\quad\varphi := \psi$;
$\quad$ if ℓ is a literal in ψ and $\ell \overset{\mathsf{T2}}{\mapsto} p$ then
$\quad\quad \psi := \psi$ with ℓ replaced by p
until $(\varphi = \psi)$
$\psi^S :=$ disj_form(ψ).

Figure 3.2: The Procedure REDUCTION2.

$r \subseteq s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_s = x_r \cup x_s$
$r = s \circ t$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_t = t \wedge x_r = x_s \circ x_t$ $\{r, s, t\} \not\subseteq \mathcal{V}$ and $\circ \in \{\cup, \setminus\}$
$r \neq s$	$\mapsto$	$(x_r = r \wedge x_s = s \wedge w_{rs} \in x_s \wedge w_{rs} \notin x_r) \vee$ $(x_r = r \wedge x_s = s \wedge w_{rs} \notin x_s \wedge w_{rs} \in x_r)$
$r \in s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_r \in x_s$ $\{r, s\} \not\subseteq \mathcal{V}$
$r \notin s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_r \notin x_s$ $\{r, s\} \not\subseteq \mathcal{V}$
$x = y$	$\mapsto$	$\psi[x/y]$

Figure 3.3: The rules T2 for the Procedure REDUCTION2.

It is easy to see that each purely existentially quantified formula $\psi \in \mathcal{S}_\exists$ is equivalent to a formula ψ^S in the class $\mathcal{S}$. The procedure REDUCTION2 (the procedure REDUCTION1 has been presented in Chapter 2) presented in Figure 3.2 allows, given a formula $\psi \in \mathcal{S}_\exists$, to obtain such an equivalent formula ψ^S.

Notice that now the empty set $\emptyset$, whose existence is stated from (N), can be expressed using a formula of $\mathcal{S}$:

$$x = x \setminus x.$$

It is easy to prove that:

$$\Omega^- ENS \vdash \forall x\,(x = x \setminus x \leftrightarrow \forall y\,(y \notin x)).$$

This was not true in the previous chapter because we had not (E) and (N). Moreover, now we can correctly define

$$y \cap z = w \leftrightarrow w \subseteq y \wedge w \subseteq z \wedge y \setminus z = w_1 \wedge y \setminus w = w_1).$$

Example 3.2 *Consider the formula*

$$x = x \cup y \wedge x \neq y.$$

It is satisfied in a model of $\Omega^- ENS(AFA)$ if and only if

y is a proper subset of x.

The procedure REDUCTION2 *maps it into*

$$(x = x \cup y \wedge w_{xy} \in y \wedge w_{xy} \notin x) \vee (x = x \cup y \wedge w_{xy} \notin y \wedge w_{xy} \in x).$$

3.3 The Decision Procedure for the Basic Case

In Figure 3.4 we describe a procedure $\mathsf{de-Boole}$ which transforms a given formula φ of $\mathcal{S}$ in order to decide its satisfiability. The procedure makes use of the two functions,

$\mathsf{de-Boole}(\varphi)$:
$\varphi := \in\!-\mathsf{introduction}(\mathsf{Extensionalize}(\in\!-\mathsf{introduction}(\varphi)))$.

Figure 3.4: The procedure $\mathsf{de-Boole}$.

$\in\!-\mathsf{introduction}, \mathsf{Extensionalize} : \mathcal{S} \longrightarrow \mathcal{S}$, described in Figure 3.5 and in Figure 3.7. The function $\in\!-\mathsf{introduction}$ maps a formula φ into a new formula $\in\!-\mathsf{introduction}(\varphi)$ which is equivalent to φ with respect to the theory $\Omega^{-}ENS$. The rules applied to compute $\in\!-\mathsf{introduction}(\varphi)$ (rules of F) are presented in Figure 3.6. The aim of this function is to spell out the membership relation, according to the axioms which define the operators. Similarly, the function $\mathsf{Extensionalize}$ maps a formula ψ into an equivalent one, with respect to the theory $\Omega^{-}ENS$, making use of the rule S in Figure 3.8. The rule applied by the function $\mathsf{Extensionalize}$ to ψ is based on the Extensionality axiom (E). This rule is applied only when it is necessary to check the satisfiability: it introduces a new variable and hence, it is a non-analytical rule which, if not controlled, can lead to non-termination. We apply the rule in such a way that the number of new variables introduced is bounded by the number of variables of the formula ψ, see (Lemma 3.3). The function $\mathsf{disj\_form}$ maps a formula to its disjunctive normal form.

```
∈−introduction(φ1 ∨ ... ∨ φn) :
for j = 1 to n do
    ψ := φj;
    repeat
        τ := ⊤
        for all ℓ subformulae of ψ do
            if there is i ∈ F and σ such that ℓ = li[σ] then
                if ri[σ] is not subformula of ψ then
                    ψ := ψ ∧ ri[σ];
                    τ := ⊥;
    until τ;
φ := disj_form(φ1 ∨ ... ∨ φn);
return φ.
```

Figure 3.5: The Function $\in\!-\mathsf{introduction}$.

Lemma 3.3 (Termination) *The procedure* $\mathsf{de-Boole}(\varphi)$ *terminates in at at most* $2n^6 - 4n^5 + 6n^4 - 3n^3 + 3n^2$ *steps and the number of variables in the output formula is at most* $n^3 - n^2 + n$, *where* $n = |FV(\varphi)|$.

Proof. The function $\in\!-\mathsf{introduction}(\varphi)$ does not introduce new variables: it adds to φ literals of the form $x \in y$ or $x \notin y$, where $x, y \in FV(\varphi)$, so it can add at most $2n^2$ literals, if $n = |FV(\varphi)|$. The function $\mathsf{Extensionalize}(\varphi)$ introduces new variables, but it makes only use of the literals of φ, hence it must be $x, y, z \in FV(\varphi)$ and $x \not\equiv z$. So the maximum number of

$$\frac{y \cup z = w \quad x \in y}{x \in w}\ (1_\cup) \qquad \frac{y \cup z = w \quad x \in w}{x \in y \vee x \in z}\ (2_\cup)$$

$$\frac{y \cup z = w \quad x \notin y}{x \notin z \wedge x \notin w \vee x \in z \wedge x \in w}\ (3_\cup) \qquad \frac{y \cup z = w \quad x \notin z}{x \notin y \wedge x \notin z}\ (4_\cup)$$

$$\frac{y \setminus z = w \quad x \in y}{x \notin z \wedge x \in w \vee x \in z \wedge x \notin w}\ (1_\setminus) \qquad \frac{y \setminus z = w \quad x \in z}{x \notin w}\ (2_\setminus)$$

$$\frac{y \setminus z = w \quad x \in w}{x \in y \wedge x \notin z}\ (3_\setminus) \qquad \frac{y \setminus z = w \quad x \notin y}{x \notin w}\ (4_\setminus)$$

$$\frac{y \setminus z = w \quad x \notin w}{x \notin y \vee x \in z}\ (5_\setminus) \qquad \frac{x \in y \wedge x \notin y}{\perp}\ (\perp)$$

The rules $(1_\cup)$ and $(3_\cup)$ are applied to both y and z.

Figure 3.6: The rules F for the function $\in$−introduction.

Extensionalize($\varphi_1 \vee \ldots \vee \varphi_n$) :

for $j = 1$ to n do
    $\psi := \varphi_j$;
    for all ℓ subformulae of φ_j do
        if there is $i \in \mathsf{S}$ and σ such that $\ell = l_i[\sigma]$ then
            if $r_i[\sigma]$ is not subformula of ψ then
                $\psi := \psi \wedge r_i[\sigma]$;
    $\varphi_j := \psi$
$\varphi :=$ disj_form($\varphi_1 \vee \ldots \vee \varphi_n$);
return φ.

Figure 3.7: The Function Extensionalize.

steps and variables introduced by Extensionalize(φ) are $n^2(n-1)$, if $n = |FV(\varphi)|$. Applying these remarks to $\in$−*introduction*(*Extensionalize*($\in$−*introduction*(φ))) we obtain that the number of steps is at most $2n^2 + n^2(n-1) + 2(n^2(n-1)+n)^2$, while the number of variables is at most $n^2(n-1)+n$. □

Lemma 3.4 (Correctness and Completeness) *Let $\psi = \psi_1 \vee \ldots \vee \psi_m$ be the output formula of* de − Boole(φ). *There exists a model of $\Omega^- ENS(AFA)$ which satisfies ψ if and only if there exists $j \in \{1, \ldots, m\}$ such that in ψ_j there are no literals of the form $\perp$.*

Proof. If there exists a model of $\Omega^- ENS$(AFA) and ψ, then it cannot be the case that for all $j \in \{1, \ldots, m\}$ in ψ_j there is a literal of the form $\perp$, since this kind of literal is only added when a situation of the form $x_j \in y_j \wedge x_j \notin y_j$ is detected.

Conversely we can assume that $\psi = \psi_1$ and there are no literals of the form $\perp$ in ψ. We prove that ψ is satisfiable in every model of $\Omega^- ENS$(AFA).

Let us consider for each variable x in ψ which appears at least in a literal of the form $y_i \in x$, the equation:

$$x = \{y_1, \ldots, y_n\}$$

$$\frac{x \in y \quad z \notin y}{n \in x \wedge n \notin z \vee n \notin x \wedge n \in z}\ (E)$$

Figure 3.8: The rule S for the function Extensionalize.

where $y_1, \ldots, y_n$ are all the variables that appear in literals of the form $y_i \in x$ in ψ.

Let $\tau = max\{n \mid y_1 \in x, \ldots, y_n \in x \text{ occur in } \psi\}$.

For each variable x in ψ which does not appear in any literal of the form $y \in x$ consider the equation:

$$x = \{\emptyset, \{\emptyset\}, \ldots \{\emptyset\}^{r_x}\}$$

where $r_x > \tau$, and if $x \not\equiv y$, then $r_x \neq r_y$.

(AFA) ensures that the system which contains all the equations considered has a unique solution $\sigma = [x_1/\bar{x}_1, \ldots, x_n/\bar{x}_n]$. We prove that the solution σ satisfies ψ.

First notice that the n's introduced by the function Extensionalize can only appear on the l.h.s. of $\in$-literals in ψ; hence, all their defining equations are of the form $n = \{\emptyset, \{\emptyset\}, \ldots \{\emptyset\}^{r_n}\}$. This allows us to say that:

$$\begin{array}{l}\text{if } n \text{ has been introduced by Extensionalize, then} \\ \text{the solution } \sigma \text{ of the system satisfies } \forall x \in FV(\psi)(\bar{x} \neq \bar{n}).\end{array} \tag{3.2}$$

In order to prove it let us consider that:

- if also x has been introduced by Extensionalize, then $r_n \neq r_x$ ensures that $\bar{x} \neq \bar{n}$, since they have a different number of elements;
- else $\bar{x}$ has at most τ elements while $\bar{n}$ has at least $\tau + 1$ elements.

Moreover, we can prove that:

$$\begin{array}{l}\text{if } t \in \bar{x} \text{ and } |t| > \tau, \text{ then} \\ \text{there exists only one } y \in FV(\psi) \text{ such that } t = \bar{y}.\end{array} \tag{3.3}$$

This is an immediate consequence of the fact that the solution σ is injective on sets with more than τ elements.

If $x \in y$ is a literal in ψ it is satisfied by σ, since in the system there is the equation $x = \{\ldots, y, \ldots\}$.

If $x \notin y$ is a literal in ψ and x has not been introduced by the function Extensionalize, then for each literal $z \in y$ in ψ either the literals $n \in x \wedge n \notin z$, or the literals $n \notin x \wedge n \in z$, are in ψ. In the first case $\bar{n}$ is an element of $\bar{x}$ different from all the elements of $\bar{z}$, from (3.2), and hence $\bar{x} \neq \bar{z}$. In the second case, $\bar{n}$ is an element of $\bar{z}$ different from all the elements of $\bar{x}$, from (3.2), and hence $\bar{x} \neq \bar{z}$. We have obtained that for all $\bar{z} \in \bar{y}$ it holds $\bar{x} \neq \bar{z}$, i.e. $\bar{x} \notin \bar{y}$.

If $x \notin y$ is a literal in ψ and x has been introduced by the function Extensionalize, then, from (3.2) we have that $\bar{x} \neq \bar{z}$, for all $\bar{z} \in \bar{y}$, hence the literal is satisfied.

If $x \cup y = z$ is a literal in ψ, then:

- if $t \in \bar{x}$ (or $t \in \bar{y}$), then $t = \bar{w}$ and $w \in x$ (or $w \in y$) is a literal in ψ, in this case the rule $(1_\cup)$ of F (see Figure 3.6) ensures that $w \in z$ is a literal in ψ, hence $t \in \bar{z}$;
- if $t \in \bar{z}$, then $t = \bar{w}$ and $w \in z$ is a literal in ψ, hence the rule $(2_\cup)$ of F ensures that either $w \in x$ or $w \in y$ are literals in ψ, i.e. either $t \in \bar{x}$ or $t \in \bar{y}$.

If $x \setminus y = z$ is a literal in ψ, then:

- if $t \in \bar{x}$ and $t \notin \bar{y}$, then there exists a variable w such that $t = \bar{w}$, $w \in x$ is a literal in ψ, and $w \in y$ is not a literal in ψ, in this case the rule $(1_\setminus)$ of F ensures that $w \in z$ is a literal in ψ, hence $t \in \bar{z}$;
- if $t \in \bar{z}$, then it must be $t = \bar{w}$ with $w \in z$ literal in ψ, in this case the rule $(2_\setminus)$ of F ensures that also $w \in x$ and $w \notin y$ are literals in ψ, hence, since we have already proved that all the literals of the form $w \notin y$ are satisfied, $t \in \bar{x}$ and $t \notin \bar{y}$.

□

Theorem 3.5 *Let $\psi = \psi_1 \vee \ldots \vee \psi_m$ be the output formula of* $\mathsf{de} - \mathsf{Boole}(\varphi)$. *It holds that $\Omega^- ENS(AFA) \Vdash \varphi$ if and only if there exists $j \in \{1, \ldots, m\}$ such that in ψ_j there are no literals of the form $\bot$.*

Proof. It follows immediately from Lemma 3.4. □

So the procedure $\mathsf{de} - \mathsf{Boole}$, together with the procedure REDUCTION2, can be used to decide the class of purely existentially quantified formulae of $\mathcal{L}$: given a purely existentially quantified formula ψ of $\mathcal{L}$, in order to decide its satisfiability first we convert it into a formula ψ^S in the class $\mathcal{S}$ using the procedure REDUCTION2, then we apply $\mathsf{de} - \mathsf{Boole}$ to ψ^S, and we check for a disjunct without occurrences of $\bot$.

Example 3.6 *Consider the formula*

$$x = x \cup y \wedge x \neq y$$

in Example 3.2. The procedure REDUCTION2 *maps it into*

$$(x = x \cup y \wedge w_{xy} \in y \wedge w_{xy} \notin x) \vee (x = x \cup y \wedge w_{xy} \notin y \wedge w_{xy} \in x).$$

The procedure $\mathsf{de} - \mathsf{Boole}$ *applied to the first disjunct returns $\bot$ after two steps. It cannot be the case that there is an element of y which is not an element of x.*

The procedure $\mathsf{de} - \mathsf{Boole}$ *applied to the second disjunct returns*

$$x = x \cup y \wedge w_{xy} \notin y \wedge w_{xy} \in x,$$

which is clearly satisfiable.

We conclude this section by observing that a rather simple way to improve the procedure presented above consists in using a combinatorial result from [PPR93] in order to reduce the number of variables introduced by Extensionalize.

3.4 The Language and the Theory in the Case with Powerset

If we add the powerset operator to the language used in the first part of this chapter we have to introduce rules which could lead to the generation of both infinite ascending and infinite descending chains of memberships. Consider, for instance, the formula

$$x = \wp(x) \wedge y \in x.$$

It is necessary to put in x also the singleton of y, since this is a subset of x, and hence, the singleton of the singleton of y, and so on, i.e. we have the infinite ascending chain:

$$n_1 = \{y\} \wedge n_1 \in x,\ n_2 = \{n_1\} \wedge n_2 \in x,\ \ldots$$

While, if we consider the formula

$$y = \wp(x) \wedge y \in x \wedge w \in y$$

the Extensionality could generate the following infinite descending chain

$$n_1 \in w \wedge n_1 \notin x,\ n_2 \in n_1 \wedge n_2 \notin y,\ n_3 \in n_2 \wedge n_3 \notin x,\ \ldots$$

in which: n_1 differentiates w and x, since w is not a subset of x; n_2 differentiates n_1 and y, since n_1 is not an element of x, while y is an element of x, hence they have to be different; n_3 differentiates n_2 and x, since n_2 is not a subset of x; and so on. The introduction of atoms and the consequent variation of the language (from functional to predicative) allow us to *push forward* the problem of the infinite chains: we are able to define a procedure which works correctly both in the case with and in the case without atoms, we are able to exhibit a model with atoms for a closed tableaux. It could be possible with more sophisticated techniques or with some restrictions on the formulae to exhibit also a model without atoms: a way to do this could be to replace the atoms we use in our model with *pure* sets. The idea behind our procedure is to use the atoms to avoid the generations of infinite descending chains: two atoms are always different, and it is not necessary to introduce new elements to witness this fact. It turns out that the atoms avoid also the generation of some ascending chains.

The language we use in the second part of this chapter is the first-order language

$$\mathcal{L} = \langle \mathcal{F}, \mathcal{C}, \mathcal{P}, \mathcal{V} \rangle,$$

with

$$\mathcal{F} = \emptyset, \quad \mathcal{C} = \{a_i \,|\, i \in \mathbb{N}\}, \quad \mathcal{P} = \{\cup, \setminus, \wp, \in, \subseteq, =\},$$

and $\mathcal{V}$ a denumerable set of variables.

Hence, here $\cup, \setminus, \wp$, as it was in Chapter 2, are predicate symbols. Moreover we have a denumerable set of constant symbols: the *atoms*. It is just the presence of atoms which *requires* to move from functional to predicate symbols. Which is the result of the union of two atoms? How many and which atoms we put in the powersets? The choice to move to predicates allows us to do not impose restrictions.

We have already introduced (at_a) and (E_a) in Section 2.6 together with some historical notes. An axiomatization for the first-order theory $\Omega_{Pa}E_aN_aS$ (we intend $D_{(a,b)}$ as a part of E_a, hence we do not explicitly mention it in the name of the theory) is presented in Figure 3.9.

The sub-theory $\Omega_{Pa}E_a$ of $\Omega_{Pa}E_aN_aS$ is exactly the theory we met at the end of Chapter 2: here we use an equivalent version of (E_a).

The main differences between the theories $\Omega_{Pa}E_aN_aS$ and $\Omega^- ENS$ are the following:

- the axioms (N_a) and $(D_{(a,b)})$ ensure the existence of infinitely many atoms, i.e. sets without elements;
- the Extensionality criterion for equality (clearly) does not apply to atoms.

$(\cup)$	$\forall y,\, z,\, w\, (\cup(y,z,w) \leftrightarrow \forall x\, (x \in w \leftrightarrow x \in y \vee x \in z))$
$(\setminus)$	$\forall y,\, z,\, w\, (\setminus(y,z,w) \leftrightarrow \forall x\, (x \in w \leftrightarrow x \in y \wedge x \notin z))$
$(\subseteq)$	$\forall x,\, y\, (x \subseteq y \leftrightarrow \forall z\, (z \in x \rightarrow z \in y))$
(at_a)	$\forall x (\mathsf{at}(x) \leftrightarrow \forall y (y \notin x))$
$(\wp_a)$	$\forall y\, z\, (\wp(y,z) \leftrightarrow \forall x\ \ ((x \subseteq y \wedge \neg \mathsf{at}(x) \rightarrow x \in z)) \wedge$ $(x \in z \rightarrow x \subseteq y))$
(E_a)	$\forall x,\, y\, (\neg \mathsf{at}(x) \rightarrow (x = y \leftrightarrow \forall z\, (z \in x \leftrightarrow z \in y)))$
(N_a)	$\mathsf{at}(a)$ for all $a \in \mathcal{C}$
$(D_{(a,b)})$	$a \neq b$ for all $a,\, b \in \mathcal{C}$ with $a \not\equiv b$
(S)	$\forall x\, \exists y\, \forall z\, (z \in y \leftrightarrow z = x)$

Figure 3.9: The axiomatization of $\Omega_{Pa} E_a N_a S$.

As a consequence of these two facts, as it is explained in the following remark, the operators $\cup, \setminus$,and $\wp$ are not functional any more.

Remark 3.7 *The predicate symbol $\cup$ is not functional in the case where its arguments are atoms, as follows from:*

$$\Omega_{Pa} E_a N_a S \vdash \forall x, y, z\, (\mathsf{at}(x) \wedge \mathsf{at}(y) \wedge \mathsf{at}(z)) \rightarrow \cup(x,y,z)).$$

In particular, this is the only case in which it is not functional, since:

$$\Omega_{Pa} E_a N_a S \vdash \forall x, y, z, w\, ((\cup(x,y,z) \wedge \cup(x,y,w) \wedge z \neq w) \rightarrow (\mathsf{at}(x) \wedge \mathsf{at}(y))).$$

The predicate symbol $\setminus$ is not functional if its first argument is a subset of its second argument. The predicate symbol $\wp$ always admits infinitely many solutions, as follows from:

$$\Omega_{Pa} E_a N_a S \vdash \forall x, y, z((\wp(x,y) \wedge y \subseteq z \wedge \forall w\, (w \in z \wedge w \notin y \rightarrow \mathsf{at}(w))) \rightarrow \wp(x,z)).$$

Moreover, two different powersets of the same set can only differ for some atoms:

$$\Omega_{Pa} E_a N_a S \vdash \forall x, y, z\, ((\wp(x,y) \wedge \wp(x,z)) \rightarrow \forall w\, (w \in z \wedge w \notin y \rightarrow \mathsf{at}(w))).$$

Definition 3.8 (System) *A* system *S of $\mathcal{L}$ is a finite set of equations of the form*

$$x = \{y_1, \ldots, y_n\} \qquad or \qquad x = a$$

where $x, y_1, \ldots, y_n \in \mathcal{V}$, and $a \in \mathcal{C}$, such that for each variable x in S, there is exactly one equation in S in which x occurs as the l.h.s..

In our construction we will also make use of the following adaptation (AFA_a) of the anti-foundation axiom:

Every system of $\mathcal{L}$ has a unique solution.

We use $\Omega_{Pa} E_a N_a S(\mathrm{AFA}_a)$ to refer to the theory $\Omega_{Pa} E_a N_a S$ together with the axiom (AFA_a). (AFA_a) is not a generalization of (AFA): each system corresponds to a finite graph. This means that $(\mathrm{AFA})_a$ has not the same power as (AFA) on infinite graphs.

We consider the finite satisfiability problem for the class of purely existentially quantified formulae of the language $\mathcal{L}$.

3.5 A Model of $\Omega_{Pa}E_aN_aS(\mathbf{AFA}_a)$

A rather natural question at this point is whether or not $\Omega_{Pa}E_aN_aS(\text{AFA}_a)$ has a model: the following construction provides one model. Intuitively the model we propose is obtained from a model of (AFA): we remove from it the well-founded elements of rank less than α and we use the well-founded sets of rank α to interpret the atoms. It is not necessary to use permutations of the universe (see [For95]), since we start from a model with non-well-founded sets.

Let α be an infinite ordinal, let $V_{<\alpha}$ be the set of all well-founded sets of rank less than α, let $V_{=\alpha}$ the set of all well-founded sets of rank α, and let $\mathcal{U}'_\alpha = \mathcal{U} \setminus V_{<\alpha}$ be the universe of all non-well-founded sets (see [Acz88]) not belonging to $V_{<\alpha}$. We use $\mathcal{W}_\alpha$ to denote the quotient $\mathcal{U}'_\alpha / \cong$, where:

$$\forall x, y \in \mathcal{U}'_\alpha (x \cong y \text{ iff } x \notin V_{=\alpha} \wedge y \notin V_{=\alpha} \wedge \forall z \in \mathcal{U}'_\alpha (z \in x \leftrightarrow z \in y))$$

We consider the structure $\mathcal{M}_\alpha$ for the language $\mathcal{L}$ with support $\mathcal{W}_\alpha$ and interpretation function $()'$ defined as follows:

$$\begin{array}{lll}
x \in' y & \text{iff} & x \in y \\
\cup'(x,y,z) & \text{iff} & \begin{cases} z = x \cup y & \text{if } \exists w\,(w \in' x \vee w \in' y) \\ z \in V_{=\alpha} & \text{otherwise} \end{cases} \\
\setminus'(x,y,z) & \text{iff} & \begin{cases} z = x \setminus y & \text{if } \exists w\,(w \in' x \wedge w \notin' y) \\ z \in V_{=\alpha} & \text{otherwise} \end{cases} \\
x \subseteq' y & \text{iff} & \begin{cases} x \subseteq y & \text{if } \exists w\,(w \in' x) \\ x \in V_{=\alpha} & \text{otherwise} \end{cases} \\
\wp'(x,y) & \text{iff} & \forall w\,(w \in' y \rightarrow w \in \wp(x) \vee w \in V_{=\alpha}) \wedge \\
 & & \forall w\,((w \in \wp(x) \wedge w \notin V_{=\alpha}) \rightarrow w \in' y) \\
a'_i = w_i & & \text{for all } a_i \in \mathcal{C}, \text{ with } w_i \in V_{=\alpha}, w_i \not\equiv w_j
\end{array}$$

Theorem 3.9 (Consistency) *Let α be an infinite ordinal.*

$$\mathcal{M}_\alpha \textit{ is a model of } \Omega_{Pa}E_aN_aS(AFA_a).$$

Proof. The axioms (N_a) are satisfied, since in $\mathcal{M}_\alpha$ the well-founded sets of rank α have no elements. Also the axioms $(D_{(a,b)})$ are satisfied, since $|V_{=\alpha}| \geq |\mathcal{C}|$ and the elements of $V_{=\alpha}$ have not been collapsed in $\mathcal{W}_\alpha$. The axiom (S) is satisfied, since if x is an element of $\mathcal{U} \setminus V_{<\alpha}$, then $\{x\}$ is also an element of $\mathcal{U} \setminus V_{<\alpha}$. It is immediate to see that if $\neg\mathsf{at}(x)$, then it cannot be $x \in V_{=\alpha}$, and hence there exists w in $\mathcal{U} \setminus V_{<\alpha}$ such that $w \in' x$. If $\forall z \in \mathcal{W}_\alpha (z \in' x \leftrightarrow z \in' y)$, then the definition of $\mathcal{W}_\alpha$ ensures that $x = y$, when x has at least one element, hence (E) is satisfied. The axiom $(\cup)$ is satisfied, since if x and y have no elements, then the relation $\cup'(x,y,z)$ holds if and only if z has no elements, otherwise $\cup'(x,y,z)$ holds if and only if z has exactly all the elements of x and y. The fact that the axioms $(\setminus), (\subseteq), (\wp)$ are satisfied follows from similar considerations. Now we prove that $\mathcal{M}_\alpha$ satisfies $(\text{AFA})_a$. Let S be a system in the variables $x_1, \ldots, x_n$, consider the system S' in which each equation of the form $x_i = a_i$, with $a_i \in \mathcal{C}$ has been replaced by $x_i = w_i$, where w_i is the well-founded set of rank α such that $a'_i = w_i$. (AFA) ensures that S' has a unique solution $\sigma = [x_1/\bar{x}_1, \ldots, x_n/\bar{x}_n]$ in $\mathcal{U}$. If $\bar{x}_j$ is a well-founded set in the solution of S', then $\bar{x} \notin V_{<\alpha}$, as can be proved by induction on the number of variables in S'. Hence the solution σ is in $\mathcal{U}'_\alpha$, moreover this is a solution of S in $\mathcal{M}_\alpha$. (AFA) guarantees also that S' has only one solution, and this implies that S cannot have another solution in $\mathcal{M}_\alpha$. $\square$

3.6 The Formulae in the Case with Powerset

Let φ be a conjunction of literals of the form:

$$x \in y,\ x \notin y,\ \cup(x, y, z),\ \backslash(x, y, z),\ \wp(x, y) \tag{3.4}$$

where x, y, z are variables. The class $\mathcal{S}'$ is the class of all conjunctions of literals of the form (3.4).

Each purely existentially quantified formula ψ of $\mathcal{L}$ is equivalent to a disjunction of formulae $\psi_1^{S'} \vee \ldots \vee \psi_n^{S'}$ in the class $\mathcal{S}'$. Given a formula ψ, let $\psi_1 \vee \ldots \vee \psi_n$ be its disjunctive normal form. The procedure REDUCTION3 presented in Figure 3.10 allows, given a formula ψ_j which is a conjunction of literals of $\mathcal{L}$, to obtain a formula $\psi_j^{S'}$ in $\mathcal{S}'$.

```
REDUCTION3(ψ) :
S := FV(ψ);
repeat
      φ := ψ;
      if (a ∈ C and a in ψ then
            let x ∈ V \ FV(ψ) in ψ := ψ[a/x];
until (φ = ψ);
for each {x, y} ⊆ FV(ψ) \ S do
      ψ := ψ ∧ x ≠ y;
repeat
      φ := ψ;
      if ℓ is a literal in ψ and ℓ ↦(T3) p then
            ψ := ψ with ℓ replaced by p
until (φ = ψ)
repeat
      φ := ψ;
      if x = y is a literal in ψ then
            ψ := (ψ \ (x = y))[x/y]
until (φ = ψ).
```

Figure 3.10: The Procedure REDUCTION3.

$x \subseteq y$	$\mapsto$	$\cup(x, y, y)$	
$x \neq y$	$\mapsto$	$x \in z \wedge y \notin z$	with $z \in \mathcal{V} \setminus FV(\psi)$

Figure 3.11: The rules T3.

Corollary 3.10 *If ψ is a conjunction of literals of $\mathcal{L}$ and $\psi^{S'}$ is the output formula of the procedure* REDUCTION3*, then $\psi^{S'}$ is in $\mathcal{S}'$ and ψ is satisfiable in a model of $\Omega_P EN_a S(AFA_a)$ if and only if $\psi^{S'}$ is satisfiable in a model of $\Omega_P EN_a S(AFA_a)$.*

Proof. Clearly the formula $\psi^{S'}$ is in the class $\mathcal{S}'$. Every model of ψ is also a model of $\psi^{S'}$. If $\mathcal{M}$ is a model of $\Omega_P EN_a S(\mathrm{AFA}_a)$ and σ a valuation which satisfies $\psi^{S'}$, then in order to obtain a valuation which satisfies ψ it is sufficient to consider an appropriate permutation of the constant symbols $a_i \in \mathcal{C}$ which occur in σ. $\square$

3.7 A Decision Procedure for the Case with Powerset

In the following we present a decision procedure for the satisfiability problem for the class of purely existentially quantified formulae of the language $\mathcal{L}$, with respect to the theory $\Omega_{Pa}E_aN_aS(\mathrm{AFA}_a)$. In particular, we show how to deal with the restricted class $\mathcal{S}'$, since we already know that the whole class of purely existentially quantified formulae can be reduced to disjunctions of formulae in the class $\mathcal{S}'$. While in Section 3.3 we have presented a deterministic procedure, here, for the sake of simplicity, we define a non-deterministic procedure: instead of considering all the possible disjuncts, each time we make a choice and we add only one of them.

The procedure $\mathsf{de}-\wp\mathsf{Boole}$ (Figure 3.12), allows to decide the finite satisfiability of a formula φ of the class $\mathcal{S}'$. Two procedures, $\in\!-\wp\mathsf{introduction}$ and $\wp\mathsf{Extensionalize}$, are used

$\mathsf{de}-\wp\mathsf{Boole}(\varphi)$:
$S := FV(\varphi)$;
$\in\!-\wp\mathsf{introduction}(\varphi)$;
$\wp\mathsf{Extensionalize}(\varphi, S)$;
$\in\!-\wp\mathsf{introduction}(\varphi)$.

Figure 3.12: The procedure $\mathsf{de}-\wp\mathsf{Boole}$.

in $\mathsf{de}-\wp\mathsf{Boole}$; these procedures take a formula as global variable, hence, for instance, the formula φ in the third line is not necessarily the same as the one which occurs in the second line.

The main differences between $\in\!-\mathsf{introduction}$ and $\in\!-\wp\mathsf{introduction}$ are the eight new rules in $\in\!-\wp\mathsf{introduction}$ to deal with the $\wp$ relation symbol and the fact that the rules of $\in\!-\wp\mathsf{introduction}$ have been split into 1P and 2P in order to control termination.

The procedure $\in\!-\wp\mathsf{introduction}$ calls, inside a repeat-loop, two auxiliary procedures, $\in\!-\mathsf{Boolean}$ and $\in\!-\mathsf{Powerset}$, which transform the formula φ into a formula φ' equivalent to φ with respect to the theory Ω_PEN_aS. At the end of each iteration of this loop three sets of variables are taken into consideration to enforce termination. The purpose of the procedures $\in\!-\mathsf{Boolean}$ and $\in\!-\mathsf{Powerset}$ is to apply the rules presented in Figures 3.15 and 3.17. The rules applied by the procedure $\in\!-\mathsf{Boolean}$ are almost the same as the ones applied by the function $\in\!-\mathsf{introduction}$ presented in Section 3.3. Two new rules have been introduced to deal with literals of the form $y = \{x_1, \ldots, x_h\}$ which, even though not literals of the language, are useful to explain the procedures. Moreover here the rules $(1_\cup)$ and $(3_\cup)$ of 1P (see Figure 3.15) need to be applied to both y and z. The rules 2P applied by $\in\!-\mathsf{Powerset}$ deal with literals of the form $\wp(x, y)$. These rules cannot be applied without control, since they can lead to non-termination. The sets $N_n[r]$, $N_h[r]$, and $Inc[r]$ in the procedure $\in\!-\wp\mathsf{introduction}$ are used to check that there are no loops in the rules applied by $\in\!-\mathsf{Powerset}$: the set $N_n[r]$ collects the sets of variables into which a new variable n (a variable introduced by the second rule of 2P) has been added; the set $N_h[r]$ has the same purpose as $N_r[n]$ w.r.t. the variables h introduced by the rule $(2_{\{\}})$ of 2P; the set $Inc[r]$ collects the pairs (x, y) such that, at step r, x is still a subset of y. As follows from Lemma 3.12 the condition we impose over these sets ensures termination, while Lemma 3.14 proves that $\exists s < r\, Pict[s] = Pict[r]$ leads only to infinite solutions. The rule $(2_\wp)$ of 2P, see Figure 3.17, must be applied once for each subset of literals $z_i \in y$ of the formula. For each new variable ℓ introduced by the rules $(2_\wp)$ and $(2_{\{\}})$ of 2P, the last two rules of 2P decide to which variables of φ the variable ℓ belongs.

$\in - \wp$introduction(φ) :
$FV := FV(\varphi)$;
$B := \perp$;
$r := 0$;
repeat
  $\psi := \varphi$;
  $r := r + 1$;
  $\in -$Boolean(φ);
  $\in -$Powerset(φ);
  $New := \{(x, y) \mid x, y \in S$ and $(x \circ y)$ in $\varphi \setminus \psi$ with $\circ = \in, \notin\}$;
  $New := New \cup \{k_{xy} \mid k_{xy} \in FV(\varphi) \setminus FV(\psi)\}$;
  if $New \neq \emptyset$ then $r := 0$;
  $N_n[r] := \{V_n \mid n \in FV(\varphi) \setminus FV(\psi)\}$;
  $N_h[r] := \{V_{h_{xn}} \mid h_{xn} \in FV(\varphi) \setminus FV(\psi)\}$;
  $Inc[r] := \{(x, y) \mid x, y \in S$ and if$(z \in x)$ in φ, then $(z \in y)$ in $\varphi\}$;
  $Pict[r] := \langle N_n[r], N_h[r], Inc[r]\rangle$;
  if $(\exists s < r\ Pict[s] = Pict[r])$ then
    $B := \top$;
until $(\varphi = \psi \vee B)$;
if B then $\varphi := \perp$.
return φ.

Figure 3.13: The Procedure $\in - \wp$introduction.

The procedure $\wp$Extensionalize is the analogue of Extensionalize in Section 3.3. The new variables introduced by this procedure can always be instantiated to atoms.

Example 3.11 *Each atom $a_i \in \mathcal{C}$ is a solution of the formula*

$$\wp(x, x).$$

Also the formula

$$\wp(x, x) \wedge y \in x$$

admits finite solutions. In this case one of the output formulae of our procedure is

$$\wp(x, x) \wedge y \in x \wedge y = \{y\},$$

which is satisfied by the solution of the system

$$x = \{y\} \wedge y = \{y\}.$$

The fact that $\wp(x, x)$ admits solutions is a consequence of the fact that it can be the case that some atom belongs to the second argument of a $\wp$-literal. On the other hand, if we consider the formula

$$\wp(x, x) \wedge y \in x \wedge y \notin y,$$

there are no models of $\Omega_P EN_a S(AFA_a)$ in which this formula has finite solutions:

- *$\{y\}^i$ belongs to x, for all $i \in \mathbb{N}$;*

```
∈−Boolean(φ) :
repeat
      τ := ⊤;
      for all ℓ subformulae of φ do
            if there is i ∈ 1P and σ such that ℓ = l_i[σ] then
                  if r_i[σ] is not subformula of φ then
                        φ := φ ∧ r_i[σ];
                        τ := ⊥;
until τ;
return φ.
```

Figure 3.14: The Procedure $\in\!-\mathsf{Boolean}$.

$\dfrac{\cup(y,z,w) \quad x \in y}{x \in w}\ (1_\cup)$	$\dfrac{\cup(y,z,w) \quad x \in w}{x \in y \mid x \in z}\ (2_\cup)$
$\dfrac{\cup(y,z,w) \quad x \notin y}{x \notin z \wedge x \notin w \mid x \in z \wedge x \in w}\ (3_\cup)$	$\dfrac{\cup(y,z,w) \quad x \notin w}{x \notin y \wedge x \notin z}\ (4_\cup)$
$\dfrac{\setminus(y,z,w) \quad x \in y}{x \notin z \wedge x \in w \mid x \in z \wedge x \notin w}\ (1_\setminus)$	$\dfrac{\setminus(y,z,w) \quad x \in z}{x \notin w}\ (2_\setminus)$
$\dfrac{\setminus(y,z,w) \quad x \in w}{x \notin z \wedge x \in y}\ (3_\setminus)$	$\dfrac{\setminus(y,z,w) \quad x \notin y}{x \notin w}\ (4_\setminus)$
$\dfrac{\setminus(y,z,w) \quad x \notin w}{x \notin y \mid x \in z}\ (5_\setminus)$	$\dfrac{y = \{x_1,\ldots,x_h\} \quad x \in y}{\varphi[x_1/x] \mid \ldots \mid \varphi[x_h/x]}\ (1_{\{\}})$
$\dfrac{x \in y \quad x \notin y}{\perp}\ (1_\perp)$	$\dfrac{y = \{x_1,\ldots,x_h\} \quad x_i \notin y}{\perp}\ (2_\perp)$

Figure 3.15: The rules $\mathsf{1P}$ for the procedure $\in\!-\mathsf{Boolean}$.

- $\{y\}^i \neq \{y\}^j$, *if* $i \neq j$.

If we consider the set of rules $\mathsf{1P} \cup \mathsf{2P}$ *and we apply them to the formula without concern for the newly introduced variables, then we obtain:*

$$y \in x,$$

hence

$$n_1 = \{y\} \wedge n_1 \in x,$$

hence

$$n_2 = \{n_1\} \wedge n_2 \in x,$$

and so on. Our procedure detects such loops and it terminates with output $\perp$.

Lemma 3.12 (Termination) *The procedure* $\mathsf{de} - \wp\mathsf{Boole}(\varphi)$ *always terminates.*

$\in$−Powerset(φ) :

$V := FV(\varphi)$;
repeat
    $\tau := \top$;
    for all ℓ subformulae of φ do
        if there is $i \in$ 2P and σ such that $\ell = l_i[\sigma]$
                and $x, y, z, z_i, w, t \in V$ then
            if $r_i[\sigma]$ is not subformula of φ then
                $\varphi := \varphi \wedge r_i[\sigma]$;
                $\tau := \bot$;
until τ;
return φ.

Figure 3.16: The Procedure $\in$−Powerset.

Proof. We prove that the procedures $\in$−$\wp$introduction and $\wp$Extensionalize always terminate.
At the end of each iteration of the repeat-loop in $\in$−$\wp$introduction, one of the following holds:

1. $\varphi = \psi$;
2. $\exists s < r \, Pict[s] = Pict[r]$;
3. $New \neq \emptyset$;
4. $\forall s < r \, Pict[s] \neq Pict[r]$.

If either 1. or 2. holds, then $\in$−$\wp$introduction terminates. Condition 3. holds only a finite number of times, since there are at most $2|FV(\varphi)|^2$ literals of the form $x \in y$ or $x \notin y$ and at most $|FV(\varphi)|(|FV(\varphi)| - 1)$ variables of the form k_{xy} which can be added to φ. So, there exists m such that for all $k \geq m$ condition 3. does not hold at the k-th iteration of the repeat-loop. Condition 4. cannot hold infinitely many times after the m-th iteration of the repeat-loop: $N_n[r], N_h[r] \in \wp(\wp(|FV(\varphi)|))$ and $Inc[r] \in \wp(|FV(\varphi)|^2)$, hence condition 4. does not hold for more than $4^{2^{|FV(\varphi)|}} 2^{|FV(\varphi)|^2}$ consecutive iterations of the loop. Hence, after at most $m + 4^{2^{|FV(\varphi)|}} 2^{|FV(\varphi)|^2}$ iterations, $\in$−$\wp$introduction terminates and $m \leq [2|FV(\varphi)|^2 + |FV(\varphi)|(|FV(\varphi)| - 1))][4^{2^{|FV(\varphi)|}} 2^{|FV(\varphi)|^2}]$.
The procedure $\wp$Extensionalize(φ) introduces new variables, but it makes use only of the literals in φ. Hence it must be the case that $x, y, z \in FV(\varphi)$ and $x \not\equiv z$. Therefore, the maximum number of steps and variables introduced by the procedure $\wp$Extensionalize(φ) is $|FV(\varphi)|^2(|FV(\varphi)| - 1)$. □

Example 3.13 *The number of new variables introduced by* de − $\wp$Boole *can blow up. As a matter of fact there are formulae with n variables whose minimal solution involves sets with* $\underbrace{2^{2^{\cdots}}}_{n}$ *elements:*

$$\wp(x_0, x_1) \wedge \ldots \wedge \wp(x_{n-1}, x_n) \wedge y \in x_0 \wedge \backslash(y, y, y).$$

Lemma 3.14 *If $\bot$ is the output formula of* $\in$−$\wp$introduction(φ)*, then φ does not have finite solutions.*

$$\frac{\wp(y,x) \quad z \in x \quad w \in z}{w \in y}\ (1_\wp) \qquad \frac{\wp(y,x) \quad (z_i \in y)_{i=1}^{h}}{n = \{z_1, \ldots, z_h\} \wedge n \in x}\ (2_\wp)$$

$$\frac{\wp(y,x) \quad z \notin x \quad w \in z \quad \text{with } z \not\equiv \{\ldots\}}{k_{zy} \in z \wedge k_{zy} \notin y}\ (3_\wp)$$

$$\frac{\wp(y,x) \quad n \notin x \quad n = \{z_1, \ldots, z_h\}}{z_1 \notin y \mid \ldots \mid z_h \notin y}\ (4_\wp)$$

$$\frac{\wp(x,y) \quad z \notin y \quad w \in x}{z \notin w}\ (5_\wp) \qquad \frac{t = \{z_1, \ldots, z_h\} \wedge \varphi[x]}{\varphi[t/x] \mid h_{xt} \in x \wedge \notin t}\ (2_{\{\}})$$

$$\frac{\top}{\bigwedge_{y \in V_n} n \in y \wedge \bigwedge_{y \in S \setminus V_n} n \notin y \text{ with } V_n \subseteq S}\ (1_S)$$

$$\frac{\top}{\bigwedge_{y \in V_{h_{xn}}} h_{xn} \in y \wedge \bigwedge_{y \in S \setminus V_{h_{xn}}} h_{xn} \notin y \text{ with } V_{H_{xn}} \subseteq S}\ (2_S)$$

Figure 3.17: The rules 2P for the procedure $\in-$Powerset.

Proof. If $\perp$ has been returned using a rule of the procedure $\in-$Boolean, then φ is not satisfiable.

If $\perp$ has been returned by the if condition of $\in-\wp$introduction, then there is a loop where new variables of type n and h_{xy} have been introduced. This means that there are two steps s and r such that $Pict[r] = Pict[s]$, with:

$$\begin{array}{llllll} n_1^s, & \ldots, & n_m^s, & h_1^s, & \ldots, & h_\ell^s \quad \text{introduced at step } s; \\ n_1^{s+1}, & \ldots, & n_o^{s+1}, & h_1^{s+1}, & \ldots, & h_v^{s+1} \quad \text{introduced at step } s+1; \\ \ldots & & & & & \\ n_1^r, & \ldots, & n_q^r, & h_1^r, & \ldots, & h_t^r \quad \text{introduced at step } r. \end{array}$$

Moreover, neither new literals of the form $x \in y$ or $x \notin y$ with $x, y \in FV(\varphi)$, nor new variables k_{xy} are introduced in the steps between step s and step r.

We prove that we can reproduce the situation of step $s+1$ at step $r+1$.

Since neither new literals $x \in y$ or $x \notin y$, nor new variables k_{xy} (with $x, y \in FV(\varphi)$) are added, all the variables n's introduced at step $s+1$ have at least one element which has been introduced at step s. If $n_j^{s+1} = \{\ldots, t^s, \ldots\}$ with t^s introduced at step s, we consider $n_j^{r+1} = \{\ldots, t^r, \ldots\}$, where t^r is an element introduced at step r which has the same characterization V_{t^r} of t^s, and we choose for n^{r+1} the set $V_{n_j^{r+1}} = V_{n_j^{s+1}}$. If $h_{xq^s}^{s+1}$ has been introduced during step $s+1$, then q^s has been introduced during step s using a literal of the form $\wp(z,w)$, moreover the literal $q^s \in w$ has been introduced at step s. The introduction of $h_{xq^s}^{s+1}$ means that all the elements of x were elements of q^s at step s, i.e. $(x,z) \in Inc[s]$, since all the elements of q^s were elements of z. From our hypothesis we have $Inc[s] = Inc[r]$, and hence there is an element q^r introduced at step r using the same rule used to introduce q^s which has exactly all the elements of x. Therefore at step $r+1$ we can introduce $h_{xq^r}^{r+1}$ and choose $V_{h_{xq^r}^{r+1}} = V_{h_{xq^s}^{s+1}}$. If we have to introduce other elements at step $r+1$ we can introduce them using one of the characterizations used at step $s+1$. Hence we have that

```
℘Extensionalize(φ, S) :
repeat
    τ := ⊤;
    for all ℓ(x, y, z) subformulae of φ such that x, y, z ∈ S do
        if there is i ∈ SP and σ such that ℓ = l_i[σ] then
            if r_i[σ] is not subformula of φ then
                φ := φ ∧ r_i[σ];
                τ := ⊥;
until τ;
return φ.
```

Figure 3.18: The procedure $\wp$Extensionalize.

$$\frac{x \in y \wedge z \notin y}{m_{xz} \in x \wedge m_{xz} \notin z \mid m_{zx} \notin x \wedge m_{zx} \in z} \quad (\mathsf{SP})$$

Figure 3.19: The rule SP for the procedure $\wp$Extensionalize.

$New_n[r+1] = New_n[s+1]$ and $New_h[r+1] = New_h[s+1]$, since we have introduced enough new variables to cover $New_n[s+1]$ and $New_h[h+1]$, and all the choices have been performed inside these sets. Moreover $Inc[r+1] = Inc[s+1]$, since this follows from the fact that all the new variables introduced are added to the same subsets of variables used at step $s+1$. Hence, if the procedure does not fail, it loops forever and builds a solution which is not finite: all the new variables introduced are necessary to satisfy the formula and at each step we try to identify the new variables n's with the old variables. □

Lemma 3.15 *Let ψ be the output formula of* $\mathsf{de} - \wp\mathsf{Boole}(\varphi)$. *If in ψ there are no literals of the form $\perp$, then in every model of $\Omega_P EN_a S(AFA_a)$ ψ is finitely satisfiable.*

Proof. Let us consider the following sets of equations:

- $x = \{y_1, \ldots, y_h\}$ for each variable x in ψ such that in ψ there is at least one literal of the form $y_j \in x$, where $y_1, \ldots, y_h$ are all the variables which appear in $y_i \in x$ in ψ;
- $x = a_x$ for each variable x in ψ which does not appear in any literal of the form $y \in x$ the equation, where a_x is an atom and $a_x \neq a_y$, if $x \not\equiv y$.

(AFA_a) ensures that the system containing all the above equations has always a (unique) solution $\sigma = [x_1/\bar{x}_1, \ldots, x_n/\bar{x}_n]$. This σ satisfies φ, as it is proved in Lemma 3.15. □

From this lemma we have:

Theorem 3.16 (Correctness and Completeness) *Let the formulae $\psi_1, \ldots, \psi_m$ be the non-deterministic outputs of the procedure* $\mathsf{de} - \wp\mathsf{Boole}(\varphi)$. *In every model of the theory $\Omega_P EN_a S(AFA_a)$, the formula φ has a finite solution if and only if there exists $j \in \{1, \ldots, m\}$ such that in ψ_j there are no literals of the form $\perp$.*

This means that the procedures $\mathsf{de} - \wp\mathsf{Boole}$ and $\wp\mathsf{REDUCTION3}$ can be used to decide the class of purely existentially quantified formulae of $\mathcal{L}$.

3.8 Concluding Remarks

The procedure we have presented to decide the finite satisfiability of purely existentially quantified formulae with a powerset-like predicate, makes use of rules which hold also in stronger theories, i.e. set theories without atoms, such as $\Omega^- NS(\mathrm{AFA}) \cup (\wp)$. A possible way to apply the procedure for deciding the finite satisfiability in the context of set theories with Extensionality is the following:

- if the output of the procedure is $\perp$, then the formula is not finitely satisfiable, even without atoms;
- if the output is different from $\perp$ and σ is a solution of the output formula in a model of $\Omega_P EN_a S(\mathrm{AFA}_a)$, then it is necessary to find a way to replace the atoms in order to move to a solution without atoms.

When we are working in the context of a well-founded set theory we also have to check that the membership relation is not circular. In this context the decision result presented in [COU01] ensures that a formula is satisfiable if and only if it is finitely satisfiable and the result also gives a bound on the rank of a minimal solution.

4

Universal Quantifiers

In this chapter we refer to the theory Ω introduced in Section 1.2.7. This theory, as already mentioned, is at the ground of the $\Box$-as-$\wp$ translation: is a minimal set theory which allows to set-theoretically encode in First-Order Logic the Kripke semantics.

Moreover, from a purely set-theoretical point of view, if Ω_P was the predicative theory which allows to analyse the interplay among the set-operators pruning them from matters due to the assumption of Extensionality, Regularity, or any form of anti-foundation, then Ω is one of the most natural theories in which we can push ahead this analysis: we add the complications following from the functionality of the operators.

The class of formulae we consider here is one of the possible classes to analyse after the purely existentially quantified class (we add restricted universal quantifiers), but it is also based on syntactical restrictions suggested by the $\Box$-as-$\wp$ translation. In particular, it is motivated by the results presented in [DMP96]. In Section 4.2 we briefly recall such results which are based on a stronger theory Ω', while in Section 4.4 we extend the results to the theory Ω.

We present a tableau procedure to study the satisfiability of the formulae in the class we introduce w.r.t. models of Ω. In general, the procedure could not terminate (the class we consider has not the small model property). In the cases in which the procedure terminates producing a branch which does not end with $\perp$, and in the cases in which the procedure does not terminate we are able to exhibit a model of Ω in which the formula under consideration is satisfiable. If we would be able to determine a condition which recognizes whether we are entered into an infinite branch, we could immediately turn our procedure into a decision procedure. For a subclass of formulae we are able to determine a termination condition. The difficulties in finding a termination condition depend on the facts that: at each iteration we add new variables representing the elements of the variable x (the translation of the frame); it is necessary to introduce literals which involve these new variables in order to satisfy the constraints imposed by the universal quantified part of the formula on the elements of x; it is difficult to recognize under which conditions it is possible to introduce on the new variables literals similar to the ones introduced during previous iterations.

4.1 The Language and the Theory

In this chapter we consider the first-order language

$$\mathcal{L} = \langle \mathcal{F}, \mathcal{C}, \mathcal{P}, \mathcal{V} \rangle,$$

with

$$\mathcal{F} = \{\cup, \setminus, \wp\}, \quad \mathcal{C} = \emptyset, \quad \mathcal{P} = \{\in, \subseteq\},$$

and $\mathcal{V}$ a denumerable set of variables.

Let Ω be the theory in Figure 4.1.

$(\cup)$	$\forall x,\, y,\, z\, (x \in y \cup z \leftrightarrow x \in y \vee x \in z)$
$(\setminus)$	$\forall x,\, y,\, z\, (x \in y \setminus z \leftrightarrow x \in y \wedge x \notin z)$
$(\subseteq)$	$\forall x,\, y\, (x \subseteq y \leftrightarrow \forall z\, (z \in x \rightarrow z \in y))$
$(\wp)$	$\forall x,\, y\, (x \in \wp(y) \leftrightarrow x \subseteq y)$

Figure 4.1: The Theory Ω.

Let $\mathcal{S}_{\exists\forall}$ be the class of formulae of the form

$$x \subseteq \wp(x) \wedge \varphi_1 \wedge \ldots \wedge \varphi_n \wedge w \in x \wedge w \notin t(x, \vec{z})$$

where

$$\varphi_i = \forall \vec{y} \in x(p^i_1(\vec{y}, \vec{v}) \vee \ldots \vee p^i_{m_i}(\vec{y}, \vec{v})) \tag{4.1}$$

with $p^i_j(\vec{y})$ in one of the following forms

$$\begin{array}{c} y_h \in y_k,\ y_h \notin y_k,\ y_h \neq y_k, y_h = y_k \\ y_h \in v_k,\ y_h \notin v_k,\ y_h \neq v_k, y_h = v_k \\ v_h \in y_k,\ v_h \notin y_k,\ v_h \neq y_k, v_h = y_k \end{array}$$

and

$$t(x, \vec{z}) ::= z_j \mid x \mid t_1 \cup t_2 \mid x \setminus t_1 \mid \wp(t_1). \tag{4.2}$$

We study the satisfiability problem of formulae in $\mathcal{S}_{\exists\forall}$ in models of Ω.

In the next section we motivate our choice for the class of formulae.

4.2 Motivations

In Chapter 1 (Section 1.2) we gave the definition of the set-theoretic translation $\gamma^s(x)$ of a modal formula $\gamma(p_1, \ldots, p_n)$. One of the results in [DMP95] is that if Λ is the complete modal logic obtained by adding one axiom Ax to the modal logic K, then

$$\vdash_\Lambda \gamma \quad \text{iff} \quad \Omega \vdash \forall x(x \subseteq \wp(x) \wedge Ax^s(x) \rightarrow \gamma^s(x)).$$

This means that the formula $\forall x(x \subseteq \wp(x) \wedge Ax^s(x) \rightarrow \gamma^s(x))$ is the formal counter-part to the Kripke semantics and that the theory Ω is a sort of minimal theory in which it is possible to formalize such semantics. One of the main feature of the above translation ($\Box$-as-$\mathcal{P}$ translation) is its ability to deal with any modal logic, regardless the fact that such a logic is first-order definable or not. On the other hand, the results presented in [DMP96] were also directed to a study of the possibilities of transferring decidability results to and from Modal Logic and Set Theory in the case of first-order definable, and in particular first-order complete, modal logics.

Λ is a first-order complete modal logic (which extends K) with first-order correspondent δ in the language $\{R, =\}$ (R is the accessibility relation) if it holds that

$$\vdash_\Lambda \gamma \qquad \text{iff} \qquad \gamma \text{ is valid in every frame in which } \delta \text{ is valid.}$$

Notice that δ is a sentence (i.e., the set of free variables in δ is empty, $FV(\delta) = \emptyset$). It is possible to translate δ into a set-theoretic formula using the following definition.

Definition 4.1 *Let δ be a first-order formula in the language $\{R, =\}$ and let x be a variable which does not occur in δ. The formula $\delta^s(x)$ is the formula we obtain from δ by:*
1. replacing all the occurrences of R by $\in$;
2. replacing all the $\forall y$ and all the $\exists y$ by $\forall y \in x$ and $\exists y \in x$ respectively.

Using a suitable extension $\bar{\Omega}$ of Ω (e.g. ZF$^-$) it is possible to prove that if Λ is a first-order complete modal logic (which extend K) with first-order correspondent δ, then

$$\vdash_\Lambda \gamma(p_1, \ldots, p_n) \quad \text{iff} \quad \bar{\Omega} \vdash \forall x(x \subseteq \wp(x) \land \delta^s(x) \to \gamma^s(x)). \tag{4.3}$$

In particular in [DMP96] the result has been proved using the theory Ω' (i.e., $\Omega + Cc + s$). Spelling out all the variables and the meaning of the formulae, introduced in the translation, we have that if the class of formulae of the form

$$\forall x \forall \vec{z} \forall \vec{v}(x \subseteq \wp(x) \land \forall \vec{y} \in x(\alpha^s(x, \vec{y}, \vec{v})) \to x \subseteq \gamma^*(x, \vec{z})), \tag{4.4}$$

where α is a quantifier-free conjunction of clauses over the language $\{R, =\}$, were decidable over Ω', then all first-order complete modal logics whose first-order correspondent is in the class $\exists^*\forall^*$ would be decidable.

In the following sections we prove that in (4.3), and hence in (4.4), it is possible to replace Ω' with the much more elementary Ω. Then we present a procedure which deals with this last decidability problem. We prove that when δ is in $\exists^*\forall^2$ and α does not contain equalities we are able to decide the satisfiability problem adding a termination condition to our procedure. In this way we obtain that using a *unique* decision procedure, a sort of uniform method to build canonical counter-examples, we can uniformly decide over a large class of modal logics, containing K, T, $S4$, $S5$, $S4.3$.

4.3 From Graphs to Models of Ω

In order to build a model of Ω in which an unquantified formula ψ is satisfiable, a natural idea is to build a graph whose nodes are the variables in ψ and whose edges satisfy the atoms of ψ by corresponding to the membership relation (see Chapters 2 and 3). Then, the graph need to be embedded into a model of Ω. In this section we define how to build a model of Ω out of a graph by a *saturation* procedure on the graph in order to satisfy the axioms of Ω. We will use the result we give in this section both to extend to Ω the validity of (4.3) above and to prove the correctness of our procedure Check designed to verify the satisfiability of formulae in the class $\mathcal{S}_{\exists\forall}$ (see Section 4.5). The procedure Check in a certain sense generalizes such kind of construction dealing with formulae in the class we want to decide.

Let $G = \langle N, \gtrdot \rangle$ be a graph, with N possibly infinite. The graph G_ω is defined exactly as the graph G_{ω_P} presented in Section 2.4 Equation 2.2 (we use here ω instead of ω_P because of the name of the underling theory). What we have to redefine here (we cannot reuse the definition of Chapter 2) is the way in which we read G_ω as an interpretation of $\mathcal{L}$. Let τ be a well-ordering over N_ω. From G_ω we obtain an interpretation of $\mathcal{L}$ defining $\forall v_1, v_2, \in N_\omega$:

- $u \in v$ if and only if $u \lessdot_\omega v$;
- $v \cup u = w$, with w the min (w.r.t. τ) such that $\forall x(x \in w \leftrightarrow x \in v \vee x \in u)$;
- $v \setminus u = w$, with w the min (w.r.t. τ) such that $\forall x(x \in w \leftrightarrow x \in v \wedge x \notin u)$;
- $u \subseteq v$ if and only if $\forall x(x \in u \rightarrow x \in v))$;
- $\wp(v) = w$, with w the min (w.r.t. τ) such that $\forall x(x \in w \leftrightarrow x \subseteq v)$.

We use G_ω to refer to this interpretation of $\mathcal{L}$, forgetting that it is τ-dependent. We have to prove that G_ω is a model of Ω and that G is a complete subgraph in G_ω. First we prove that this last property holds for all the G_m, hence we use this fact to prove that G_ω is a model of Ω.

Lemma 4.2 *For all $m \in \mathbb{N}$, G_m is a complete subgraph in G_ω*

Proof. Let $k > m$, the edges that are in $\gtrdot_k \setminus \gtrdot_m$ are of the form $\langle t, w \rangle$ with $t \notin N_{k-1}$, since this is the first condition in the definition of $\gtrdot_k$. From $m \leq k-1$ we have $N_m \subseteq N_{k-1}$, hence $t \notin N_n$, which gives us the thesis. □

Another important property of our construction is that we never add a node whose set of outgoing edges is equal to the set of outgoing edges of a node which was already in the graph at the previous iteration.

Lemma 4.3 *Let u and v be two nodes in G_ω, such that $\{x \mid x \lessdot_\omega v\} = \{x \mid x \lessdot_\omega u\}$. If $u \in N_m$, then $v \in N_m$.*

Proof. It is a consequence of the fact that in G_k we never add a node whose set of outgoing edges is equal to that of a node in G_{k-1}, and of Lemma 4.2. □

Lemma 4.4 *G_ω is an interpretation of $\mathcal{L}$, and a model of Ω.*

Proof. First we have to prove that $\cup, \setminus$ are defined over $N_\omega \times N_\omega$, and $\wp$ is defined over N_ω. Let v and u be two nodes in G_ω, let $m \in \mathbb{N}$ be such that $\{v, u\} \subseteq N_m$ and $\{v, u\} \not\subseteq N_{m-1}$, and let $n = 4\lceil m/4 \rceil$ (the least multiple of 4 greater than m).

We prove by contradiction that it cannot be the case that $W_\omega = \{w \in N_\omega \mid \forall x(x \lessdot_\omega w \leftrightarrow x \lessdot_\omega v \vee x \lessdot_\omega u)\}$ is empty. If, by contradiction, we assume $W_\omega = \emptyset$, then we have $W_\omega \cap N_n = \emptyset$. In our definition of N_{n+1} this implies that $t = \langle v, u, \cup \rangle$ is in N_{n+1}. From the definition of $\gtrdot_{n+1}$ we have that $t \in W_{n+1}$, where $W_{n+1} = \{w \in N_{n+1} | \forall x(x \lessdot_{n+1} w \leftrightarrow x \lessdot_{n+1} v \vee x \lessdot_{n+1} u)\}$. From Lemma 4.2 we have that since $v, u, t \in N_{n+1}$ it holds that $\{x \,|\, x \lessdot_\omega v\} = \{x \,|\, x \lessdot_{n+1} v\}$, $\{x \mid x \lessdot_\omega u\} = \{x \mid x \lessdot_{n+1} u\}$, and $\{x \mid x \lessdot_\omega t\} = \{x \mid x \lessdot_{n+1} t\}$, hence $t \in W_{n+1}$ implies $t \in W_\omega$, which gives us a contradiction.

A similar argument can be used to prove that $\setminus$ is defined over $N_\omega \times N_\omega$.

Now we prove that $U_\omega = \{w \in N_\omega \mid \forall x(x \lessdot_\omega w \leftrightarrow \forall y(y \lessdot_\omega x \rightarrow y \lessdot_\omega v)\}$ is not empty. Let $S = \{x \in N_\omega \mid \forall y(y \lessdot_\omega x \rightarrow y \lessdot_\omega v)\}$. First we prove that $S \subseteq N_{n+3}$. Let us assume by contradiction that there exists $x \in S \setminus N_{n+3}$. Since $v \in N_n$ by Lemma 4.2 we have that all the outgoing edges from v are in $\gtrdot_n$, hence from the definition of G_{n+3} we have that there exists at least one node $x' \in N_{n+3}$ such that $\{y \mid y \lessdot_\omega x\} = \{y \mid y \lessdot_{n+3} x'\}$. From Lemma 4.3 we have that $x \in N_{n+3}$. Now, coming back to our proof of $U_\omega \neq \emptyset$, let us assume by contradiction that this is not the case. Then we have $U_\omega \cap N_{n+3} = \emptyset$. In our definition of N_{n+4} this implies that $t = \langle v, \wp\rangle \in N_{n+4}$, moreover from the definition of $\gtrdot_{n+4}$ we have that $x \lessdot_{n+4} t$ if and only if $\forall y(y \lessdot_{n+4} x \rightarrow y \lessdot_{n+4} v)$. Since $S \subseteq \gtrdot_{n+3}$ we have that $S = \{x \mid x \lessdot_{n+4} t\}$. From Lemma 4.2, since $t \in N_{n+4}$ we obtain that $S = \{x \mid x \lessdot_\omega t\}$, which implies $t \in U_\omega$. This is a contradiction, hence we have the thesis.

The definition of the interpretation immediately ensures that G_ω is a model of Ω. □

4.4 Ω and Modal Logic

In Section 4.1 we recalled the result presented in [DMP96]. Here we precisely extend to Ω that result (4.3). In particular, this motivate the specific satisfiability problem we deal with in the rest of the chapter.

Let Λ be a first-order complete modal logic (which extends K) with first-order correspondent δ in the language $\{R, =\}$.

Proposition 4.5 (Correctness and Completeness) *Let $\gamma(p_1, \ldots, p_n)$ be a modal formula, we have that*

$$\vdash_\Lambda \gamma(p_1, \ldots, p_n) \textit{ iff } \Omega \vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \ldots, z_n))).$$

Proof. If $\Omega \vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \ldots, z_n)))$, then $\bar{\Omega} \vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \ldots, z_n)))$, where $\bar{\Omega}$ can be for instance the theory Ω' used in [DMP96] (it is sufficient to use a theory which is strong enough). From this last using Proposition 5.1 in [DMP96] we have that $\vdash_\Lambda \gamma(p_1, \ldots, p_n)$.

We prove the left to right direction proving that if $\Omega \not\vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \ldots, z_n)))$, then $\not\vdash_\Lambda \gamma(p_1, \ldots, p_n)$.

Since Λ is first-order complete, with first-order correspondent δ we have to find a frame $\mathcal{F} = \langle W, R\rangle$ in which δ is valid and in which there exists a world w and a valuation V such that $\langle \mathcal{F}, V\rangle, w \Vdash \neg\gamma(p_1, \ldots, p_n)$. Since $\Omega \not\vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \ldots, z_n)))$, we have that there exists a model $\mathcal{M}$ of Ω and a valuation θ such that $\mathcal{M} \models (x \subseteq \wp(x) \wedge \delta^s(x) \wedge v \in x \wedge v \notin \gamma^*(x, z_1, \ldots, z_n))[\theta]$. Let $W = \{w \mid w \in \theta(x)\}$ and put $w_1 R w_2$ if and only if $w_2 \in w_1$ holds in $\mathcal{M}$. We consider the frame $\mathcal{F} = \langle W, R\rangle$ and we prove that $\mathcal{M} \models \delta^s(\theta(x))$ implies that δ is valid in the first-order model $\mathcal{F}$. Notice that $W = \theta(x) \subseteq M$, hence a valuation σ on $\mathcal{F}$ is also a valuation on $\mathcal{M}$. We have that for all $w_1, w_2 \in W$, $w_1 = w_2$ holds in W iff it is true in $\mathcal{M}$ (in particular in $\theta(x)$) and $w_1 R w_2$ iff $w_1 \ni w_2$ is true $\mathcal{M}$ (in particular in $\theta(x)$). Hence the map $i : \mathcal{F} \rightarrow \mathcal{M}$ defined as $i(w) = w$ is a first-order embedding (it turns R into $\ni$). Since $i[W] = \theta(x)$, if we consider $\theta(x)$ has a submodel of $\mathcal{M}$ we have that i is an isomorphism. This implies that if δ is a formula on the first-order language $\{R, =\}$ and δ' is the formula we obtain replacing R with $\ni$ we have that $\mathcal{F} \models \delta$ if and only if $\theta(x) \models \delta'$. If now we replace all the $\exists y$ and $\forall y$ with $\exists y \in x$ and $\forall y \in x$ respectively we obtain that $\theta(x) \models \delta'$ if and only if $M \models \delta^s(\theta(x))$ (relativization).

Let $w = \theta(v)$ and let V be the valuation such that $V(p_i) = \theta(z_i)$. We prove by induction on $\gamma(p_1, \ldots, p_n)$ that

$$w \in \theta(x) \wedge w \notin \gamma^*(\theta(x), \theta(z_1), \ldots, \theta(z_n)) \text{ iff } \langle \mathcal{F}, V\rangle, w \Vdash \neg\gamma(p_1, \ldots, p_n).$$

If $\gamma = p$, then we have $w \in \theta(x)$ and $w \notin \theta(x) \setminus \theta(z_i)$ iff $w \in W$ and $w \notin V(p_i)$, which is the thesis. If $\gamma = \gamma_1 \vee \gamma_2$, then we have that $w \in \theta(x)$ and $w \notin (\gamma_1^* \cup \gamma_2^*)[\theta]$ iff $w \in W$ and $w \notin \gamma_1^*[\theta]$ and $w \notin \gamma_2^*[\theta]$, which by inductive hypothesis give us the thesis. If $\gamma = \neg\gamma_1$, then we obtain the thesis by inductive hypothesis. Let $\gamma = \Box\gamma_1$. If $w \in \theta(x)$ and $w \notin \wp(\gamma_1^*)[\theta]$, then there exists $w_1 \in w$ such that $w_1 \notin \gamma_1^*[\theta]$. Since $\theta(x) \subseteq \wp(\theta(x))$ we obtain that $w_1 \in \theta(x)$, hence by inductive hypothesis $\langle \mathcal{F}, V \rangle, w_1 \Vdash \neg\gamma_1$. Moreover from $w_1 \in w$ we have wRw_1, hence $\langle \mathcal{F}, V \rangle, w_1 \Vdash \Diamond\neg\gamma_1$, i.e. the thesis. If $\langle \mathcal{F}, V \rangle, w \Vdash \neg\Box\gamma_1$, then there exists w_1 such that wRw_1 and $\langle \mathcal{F}, V \rangle, w_1 \Vdash \neg\gamma_1$, hence we obtain the thesis in a similar way. $\Box$

Now if we want to decide
$$\vdash_\Lambda \gamma,$$
from Proposition 4.5 it follows that we can equivalently decide
$$\Omega \vdash \forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \dots, z_n))).$$
This is equivalent to decide the unsatisfiability in Ω of
$$\neg(\forall x(x \subseteq \wp(x) \wedge \delta^s(x) \rightarrow \forall \vec{z}(x \subseteq \gamma^*(x, z_1, \dots, z_n)))).$$
Bringing this formula towards its prenex normal form, we obtain that we can equivalently decide the unsatisfiability in Ω of
$$\exists x \exists \vec{z} \exists w(x \subseteq \wp(x) \wedge \delta^s(x) \wedge w \in x \wedge w \notin \gamma^*(x, z_1, \dots, z_n)).$$
When δ is of the form $\exists^*\forall^*\alpha$ we can give to the above formula the equivalent form
$$\exists x \exists \vec{z} \exists w \exists \vec{v}(x \subseteq \wp(x) \wedge \forall \vec{y} \in x(\alpha^s(x, \vec{y}, \vec{v})) \wedge w \in x \wedge w \notin \gamma^*(x, z_1, \dots, z_n)).$$
The above argument shows that:

Proposition 4.6 *If the satisfiability problem w.r.t. Ω of the class of formulae of the form*
$$\exists x \exists \vec{z} \exists w \exists \vec{v}(x \subseteq \wp(x) \wedge \forall \vec{y} \in x(\alpha^s(x, \vec{y}, \vec{v})) \wedge w \in x \wedge w \notin \gamma^*(x, z_1, \dots, z_n)),$$
where α^* is quantifier-free, is decidable, then the decidability of all the first-order complete modal logics with first-order correspondent in the class $\exists^*\forall^*$ follows.

In the next two sections we study the above satisfiability problem. In the general case we are able to give a procedure which could not terminate. In the case δ is in the class $\exists^*\forall^2$ and in α^s there are no positive equality literals we are able to add a test to the procedure in order to obtain a procedure which decides the satisfiability problem.

4.5 The Tableau Satisfiability Procedure

As we explained in the previous section, the class of formulae we are interested in is the class $\mathcal{S}_{\exists\forall}$ of formulae of the form
$$x \subseteq \wp(x) \wedge \varphi_1 \wedge \dots \wedge \varphi_n \wedge w \in x \wedge w \notin t(x, \vec{z})$$
where
$$\varphi_i = \forall \vec{y} \in x(p_1^i(\vec{y}, \vec{v}) \vee \dots \vee p_{m_i}^i(\vec{y}, \vec{v})) \tag{4.5}$$

with $p_j^i(\vec{y})$ in one of the following forms

$$\begin{array}{l} y_h \in y_k,\ y_h \notin y_k,\ y_h \neq y_k, y_h = y_k \\ y_h \in v_k,\ y_h \notin v_k,\ y_h \neq v_k, y_h = v_k \\ v_h \in y_k,\ v_h \notin y_k,\ v_h \neq y_k, v_h = y_k \end{array}$$

and

$$t(x,\vec{z}) ::= z_j \mid x \mid t_1 \cup t_2 \mid x \setminus t_1 \mid \wp(t_1). \tag{4.6}$$

We want to test whether there exists at least one model of Ω in which a formula ψ of this class is satisfiable. Notice that $FV(\psi) = \{x, w, \vec{v}, \vec{z}\}$. Let us recall the connection between this satisfiability problem and the problem $\vdash_\Lambda \gamma$, when Λ has δ as first-order correspondent:

- the variable x represent the *frame* we are looking for, i.e. a frame in which δ is valid and γ is not;
- $\varphi_1 \wedge \ldots \wedge \varphi_n$ is our δ^s;
- the variables y's and v's are *worlds* of our frame x;
- the variable w represent the world in which we want to *falsify* γ;
- $t(x,\vec{z})$ is the $\Box$-as-$\mathcal{P}$ translation of γ, i.e. it is *the set of world at which* γ *is true*;
- the z's represent the *values* (set of worlds) we must assign to the propositional literals in γ in order to make that w falsify γ.

The main idea behind our decision procedure is the following: we build the frame (set) x starting from its world (element) w; at each iteration we add the new worlds (elements) that are necessary to satisfy δ and to ensure that w does not satisfy γ (w does not belong to γ^*).

The fact that we look at the problem from a set-theoretic perspective gives us the advantage that:

- in order to add the new elements to x (and to the subterms of $t(x,\vec{z})$, when it is necessary), we can use the rules we obtain from the natural meaning that Ω gives to the set-operators;
- in order to ensure termination it is sufficient to check to which subsets of x we are adding elements: if these subsets are already *big enough* we can stop our procedure.

In order to decide whether a formula in the class $\mathcal{S}_{\exists\forall}$ is satisfiable it is convenient to deal with a larger class $\mathcal{S}$ of formulae.

Definition 4.7 (Class $\mathcal{S}$) *Let $\{x, z_1, \ldots, z_p, v_1, \ldots, v_q\}(= \{x, \vec{z}, \vec{v}\})$ be a finite set of distinct variables, and $Term = \{t_1(x,\vec{z}), \ldots, t_s(x,\vec{z})\}$ be a finite set of terms of the form (4.6). Let $A, B \subseteq I_q \times I_s$ and $C, D, E \subseteq I_q \times I_q$.*[1] *A formula $\psi(x,\vec{z},\vec{v})$ is in the class $\mathcal{S}$ if and only if it is of the form*

$$\psi_\forall(x,\vec{v}) \wedge x \subseteq \wp(x) \wedge \psi_{\mathsf{term}}(x,\vec{z},\vec{v}) \wedge \psi_{\mathsf{var}}(\vec{v}),$$

where

$$\psi_\forall(x,\vec{v}) = \bigwedge_{i=1}^{n} \varphi_i,$$

[1] Let $n \in \mathbb{N}$, $n > 0$, we use I_n to refer to $\{1, \ldots, n\}$.

with the φ_i's of the form (4.5),

$$\psi_{\mathsf{term}}(x,\vec{z},\vec{v}) = \bigwedge_{\langle a_1,a_2\rangle\in A} (v_{a_1} \notin t_{a_2}(x,\vec{z})) \wedge \bigwedge_{\langle b_1,b_2\rangle\in B} (v_{b_1} \in t_{b_2}(x,\vec{z})),$$

and

$$\psi_{\mathsf{var}} = \bigwedge_{\langle c_1,c_2\rangle\in C} (v_{c_1} \notin v_{c_2}) \wedge \bigwedge_{\langle d_1,d_2\rangle\in D} (v_{d_1} \in v_{d_2}) \wedge \bigwedge_{\langle e_1,e_2\rangle\in E} (v_{e_1} \neq v_{e_2}).$$

The fact that in ψ_{var} we do not have equalities is not a restriction: it is possible to map a formula in which there are equalities of the form $v_h = v_k$ into an equivalent formula in $\mathcal{S}$. The class $\mathcal{S}_{\exists\forall}$ is a subclass of $\mathcal{S}$. A formula in $\mathcal{S}$ still have a modal interpretation: we are looking for a frame x whose accessibility relation satisfies certain conditions ($\psi_\forall \wedge \psi_{\mathsf{var}}$) and in which there are worlds which do not satisfy and do satisfy certain modal formulae (ψ_{term}).

We now start to describe a procedure to test the satisfiability of formulae of $\mathcal{S}$ w.r.t. the theory Ω. Let us consider the set of rules, $\exists$-reductions, in Figure 4.2. The notation used

$$\frac{w \notin s_1 \cup s_2}{w \notin s_1 \wedge w \notin s_2}\ (1_\exists) \qquad \frac{w \notin k \setminus s_1}{w \notin k \mid w \in s_1}\ (2_\exists)$$

$$\frac{w \notin \wp(s_1)}{w_1 \in w \wedge w_1 \notin s_1}\ (3_\exists) \qquad \frac{w_1 \in w \quad w \in k \quad k \subseteq \wp(k)}{w_1 \in k}\ (4_\exists)$$

$$\frac{w \in s_1 \cup s_2}{w \in s_1 \mid w \in s_2}\ (5_\exists) \qquad \frac{w \in k \setminus s_1}{w \in k \wedge w \notin s_1}\ (6_\exists)$$

$$\frac{w_1 \in w \quad w \in \wp(s_1)}{w_1 \in s_1}\ (7_\exists)$$

Figure 4.2: The rules $\exists$-reductions.

in rules ($2_\exists$) and ($5_\exists$) means that it is necessary to choose (non-deterministically) one out of two alternatives. The rule ($4_\exists$) is needed only to deal with the inclusion $x \subseteq \wp(x)$, since in our class of formulae there are no other inclusions. Notice that rule ($3_\exists$) is the only one which leads to the introduction of a new variable. In these rules we make use of the variables w, w_1, s_i, k. When we apply these rules to formulae in $\mathcal{S}$ we have that: w's, w_1's correspond to variables in $\vec{v}$; the variables s_i's correspond to terms of $Term$; the variable k corresponds to x. Later we will use variables $\vec{h}$, which for formulae in $\mathcal{S}$ correspond to the variables $\vec{y}$.

Intuitively, what we want to do is to apply all the rules of $\exists$-reductions, until a fix-point is reached. Rule ($3_\exists$) can cause a problem since it introduces new variables. However, in ψ there are only a finite number of $\wp$ operators and, moreover, the r.h.s. of ($3_\exists$) is of 'lower complexity' than its l.h.s.; hence the fix-point will always be reached in a finite number of iterations.

Our next problem is that we must apply the $\exists$-reductions rules together with other rules we will introduce later in this section. It is an uncontrolled interaction of the $\exists$-reductions with these other rules that can lead to non-termination. This is why we introduce a procedure in which we control the application of the $\exists$-reductions. The procedure apply-$\exists$ (Figure 4.3) takes as input a formula ψ in the class $\mathcal{S}$, a variable v, and a term t, and allows to apply rule ($3_\exists$) only if $\sigma(w) = v$ and $\sigma(\wp(s_1)) = t$. In Figure 4.4 we give the second set of rules we

```
apply-∃(ψ, v, t):
repeat
  τ := ⊤;
  for all ℓ subformulae of ψ do
      if there is i ∈ {1∃, 2∃, 4∃, 5∃, 6∃, 7∃} and σ such that ℓ = l_i[σ] then
          if r_i[σ] is not subformula of ψ then
              ψ := ψ ∧ r_i[σ];
              τ := ⊥;
      if ℓ = v ∉ t and t = ℘(t1) then
          if there is not v1 such that
            (v1 ∈ v ∧ v1 ∉ t1) is subformula of ψ then
                ψ := ψ ∧ v1 ∈ v ∧ v1 ∉ t1;
                τ := ⊥;
until τ;
return ψ.
```

Figure 4.3: The Procedure apply-$\exists$.

need and by the procedure apply-$\forall$ (Figure 4.5) we explain how we control the application of these rules. apply-$\forall$ takes a formula ψ as input and applies the rule in $\forall$-reductions until a

$$\frac{\vec{w} \in k \quad \forall \vec{h} \in k(p_1 \vee \ldots \vee p_m)}{p_1[\vec{h}/\vec{w}] \mid \ \ldots \ \mid p_m[\vec{h}/\vec{w}]} \ (1_\forall)$$

Figure 4.4: The rule $\forall$-reductions.

fix-point is reached. It also performs a collapsing of the variables on which an equality has been introduced. It is clear that a sufficient large number of applications of $\forall$-reductions to a formula ψ always reaches a fix-point (immediate consequence of the fact that this rule never adds new variables). As we mentioned before, a problem with termination could arise when we mix $\exists$-reductions and $\forall$-reductions.

To conclude, we need to give the rules to close the branches of the tableau: such general-reductions are presented in Figure 4.6. and it is clear how to apply them to a formula ψ in $\mathcal{S}$. The procedure in Figure 4.7 gives the details. Notice that if ψ is a formula in $\mathcal{S}$, then also apply-$\exists(\psi, w, s)$, apply-$\forall(\psi)$, and apply-gen(ψ) are in $\mathcal{S}$.

We introduce an auxiliary function part, which takes as input three sets and returns a Boolean value. part(A, B, C) returns $\top$ if and only if A and B partition C into one or two blocks.

$$\mathsf{part}(A, B, C) := (A \cap B = \emptyset \text{ and } A \cup B = C).$$

The procedure Check (Figure 4.8), combines all the subroutines introduced up to this point in our general tableau procedure. Let us assume that in ψ_{term} there are m terms of the form $t(x, \vec{z})$ and that the set T of all their subterms has cardinality p. For each variable v in $\vec{v}$ we can guess to which elements of T v has to belong (the set EC_v) and, hence, to which elements of T v does not belong (the set NC_v). From the point of view of Kripke semantics

```
apply-∀(ψ):
repeat
    τ := ⊤;
    for all ∀y_1,...,y_h ∈ x(p_1 ∨ ... ∨ p_m) subformulae of ψ do
        for all (v_1 ∈ x ∧ ... ∧ v_h ∈ x) subformulae of ψ do
            if there is not k ≤ m such that
              p_k[y⃗/v⃗] is subformula of ψ then
                  let k ≤ m;
                  ψ := p_k[y⃗/v⃗] ∧ ψ;
                  τ := ⊥;
until τ;
return ψ.
for all (v_1 = v_2) subformulae of ψ do
    ψ := ψ[v_1/v_2];
```

Figure 4.5: The Procedure apply-∀.

$$\frac{w_1 \in s_1 \quad w_1 \notin s_1}{\bot}\ (1_g) \qquad \frac{w_1 \neq w_1}{\bot}\ (2_g)$$

Figure 4.6: The rules general-reductions.

this guess corresponds to guess the subformulae of γ which holds at v. The $\exists$-reductions rules infer, using EC_v and NC_v, the membership relations and together with the general-reductions rules, they check the consistency of EC_v. The $\forall$-reductions rules add the membership literals which are necessary if we want that x satisfies $\psi_\forall$. Modally this corresponds to the fact that these rules *adjust* the frame in such a way that δ is valid. We know that rule $(3_\exists)$ can be applied using v at most p times, hence we put p occurrences of v in the queue *new*. Its meaning in modal logic's terms is that since there are at most p occurrences of $\Box$ in γ, we need to consider at most p immediate successors of v. At each iteration of the repeat-loop at most a new variable u is added. We *initialize* the values EC_u and NC_u and we add p occurrences of u in the queue *new*; the $\langle v, i\rangle$ which is taken out from the queue *new* is put in the queue *old*. We will explain later the meaning of the two commented line in the procedure: they will ensure the termination.

The rules in $\exists$-reductions, applied by the procedure apply-$\exists$, build the solution adding new elements (variables) when it is necessary. Moreover, they establish to which subterms of $t(x, \vec{z})$ these elements (variables) have to belong.

The rule in $\forall$-reductions, applied by the procedure apply-$\forall$, ensures that all the new elements of x satisfy the universal conditions which are in $\psi_\forall$.

The rules in general-reductions, applied by the procedure apply-gen, determine whether there is a contradiction in the formula. Such a contradiction could derive from a wrong guess of the sets EC's and NC's or from the fact that the formula is not satisfiable. Intuitively, since all the possible choices are taken into consideration, if all the non-deterministic branches of Check(ψ) terminates with $\psi = \bot$, then the formula ψ is not satisfiable.

```
apply-gen(ψ):
for all ℓ subformulae of ψ do
    if there is i ∈ {1,2}, and σ such that ℓ = l_i[σ] then
        ψ := ⊥.
return ψ.
```

Figure 4.7: The Procedure apply-gen.

```
Check(ψ(x, z⃗, v⃗)):
V := {v | v ∈ v⃗};
T := {t' | t(x, z⃗) term in ψ_term, t' subterm of t(x, z⃗) or t' ∈ V ∪ {x}};
p := |T|;
let ord be an ordering on T;
new := [];
old := [];    % only for termination
Inf := ⊥;    % only for termination
for all v ∈ V do    % only for termination
    E_v := {t' | t' ∈ T and v ∈ t' is a subformula of ψ_term};
    N_v := {t' | t' ∈ T and v ∉ t' is a subformula of ψ_term};
    let ⟨EC_v, NC_v⟩ such that
        EC_v ⊇ E_v and NC_v ⊆ N_v and part(EC_v, NC_v, T);
    ψ := ψ ∧ ⋀_{t'∈EC_v} v ∈ t' ∧ ⋀_{t'∈NC_v} v ∉ t'
    new := append(new, [⟨v,1⟩, ..., ⟨v,p⟩]);
repeat
    φ := ψ;
    ⟨v,i⟩ := head(new);
    new := cons(new);
    old := append(old, [⟨v,i⟩]);    % only for termination
    apply-∃(ψ, v, ord(i));
    apply-∀(ψ);
    if u ∈ FV(ψ) \ FV(φ) then    % only for termination
        E_u := {t' | t' ∈ T and u ∈ t' is a subformula of ψ_term};
        N_u := {t' | t' ∈ T and u ∉ t' is a subformula of ψ_term};
        let ⟨EC_u, NC_u⟩ such that
            EC_u ⊇ E_u and NC_u ⊆ N_u and part(EC_u, NC_u, T);
        ψ := ψ ∧ ⋀_{t'∈EC_u} u ∈ t' ∧ ⋀_{t'∈NC_u} u ∉ t'
        new := append(new, [⟨u,1⟩, ..., ⟨u,p⟩]);
    apply-gen(ψ);
%   if {w | ⟨w,j⟩ ∈ new} ∩ V = ∅ then
%       if {EC_w | ⟨w,j⟩ ∈ old} ⊇ {EC_w | ⟨w,j⟩ ∈ new} and i = p then
%           Inf := ⊤;
until φ = ψ ∨ ψ = ⊥ ∨ Inf.
```

Figure 4.8: The Procedure Check.

Lemma 4.8 (Completeness) *If all the non-deterministic branches of the* Check *procedure on input ψ terminate with $\psi = \bot$, then ψ is not satisfiable in any model of Ω.*

Proof. We prove that if ψ is satisfiable in a model of Ω, then we are able to apply Check on ψ with only one iteration of the repeat-loop obtaining a formula ψ', different from $\perp$ and which is satisfiable in a model of Ω. Let M be the model of Ω in which ψ is satisfiable. Let σ be the valuation which satisfies ψ in Ω. For all the variable $v \in V$ we put $EC_v = \{t' \mid t' \in T \text{ and } \sigma(v) \in \sigma(t')\}$ and $NC_v = T \setminus EC_v$. Each time that a rule in $\exists$-$\mathsf{reductions}$ or $\forall$-$\mathsf{reductions}$ requires to choose a literal among $r_1, \dots, r_n$ and to add it to ψ, we choose a literal r_j such that σ satisfies r_j. Since all the rules are theorems in Ω and σ satisfies ψ it cannot be the case that such a literal does not exist. Let u be the variable we have added using rule $(3_\exists)$ on v and $\wp(t_1)$. In ψ we have $v \notin \wp(t_1)$, hence there exists u' in $\sigma(v) \setminus \sigma(t_1)$. Let $EC_u = \{t' \mid t' \in T \text{ and } \sigma(u') \in \sigma(t')\}$ and $NC_u = T \setminus EC_u$. The model M and the valuation σ still satisfy the formula ψ' we have obtained adding all these literals to ψ. Hence it cannot be the case that using the $\mathsf{general}$-$\mathsf{reductions}$ we obtain $\perp$. Since ψ' is satisfiable we can use the same reasoning and apply to it one iteration of the repeat-loop, obtaining a formula ψ'' different from $\perp$ and satisfiable.

We have proved that if ψ is satisfiable we are able to produce a branch of $\mathsf{Check}(\psi)$ which does not terminate with $\psi = \perp$, i.e., the thesis. $\square$

The third rule of $\exists$-$\mathsf{reductions}$ introduces a new variable, hence one, or more, of the non-deterministic branches of $\mathsf{Check}(\psi)$ could not terminate. This happens, for instance, when we are dealing with a formula ψ of $\mathcal{S}$ which has only solutions with an infinite number of elements.

From a set-theoretic point of view, recalling that we are not assuming Extensionality and that the Regularity axiom has not been replaced by any form of anti-foundation, we start by showing how to build a solution in the case that one of the possible non-deterministic branches of $\mathsf{Check}(\psi)$ terminates with $\psi \neq \perp$ or does not terminates. Let us use the term *infinitary formula* [2] to refer to a formula in our language in which we allow conjunctions of infinitely many literals. We extend the class $\mathcal{S}$ to the class $\mathcal{S}_\omega$ of generalized formulae in which we allow an infinite number of $\vec{v}$ variables. In particular $\psi^\omega(x, \vec{z}, \vec{v})$ is in $\mathcal{S}_\omega$ if and only if $\psi^\omega(x, \vec{z}, \vec{v})$ is a generalized formula of the form

$$\psi^\omega_\forall(x, \vec{v}) \wedge x \subseteq \wp(x) \wedge \psi^\omega_{\mathsf{term}}(x, \vec{z}, \vec{v}) \wedge \psi^\omega_{\mathsf{var}}(\vec{v}),$$

where $\psi^\omega_{\mathsf{var}}(\vec{v})$ and $\psi^\omega_{\mathsf{term}}(x, \vec{z}, \vec{v})$ can be conjunctions of ω literals, provided that there are only a finite number of variables in $\vec{z}$.

Consider the formula ψ^ω obtained after at most ω iterations of the repeat-loop of $\mathsf{Check}(\psi)$. Each model of Ω in which ψ^ω is satisfied is also a model of Ω in which ψ is satisfied, since ψ is a subformula of ψ^ω.

Let $G(\psi) = \langle N(\psi), \gtrdot(\psi) \rangle$ be the following graph:

$$\begin{array}{rcl} N(\psi) & = & FV(\psi^\omega) \\ \gtrdot(\psi) & = & \{u_2 \lessdot u_1 \mid u_2 \in u_1 \text{ is an atom in } \psi^\omega\}. \end{array}$$

Let $G_\omega(\psi)$ be a model of Ω obtained from $G(\psi)$ as described in Section 4.3.

Lemma 4.9 *All the literals that are in $\psi^\omega \setminus \psi_\forall$ are in one of the following forms:*

- $v = v$, *with* $v \in \mathcal{V}$;
- $v_1 \neq v_2$, *with* $\{v_1, v_2\} \subseteq \mathcal{V}$ *and* $v_1 \not\equiv v_2$;

[2] The use of an infinitary language here is justified by convenience in dealing with quantifiers only, being easily eliminable from our argument.

- $v \in s$, *with* $v \in \mathcal{V}$;
- $v \notin s$, *with* $v \in \mathcal{V}$;
- $x \subseteq \wp(x)$.

Proof. It is only the application of a rule in $\forall$-reductions which can introduce a literal of the form $v_1 = v_2$ with $\{v_1, v_2\} \subseteq \mathcal{V}$. When this is the case, the last for all instruction in apply-$\forall$ performs a substitution of all the occurrences of v_1 with v_2, hence the literal is mapped into $v_2 = v_2$.

Literals of the form $s_1 \neq s_2$ are generated only by $\forall$-reductions, hence it must be the case that both s_1 and s_2 are variable. The second rule in general-reductions ensures that if there is a literal of the form $v_1 \neq v_2$, then v_1 and v_2 are distinct.

It is also easy to see that the l.h.s. in the membership literals are always variables. □

The above result is nothing but a formal proof of the fact that the rules in Check maps formulae of the class $\mathcal{S}$ into formulae of the class $\mathcal{S}$ (formulae of the class $\mathcal{S}_\omega$, if we allow infinite branches).

Lemma 4.10 *$G_\omega(\psi)$ is a model of Ω and a model of ψ^ω.*

Proof. The fact that $G_\omega(\psi)$ is a model of Ω is a consequence of its definition, see Lemma 4.4.

To prove that $G_\omega(\psi)$ is a model of ψ^ω we consider the evaluation assigning to each free variable v in ψ^ω the node v of $G(\psi)$.

First of all we prove that all the literals which are in $\psi^\omega \setminus \psi_\forall$ are satisfied.

The literals of the form $v = v$ are tautologies.

All the literals of the form $v_1 \neq v_2$ are satisfied since v_1 and v_2 are distinct nodes in $G(\psi)$.

As far as the literals of the forms $v \in s$ and $v \notin s$ are concerned we proceed by induction on $|s|$. If s is a variable, then the result is a consequence of the fact that all the outgoing edges from s which are in $G_\omega(\psi)$ are also in $G(\psi)$ and vice-versa (Lemma 4.4). If $s = s_1 \cup s_2$ and the literal is $v \in s$, then in ψ^ω there is also $v \in s_1$ (or $v \in s_2$) and, by inductive hypothesis, this is satisfied. Hence the thesis follows since $G_\omega(\psi)$ is a model of Ω. In a similar way we obtain the thesis when $s = s_1 \cup s_2$ and the literal is $v \notin s$. If $s = s_1 \setminus s_2$ and the literal is $v \in s$, then in ψ^ω there are also the literal $v \in s_1$ and $v \notin s_2$ and these are satisfied by inductive hypothesis, hence we are done. The case in which the literal is $v \notin s$ is similar. If $s = \wp(s_1)$ and the literal is $v \in s$, then for all the literals $v_1 \in v$ in ψ^ω we have added $v_1 \in s_1$, hence v is a subset of s_1, i.e. it is one of the elements of s. If $s = \wp(s_1)$ and the literal is $v \notin s$, then in ψ^ω there are two literals of the form $v_1 \in v$ and $v_1 \notin s_1$, these are satisfied by inductive hypothesis, hence we have the thesis.

The literal $x \subseteq \wp(x)$ is satisfied, since we rule $(3_\exists)$ ensures it.

From Lemma 4.9 we know that these are all the possible cases.

Now we have to prove that the universally quantified part, $\psi^\omega_\forall$, of ψ^ω is satisfied. We have that $\psi_\forall = \psi^\omega_\forall$. In $G(\psi)$ the elements of x are only nodes associated to free variables of ψ^ω. All the literals that are in ψ^ω, containing these variables are satisfied (as we have proved above). The rule in $\forall$-reductions ensures that from these two fact we have the thesis. □

Theorem 4.11 *If there exists a non-deterministic branch of* Check(ψ) *which terminates with $\psi \neq \bot$ or which does not terminate, then ψ is satisfiable in a model of Ω.*

Proof. It is a consequence of Lemma 4.10. $\square$

Corollary 4.12 *A formula ψ in the class $\mathcal{S}$ is satisfiable in a model of Ω if and only if not all the non-deterministic branches of* $\mathsf{Check}(\psi)$ *terminates with $\psi = \perp$.*

Proof. The left to right direction is a consequence of Lemma 4.8, the right to left direction is a consequence of Theorem 4.11. $\square$

Notice that in the case Check terminates after a finite number of iterations with $\psi \neq \perp$, we have implicitly proved that ψ has a finite model. In this case the whole model $G_\omega(\psi)$ is infinite but the part of model (the graph) used to give a solution is finite, since in a finite number of iterations we can only add a finite number of new variables.

There are formulae in the class $\mathcal{S}_{\exists\forall}$ which admits only infinite models. Consider for instance the formula

$$x \subseteq \wp(x) \wedge \forall y \in x(y \notin y) \wedge \forall y_1, y_2, y_3 \in x((y_1 \in y_2 \wedge y_2 \in y_3) \rightarrow y_1 \in y_3) \wedge \\ w \in x \wedge w \notin (x \setminus \wp(x \setminus \wp(x \setminus x))) \cup \wp(x \setminus x).$$

It has only infinite models since: the universal quantified part of the formula impose that all the elements of x are irreflexive and transitive, and this implies that there cannot be cycles of memberships; the literals on w imply that $w \in \wp(x \setminus \wp(x \setminus x)))$ and $w \notin \wp(x \setminus x)$; from the second we obtain that there exists $w_1 \in w$ such that $w_1 \notin x \setminus x$; using the first on w_1 we obtain that $w_1 \in x \setminus \wp(x \setminus x)$, hence $w_1 \notin \wp(x \setminus x)$; this last implies that there exists $w_2 \in w_1$ such that $w_2 \notin x \setminus x$; from the transitivity we obtain that $w_2 \in w$, hence using a similar reasoning we generate $w_3 \in w_2$; and so on. This means that in order to satisfy the formula x has to include at least one infinite descending chain. Notice that this formula came from modal logic, in particular it means that we are looking for a transitive irreflexive frame in which the formula

$$\Diamond\Box\perp \vee \Box\perp$$

is not valid.

4.6 Correctness and Termination of TCheck

In the case that $\psi_\forall$ is in the class $\forall^2$ with no positive literals of equality (we use $\mathcal{S}^2_{no=}$ to refer to this class) we modify the procedure Check in order to obtain a procedure TCheck which decides the satisfiability problem. This means that in this particular case we are able to add a termination condition which ensures the existence of a solution. The procedure TCheck is obtained from Check by simply removing the comment symbols % at the beginning of the last three lines of the repeat-loop.

Lemma 4.13 *Let ψ be a formula in $\mathcal{S}$ such that in $\psi_\forall$ there are no literals of the form $y_h = y_k, y_h = v_k, v_h = y_k$, and let ψ^ω be a formula obtained after at most ω iterations of* $\mathsf{Check}(\psi)$. *In ψ^ω there are no literals of the form $v = v$.*

Proof. It is possible to obtain a literal of the form $v = v$ only if it is possible to obtain a literal of the form $v_1 = v_2$. This is not possible when we assume that in $\psi_\forall$ there are no equalities. $\square$

Lemma 4.14 (Termination) *Let ψ be a formula in $\mathcal{S}^2_{no=}$. The procedure* TCheck *on input ψ always terminates.*

Proof. Since ψ is in $\mathcal{S}^2_{no=}$, from Lemma 4.13, we have that no equalities between new variables and variables in V are generated. Hence, each variable in V is never added to *new* more than p times, so after $p * |V|$ iterations of the repeat-loop it is always true that $\{w \mid \langle w, j\rangle \in new\} \cap V = \emptyset$ (this is the first condition we add in TCheck). Let ψ be a formula in $\mathcal{S}$, and ψ_i be the formula obtained after the i-th iteration of the repeat-loop of TCheck(ψ), with $i > p * |V|$. We say that two variables u and v in ψ_i are *similar*, $u \sim v$ if $EC_u = EC_v$ (hence, $NC_u = NC_v$). The set T (defined in the first line of Check) is a finite set. Let $t = |T|$. There are at most 2^t possible choices for EC_v. So there are at most 2^t equivalence classes in $\sim$.

From Lemma 4.13 we have that variable is never put in the queue *new* more than p times.

Since there are a finite number of blocks, after a finite number of iterations we have that there exists a set S of blocks such that:

- if $s \in S$, then there is at least a variable v_s in s such that $\langle v_s, 1\rangle, \ldots, \langle v_s, p\rangle$ are in the queue *out*;
- all the variables in the queue *new* belong to a block s of S;
- the head of *new* is of the form $\langle v, 1\rangle$.

When this situation arises we assign $\top$ to the variable Inf. Hence after a finite number of iterations TCheck terminates. $\square$

Lemma 4.15 *If there exists a non-deterministic branches of* TCheck(ψ) *which terminates with $Inf = \bot$ and which maps the formula ψ into the formula ψ', then there exists a non-deterministic branch of* Check(ψ) *which terminates and maps the formula ψ into the formula ψ'.*

Proof. It is a consequence of the fact that if $Inf = \bot$ at the end of a non-deterministic branch of TCheck, then this branch is equivalent to a branch of Check. $\square$

Lemma 4.16 *If there exists a non-deterministic branch of* TCheck(ψ) *which terminates with $Inf = \top$, then there exists a branch of* Check(ψ) *which does not terminate.*

Proof. Let us assume that we are considering a non-deterministic branch of TCheck(ψ) which terminates with $Inf := \top$ after k iterations of the repeat-loop. We have to exhibit a branch of Check(ψ) which does not terminate.

It is immediate to observe that the branch of TCheck(ψ) coincide with an initial branch of Check(ψ) in which we are at the end of the k-th iteration of the repeat-loop. We show that we are able to extend this branch of Check(ψ) adding a further iteration of the repeat-loop at the end of which ψ is not equal to $\bot$, and $\{EC_w \mid \langle w, j\rangle \in old\} \supseteq \{EC_w \mid \langle w, j\rangle \in new\}$. We do not use the fact that the head of *new* is of the form $\langle v, 1\rangle$, so that we can immediately generalize this result.

Let $\langle v, j\rangle$ be the head of the queue *new* at the $(k+1)$-th iteration. There exists a variable w such that $EC_w = EC_v$, $NC_w = EC_v$ and all the pairs $\langle w, i\rangle$, with $1 \leq i \leq p$ are in *old*.

Let w_j be the element of w introduced when $\langle w, j\rangle$ has been processed, i.e. applying the rule $(3_\exists)$ on w and t_j. We introduce the new variable v_j applying the rule $(3_\exists)$ on v and t_j. We are sure that this is possible since $NC_w = NC_v$.

We choose $EC_{v_j} = EC_{w_j}$ and $NC_{v_j} = NC_{w_j}$.
Let $\varphi = \psi$. We have to choose the literals we add to ψ, then we have to prove that our choice coincides with one of the possible executions of one iteration of the repeat-loop on φ, and that we have not obtained $\perp$. Let $W = (FV(\psi) \setminus \{\vec{z}, x\}) \cup \{v_j\}$. Let $\sigma : W \mapsto W$ defined as

$$\begin{array}{lcll} \sigma(v_j) & = & w_j & \\ \sigma(v) & = & w & \\ \sigma(u) & = & u & \text{otherwise} \end{array}$$

We add the following literals to ψ:

- $v_j \in u$ iff $\sigma(v_j) \in \sigma(u)$ is in φ;
- $v_j \notin u$ iff $\sigma(v_j) \notin \sigma(u)$ is in φ;
- $v_j \ni u$ iff $\sigma(v_j) \ni \sigma(u)$ is in φ;
- $v_j \not\ni u$ iff $\sigma(v_j) \not\ni \sigma(u)$ is in φ;
- $v_j \neq u$ iff $\sigma(v_j) \neq \sigma(u)$ is in φ;
- $v_j \neq u$ iff $\sigma(v_j) \neq \sigma(u)$ is in φ;

We also add in ψ all the literals of the form $v_j \in t$ such that $t \in EC_{v_j}$ and $v_j \notin t$ such that $t \in NC_{v_j}$.

First, we prove that the formula ψ can be obtained applying one iteration of the repeat-loop to the formula φ. From our definitions of the procedures apply-$\forall$, apply-$\exists$, apply-gen we have that we can only add to φ literals which involve v_j. This is a consequence of the fact that v_j is the only new variable we add, and all the literals which could be added concerning the old variables are already in φ.

It cannot be the case that we have not added a literals of the form $v_j \in t(x, \vec{z})$ or $v_j \notin t(x, \vec{z})$, since $EC_{v_j} = EC_{w_j}$ and (EC_{v_j}, N_{v_j}) is a partition of T. Hence we have added all the literals added in a non-deterministic branch of apply-$\exists$.

Let us assume by contradiction that there is a subformula of ψ of the form $\forall y_1, y_2 \in x(p_1 \vee \ldots \vee p_m)$ such that there exist u_1 and u_2 in ψ and no one of the $p_a(u_1, u_2)$'s is in ψ. If both u_1 and u_2 are different from v_j we have a contradiction, since on u_1 and u_2 we add already applied apply-$\forall$. Without loosing generality we can assume that $u_1 = v_j$. If $u_2 = v_j$, then there is an index a such that $p_a(w_j, w_j) = p_a(\sigma(v_j), \sigma(v_j))$ is in φ, hence $p_a(v_j, v_j)$ is in ψ. If $u_2 = w_j$, then there is an index a such that $p_a(w_j, w_j) = p_a(\sigma(v_j), \sigma(w_j))$ is in φ, hence $p_a(v_j, w_j)$ is in ψ. If $u_2 = v$, then there is an index a such that $p_a(w_j, w) = p_a(\sigma(v_j), \sigma(v))$ is in φ, hence $p_a(v_j, v)$ is in ψ. If $u_2 = w$, then there is an index a such that $p_a(w_j, w) = p_a(\sigma(v_j), \sigma(w))$ is in φ, hence $p_a(v_j, w)$ is in ψ. If $u_2 \neq v_j, w_j, v, w$, then is an index a such that $p_a(w_j, u_2) = p_a(\sigma(v_j), \sigma(u_2))$ is in φ, hence $p_a(v_j, u_2)$ is in ψ. In all the possible cases we obtain a contradiction, hence the thesis follows.

Now we have to prove that it cannot be the case that we have introduced a pair of literals of the form

$$v_j \in s \wedge v_j \notin s \text{ or } s \in v_j \wedge s \notin v_j \text{ or } v_j \neq v_j.$$

Let us assume we have introduced a pair of literals of the form $v_j \in s \wedge v_j \notin s$, then using σ we obtain: if $s = v_j$, this would imply that in φ we have $w_j \in w_j \wedge w_j \notin w_j$; if $s = v$, this would imply that in φ we have $w_j \in w \wedge w_j \notin w$; if $s = w$, in φ we have $w_j \in w \wedge w_j \notin w$; if $s = w_j$ this would imply that in φ we have $w_j \in w_j \wedge w_j \notin w_j$; if $s \neq v_j, w_j, v, w$, this would imply that in φ we have $w_j \in s \wedge w_j \notin s$. In all the possible cases we obtain that there is a contradiction in φ, which is not possible. A similar argument can be used in the other cases.

At the end of iteration k we had $Inf := \top$, hence there is a variable h_j such that $h_j \sim w_j$ and $\langle h_j, 1\rangle, \dots \langle h_j, p\rangle$ are in *old*. Since $v_j \sim w_j$, we have that $h_j \sim v_j$. So, we have obtained also $\{EC_w \mid \langle w, j\rangle \in old\} \supseteq \{EC_w \mid \langle w, j\rangle \in new\}$.

Hence, we have obtained an initial branch of $\mathsf{Check}(\psi)$ with $(k+1)$ iteration of the repeat-loop, at the end of which ψ is different from $\perp$ and $\{EC_w \mid \langle w, j\rangle \in old\} \supseteq \{EC_w \mid \langle w, j\rangle \in new\}$. By induction, if we assume that we have extended this branch till the h-th iteration with $h > k$ we can prove, using the same argument (we have not used $\mathsf{head}(new) = \langle v_j, 1\rangle$), that we are able to extend it with one more iteration. This implies that we are able to build a branch of $\mathsf{Check}(\psi)$ which does not terminates. □

Lemma 4.17 *If all the non-deterministic branches of* TCheck *map* ψ *into* $\perp$*, then all the non-deterministic branches of* Check *terminate and map* ψ *into* $\perp$*.*

Proof. It is an immediate consequence of the way we obtain TCheck from Check. □

Theorem 4.18 (Correctness and Completeness) *Let* ψ *be a formula in the class* $\mathcal{S}^2_{no=}$*. ψ is satisfiable in a model of Ω if and only if there exists a non-deterministic branch of* TCheck *at the end of which* $\psi \neq \perp$*. Moreover, if there exists a non-deterministic branch of* TCheck *at the end of which* $Inf = \perp$ *and* $\psi \neq \perp$*, then* ψ *is finitely satisfiable in a model of* Ω*.*

Proof. For the first part of the theorem, we prove that ψ is not satisfiable in any model of Ω if and only if all the non-deterministic branches of TCheck map ψ into $\perp$. From Theorem 4.11 we have that if ψ is not satisfiable in any model of Ω, then all the non-deterministic branches of Check terminate and map ψ into $\perp$. Hence, from Lemma 4.16 it cannot be the case that there exists a branch of TCheck which terminates with $Inf = \top$. So from Lemma 4.15 and from the fact that all the non-deterministic branches of Check terminate mapping ψ into $\perp$, we obtain that all the non-deterministic branches of TCheck map ψ into $\perp$. If all the non-deterministic branches of TCheck map ψ into $\perp$, then from Lemma 4.17 we obtain that all the non-deterministic branches of Check terminate and map ψ into $\perp$. Hence from Lemma 4.8 we have that ψ is not satisfiable in any model of Ω.

The second part of the theorem is a consequence of Lemma 4.15 and of the observation above Theorem 4.11. □

Notice that in order to force termination in this particular case we check to which subsets of x the new variables we are adding belong. When we are in a situation in which all the variables we have to process present a situation similar to variables already processed we are able to generate an infinite branch. This is similar to what we do in Chapter 3, where we analyse what we call *pictures*. But in that case from the existence of an infinite branch we are not able to deduce the existence of a solution. That problem there is mainly caused by the presence of the $\wp$ operator on the l.h.s. of inclusions. This never arises when the formulae we consider are translations of modal formulae.

Notice that we have not proved the small model property for the class $\mathcal{S}^2_{no=}$. Our conjecture is that the property holds because of two reasons: the most simple formula that we are able to exhibit which has only infinite models (showed at the end of the previous section) involves 3 universal quantifiers; our proof of Lemma 4.16 suggests also a way to quotient an infinite model.

4.7 Modal Consequences

Definition 4.19 *Let Λ be a first-order complete modal logic with first-order correspondent δ. If δ is of the form $\exists^*\forall^2\alpha$ such that α is in conjunctive normal form and there are not positive literals of equality in α, then we say that Λ is an* $\exists^*\forall^2$-equality-free *modal logic.*

Corollary 4.20 *If Λ is an $\exists^*\forall^2$-equality-free modal logic, then Λ is decidable.*

Some $\exists^*\forall^2$-equality-free modal logics are K, T, B.

In the case that Λ is a first-order complete modal logic with first-order correspondent δ in the class $\exists^*\forall^*$ it is possible to use the procedure Check to test $\vdash_\Lambda \gamma$. In the way that the procedure is presented here it is possible that it does not terminate. However, what is interesting is that it allows to deal in a uniform way with a large class of modal logics: we do not have to define a set of tableau rules for each modal logic; there is only one rule, the rule $(3_\exists)$, which takes care of the introduction of new worlds; while the rule in $\forall$-reduction manages the *encoding* of the axioms of the modal logic. Some famous modal logics in this class are $K4$, $S4$, $S5$, $K4.3$, $S4.3$ (no branching to the right).

4.8 Concluding Remarks

The results presented in this chapter go towards establishing a strong link between decision problems in Set Theory and Modal Logic. The two main tools used are the $\Box$-as-$\mathcal{P}$ translation method and the tableau technique, intended as a general tool for coordinating a semantic and syntactic interplay in the analysis of a formula. The set-theoretic approach, on the one hand, guarantees an entirely uniform point of view on modal logics characterized via Hilbert's axioms. Such a point of view, on the other hand, allows a tableau-like analysis that, abstracting from the specific modal logic under consideration concentrates on the model construction in completely general and intuitive (set-theoretic) terms. In this respect consider, for example, the fact that the only rule introducing new variables in our procedure is $(3_\exists)$.

The decidability result presented here is an example of the possible fruitfulness of the approach and we are convinced that such a result can be generalized to the entire class of modal logic with first-order correspondent of the form $\exists^*\forall^*$. Major stumbling block towards this achievement is the specification of the correct termination condition of the tableau procedure presented in Section 4.5. In the determination of such a termination condition we are sure that we cannot exploit the small model property since we showed an example coming from modal logic which has only infinite solutions.

II

Quotients of Non-Well-Founded Sets

Introduction to Part II

It seemed that in order to understand common knowledge (a crucial feature of communication), circular propositions, various aspects of perceptual knowledge and selfawareness [...] we either had to give up the tools of set theory which are so well loved in mathematical logic, or we had to enrich the conception of set, finding one that admits of circular sets, at least. (J. Barwise)

In this part of the thesis we address the algorithmic problem of finding the minimal representation of a set in a non-well-founded set theory in which the anti-foundation axiom (AFA) (cf. [Acz88, FH83] and Chapter 3) is assumed. The Extensionality axiom—saying that two objects are equal if and only if they contain exactly the same elements—is the standard criterion for establishing equality between well-founded sets. If Extensionality is assumed it is immediate to see that, for example, different acyclic graphs can represent the same set. However, Extensionality leads to a cyclic argument (no wonder!) whenever one tries to apply it as a test to establish whether two cyclic graphs represent the same non-well-founded set.

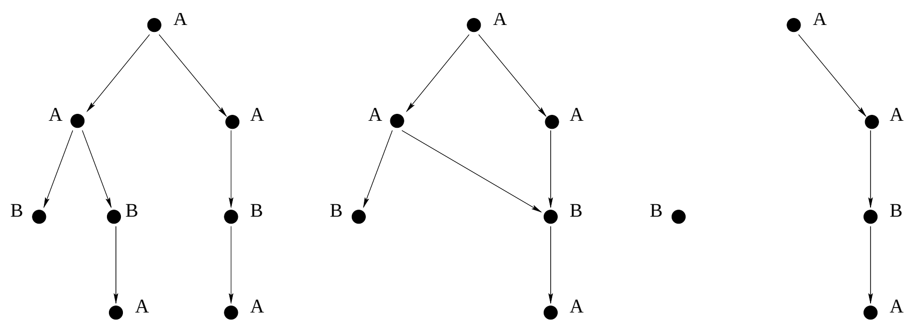
Example 1 *Let a and b be the two non-well-founded sets represented in the figure below*

Using Extensionality it is not possible to prove that $a = b$, since we obtain

$$a = b \leftrightarrow a = b.$$

To this end a condition (*bisimulation*) on graphs can be stated in accordance with Extensionality: two graphs are bisimilar if and only if they are representations of the same set. In particular bisimulation is the operational counter-part to the anti-foundation axiom (AFA).

We also consider the problem of computing the minimum representation of a non-well-founded set w.r.t. the maximum simulation relation, i.e. the *Maximum Simulation Problem.* The definition of simulation is very similar to the one of bisimulation and we start to study the problem in order to analyse if the same circle of ideas we apply to bisimulation were applicable to simulation.



Labelled Graph Bisimulation Simulation

It is difficult to accurately list all the fields in which, in one form or another, the notion of *bisimulation* was introduced and now plays a central role. Among the most important ones are: Modal Logic, Concurrency Theory, Formal Verification, and Set Theory. In Modal Logic this notion was introduced by van Benthem (cf. [Ben76]) as an equivalence principle between Kripke structures. In Concurrency Theory it was introduced by Milner and Park for testing observational equivalence of the Calculus of Communicating Systems (CCS). In particular in [Par81] a previous notion of automata simulation by Milner is refined in the context of omega regular languages for concurrency. In [Mil80] (weak and strong) bisimulation is proposed. In Set Theory, it was introduced by Forti and Honsell (cf. [FH83]) as a natural principle replacing Extensionality in the context of non-well-founded sets. We already met in Chapter 3 the anti-foundation axiom (AFA), which is the axiomatic counter-part of the notion of bisimulation. Other examples of scenarios in which the problem of finding the minimal representation of a set of cyclic graphs is particular relevant can be found in the notion of rational tree which is at the basis of Prolog III (see [Col90]), and in the problem of representing analysis of terms through type graphs (see [CvHC95]).

A great interest in the notion of bisimulation cames from the area of Model Checking. The main disadvantage of Model Checking is the so-called *state explosion* that can occur if the system being verified has many components that can make transitions in parallel. In this case the number of global system states may grow exponentially with the number of processes. Different techniques have been developed and are still under active investigation to solve this problem: OBDD's and Symbolic Model Checking [McM93], Abstract Model Checking [Dam96], partial order reductions [CGP99], equivalence reductions [LY92, HHK95]. Several existing verification tools make use of bisimulation in order to minimize the state spaces of systems description. Moreover, bisimulation serves as a means of checking equivalence between transition systems. The verification environment XEVE [Bou98] provides bisimulation tools which can be used for both minimization and equivalence test. In general, in the case of explicit-state representation, the underlying algorithm used is the one proposed by Kanellakis and Smolka [KS90], while Bouali and de Simone algorithm [BdS92] is used in the case of symbolic representation. As for the criticism on the use of bisimulation algorithms in Model Checking[3], Fisler and Vardi observe in [FV99] that *bisimulation minimization does not appear to be viable in the context of invariance verification*, but in the context of compositional verification it *makes certain problems tractable that would not be so without minimization* [ASSB94, Rah98].

[3]Model Checking for CTL is linear ($O(|\geqslant| + |N|)$, while the bisimulation computation is more expensive ($O(|\geqslant| \log |N|)$).

The notion of simulation, which is very similar (less demanding, in fact) to the notion of bisimulation, is also largely used in Model Checking to reduce the number of states in the model of the system: it preserves the universal fragment of CTL* and it reduces the state spaces more than bisimulation. The most famous algorithms used in model checkers to test simulation are the one proposed in [HHK95] and the one of [BG00]. The similarity between the definition of simulation and the definition of bisimulation in particular attract our attention to the study of simulation in the context of non-well-founded sets.

As far as the algorithmic results present in the literature for the solution of the bisimulation problem are concerned we can say that the first significant result is in [Hop71], where Hopcroft presents an algorithm for the minimization of the number of states in a given finite-state automaton. The problem is equivalent to that of determining the coarsest partition of a set *stable* with respect to a finite set of functions. A variant of this problem is studied in [PTB85], where it is shown how to solve it in linear time in case of a single function. Finally, in [PT87] Paige and Tarjan solved the problem for the general case (i.e., bisimulation) in which the stability requirement is relative to a relation $\Rightarrow$ (on a set N) with an algorithm whose complexity is $O(|\Rightarrow| \log |N|)$.

The main feature of the linear solution to the single function coarsest partition problem (cf. [PTB85]), is the use of a *positive* strategy in the search for the coarsest partition: the starting partition is the partition with singleton classes and the output is built via a sequence of steps in which two or more classes are merged. Instead, Hopcroft's solution to the (more difficult) many functions coarsest partition problem is based on a (somehow more natural) *negative* strategy: the starting partition is the input partition and each step consists of the split of all those classes for which the stability constraint is not satisfied. The interesting feature of Hopcroft's algorithm lies in its use of a clever ordering (the so-called *process the smallest half* ordering) for processing classes that must be used in a split step. Starting from an adaptation of Hopcroft's idea to the relational coarsest partition problem, Paige and Tarjan succeeded in obtaining their fast solution [PT87]. The algorithm presented in [BFH90] by Bouajjani, Fernandez, and Halbwachs is based on the naïve negative strategy, but it is tailored for on-the-fly model checkers, since on each iteration it stabilizes only reachable blocks with respect to all blocks. This is improved by Lee and Yannakakis in [LY92], where only reachable blocks are stabilized with respect to reachable blocks only.

As far as the simulation problem is concerned several polynomial-time algorithms for computing similarity quotients have been proposed: the ones presented in [Blo89], [CPS93], and [CS92] achieve time complexities of the orders $O(|N|^6 |\Rightarrow|)$, $O(|N|^4 |\Rightarrow|)$, and $O(|\Rightarrow|^2)$, respectively. The best time complexity for algorithms computing the maximal simulation is the one presented in [HHK95] and [BP95] ($O(|N||\Rightarrow|)$).

All of the algorithms just mentioned ([Blo89], [CPS93], [CS92], [BP95], [HHK95]) obtain the similarity quotient as a by-product of the computation of the entire similarity relation on the set of states N. Their space complexity is then limited from below by $O(|N|^2)$.

Recently Bustan and Grumberg proposed an algorithm which gives in output the quotient structure with respect to the maximum simulation relation $\equiv_s$ and the simulation relation among the classes of $M = N/_{\equiv_s}$, without computing the entire simulation on N. The space requirements (often more critical, especially in the field of verification) of this algorithm depend on the size of M and are lower than the ones of the algorithms in [Blo89], [CPS93],

[CS92], [BP95], and [HHK95]. In more detail, the so-called *Partitioning Algorithm* described in [BG00] uses only $|M|^2 + |N|\log(|M|)$ space. The time complexity of the *Partitioning Algorithm* is instead rather heavy: it is $O(|M|^4(|\geqslant| + |M|^2) + |M|^2|N|(|N| + |M|^2))$.

Taking the algorithmic results on bisimulation mentioned above as starting points, in Chapter 5 we present a procedure that integrates positive and negative strategies to obtain the algorithmic solution to the bisimulation problem and, hence, to the relational coarsest partition problem. The strategy we develop is driven by the set-theoretic notion of *rank* of a set. The algorithm we propose uses [PTB85] and [PT87] as subroutines and terminates in linear time in many cases, for example when the input problem corresponds to a bisimulation problem on acyclic graphs (well-founded sets). It operates in linear time in other cases as well and, in any case, it runs at a complexity less than or equal to that of the algorithm by Paige and Tarjan [PT87]. Moreover, the partition imposed by the rank allows to process the input without storing the entire structure in memory.

The Ordered Binary Decision Diagrams (OBDD's) are the data-structures at the basis of Symbolic Model Checking. In [McM93] McMillan proposed a symbolic technique for model checking which uses OBDD's to represent set of states and fix-point computations to determine the set of states which satisfy the specification. This technique makes possible to model check systems with more than 10^{100} states (this is not possible representing explicitly the set of states). The importance of the symbolic representation of state spaces in Model Checking pushes us, in Chapter 6, to reconsider the problem of computing and representing the bisimulation on a given non-well-founded set from a different point of view: we compute the maximum bisimulation using a positive strategy and obtain a compact OBDD's representation of the minimized structure at the same time. If on the one hand the OBDD data-structures employed in symbolic model checking allow the representation in memory of much larger structures and set new standards for the size of Kripke structures to which Model Checking can be applied, on the other hand, it is not always immediate to map the further space-saving method of bisimulation reduction on OBDD-representation of Kripke structures: especially designed algorithms must be proposed for this purpose and different complexity parameters must be considered [FV99]. The method we propose in Chapter 6 aims at carrying out the activity of bisimulation computation and OBDD-representation in *one single step*. Starting from Ackermann's encoding [Lev79, Ack37] of well-founded sets as natural numbers, we propose an extension of such an encoding to non-well-founded hereditarily finite sets which naturally suggests an OBDD-representation.

In Chapter 7 we analyse which are difficulties that arise if we try to apply an idea similar to the extension of the Ackermann's encoding presented in Chapter 6 to the computation of simulation. Then, we propose an encoding of the simulation problem as constraint over sets. This allows us both to use already existing constraint-solvers to compute the maximum simulation and to exploit the notion of rank in order to perform the computation incrementally. Moreover, we obtain an OBDD-representation similar to the one presented in Chapter 6.

As conclusion of our work, in Chapter 8 we propose a rephrasing of the notion of simulation in purely algebraic terms as for bisimulation: we define the *generalized coarsest partition problem*. On this ground, we propose an algorithm for determining the quotient w.r.t. the maximum simulation that improves (significantly) on either space or time complexity with respect to the best known algorithms for the same problem.

Notice that, since the main application which has attracted our attention in the computation of bisimulation and simulation is Model Checking, and in Model Checking the state spaces are finite, in the rest of this thesis we consider only *finite non-well-founded sets*. Extensions of our work to the infinite case are in progress and found interesting applications in Model Checking of Hybrid Systems, where the state spaces are infinite.

5

Bisimulation by Rank

In this chapter we introduce the notion of bisimulation on rooted graphs which represent non-well-founded sets. The notion of bisimulation is the operational counter-part to the anti-foundation axiom (AFA) which allows to establish equalities between non-well-founded sets. Each graph represents a non-well-founded set, but it can be the case that two different graphs represent the same set. In particular, two graphs represent the same set if and only if they are bisimilar. Given a graph $G = \langle N, \gtrdot, r \rangle$, which represents a set s, in order to find a minimal representation of s (minimal in term of nodes and edges) it is sufficient to compute the maximum bisimulation relation $\equiv$ over G. The relation $\equiv$ is always an equivalence relation and the quotient graph $G/\equiv$ is the minimum representation of s. With the terminology *Bisimulation Problem* we refer to the problem of finding the maximum bisimulation relation over a graph: this can be easily used also to test whether two graphs are bisimilar.

The notion of bisimulation can be introduced in the context of graphs with labels on the nodes and on the edges, while we work on unlabelled graphs. Hence, it becomes important to show that each bisimulation problem a graph with labels can be mapped into an equivalent bisimulation problem on a graph without labels: new nodes are introduced in order to remove the labels.

The best algorithms in the literature to compute bisimulations are based on a characterization of the bisimulation problem as relational coarsest stable partition problem: instead of computing the pairs of nodes that are in relation through $\equiv$, the blocks of the partition induced by $\equiv$ on the set of nodes are determined (this requires less space); the *process the smallest half* policy introduced by Paige and Tarjan in [PT87] allows to obtain an algorithm which works in time $O(|\gtrdot| \log |N|)$. The algorithms which determine $G/\equiv$ as solution of a relational coarsest stable partition problem start with all the nodes in a unique block and split a block into two or more parts each time that a condition is not respected: usually, given two blocks B and C, the block C is split into $C_1 = C \cap \lessdot(B)$ and $C_2 = C \setminus \lessdot(B)$.

Starting with this knowledge we analyse how it is possible to use the set-theoretical point of view, and in particular the set-theoretical notion of rank, in the solution of the bisimulation problem. The notion of rank, well known in the well-founded case, suggests in a natural way an initial partition of the set of nodes finer than the ones that are usually used and an order to follow in the choice of the blocks to be used to perform the split operations. In particular this allows in the case of well-founded sets to:

- work during each iteration on a subgraph of the original graph;

- perform the computation in linear time.

Keeping in mind the characteristics of the notion of rank that in the well-founded case allow to obtain such results, we extend the notion of rank to the non-well-founded case, obtaining an algorithm to solve the bisimulation problem based on the notion of rank whose main advantages are that it works on subgraphs and it computes in linear time on some parts of the graph.

Moreover, we discuss some other possible optimization of our rank-based algorithm, which are always suggested from the fact that we use a set-theoretical point of view.

In Section 5.7 we present the results of some testing made to compare the performances of our algorithm with the ones of the algorithm of Paige and Tarjan.

5.1 Bisimulation in a Set-theoretic Perspective

In Chapter 1 we have already explained how graphs represent sets, however we recall here some basic notions from [Acz88].

Definition 5.1 (Rooted Graph) *A (direct)* rooted graph *is a triple $G = \langle N, \gtrdot, r\rangle$, where N is the set of nodes, $\gtrdot \subseteq N \times N$ is the set of edges, and $r \in N$ is a node such that all the other nodes in N are reachable from r, i.e.*[1]*:*

$$\forall n \in N \exists n_1, \ldots, n_h (r \gtrdot n_1 \wedge \ldots \wedge n_h \gtrdot n).$$

Each rooted graph (or graph for short) represents a non-well-founded set in the following sense:

$$\forall m, n \in N(m \in n \leftrightarrow m \lessdot n).$$

In particular, the non-well-founded set represented by a rooted graph is the non-well-founded set represented by the root r. However, each node n of a graph represents a non-well-founded set: consider only the subgraph formed by the nodes reachable from n. Notice also that the root r represents a well-founded set if and only if G is acyclic. In Figure 5.1 we show an example of a non-well-founded set and an example of a well-founded-set.

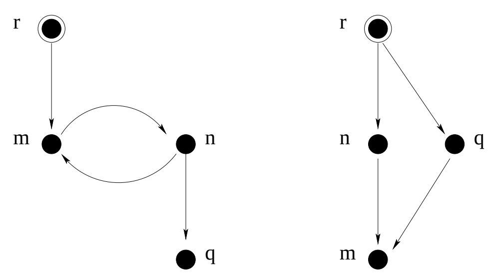


Figure 5.1: Sets as Graphs.

In order to move from the representation of non-well-founded sets as graphs to a representation using systems it is sufficient to:

- assign a variable (name) x_n to each node n of the graph;

[1] In this part of the thesis we always refer to finite graphs.

- if $\{m \mid n \gg m\} = \{m_1, \ldots, m_j\}$, then add to the system the equation

$$x_n = \{x_{m_1}, \ldots, x_{m_j}\}.$$

Each node n represents the non-well-founded set which is assigned to x_n in the unique solution of the system[2].

Example 5.2 *From the non-well-founded set on the left side of Figure 5.1 we obtain the following system*

$$\begin{aligned} x_r &= \{x_m\} \\ x_m &= \{x_n\} \\ x_n &= \{x_m, x_q\} \\ x_q &= \emptyset \end{aligned}$$

The well-founded set on the right side of Figure 5.1 gives us the system

$$\begin{aligned} x_r &= \{x_n, x_q\} \\ x_n &= \{x_m\} \\ x_q &= \{x_m\} \\ x_m &= \emptyset \end{aligned}$$

We are now ready to introduce the notion of bisimulation which allows to both establish equalities between sets and determine a minimum representation for a set.

Definition 5.3 (Bisimulation) *Given $G_1 = \langle N_1, \gg_1, r_1 \rangle$ and $G_2 = \langle N_2, \gg_2, r_2 \rangle$ a relation $B \subseteq N_1 \times N_2$ is a* bisimulation *between G_1 and G_2 if and only if*

(r) $r_1 B r_2$;

(f) $mBn \wedge m \gg_1 m' \Rightarrow \exists n'(n \gg_2 n' \wedge m'Bn')$;

(b) $mBn \wedge n \gg_2 n' \Rightarrow \exists m'(m \gg_1 m' \wedge n'Bm')$.

Given a graph $G = \langle N, \gg \rangle$, a relation B on N is said to be a bisimulation *on G if and only if*

(f) $mBn \wedge m \gg m' \Rightarrow \exists n'(n \gg n' \wedge m'Bn')$;

(b) $mBn \wedge n \gg n' \Rightarrow \exists m'(m \gg m' \wedge n'Bm')$.

In the above definition (r) stands for root, (f) for forward, and (b) for backward.

Given two non-well-founded sets s_1 and s_2 represented by G_1 and G_2, we will check whether they are equal by checking if there exists a bisimulation B between G_1 and G_2. The equalities between non-well-founded sets established using bisimulation still satisfy the Extensionality axiom, but, moreover, they satisfy the anti-foundation axiom (AFA).

The notion of bisimulation on a graph G give us a way to determine the minimum graph (w.r.t. the number of nodes and edges) representing a non-well-founded set. The nodes in such a minimum graph are equivalence classes of nodes of G and the fact that the edges can be put in a *unique* way is ensured by the property $(\flat)$. This is important when we compare the notion of bisimulation and the notion of simulation: in the case of simulation it is not so immediate to determine which is the minimum set of edges (see Chapter 7).

[2]Assuming (AFA) each system has exactly one solution.

Theorem 5.4 (Maximum Bisimulation) *Given a graph G there exists always a maximum bisimulation $\equiv$ on G. The bisimulation $\equiv$ is an equivalence relation.*

Proof. It is immediate to prove that if B_1 and B_2 are two bisimulations over G, then $B_1 \cup B_2$ is a bisimulation over G. It follows that the union of all the bisimulations over G is the maximum bisimulation over G.

From the fact that $\equiv$ is the union of all the bisimulations over G and that the identity relation on N is a bisimulation it follows that $\equiv$ is an equivalence relation. □

Since $\equiv$ is an equivalence relation it is possible to consider the quotient $N/\equiv$, whose elements are the equivalence classes $[n]$ with $n \in N$. An important property of $\equiv$ (and of all the bisimulations that are equivalence relations) is that

$$\begin{array}{l} \forall n, m(\exists n_1, m_1(n_1 \in [n] \wedge m_1 \in [m] \wedge n_1 \gtrdot m_1) \rightarrow \\ \qquad \forall n_2(n_2 \in [n] \rightarrow \exists m_2(m_2 \in [m] \wedge n_2 \gtrdot m_2))). \end{array} \tag{$\flat$}$$

This allows us to define the quotient graph $G/\equiv = \langle N_\equiv, \gtrdot_\equiv, r_\equiv \rangle$ as follows

$$\begin{array}{rcl} N_\equiv & = & N/\equiv \\ [n] \gtrdot_\equiv [m] & \Leftrightarrow & \exists m_1(m_1 \in [m] \wedge n \gtrdot m_1) \\ r_\equiv & = & [r] \end{array}$$

Theorem 5.5 (Minimum Representation) *Let $G = \langle N, \gtrdot, r \rangle$ be a graph which represent a set s. The graph $G/\equiv$ is the minimum representation of s.*

Proof. It can be proved that the relation B defined as

$$\forall n \in N(nB[n])$$

is a bisimulation between G and $G/\equiv$, hence they represent the same set, namely s.

Let $G_1 = \langle N_1, \gtrdot_1, r_1 \rangle$ be a graph and B_1 be a bisimulation between G and G_1. The relation B_1' defined as

$$\forall n, m \in N(nB_1'm \leftrightarrow \exists k \in N_1(nB_1k \wedge mB_1k)).$$

is a bisimulation. Since B_1' is included in $\equiv$ it holds that $|N/\equiv| \leq N_1$. □

The maximum bisimulation over a graph G can be used also to test whether there exists a bisimulation between two rooted graphs G_1 and G_2. Given two rooted graphs $G_1 = \langle N_1, \gtrdot_1, r_1 \rangle$ and $G_2 = \langle N_2, \gtrdot_2, r_2 \rangle$, with $N_1 \cap N_2 = \emptyset$ and $r_{12} \notin N_1 \cup N_2$, consider the graph $G_{12} = \langle N_{12}, \gtrdot_{12}, r_{12} \rangle$, where

$$N_{12} = N_1 \cup N_2 \cup \{r_{12}\}$$

$$n \gtrdot_{12} m \quad \text{iff} \quad n \gtrdot_1 m \vee n \gtrdot_2 m \vee (n = r_{12} \wedge m = r_1) \vee (n = r_{12} \wedge m = r_2)$$

In order to obtain G_{12} it is necessary to add a new node r_{12} with two outgoing edges, reaching the roots of the two graphs.

Theorem 5.6 *Let G_1 and G_2 be two graphs and let G_{12} be the graph obtained from G_1 and G_2 as described above. There exists a bisimulation between G_1 and G_2 if and only if in the maximum bisimulation $\equiv_{12}$ over G_{12} the two roots r_1 and r_2 are in relation.*

Proof. It is a consequence of the fact that each bisimulation between G_1 and G_2 is a bisimulation over G_{12} and of the fact that from a bisimulation over G_{12} it is possible to obtain a bisimulation between G_1 and G_2. □

Hence, we consider the problem of finding the minimum graph bisimilar to a given one. Notice that in this context the root of the graph is not more important than the other nodes of the graph, so from now on we will consider graphs of the form $G = \langle N, \gtrdot \rangle$, without specifying the root.

First of all, in the following section we justify the fact that it is not restrictive to consider the bisimulation problem in the context of graphs without labelling on the nodes and on the edges. Then we discuss the relationships between the bisimulation problem and the *relational coarsest partition problem*, and we introduce our new algorithm.

5.2 From Labelled to Unlabelled Graphs

The graphs on which the bisimulation reductions are applied in Model Checking are the Kripke structures (or state spaces) that model the systems under consideration. Hence, these graphs have labels on the nodes or on the edges or on both nodes and edges.

The most general definition of labelled graph which can be given is the following.

Definition 5.7 (2-Labelled Graph) *Let Act be a finite sets of labels. A* 2-labelled graph *G is a tuple of the form $\langle N, \gtrdot, \Sigma, act \rangle$, where*

- *N is the set of nodes;*
- *$\gtrdot \subseteq N \times N$ is the set of edges;*
- *Σ is a partition of N (labelling of the nodes);*
- *$act : N \rightarrow Act$ is the labelling of the edges.*

The name *Act* for the set of labels of the edges comes from the area of CCS, where the labels on the edges represent the *actions* performed by the process. We will use the notation $n \xrightarrow{a} n'$ to denote the fact that there is an edge from n to n' whose label is a. We will say that a node n has label α if it belongs to the class α in Σ, and that two nodes n and m have the same label (say α) if they belong to the same class α in the partition Σ. The definition of the notion of bisimulation in the case of 2-labelled graphs is the following.

Definition 5.8 (Bisimulation on 2-Labelled Graphs) *Given a 2-labelled graph $G = \langle N, \gtrdot, \Sigma, act \rangle$, a relation $B \subseteq N \times N$ is a bisimulation over G if it satisfies the following conditions:*

(l) $nBm \wedge n \in \alpha \in \Sigma \Rightarrow m \in \alpha \in \Sigma$;

(f) $nBm \wedge n \xrightarrow{a} n' \Rightarrow \exists m'(m \xrightarrow{a} m' \wedge n'Bm')$;

(b) $nBm \wedge m \xrightarrow{a} m' \Rightarrow \exists n'(n \xrightarrow{a} n' \wedge n'Bm')$.

It is possible to map a labelled graph G into a graph G'' by adding new nodes in such a way that the restriction of the maximum bisimulation on G'' to G is the maximum bisimulation on G. The first step is to remove the labels on the edges.

Definition 5.9 (Labelled Graphs) *A* labelled graph *$G = \langle N, \gtrdot, \Sigma \rangle$ is a triple in which:*

- *N is the set of nodes;*
- *$\gtrdot \subseteq N \times N$ is the set of edges;*
- *Σ is a partition of N (labelling of the nodes).*

Obviously the definition of bisimulation on labelled graphs goes as follows:

Definition 5.10 (Bisimulation on Labelled Graphs) *Let $G = \langle N, \gtrdot, \Sigma \rangle$ be a labelled graph. A relation $B \subseteq N \times N$ is a bisimulation over G if it satisfies the following conditions:*

(l) $nBm \wedge n \in \alpha \in \Sigma \Rightarrow m \in \alpha \in \Sigma$;

(f) $nBm \wedge n \gtrdot n' \Rightarrow \exists m'(m \gtrdot m' \wedge n'Bm')$;

(b) $nBm \wedge m \gtrdot m' \Rightarrow \exists n'(n \gtrdot n' \wedge n'Bm')$.

Below we introduce a technique to remove the labels on the edges of a 2-labelled graph and obtain a labelled graph *preserving* the maximum bisimulation.

Definition 5.11 *Given a 2-labelled graph $G = \langle N, \gtrdot, \Sigma, act \rangle$ consider the labelled graph $G' = \langle N', \gtrdot', \Sigma' \rangle$ obtained as follows:*

- $\Sigma' = \Sigma \cup Img(act)$
- *for each edge $n \xrightarrow{a} m$ consider a new node $\langle n, a, m \rangle$ with label a;*
- *replace each edge $n \xrightarrow{a} m$ with the edges $n \to \langle n, a, m \rangle$ and $\langle n, a, m \rangle \to m$.*

Example 5.12 *Considering the 2-labelled graph G in Figure 5.2 on the left we obtain the labelled graph G' in the same figure on the right.*

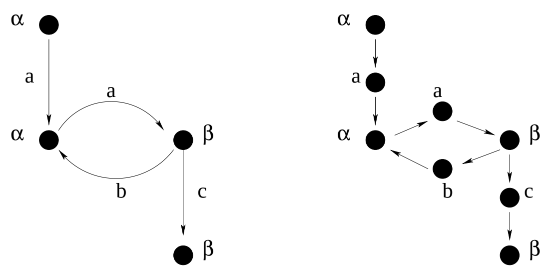


Figure 5.2: 2-Labelled Graph and its unlabelling.

Theorem 5.13 *Let G be a 2-labelled graph, and G' be the graph obtained from G as described in Definition 5.11. Let $\equiv'$ be the maximum bisimulation on G'. The restriction of $\equiv'$ to the nodes of G is the maximum bisimulation $\equiv$ on G.*

Proof. We prove that each bisimulation B over G can be extended to a bisimulation B' on G' and vice-versa each bisimulation B' over G' can be restricted to a bisimulation B on G.

Given a bisimulation B on G consider its extension to G' defined as

$$\forall \langle n, a, m \rangle, \langle p, b, q \rangle \in N' \setminus N(\langle n, a, m \rangle B' \langle p, b, q \rangle \leftrightarrow (nBp \wedge mBq \wedge a = b)).$$

The fact that B' is a bisimulation over G' is an immediate consequence of the fact that the nodes introduced in G' have only one outgoing edge and one ingoing edge.

Given a bisimulation B' over G' consider its restriction B over G. It is easy to prove that B is a bisimulation over G.

As a consequence the restriction of the maximum bisimulation over G' to G is the maximum bisimulation over G. $\square$

As second step we show how to remove the labels from the nodes moving from the graph G' to the graph G''.

Definition 5.14 (Chain) *Let $n \in \mathbb{N}$ be a positive natural number. The n-*chain *$\{\emptyset\}^n$ is the graph $\langle I_n, \gtrdot_n \rangle$, where*

$$I_n = \{0, 1, \dots, n\}$$
$$i \gtrdot_n j \quad \textit{iff} \quad j = i + 1$$

The fact that we use $\{\emptyset\}^n$ to refer to the n-chain is not casual: the n-chain is the representation of the set $\{\emptyset\}^n = \{\{\emptyset\}^{n-1}\}$ (and $\{\emptyset\}^0 = \emptyset$). In Figure 5.3 we show the 1-chain, the 2-chain, and the 3-chain.

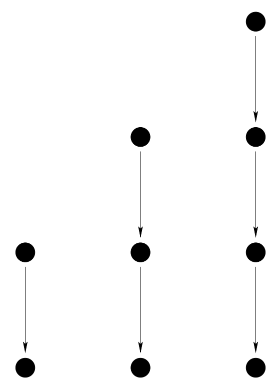


Figure 5.3: Chains.

Lemma 5.15 *If $n \neq m$, then $\{\emptyset\}^n$ is not bisimilar to $\{\emptyset\}^m$.*

Proof. By induction on n. $\square$

The idea is to remove the labels from the nodes and replace them with chains. Notice that, as we show in the example below, it is not enough to use a chain of a different length for each label.

Example 5.16 *Consider the labelled graph G' in Figure 5.4.*

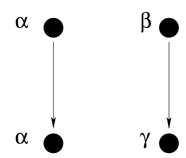


Figure 5.4: Example with Labels.

If we replace label α with a 3-chain, label β with a 4-chain, and label γ with a 2-chain (or in general the label α with a $j+1$-chain, the label β with a $j+2$ chain, and the label γ with a j-chain) we obtain the graph in Figure 5.5, in which one of the nodes which had label α (the higher one) is bisimilar to the node which had label β.

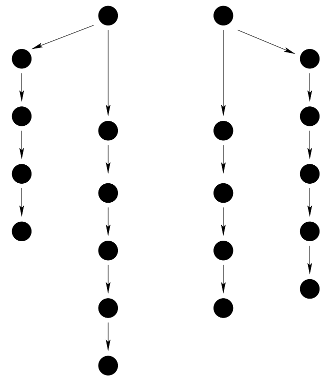


Figure 5.5: Wrong Unlabelling of the Example.

However, it is possible to solve the above problem using chains of sufficiently different lengths.

Definition 5.17 *Let $G' = \langle N', \gtrdot', \Sigma' \rangle$ be a labelled graph. Let $\Sigma' = \{b_1, \ldots, b_h\}$ be the sets of labels and let $|E'| = k$. The graph $G'' = \langle N'', \gtrdot'' \rangle$ is obtained from G' by replacing each label b_i with a $(k+1) * i$-chain.*

Example 5.18 *Using Definition 5.17 the graph G'' we obtain from the graph G' in Figure 5.4 is the graph shown in Figure 5.6, in which, correctly, the nodes which had label α are not bisimilar to the nodes with labels β and γ.*

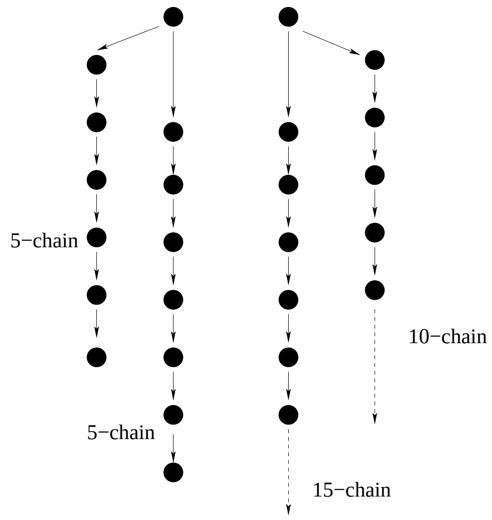


Figure 5.6:

Figure 5.7: Right Unlabelling of the Example.

Theorem 5.19 *Let G' be a labelled graph, and G'' be the graph obtained from G' as described in Definition 5.17. Let $\equiv''$ be the maximum bisimulation on G''. The restriction of $\equiv''$ to the nodes of G' is the maximum bisimulation $\equiv'$ on G'.*

Proof. It is immediate to prove that a bisimulation B' on G' can be extended to a bisimulation B'' on G''.

In order to prove that the restriction to G' of a bisimulation B'' on G'' is a bisimulation on G' we have to prove that

$$nB''m \to [n]_\Sigma = [m]_\Sigma,$$

since the other conditions of the definition of bisimulation follow immediately from the fact that B'' is a bisimulation on G''. Let us assume by contradiction that $[n]_\Sigma = b_i$ and $[m]_\Sigma = b_j$ with $i \neq j$. There is a chain of length $(k+1)*i$ starting from n, hence we must find a chain of length $(k+1)*i$ starting from m. There is a chain of length $(k+1)*j$ starting from m, and all the other chain starting from m have a length which is a number of the form $r+1+(k+1)*s$ with $r \leq k-1$, hence the length of these chains is not a multiple of $(k+1)$, i.e. they cannot have length $(k+1)*i$, which is a contradiction. $\square$

Hence, we have shown in this section that given a 2-labelled graph G or a labelled graph G', it is always possible to move to a graph G'' without labels and compute the maximum bisimulation on it. For this reason, from now on we consider the problem of finding the maximum bisimulation on graphs without labels. Notice that even if G and G' have only one strongly connected component, this is not the case for the graph G''. This observation will become important later, since we will present an algorithm which works better in term of space and time when there are more strongly connected components.

5.3 Relational Coarsest Partition Problem

In [PTB85] Paige Tarjan and Bonic are probably the firsts who reformulate the bisimulation problem as a relational coarsest partition problem, also if the main result in their work is a linear algorithm to solve the single function coarsest partition problem.

The notion of bisimulation can be connected in [PTB85] with the notion of *stability* formally defined as follows:

Definition 5.20 (Stability) *Let $\gtrdot$ be a relation on the set N, $\lessdot$ its inverse relation, and P a partition of N. Then P is said to be* stable *with respect to $\gtrdot$ iff for each pair B_1, B_2 of blocks of P, either $B_1 \subseteq \lessdot(B_2)$ or $B_1 \cap \lessdot(B_2) = \emptyset$.*

Notice that given a partition P we will refer to its elements indifferently as *blocks* or as *equivalence classes*: the name *blocks* cames from the fact that the elements of the partition are sets (blocks) of nodes, while the name *equivalence classes* is connected to the fact that each partition individuates an equivalence relation and vice-versa.

We say that a partition Q *refines* a partition P if each block B of Q is a subset of some block C of P. In particular, the *relational coarsest partition problem* consists in

given a set N, a relation $\gtrdot$ on N, and a partition P of N determine the coarsest partition Q which refines P and is stable w.r.t. $\gtrdot$.

Similarly, when $\gtrdot$ is replaced by a function $f : N \to N$ the problem is known as *single function coarsest partition problem*, while the *many function coarsest partition problem* corresponds to the case in which $\gg$ is replaced by a set of functions. This last problem is equivalent to the minimization of the number of states of a DFA (Deterministic Finite Automaton) (see [Hop71]).

In [KS90] Kannellakis and Smolka notice that the algorithm by Paige and Tarjan [PT87] that solves the relational coarsest partition problem can be used to determine the maximum bisimulation over a graph $G = \langle N, \gg \rangle$.

Theorem 5.21 *Given a graph $G = \langle N, \gg \rangle$, the solution of the coarsest partition problem over the set N, the relation $\gg$, and initial partition $P = \{N\}$, is the partition which corresponds to the maximum bisimulation $\equiv$.*

Proof. First, using the definition of bisimulation it is possible to prove that if B is a bisimulation and it is also an equivalence relation, then the partition determined by B, i.e.

$$Q(B) = \{[n]_B \mid n \in N\},$$

where

$$[n]_B = \{m \in N \mid nBm\}$$

is stable w.r.t. $\gg$ and obviously it refines $P = \{N\}$.

On the other hand, it also holds that if Q is a partition which refines P and it is stable w.r.t. $\gg$, then the equivalence relation that it determines, i.e.

$$nB_Q m \Leftrightarrow \exists B \in Q(n \in B \wedge m \in B),$$

is a bisimulation over G.

Hence, it follows that the coarsest partition which refines P and is stable w.r.t. $\gg$ corresponds to the maximum bisimulation. $\square$

In the case of the minimization of the number of states of a DFA Hopcroft in [Hop71] present an algorithm working in time $O(|N| \log |N|)$ which solves the many function coarsest partition problem (one function for each character of the alphabet of the DFA). Hopcroft's algorithm consists in the computation of a greatest fix-point starting from the bottom in the lattice of the partitions. The fact that a greatest fix-point is computed is also addressed as the fact that a *negative* strategy is implemented: start with the coarsest possible P, keep on splitting some class whenever is <u>not</u> stable. Moreover, Hopcroft's algorithm is based on the *process the smallest half* policy.

In the more complex case of the relational coarsest partition problem the Paige-Tarjan's algorithm, presented in [PT87], solves the problem using a negative strategy and a non-trivial generalization of the process the smallest half policy, in time $O(| \gg | \log |N|)$. In particular, when we say that a block C is split (or refined) using a *splitter* block B we mean that in the partition P we replace C by

$$C_1 = C \cap \ll(B) \quad \text{and} \quad C_2 = C \setminus \ll(B).$$

The process the smallest half policy establishes that if a block S is split into B and $S \setminus B$, and B has less elements than $S \setminus B$, then we have to use B as splitter and not $S \setminus B$. In their generalization of this policy Paige and Tarjan determine how to use B also when it is necessary to use as splitter $S \setminus B$.

A *positive strategy* for the relational coarsest partition problem would consists in starting with the finest partition in which all the nodes are separate and replace two blocks with their union whenever it is possible to prove that in the result their elements have to belong to the same block. A positive strategy has been presented by Paige Tarjan and Bonic for the single function coarsest partition problem, but no one has been able untill now to generalize it to the relational coarsest partition problem.

We have not mentioned the many relations coarsest partition problem, since this is substantially equivalent to the single relation case and, in particular, using our transformation of 2-labelled graphs into labelled graphs it can be reduced to this last. The many relations coarsest partition problem corresponds to the minimization of the number of states of a NFA (Non-deterministic Finite Automaton), but differently from the deterministic case, in the non-deterministic case it can happen that there exist two different NFA's which recognize the same language and whose minimizations are different.

5.4 Bisimulation on Well-Founded Sets

We now start to present our algorithm by considering the well-founded case, i.e. acyclic graphs.

The negative strategy used in [PT87] (and in all the other partitioning algorithms present in the literature) is *blind* with respect to the relation in the following sense: the splitter B is chosen without taking into account the relation $\gg$, the only thing which is taken into account is the cardinality of B.

We take advantage of the set-theoretic point of view of the problem in order to develop a selection strategy for the splitters depending on the relation $\gg$. Making use of the ordering induced by the notion of *rank* we start from a partition which is a refinement of the coarsest one; then we choose the splitters using the ordering induced by the rank. These two ingredients allow to obtain a linear-time algorithm to compute the maximum bisimulation on a graph representing a well-founded set.

Definition 5.22 (Well-Founded Rank) *Let $G = \langle N, \gg \rangle$ be an acyclic graph. The* rank *of a node n is recursively defined as follows:*

$$rank(n) \quad = \quad \begin{cases} 0 & \textit{if } n \textit{ is a leaf} \\ 1 + \max\{rank(m) \mid n \gg m\} & \textit{otherwise} \end{cases}$$

It is immediate to verify that the notion of rank determines a partition between the nodes which is coarsest than the maximal bisimulation. Moreover the following holds.

Lemma 5.23 *Let G be an acyclic graph and $m, n \in N$ be two nodes of G.*

$$m \equiv n \quad \Rightarrow \quad rank(m) = rank(n).$$

Proof. By induction on $rank(m)$. □

The converse, of course, is not true. For instance, $\{\emptyset, \{\emptyset\}\}$ and $\{\{\emptyset\}\}$ have both rank 2 but are not equal (i.e. their associated graphs are not bisimilar).

Let P be a partition of N such that each block B in P is such that for all pair of nodes $m, n \in B$, $rank(m) = rank(n)$; then clearly every refinement of P fulfills the same property. On the ground of this observation it is natural to assign to a block B the rank of its elements.

The algorithm consists of the macro steps described in Figure 5.8, where we rely on a rather natural implementation of the activity of *collapsing* a set of nodes: replace a set of

BISIMULATION-WF($G = \langle N, \gg \rangle$):
1. for $n \in N$ do compute $rank(n)$; — *compute the ranks* 2. $\rho := \max\{rank(n) \mid n \in N\}$; 3. for $i = 0, \ldots, \rho$ do $B_i := \{n \in N \mid rank(n) = i\}$; 4. $C := \{B_i \mid i = 0, \ldots, \rho\}$; — *$C$ is the partition to be refined initialized with the B_i's* 5. for $i = 0, \ldots, \rho$ do (a) $D_i := \{X \in C \mid X \subseteq B_i\}$; — *determine the blocks currently at rank i* (b) for $X \in D_i$ do $G :=$ collapse(G, X) — *collapse nodes at rank i* (c) for $n \in N \cap B_i$ do — *refine blocks at higher ranks* for $C' \in C$ and $C' \subseteq B_{i+1} \cup \ldots \cup B_\rho$ do $C := (C \setminus \{C'\}) \cup \{C' \cap \lessdot(\{n\}), C' \setminus \lessdot(\{n\})\}$

Figure 5.8: Algorithm for the Well-Founded case.

nodes with one single node. Hence, $G :=$ collapse(G, X) means that in N all the nodes which belong to X are replaced by a single node $x \in X$. This implies that after step 5.(b) B_i is no more included in N and for this reason in step 5.(c) we consider $n \in N \cap B_i$.

Step 1. can be performed in time $O(|N| + |\gg|)$ by a depth-first visit of the graph. Despite of the nesting of For-loops the following holds.

Theorem 5.24 (Correctness - Complexity) *The algorithm* BISIMULATION-WF *computes correctly the minimal bisimilar graph of its input acyclic graph $G = \langle N, \gg \rangle$ and its time complexity is $O(|N| + |\gg|)$.*

Proof. The correctness result follows from Lemma 5.23 and from the fact that if $m \not\equiv n$ with $rank(m) = rank(n) = i$, then either there is $m' \in m$ which is not bisimilar to any element of n, or there is $n' \in n$ which is not bisimilar to any element of m. Hence, at the i-th iteration the two nodes belong to different blocks of the current partition.

The complexity result is a consequence of the following facts:

- each node n is considered at most once in (b);
- the incoming edges are considered only once;
- for an efficient implementation of the collapsing procedure and of (c), it is sufficient to retain the counter-image $\lessdot(n)$ of each node n. Notice that collapsing the nodes $n_1, \ldots, n_k$ into a node n_0 can be done in time $O(\lessdot(\{n_1, \ldots, n_k\}))$, and that the collapsing phase is done once per level
- Some reasonable list-manipulation procedure is needed for correctly and quickly storing partitions.

$\square$

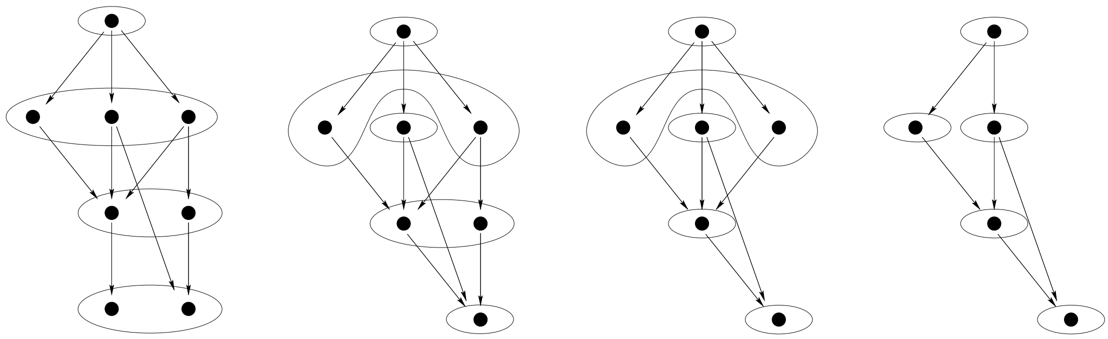


Figure 5.9: Minimization Process.

An example of computation of the above algorithm can be seen in Figure 5.9, where we emphasize the state of the partition C at each iteration within point (5).

In all the examples we present in this chapter we use the first column of the picture for the initial graph, while the other columns represent (from left to right) the graph during the subsequent iterations of the algorithm.

Those who are familiar with OBDDs ([Bry86]) or with k-layered DFA's ([HP99]) can read our algorithm for the well-founded case as a generalization of the minimization algorithm for k-layered DFA. In the well-founded case we admit that a node at the i-th layer may reach a node at the j-th layer with $j > i$.

5.5 Rank and Bisimulation on Non-Well-Founded Sets

We describe here the algorithm for the non-well-founded case. First of all, the presence of cycles causes the usual notion of rank (cf. Definition 5.22) to be not adequate: an extension of the notion of *rank* must be defined, and to this end, the following notion will be useful

Definition 5.25 (Strongly Connected Components) *Given a graph $G = \langle N, \gtrdot \rangle$, the graph $G^{scc} = \langle N^{scc}, \gtrdot^{scc} \rangle$ is the graph obtained as follows:*

nodes *Strongly connected components of G are biunivocally associated to nodes of N^{scc}.*

edges *An edge $\langle m, n \rangle$ is in $\gtrdot^{scc}$ if and only if $m \neq n$ and there is an edge in $\gtrdot$ going from a node of the strongly connected component associated to m to a node of the strongly connected component associated to n.*

Given a node $n \in N$, we refer to the node of G^{scc} associated to the strongly connected component of n as $c(n)$.

Observe that G^{scc} is acyclic and if G is acyclic then G^{scc} is (isomorphic to) G.

We need to distinguish between the well-founded part and the non-well-founded part of a graph G.

Definition 5.26 (Well-founded Part) *Let $G = \langle N, \gtrdot \rangle$, given a node $n \in N$ we define $G(n) = \langle N(n), \gtrdot_{\restriction N(n)} \rangle$ as the subgraph of G of the nodes reachable from n.*

We define $WF(G)$, the well-founded part of G, as

$$WF(G) = \{n \in N \mid G(n) \text{ is acyclic}\}.$$

Definition 5.27 (Cyclic Rank) *Given a graph $G = \langle N, \gtrdot \rangle$, then the* rank *of a node is recursively defined as follows:*

$$rank(n) = \begin{cases} 0 & \text{if } n \text{ is a leaf in } G \\ -\infty & \text{if } c(n) \text{ is a leaf in } G^{scc} \text{ and } n \text{ is not a leaf in } G \\ \max(\{1 + rank(m) \mid c(n) \gtrdot^{scc} c(m), m \in WF(G)\} \cup & \\ \quad \{rank(m) \mid c(n) \gtrdot^{scc} c(m), m \notin WF(G)\}) & o.w. \end{cases}$$

Since G^{scc} is always acyclic, the definition is correctly given. If G is acyclic then $G = G^{scc}$ and the above definition extends the one given in the well-founded case. Nodes that are mapped into leaves of G^{scc} are either equivalent to $\emptyset$ or to the non-well-founded set Ω, the unique non-well-founded set such that $\Omega = \{\Omega\}$, see Section 1.2.3 (not to be confused with the theory Ω used in the first part of this thesis). For a non-well-founded nodes different from Ω the rank is 1 plus the maximum rank of a well-founded set in its transitive closure.

We have explicitly used the graph G^{scc} to provide a formal definition of the notion of rank. However, in practice, the rank can be computed directly on G by two visits of the graph, avoiding the explicit construction of G^{scc}.

Notice that if the relation $\equiv$ holds between two nodes, then they have the same rank, even in the new sense:

Lemma 5.28 *Let m and n be nodes of a graph G:*

a) $m \equiv \Omega \Leftrightarrow rank(m) = -\infty$.

b) $m \equiv n \Rightarrow rank(m) = rank(n)$.

Proof.

a) Immediate from the characterization of Ω of [Acz88]: a graph G rooted in n represents Ω if and only if each node has at least one outgoing edge.

b) If m and n are well-founded nodes, the result follows immediately from Lemma 5.23 and from the fact that in the well-founded case the new definition of rank coincides with the old one. If one is well-founded and the other one is not, they cannot be bisimilar.

The interesting case occurs when both m and n are non-well-founded. Let $rank(m) = h$ and let a be a well-founded node reachable from m of rank $h-1$. Since $m \equiv n$ there exists a node b reachable from n such that $a \equiv b$. Since a and b are well-founded $rank(b) = rank(a)$ and this implies $rank(n) \geq rank(m)$. Symmetrically we can conclude that $rank(m) \geq rank(n)$ from which the thesis follows. $\square$

In the last part of this proof we use an argument which is nothing but a generalization of the argument we use on chains in Section 5.2.

Clearly the converse of Lemma 5.28-**b** is not true in general. Moreover the rank of $c(n)$ in G^{scc} is not necessarily equal to the rank of n in G. In the example in Figure 5.10 there are bisimilar nodes which do not belong to the same strongly connected component. The lower node has rank 0, while the other four nodes have rank 1.

The above example shows also that using as definition of rank of a node m in G the rank of $c(m)$ in G^{scc}, Lemma 5.28-**b** would not hold. In Section 5.6.2 we present other possible notions of rank. In fact we could, in general, work with any *rank* function such that $rank : N \to \mathbb{N}$ and:

- $m \equiv n \Rightarrow rank(m) = rank(n)$;

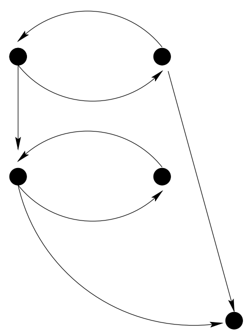


Figure 5.10: Example of nodes of ranks 0 and 1.

- it is possible to compute $\equiv_{|rank=i}$ using $\equiv_{|rank<i}$ and $\Rightarrow_{|rank=i}$, where $\equiv_{|rank=i}$ is the bisimulation between the nodes at rank i, $\equiv_{|rank<i}$ is the bisimulation between the nodes at rank less than i, $\Rightarrow_{rank=i}$ are the edges between the nodes at rank i.

Given a graph $G = \langle N, \Rightarrow \rangle$ with $\rho = \max\{rank(n) \mid n \in N\}$, we call the sets of nodes $B_{-\infty}, B_0, \ldots, B_\rho$ where $B_i = \{n \in N \mid rank(n) = i\}$, the *rank components* of G.

Since we proved in the previous section that the bisimulation relation can be computed in linear time on well-founded graphs, it is easy to see that we can use the algorithm of the well-founded case in order to process the nodes in $WF(G)$ for the general case. Hence we can assume that the input graph for the general case does not contain two different bisimilar well-founded nodes. Notice that this assumption is made only to simplify the presentation and can be discarded without affecting the correctness of the algorithm below. The algorithm for the general case is presented in Figure 5.11.

In step 1. we compute the strongly connected components that will be used to perform the computation of ranks made in the following step. As a matter of fact, the computation of strongly connected components could even be avoided but certainly speeds up the entire procedure.

In step 5. we initialize a variable C representing the computed partition using the ranks.

Collapsing nodes $a_1, \ldots, a_k$, as in step 6. and 8., consists in eliminating all nodes but a_1 and replacing all edges incident to $a_2, \ldots, a_k$ by edges incident to a_1.

Step 8. is the core of the algorithm. For each rank i we call (8.(a)) the $O(|\Rightarrow| \log |N|)$ procedure of [PT87] using the restriction of C to the nodes of rank i as initial partition. In step 8.(b) we collapse nodes of rank i according to the partition computed at the previous step. Finally step 8.(c) (and step 7. for the case $i = -\infty$) updates the partition C on nodes of rank greater than i.

On the ground of the previous observations it is easy to see that the following holds.

Theorem 5.29 (Complexity) *In $G \equiv \langle N, \Rightarrow \rangle$ is a graph, then the worst case complexity of the algorithm* BISIMULATION *is $O(|\Rightarrow| \log |N|)$.*

Proof. Step 1. can be performed in time $O(|N| + |\Rightarrow|)$ (cf. [Tar72]). Step 2. can be executed in time $O(|N|+|\Rightarrow^{scc}|) \leq O(|N|+|\Rightarrow|)$ by a depth-first visit of the graph G^{scc}. The complexity of steps 4. and 5. is clearly linear. Step 6. is performed as in the well-founded case.

BISIMULATION($G = \langle N, \gtrdot \rangle$):

1. compute G^{scc}; — *compute the s.c.c.'s*
2. for $n \in N$ do compute $rank(n)$; — *compute the ranks*
3. $\rho := \max\{rank(n) \mid n \in N\}$;
4. for $i = -\infty, 0, \ldots, \rho$ do $B_i := \{n \in N \mid rank(n) = i\}$;
5. $C := \{B_i \mid i = -\infty, 0, \ldots, \rho\}$;
   — *C is the partition to be refined initialized with the B_i's*
6. $G :=$ collapse$(G, B_{-\infty})$;
   — *collapse all the nodes of rank $-\infty$*
7. for $n \in N \cap B_{-\infty}$ do — *refine blocks at higher ranks*
      for $C' \in C$ and $C' \neq B_{-\infty}$ do
         $C := (C \setminus \{C'\}) \cup \{C' \cap \lessdot(\{n\}), C' \setminus \lessdot(\{n\})\}$
8. for $i = 0, \ldots, \rho$ do
   (a) $D_i := \{X \in C \mid X \subseteq B_i\}$;
         — *determine the blocks at rank i*
       $G_i := \langle N \cap B_i, \gtrdot \restriction N \cap B_i \rangle$;
         — *isolate the subgraph at rank i*
       $D_i :=$ Paige-Tarjan(G_i, D_i); — *process rank i*
   (b) for $X \in D_i$ do
         $G :=$ collapse(G, X) — *collapse nodes at rank i*
   (c) for $n \in N \cap B_i$ do — *refine blocks at higher ranks*
         for $C' \in C$ and $C' \subseteq B_{i+1} \cup \ldots \cup B_\rho$ do
            $C := (C \setminus \{C'\}) \cup \{C' \cap \lessdot(\{n\}), C' \setminus \lessdot(\{n\})\}$

Figure 5.11: Algorithm for the General case.

The complexity of step 7. is the same as the complexity of step 8.(c).

As for the complexity of step 8., it corresponds to the complexity of the procedure Paige-Tarjan which is $O(|\Rightarrow|\log|N|)$. This is due to the fact that all the remaining sub-steps can be implemented at a cost which is linear in the size of G_i (cf. [Hop71]). $\square$

Theorem 5.30 (Correctness) *The algorithm* BISIMULATION *on input G correctly computes the maximal bisimulation on G.*

Proof. From Lemma 5.28 we have that if $m \equiv n$, then m and n belong to the same block in the initial partition.

If $rank(m) = rank(n) = -\infty$, then from Lemma 5.28 we know that they are both bisimilar to Ω. Step 6. takes care of their collapse.

For the remaining cases, by induction on $rank(m)$, we prove that if $rank(m) = i$, then $m \equiv n$ if and only if at the i-th iteration, after step 8.(a) there exists $X \in D_i$ such that $m, n \in X$.

If $rank(m) = 0$, then it must be the case that m is a leaf of G. At the beginning all the leaves of G are in B_0. After step 7. all the leaves of G still belong to the unique block B_0. This implies that the procedure Paige-Tarjan applied to G_0 does not split the block B_0. Hence step 8.(b) collapse all the nodes of rank 0. This is equivalent to our thesis, since it is true that all the leaves of G are bisimilar (they all represent the empty set).

If $rank(m) = i$ and we assume $m \equiv n$, then for all $m' \lessdot m$ there exists $n' \lessdot n$ such that $m' \equiv n'$ and vice-versa. Hence, by inductive hypothesis, they belong to the same block X of D_i at the beginning of the $i - th$ iteration of step 8. The correctness of the procedure Paige-Tarjan (cf. [PT87]) ensures that they still belong to the same block at the i-th iteration after step 8.(a). The opposite direction is similar. $\square$

Since we have no guarantees that there are not (linearly) many nodes at the same rank, the complexity of the algorithm presented above is asymptotically equivalent to that of Paige and Tarjan. As a matter of fact, it is easy to see how to generalize the example shown in Figure 5.12 in order to obtain a graph with n non-bisimilar nodes such that $n-1$ of them have rank 1.

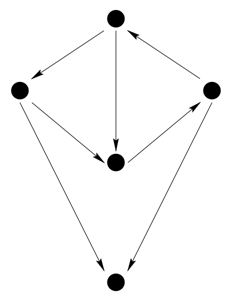


Figure 5.12: Example of worst case complexity.

However, as for the well-founded Algorithm 5.8, we take advantage of a refined initial partition and of a selection strategy of the blocks to be setected for splitting blocks of higher ranks. In a single rank, the negative strategy of Paige-Tarjan's algorithm is applied to rank components which, in general, are much smaller than the global graph. In particular, for families of graphs such that ρ is $\Theta(|N|)$ and the size of the each rank component is bounded by a constant c the global cost becomes linear. In Section 5.7 we show the results of some tests comparing our algorithm with Paige-Tarjan's algorithm. We are actually working to interface our C implementation with the model checker XEVE (see [Bou98]) in order to analyse the performances of our algorithm on graphs coming from real applications.

In the following section we discuss some possible optimizations.

5.6 Optimizations

We present two kind of optimizations which are consequence of the following two types of analysis:

- in which situations the topology of the graph at a given rank allows to apply faster strategies?
- is it possible to refine the notion of *rank* in order to start with a finer partition?

We call optimizations suggested by the first type of analysis *topological optimizations*, while optimizations suggested by the second type of analysis will be called *rank optimizations*.

5.6.1 Topological Optimizations

We present three situations in which we are able to improve the algorithm for the general case.

Let us consider the following definition.

Definition 5.31 (Multiple Node) *A node n belonging to a rank component $B_i \subseteq N$ is said to be a* multiple node *if* $|\{m \mid n \gg m, m \in B_i\}| > 1$.

If G_i has no multiple nodes, we can replace the call to Paige-Tarjan in step 8.(a) with a call to Paige-Tarjan-Bonic (cf. [PTB85]). This allows us to obtain a linear time performance at rank i, and if there are no multiple nodes in the whole graph G the entire algorithm performs in linear time.

As we say in Section 5.3 the algorithm of Paige Tarjan and Bonic solves the single function coarsest partition problem, i.e. the graph has at most one outgoing edge from each node. The strategy it implements is *positive*, since nodes are put in the same equivalence class when it is know that they are bisimilar. The tests performed to recognize whether two nodes have to belong to the same class are based on string-matching, hence, the linear time complexity is a consequence of the linear time complexity of the Knuth-Morris-Pratt algorithm (see [KMP77]).

In Figure 5.13 we present the optimized version of our algorithm with a call to the procedure Paige-Tarjan-Bonic made whenever is possible.

Theorem 5.32 (Complexity) *If $G \equiv \langle N, \gg\rangle$ is a graph with no multiple nodes, then the worst case complexity of the algorithm* BISIMULATION-OPT *is* $O(|N| + |\gg|)$.

Proof. It easily follows from the complexity analysis performed in Theorem 5.29, and from the linear complexity of Paige-Tarjan-Bonic, [PTB85]. $\square$

```
BISIMULATION-OPT(G = ⟨N, ⩾⟩):
1. compute G^{scc}; — compute the s.c.c.'s
2. for n ∈ N do compute rank(n); — compute the ranks
3. ρ := max{rank(n) | n ∈ N};
4. for i = −∞, 0, ..., ρ do B_i := {n ∈ N | rank(n) = i};
5. C := {B_i | i = −∞, 0, ..., ρ};
     — C is the partition to be refined initialized with the B_i's
6. G := collapse(G, B_{−∞});
     — collapse all the nodes of rank −∞
7. for n ∈ N ∩ B_{−∞} do — refine blocks at higher ranks
       for C' ∈ C and C' ≠ B_{−∞} do
           C := (C \ {C'}) ∪ {C ∩ ⩽({n}), C \ ⩽({n})}
8. for i = 0, ..., ρ do
   (a) D_i := {X ∈ C | X ⊆ B_i};
           — determine the blocks at rank i
       G_i := ⟨N ∩ B_i, ⩾ ↾ N ∩ B_i⟩;
           — isolate the subgraph at rank i
       if ∀x ∈ N ∩ B_i |{⟨x, y⟩ | y ∈ N ∩ B_i}| ≤ 1 — process rank i
           then D_i := Paige-Tarjan-Bonic(G_i, D_i);
           else D_i := Paige-Tarjan(G_i, D_i);
   (b) for X ∈ D_i do
           G := collapse(G, X) — collapse nodes at rank i
   (c) for n ∈ N ∩ B_i do — refine blocks at higher ranks
           for C' ∈ C and C' ⊆ B_{i+1} ∪ ... ∪ B_ρ do
               C := (C \ {C'}) ∪ {C' ∩ ⩽({n}), C' \ ⩽({n})}
```

Figure 5.13: Optimized Algorithm for the General case.

Theorem 5.33 (Correctness) *The algorithm* BISIMULATION-OPT *on input G correctly computes the maximal bisimulation on G.*

Proof.

Follows from the correctness of the algorithm for the general case (see Theorem 5.30 and the correctness of the procedure in [PTB85].

The only (minor) problem lies in the fact that in our case at any given rank we are actually working with partial functions. The original statement of the single function coarsest partition problem assumes, instead, the input functions to be total. However, it is not difficult to see how to modify the graph of a partial function into one of a total function equivalent for our purposes. For example we could add a new node with a self-loop acting as a *sink* for those nodes where the input function is not defined. $\square$

In Figure 5.14 we show a graph on which the above optimization can be performed and the overall algorithm turns out to be linear. In Section 5.7 we show some more interesting examples on which our algorithm works in linear time, while Paige-Tarjan algorithm works in time $\Omega(| \geqslant | \log |N|)$.

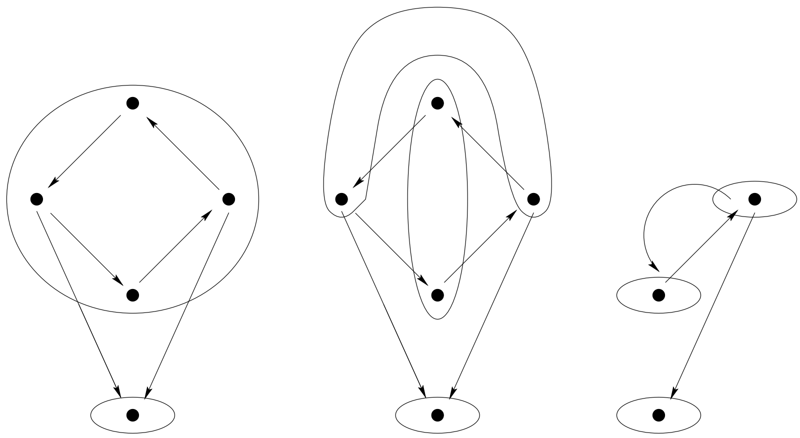


Figure 5.14: Example of the 1st topological optimization.

The crucial consideration behind the second optimization we propose is the following: the outgoing edges of a node u allow to establish to which other nodes of its rank component u is equivalent. If we have some means to know that u is not equivalent to any other nodes of its rank component we can simply delete all edges generating from u within the same B_i. The correctness of this optimization is a consequence of the fact that in the definition of bisimulation the edges generating from a node n are necessary only to determine to which nodes n is bisimilar. The deletion of a set of edges splits a rank component (i.e. we can recalculate the rank), and makes it possible to recursively apply our algorithm on a simpler case. For example, if at a given iteration there exists X in D_i which is a singleton, then all the outgoing edges generating from the node in X can be deleted. In set-theoretic terms this action corresponds to ignore some elements of a given set (those at its own rank) whenever Extensionality is already guaranteed (by elements at smaller ranks). After such a deletion we should recompute the ranks on G_i.

In some cases the application at each step of this optimization allows the algorithm to run in linear time, as shown in Figure 5.15. In general, the worst case complexity does not change. To improve the readability of the picture we have used the labels A, B, and C to distinguish the nodes which belong to the same block at the previous iteration, while the circles are used to denote rank components.

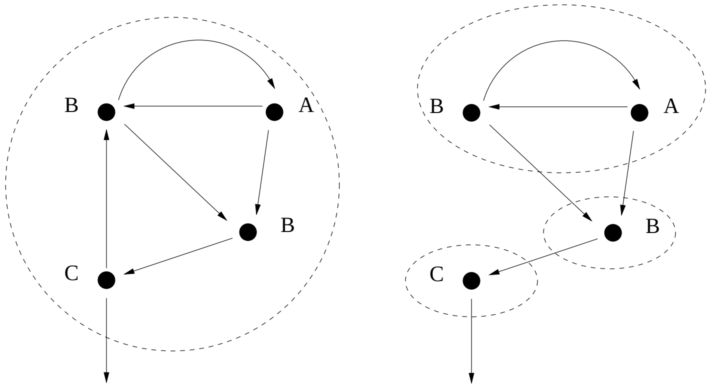


Figure 5.15: Example of the 2nd topological optimization.

The third optimization is a sort of generalization of the previous one. Even in the case in which there are no singletons in D_i some sort of optimization is possible. For example, if X in D_i is of the form $\{u_1, \ldots, u_h\}$, let us define $succ(u_j) = \{Y \mid Y \in D_i \text{ and } \exists v \in Y \langle u_j, v\rangle \in G_i\}$. It is immediate to see that if two nodes u_i and u_k of X are such that $succ(u_i) \neq succ(u_j)$, then they cannot be equivalent. In particular if there exists a node u_i such that $succ(u_i)$ is different from all the other sets $succ(u_j)$, then it is not bisimilar to any node of G_i and we can delete its outgoing edges. The set-theoretic meaning of this deletion is that, for a given $u_i \in X$, all the candidates for bisimilarity have elements witnessing the difference.

In Figure 5.16 we present a case in which we apply this optimization.

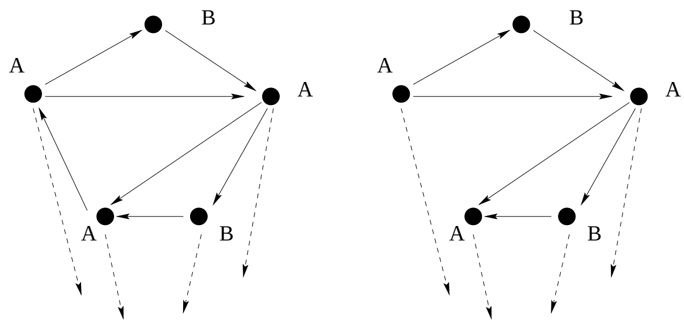


Figure 5.16: Example of the 3rd topological optimization.

The difference between this optimization and the previous one is that in the previous case the information from the nodes of rank less than i were enough to allow to delete some edges. In this case we also use information relative to the successors at the same rank of the node.

Notice that if we are considering a rank i on which no further splits are necessary, this optimization allows us to reach the solution in one step. Let $D_i = \{X_1, \ldots, X_k\}$, if for all $j \leq k$, for all $u, v \in X_j$ $succ(u) = succ(v)$, then for all $j \leq k$ all the nodes in X_j are bisimilar.

In fact it is immediate to prove that the relation defined as $u \sim v$ iff $\exists j \leq k(u \in X_j \wedge v \in X_j)$ is the maximal bisimulation.

We conclude this section by observing that the set-theoretic point of view can easily suggest further optimizations of this sort. For example, if a set is a common element of all the elements of a given X in D_i, then it can be ignored (delete all the entering edges).

5.6.2 Rank Optimizations

In Section 5.5 we gave a definition of rank in such a way to obtain that if two nodes are bisimilar, then they have the same rank (cf. Lemma 5.28). We can refine this definition, i.e. we can give a definition of rank which still satisfies the property that bisimilar nodes have the same rank, but such that there are *less* nodes with the same rank. This implies that the part of graph to be processed at each iteration is smaller and hence there is higher probability that we can apply one of the topological optimizations.

The feature that we believe is crucial in the following definitions is that the rank is not an integer number any more but is a more structured notion (e.g. a set of integers). In the second case the rank cannot be computed *a-priori*.

Clearly from a computational point of view there is a trade-off between the cost involved in determining the rank and the overall complexity of the algorithm. Such a complexity issue is under investigation.

In the first definition of *rank* on the non-well-founded part of the graph we have taken into consideration only the max of a set of ranks. We can refine this definition considering all this set of ranks.

Definition 5.34 (Cyclic Rank') *Given a graph $G = \langle N, \gtrdot \rangle$, then the* rank' *of a node is recursively defined as follows:*

$$rank'(n) \quad = \quad \begin{cases} \{0\} & \text{if } n \text{ is a leaf in } G \\ \{-\infty\} & \text{if } c(n) \text{ is a leaf in } G^{scc} \text{ and } n \text{ is not a leaf in } G \\ \{1+m' \mid c(n) \gtrdot^{scc} c(m), m \in WF(G), m' \in rank'(m)\} \cup & \\ \{m' \mid c(n) \gtrdot^{scc} c(m), m \notin WF(G), m' \in rank'(m)\} & o.w. \end{cases}$$

A possible way to compute such rank is by maintaining a bit-vector of length ρ (the maximal well-founded rank in G) for each node. Such vectors can be computed using bit-wise disjunctions over the vectors associated to the successors.

We prove that this definition of rank still satisfies the main property required to a notion of rank in order to be useful in the computation of the bisimulation.

Lemma 5.35 *Let n and n be nodes of a graph G:*

a) $m \equiv \Omega \leftrightarrow rank'(m) = \{-\infty\}$.

b) $m \equiv n \rightarrow rank'(m) = rank'(n)$.

Proof.

The argument for **a:** and the first part of the argument for **b:** are as in Lemma 5.28.

When both m and n are non-well-founded let $h \in rank'(m)$ and let a be a well-founded node reachable from m such that $h - 1 \in rank'(a)$. Since $m \equiv n$ there exists a node b reachable from n such that $a \equiv b$. Since a and b are well-founded $rank'(b) = rank'(a)$ and

Comp_Rank'($G = \langle N, \gtrdot \rangle$):

1. for $c(n) \in N^{scc}$ compute $rank(c(n)) \in G^{scc}$;
2. compute $WF(G)$; — *compute the well-founded part of G*
3. $\theta := 3 + \max\{rank(c(n)) \mid n \in WF(G)\}$;
4. $\sigma := \max\{rank(c(n)) \mid n \in G\}$;
5. for $n \in N^{scc}$ do
   for $1, \ldots, \theta$ do $rank'(n)[i] := 0$; — *initialize the bit-vectors*
6. if $rank(c(n)) = 0$ and n is a leaf in G then $rank'(n)[2] := 1$;
   — *compute rank' of the nodes bisimilar to* $\emptyset$
7. if $rank(c(n)) = 0$ and n is not a leaf in G then $rank'(n)[1] := 1$;
   — *compute rank' of the nodes bisimilar to* Ω
8. for $i = 1, \ldots, \sigma$ do — *compute rank' of the remaining nodes*
   if $rank(c(n)) = i$ then
   $A := \{c(m) \mid c(n) \gtrdot^{scc} c(m), m \in WF(G)\}$;
   $B := \{c(m) \mid c(n) \gtrdot^{scc} c(m), m \notin WF(G)\}$;
   for $j = 1, \ldots, \theta$ do
   $rank'(c(n))[j] := \bigvee_{a \in A} rank'(a[j-1]) \vee$
   $\bigvee_{b \in B} rank'(b[j])$;
9. for $n \in N$ do $rank'_d(n) = \sum_{j=1}^{\theta} rank'(c(n))[j] * 10^j$
   — *order the rank' 's*

Figure 5.17: The Procedure Comp_Rank'.

this implies $rank'(m) \subseteq rank'(n)$. Symmetrically we can conclude that $rank'(m) \supseteq rank'(n)$. □

It easy to see that the above definition of rank refines the previous one properly and that it is possible to give an order in which rank components can be correctly processed.

Theorem 5.36 (Complexity) *In $G \equiv \langle N, \gtrdot \rangle$ is a graph, then the worst case complexity of the algorithm* BISIMULATION' *is $O(|\gtrdot| \log |N|)$.*

Proof. The only step which is different from the algorithm for the general case is step 2. We prove that the complexity of Comp_Rank'(G) is $O(|\gtrdot| + |N|)$.

From Theorem 5.29 we have that step 1. of Comp_Rank'(G) can be performed in time $O(|\gtrdot^{scc}| + |N^{scc}|)$. To execute step 2. of the procedure in linear time observe that it is possible to decide whether $n \in WF(G)$ using a breath-first visit of $G(n)$. A node n is in $WF(G)$ if and only if n is a leaf or $\forall m(\langle n, m \rangle \in \gtrdot \rightarrow m \in WF(G)$.

The steps 3., 4., 5., 6., 7. are trivial.

Step 8. can be implemented using a breath-first visit of G^{scc} and a bit-sum over the vectors of the $rank'$s. □

Theorem 5.37 (Correctness) *The algorithm* BISIMULATION' *is correct whenever it uses a notion r of rank which fulfills the following three properties:*

a) the nodes collapsed in step 6. *are exactly all the nodes bisimilar to* Ω*;*

b) if $m \equiv n$, then $r(m) = r(n)$;

BISIMULATION'($G = \langle N, \gg \rangle$):

1. compute G^{scc}; — *compute the s.c.c.'s*
2. Comp_Rank'(G); — *compute the ranks*
3. $\rho := \max\{rank'_d(n) \mid n \in N\}$;
4. for $i = 10, \ldots, \rho$ do $B_i := \{n \in N \mid rank'_d(n) = i\}$;
5. $C := \{B_i \mid i = 10, 100, \ldots, \rho\}$;
   — *C is the partition to be refined initialized with the B_i's*
6. $G :=$ collapse(G, B_{10});
   — *collapse all the nodes of rank'* $\{-\infty\}$
7. for $n \in N \cap B_{10}$ do — *refine blocks at higher ranks*
   for $C' \in C$ and $C' \neq B_{10}$ do
   $C := (C \setminus \{C'\}) \cup \{C' \cap \lessdot(\{n\}), C' \setminus \lessdot(\{n\})\}$
8. for $i = 100, \ldots, \rho$ and $B_i \neq \emptyset$ do
   (a) $D_i := \{X \in C \mid X \subseteq B_i\}$;
   — *determine the blocks at* $rank'_d = i$
   $G_i := \langle N \cap B_i, \gg \restriction N \cap B_i \rangle$;
   —*isolate the subgraph to process*
   $D_i :=$ Paige-Tarjan(G_i, D_i); — *process* G_i
   (b) for $X \in D_i$ do
   $G :=$ collapse(G, X) — *collapse nodes at* $rank'_d = i$
   (c) for $n \in N \cap B_i$ do — *refine blocks at higher ranks*
   for $C' \in C$ and $C' \subseteq B_{i+1} \cup \ldots \cup B_\rho$ do
   $C := (C \setminus \{C'\}) \cup \{C' \cap \lessdot(\{n\}), C' \setminus \lessdot(\{n\})\}$

Figure 5.18: Algorithm for the General case using $rank'$.

c) if $m \lessdot n$, then $r(m) \leq r(n)$.

Proof. The result can be proved using the same considerations used to prove Theorem 5.30. □

Corollary 5.38 *The algorithm* BISIMULATION' *on input G correctly computes the maximal bisimulation on G.*

Proof. From Lemma 5.35 we have that $rank'_d$ satisfies a) and b) of Theorem 5.37. The definitions of $rank'$ and $rank'_d$ ensure that $rank'_d$ satisfies c) of Theorem 5.37. Hence, from Theorem 5.37 we have the thesis. □

The last rank optimization we sketch consists in determining the rank component on-line starting from the following observation. Let X and Y be two strongly connected components whose rank (according to Definition 5.27) is the same and such that Y is reachable from X in G^{scc}. For a given node a let us call *label* the block of C to which a belongs. X and Y can contain bisimilar nodes only if the set of labels of nodes in X is contained in the set of labels of nodes in Y.

On the ground of the above observation we could proceed by computing a $rank''$ for nodes belonging to the same strongly connected components on-line during the main loop of the algorithm.

We conclude noticing that it would be interesting to study the problem of the existence of a notion of rank (refining the ones given above) with respect to which the computation of the maximal bisimulation can be performed at optimal cost.

Our intuition is that, for a given non-well-founded set a, such an optimal notion should somehow encode the entire topology of the well-founded sets in the transitive closure of a.

5.7 Testing

We present here some tests performed on two implementations of our bisimulation algorithm. First we motivate, from a theoretical point of view, the set of tests we have chosen and then we give the results of the tests.

To the best of our knowledge there is no *official* set of benchmarks for testing an algorithm such as the one we propose. We decided to test our implementation in the context of formal verification using model checkers (such as SPIN) and considering the transition graphs they generate from a given program. In [Hol91] it is described how to check that the implementation of a protocol conforms a formal specification. To this purpose the implementation is translated into interacting finite-state machines. Usually the graphs of these machines are composed by a unique strongly connected component. The theory presented in [Hol91] finds application in the model checker SPIN [Hol97]. We have taken into consideration the transition systems generated by the examples in the SPIN package. The alphabets which label the edges of the graphs of these transition systems have usually a great number of characters. When we translate these graphs into graphs without labels on the edges (see Section 5.2), we obtain graphs on which we can perform the second optimization proposed in Section 5.6. Such an optimization allows us to delete edges in the graphs, obtaining, in a large number of cases, graphs on which our algorithm works in linear time. For example, in Figure 5.19 we show the graph we obtain for the process Cp0 of the Snooping Cache protocol.

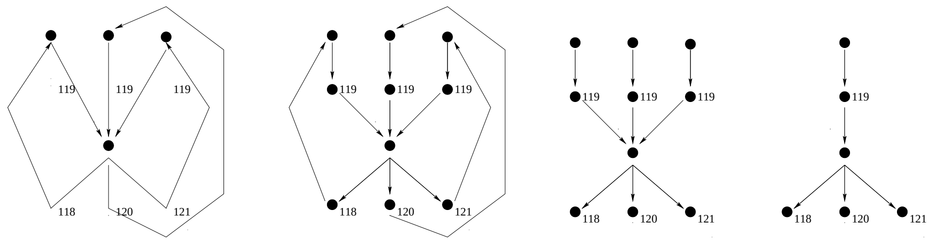


Figure 5.19: Bisimulation contraction of Cp0 from Snoopy.

We obtain similar results for all its procedures. In the first column we give the graph with labels on the edges, its translation into a graph without labels on the edges is drawn in the second column. In the third column we show the graph we obtain applying our optimization. The graph in the third column can be computed from the initial one in linear time and our algorithm determines its bisimulation contraction (fourth column) in linear time.

In Figure 5.20 we show the graphs of the process Node in the Leader protocol from the SPIN package. This protocol implement the Dolev, Klawe and Rodeh algorightm for leader election in unidirectional ring ([DKR82]). In the first row we show the graph we obtain applying the mapping described in Appendix A in order to eliminate the labels on the edges. In the second row we draw the graph we obtain by applying the second optimization described in Section 5.6. This last is an acyclic graph, hence our algorithm computes its bisimulation contraction in linear time. These considerations about the topology of the graphs coming from the context of verification suggest us some examples on which compare the performances of our algorithm with those of Paige-Tarjan' algorithm.

5.7.1 Tests on the C Implementation

The algorithm has been implemented in standard C and compiled using the C++ Builder 5 Compiler. Tests have been executed on a PIII, 600 MHz PC, OS Windows NT. In the results we compare the execution times of the Paige-Tarjan procedure ([PT87]) with those of our algorithm. Also, in this implementation in order to make the comparison meaningful, the Paige-Tarjan procedure has been implemented in the same code and the two algorithms employ the same data structures for graphs and partitions, and the same auxiliary procedures (e.g., that performing blocks splitting).

We report below the running times of the two algorithms over 6 families of graphs. In the tables, Alg denotes the running time of our algorithm, and PT denotes the running time of the Paige-Tarjan procedure. Times are expressed in seconds.

Test 1. Hopcroft. This is the graph used by Hopcroft in [Hop71] as an example of an automata on which its algorithm runs in time proportional to $|\rightarrow|\log|N|$.

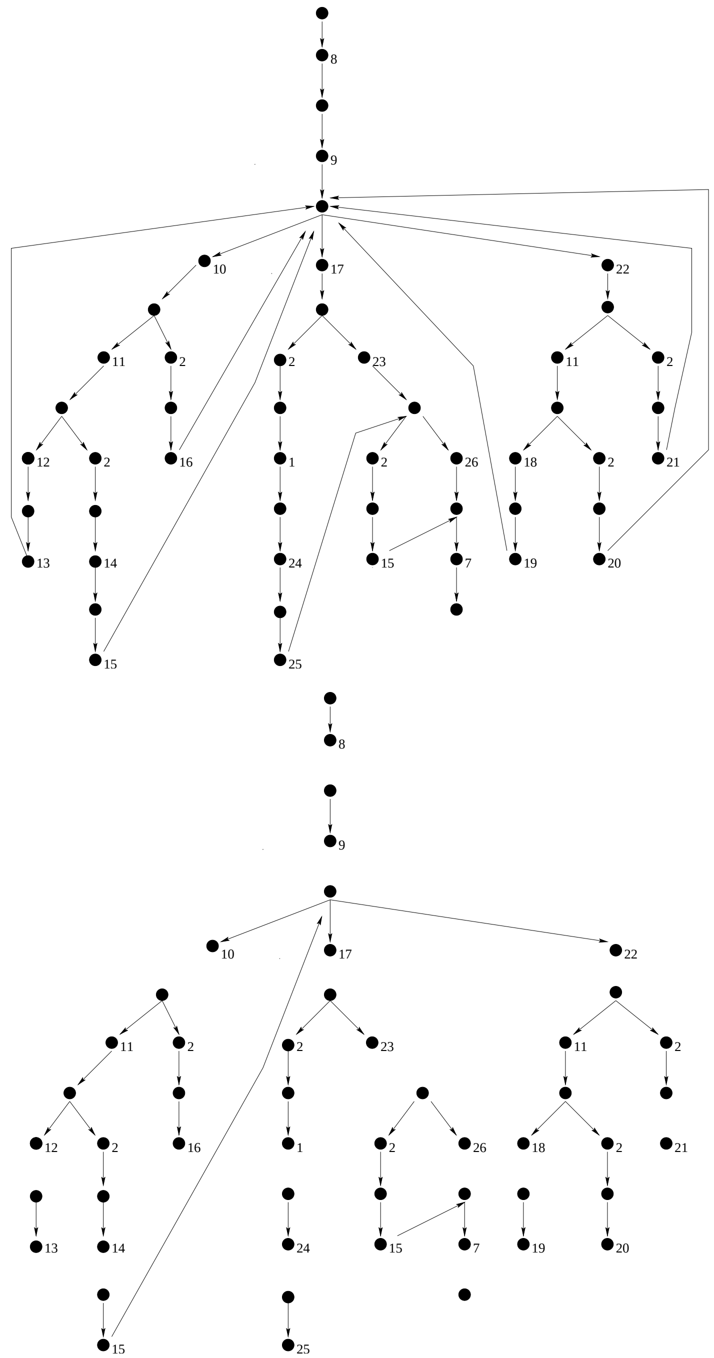
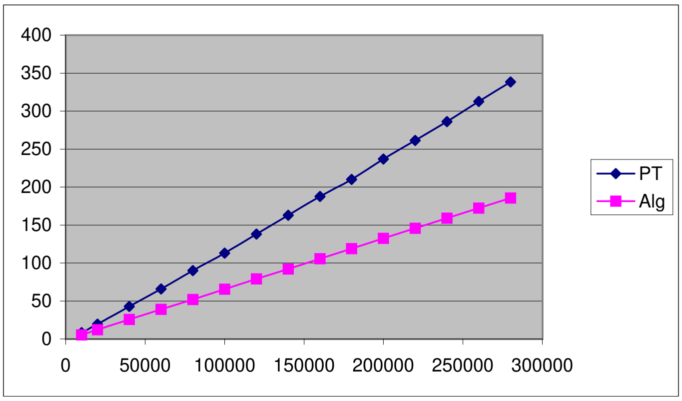
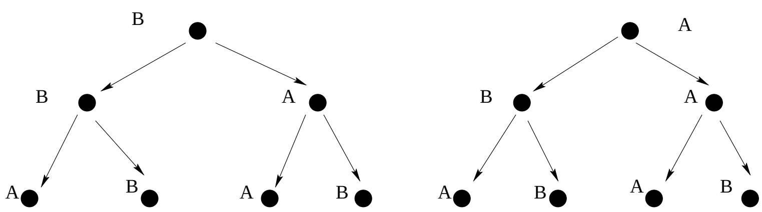


Figure 5.20: Bisimulation contraction of Node from Leader.

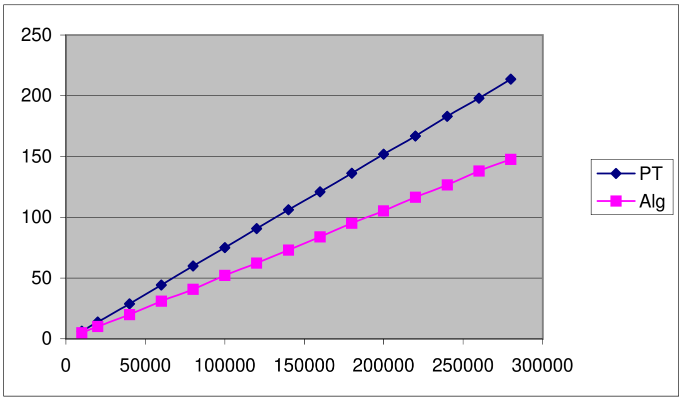
Nodes :	Alg	PT
80005	.5188	.9000
120005	.7893	1.3807
160005	1.0560	1.8773
200005	1.3250	2.3690
240005	1.5900	2.8620
280005	1.8540	3.3840



Test 2. Two Trees. Inspired by the previous Hopcroft's graph, we have tested the algorithm over graphs of the form:

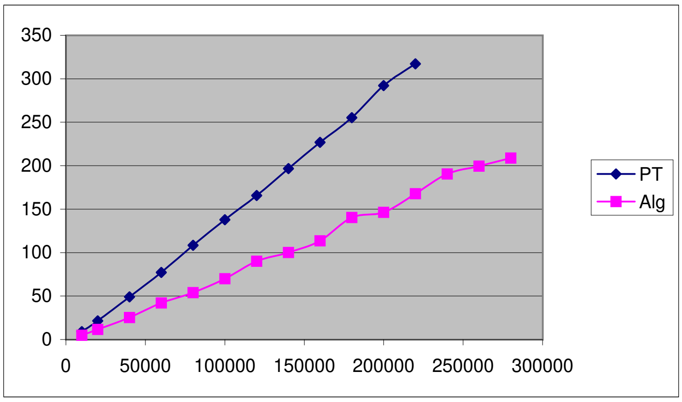


Nodes :	Alg	PT
80003	.4083	.6000
120003	.6240	.9070
160003	.8380	1.2100
200003	1.0530	1.5200
240003	1.2660	1.8300
280003	1.4760	2.1360



Test 3. Three labels function This is the graph of a function, i.e. each node has one outgoing-edge, with an initial partition of the nodes of the graph into three classes, A, B and C. Each node in $A \cup B$ reaches a node in C and each node in B reaches a node in A or a node in C. On this is the graph Paige-Tarjan's algorithm runs in time proportional to $|\geqslant| \log |N|$.

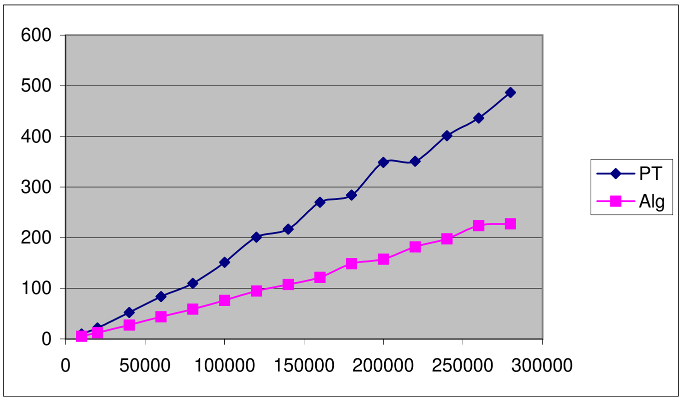
Nodes :	Alg	PT
80003	.5384	1.0852
120003	.9020	1.6575
160003	1.1353	2.2680
200003	1.4630	2.9230
240003	1.9060	ERR
280003	2.0880	ERR



The errors in Paige-Tarjan for more than 240000 nodes is due to the fact that in general Paige-Tarjan requires more memory, since it always works on the global graph.

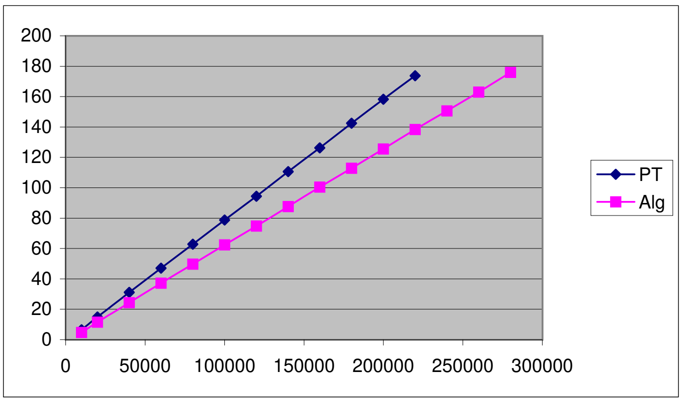
Test 4. Generalization of three labels function. This graph at each rank represents a function similar to the one in Test 3. of this section. All the nodes at rank i are also connected to nodes at rank $i-1$, hence the global graph is not the graph of a function.

Nodes :	Alg	PT
80006	.5872	1.0972
120006	.9473	2.0100
160006	1.2190	2.7000
200006	1.5740	3.4880
240006	1.9760	4.0120
280006	2.2740	4.8660



Test 5. Labelled function. This is a the graph of a function with an initial partition of the nodes chosen in such a way that there are few equivalence classes in the initial partition, but all the nodes are not bisimilar.

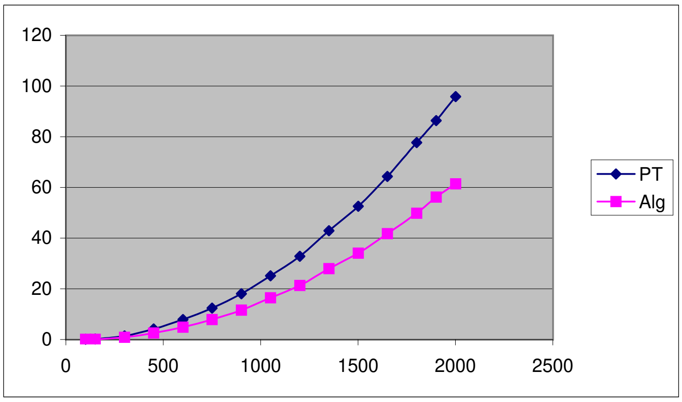
Nodes :	Alg	PT
80003	.4976	.6280
120003	.7485	.9435
160003	1.0047	1.2620
200003	1.2550	1.5830
240003	1.5060	ERR
280003	1.7600	ERR



Similarly to Test 3. of this section it is not possible to run Paige-Tarjan with more than 240000 nodes.

Test 6. Rank graph. In this graph the subgraph at rank i is a complete graph. The nodes at rank i are connected to the nodes at rank $i-1$ in such a way that all the nodes at rank i are bisimilar.

Nodes :	Alg	PT
601	.048	.079
901	.116	.181
1201	.213	.328
1501	.341	.525
1801	.498	.777
2001	.614	.958



5.8 Concluding Remarks

In this chapter we proposed algorithms to determine the minimal representation of an hereditarily finite non-well-founded set via a graph or, equivalently, to test bisimilarity between hereditarily finite non-well-founded sets. The algorithms are built making use of algorithmic solution to the relational and single function coarsest partition problems as subroutines. In the acyclic case the performance of the sketched algorithm is linear while, in the cyclic case turns out to be linear when no multiple nodes are present. In general its performance is no worst than that of the best known solution for the relational coarsest partition problem.

In [FV99], Fisler and Vardi compare three minimization algorithms with an invariance checking algorithm (which does not use minimization) and argue that the last is more efficient. The minimization algorithms they consider are those of Paige and Tarjan [PT87], of Bouajjani, Fernandez and Halbwachs [BFH90], and of Lee and Yannakakis [LY92]. An important conclusion they draw is that even if the last two algorithms are tailored to verification contexts, while the Paige and Tarjan one is not, the latter performs better. This suggests that *minimization algorithms tailored to verification settings should pay attention to choosing splitters carefully.* We have presented here an algorithm, which is not specifically tailored to verification, but whose main difference w.r.t. the Paige and Tarjan's one is that it performs better choices of the splitters and of the initial partition thanks to the use of the notion of rank. In some cases we obtain linear time runs, moreover the initial partition we use allows to process the input without storing the entire structure in memory at the same time.

Techniques to improve the complexity behavior in a wider family of graphs are under investigation and some possible optimization based on the topology of the subgraphs processed at each iteration are proposed.

In particular, the last of the topological optimization proposed can be applied considering not only the successors of a node but all the paths of length less than ℓ, for some constant ℓ. We are currently studying which are reasonable values of ℓ in order to obtain a real optimization.[3] Below is an example (cf. Figure 5.21) in which using such a kind of optimization our algorithm works in linear time. In the figure we show the result after the first iteration. In the following iteration it would possible to apply the linear strategy of the well-founded case. Applying the algorithm of Paige-Tarjan to this case it is possible to recognize that node 1 is not bisimilar to node 2 only after four iterations, while using our algorithm this is recognized after the first one because they have different rank.

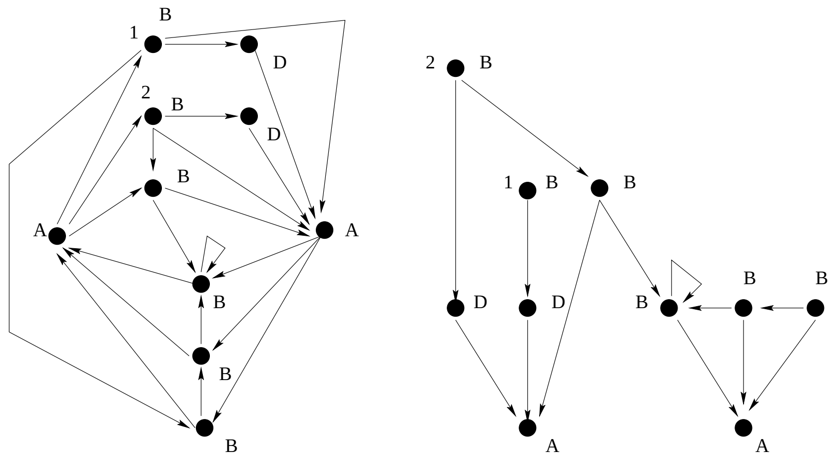


Figure 5.21: Example of a further topological optimization.

Notice also that in this example the second proposed topological optimization gives no results: the paths of length 1 do not allow us to establish that there is a node which is not bisimilar to any other node.

Other optimization, based on refinements on the introduced notion of rank are proposed and the problem of finding an optimal definition of rank with respect to the problem of determining the maximal bisimulation is under investigation.

Together with some master students we have realized an interface which allows to the C implementation of our algorithm to work with graphs written in FC2 format, this is one of the standard formats used in model checkers (see [Bou98]). Our next step will be to use FC2 graphs coming from real programs to test our algorithm. In particular, we intend to use programs written in the language ESTEREL for which the model checkers XEVE determines the models in FC2 format.

[3]If we do not put a bound on ℓ we obtain exactly the negative strategy (see Section 5.3).

Many other lines of research could be pursued using the same circle of ideas. For example, it would be interesting to extend and further study the connection and analogy with algorithms and problems considered by the scientific community studying security properties in the context of concurrent processes. Recently, connections between databases designed to easily access the web and non-well-founded sets has been pointed out in [LSY97]. For such applications, bisimulation-matching is the engine of the operational semantics of a graphical query language for web-like databases [CDQT98].

6

Ackermann Encoding, Bisimulations, and OBDD's

In this chapter we analyse the features of the data structures mainly used in Symbolic Model Checking: the ordered binary decision diagrams (OBDD's). The OBDD's were first introduced in [Lee59] and [Ake78] to represent Boolean functions, but their importance grew rapidly only when Bryant first, and McMillan later, suggested the possibility of their use in Model Checking to face the state explosion problem. In particular McMillan in [McM93] showed the effectiveness of Symbolic Model Checking (based on the use of OBDD's) to verify real systems.

First of all in this chapter we propose the use of a partitioning of the nodes of the graph in order to obtain a more compact symbolic representation. Then, we show how, when the partitioning used is exactly the one given by a notion of rank, we can in one single step compute $G/\equiv$ and provide its symbolic representation.

Actually, the order in the genesis of the material in this chapter is the opposite. We started in our analysis looking for a positive strategy to compute the maximum bisimulation. The *Ackermann encoding* gives in the well-founded case some hints in the direction of a positive strategy. The Ackermann encoding is a bijection between well-founded sets and natural numbers. It can be presented as a function from acyclic graphs into natural numbers, computed from the encoding of subgraphs and such that two nodes have the same encoding if and only if they are bisimilar.

Definition 6.1 (Ackermann Encoding) *Let $G = \langle N, \gtrdot \rangle$ be an acyclic graph, we define the* Ackermann encoding *of G, $Ack_G : N \to \mathbb{N}$ as follows:*

$$Ack_G(n) \quad = \quad \begin{cases} 0 & \textit{if } \forall m \in N \neg(n \gtrdot m) \\ \sum_{a \in A(n)} 2^a & \textit{otherwise} \end{cases}$$

where $A(n) = \{Ack_G(m) \mid n \gtrdot m\}$.

Notice that the definition makes sense since G is finite and acyclic.

It holds that

$$m \equiv n \quad \Leftrightarrow \quad Ack_G(m) = Ack_G(n).$$

Moreover, it is immediate to see that from the set of Ackermann encodings of the nodes of G it is possible to obtain $G/\equiv$. This means that in the well-founded case the Ackermann encoding provides a *numerical* algorithm to compute $G/\equiv$. The disadvantages of this method are that:

- it requires large numbers (i.e. long vectors of bits);
- it does not work on cyclic graphs.

The Ackermann encoding makes use of large number since the intention of Ackermann was to encode all the finite well-founded sets into natural numbers. Our problem is instead of a different nature: in the well-founded case, we have a fixed finite set of well-founded sets (each node in the graph represents a well-founded set) and we want to determine which ones among them are bisimilar.

The first idea we can apply in the well-founded case in order to avoid the use of large numbers is that of computing the Ackermann encoding by ranks and to reuse the codes at different ranks: if two nodes have different ranks, then we already know that they are not bisimilar (see Chapter 5), hence, without generating any confusion, we can assign to them the same encoding.

Moreover, if we look at the Ackermann encoding using the binary representation of natural numbers, it turns out, in a natural way how to use OBDD's in order to have a more compact representation. Let $a \in \mathbb{N}$ and $\underline{a}$ be its binary representation. Moreover let $\underline{a}[i]$ be the i-th digit of the binary representation of a, then

$$\underline{Ack_G(n)}[i] = 1 \quad \Leftrightarrow \quad n \gtrdot m \wedge Ack_G(m) = i.$$

Which can be represented in a more compact way using the Boolean function

$$\gtrdot(n) \quad = \quad \bigvee_{n \gtrdot m} \chi_{\{\underline{Ack_G(m)}\}}.$$

These are substantially the ideas behind the encoding we present in Section 6.3.

Finally in Section 6.4, where we generalize the encoding to the non-well-founded case: the main idea is to use two encodings for each node, depending on the side of the membership relation we are looking at.

6.1 OBDD's to Represent Graphs

The OBDD's, *ordered binary decision diagrams*, are a canonical representation of Boolean functions, i.e. two Boolean functions are equivalent if and only if they are associated to the same OBDD. They were first introduced in [Lee59] and in [Ake78], but it was the work of Bryant [Bry85] that attracted attention on the possibility of their use in *logic design verification*. All the details concerning how OBDD's represent Boolean functions can be found in [CGP99].

The way OBDD's are usually employed in Model Checking to represent the state space N, sets of states $S \subseteq N$, and the transition relation $\gtrdot$, is based on the following observations:

- we can safely assume that $N = \{0,1\}^u$, i.e. each node is a binary number;

- a set $S \subseteq N$ is a set of binary strings of length u, hence its characteristic function $\chi_S : \{0,1\}^u \to \{0,1\}$, where

$$\chi_S(s_1, \ldots, s_u) = 1 \quad \Leftrightarrow \quad \langle s_1, \ldots, s_u \rangle \in S$$

  is a binary function, which can be represented using an OBDD;

- $\gtrdot \subseteq N \times N$ is a set of binary strings of length $2u$, and hence, again, its characteristic function

$$\chi_{\gtrdot}(x_1, \ldots, x_u, y_1, \ldots, y_u) = 1 \quad \Leftrightarrow \quad \langle x_1, \ldots, x_u \rangle \gtrdot \langle y_1, \ldots, y_u \rangle$$

  is a binary function, which can be represented using an OBDD.

In particular, in the OBDD representing $\gtrdot$ (without variable reordering), the first u levels (variables) represent the codes of the source nodes, while the second u levels (variables) represent the codes of the target nodes (see Figure 6.1).

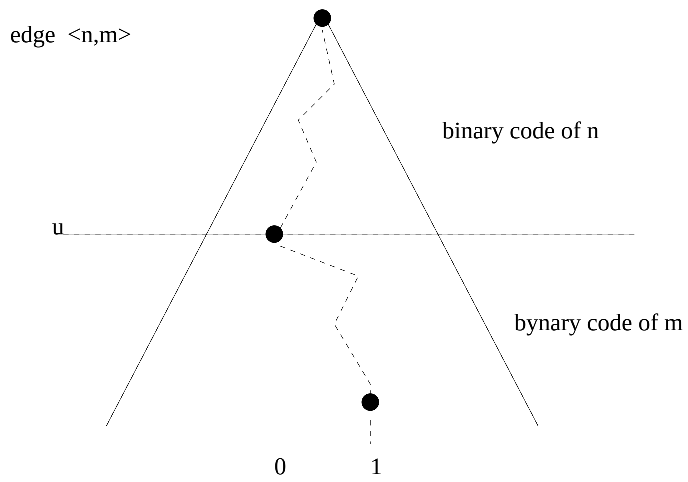


Figure 6.1: The OBDD representing $\gtrdot$.

Example 6.2 *Consider the graph G in Figure 6.2 on the left. Using the binary variables x_1 and x_2 for the code of the first node and the variables y_1 and y_2 for the code of the second node we obtain that the characteristic function of this graph is*

$$(\neg x_1 \wedge x_2 \wedge \neg y_1 \wedge \neg y_2) \vee (x_1 \wedge x_2 \wedge \neg y_1 \wedge y_2) \vee$$
$$(x_1 \wedge \neg x_2 \wedge \neg y_1 \wedge \neg y_2) \vee (x_1 \wedge \neg x_2 \wedge \neg y_1 \wedge y_2)$$

which is represented by the OBDD in Figure 6.2 on the right.

If we use this OBDD encoding to represent $G/\equiv$ (without using *variable reordering* or other minimization techniques) we always obtain that if $|N/\equiv| = 2^{v-1}$, then the higher half of the OBDD is a complete tree with $2^{v-1} - 1$ nodes at the level corresponding to y_1 (the first variable relative to the encoding of the target node) (see Example 6.2). This is because two paths of length v which converge would represent two or more different nodes whose sets of $\gtrdot$-successors are the same. However, since different nodes in $G/\equiv$ are not bisimilar,

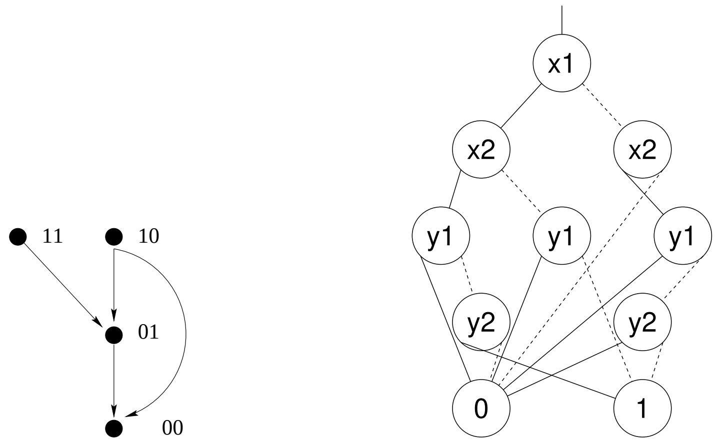


Figure 6.2: Example.

different nodes cannot have the same set of $\gg$-successors[1]. In case G is not quotiented w.r.t. bisimulation, we obtain that (without using variable reordering) in the OBDD representing $\gg$, at level v there are at least $|N/\equiv| - 1$ different nodes.

A second disadvantage of this way (rather classical) of using OBDD's to represent a graph, is given by the fact that *topological* repetitions in the structure of the graphs are not exploited in order to obtain further reductions. OBDD's reduce the dimensions of a graph *horizontally*: nodes of the OBDD at the same level are collapsed when the sub-OBDD's rooted on them are equal. A situation as the one described in the following example is, in general, not exploited.

Example 6.3 *Consider the graph in Figure 6.3 on the left. The situation in the part A and*

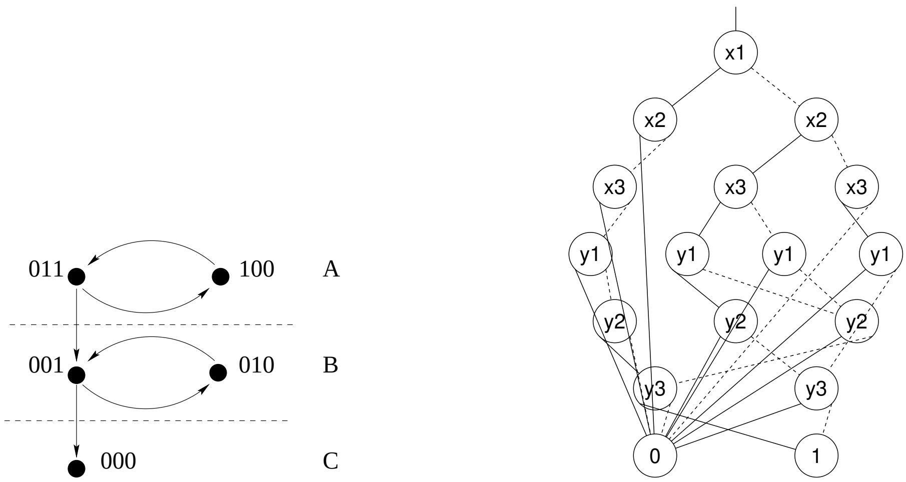


Figure 6.3: Example.

B of the graph is similar, but this has no reduction effects on the OBDD (see Figure 6.3 on the right) which at level 3 *has* $|N| - 1$ *nodes.*

[1]If there is a node without outgoing edges, then its path reaches directly 0, this is the reason for which we have $2^{v-1} - 1$ instead of 2^{v-1} nodes at level v

A *vertical* reduction is performed only when it is possible to jump some levels, which means *do not care of the values of the variables of the levels that are jumped.* But, in general the global hight of the OBDD remains $2u$, which implies that it is possible to have $O(2^{2u})$ internal nodes in it.

What we try to do in the next section is to propose an alternative way to use OBDD's in order to represent graphs in such a way to obtain:

- a vertical reduction, i.e. we deal with OBDD's shorter than $2u$;
- use topological similarity to obtain a further horizontal reduction.

6.2 An Alternative use of OBDD's to Represent Graphs

As we already observed, if we are able to work on a graph G reduced by bisimulation, we can safely assume that no two different nodes have the same set of successors with respect to $\gtrdot$. Hence, the graph can be represented as a *collection of sets of* $\gtrdot$*-successors* (one for each node).

Moreover, if we have a partition P on the the graph, each *block* B_j (P-class) of the partition can be treated separately, allowing sharing among the OBDD's used in the representation.

Below we will combine the above two ideas into an OBDD's representation minimizing the size of the data structures involved. The technique will turn out to be suitable for use even *while* the quotient structure $G/\equiv$ is determined starting from G (cf. Sections 6.3 and 6.4).

The following definitions will be used:

- let $P = \{B_1, \ldots, B_p\}$ be a partition of N;
- for each block $B_j \in P$ consider the list

  $$Dom_j \quad = \quad [m \mid \exists n \in B_j (n \gtrdot m)],$$

  of the nodes reached from a node in B_j. Let the nodes in Dom_j be ordered using their binary codes and let $|Dom_j| = h_j$;
- for each $m \in Dom_j$ let $d_j(m)$ be the binary representation of the position in which m occurs in Dom_j ($d(m)$ is in $\{0,1\}^{\lfloor \log h_j \rfloor + 1}$);
- for each $n \in B_j$ consider the Boolean function

  $$\gtrdot_j(n)(z_1, \ldots, z_{\lfloor \log h_j \rfloor + 1}) \quad = \quad \bigvee_{n \gtrdot m} \chi_{\{d_j(m)\}}.$$

The notation Dom_j comes from the fact that the Boolean functions $\gtrdot_j$'s can be seen as function from the *domain* Dom_j into $\{0, 1\}$. It is clear that if we know P, all the Dom_j's, and all the $\gtrdot_j(n)$, we are able to infer $\gtrdot$. The data structure that can be used to keep all these information are the following:

- the binary encodings for the nodes of N in which we assume that in the binary encoding of $n \in N$ the first $q = \lfloor \log p \rfloor + 1$ digits represent the block B_j to which n belongs;

- the list Dom_j is equivalent to the set of binary strings

$$SDom_j \quad = \quad \{\langle m, d_j(m)\rangle \mid m \in Dom_j\}$$

  whose characteristic function χ_{SDom_j} can be represented using an OBDD (the S in $SDom_j$ stand for the fact that $SDom_j$ is a *set*);

- the Boolean function $\geqslant_j(n)$ can be represented using an OBDD (to which n points).

We keep all the OBDD's in a unique table and we never duplicate a sub-OBDD, i.e. it is possible that the OBDD of $SDom_j$ and the OBDD of $SDom_{j'}$ share some sub-OBDD's. In particular, whenever $SDom_j$ and $SDom_{j'}$ have the same cardinality they share all the levels relative to the $d_j(m)$'s.

Notice that if $P = \{N\}$, then our representation is similar to the one presented in the previous section: for each node n we give the OBDD rooted at the end of the binary code of n. In the case $P = \{N\}$ the connection between the representation presented in the previous section and the representation we propose is similar to the connection between the adjacency-matrix representation and the adjacency-list representation.

Using our representation with a non-trivial partition P:

- the OBDD's of the functions $\geqslant_j(n)$'s are shorter than u (vertical reduction);

- it is possible that n is not bisimilar to n', but $\geqslant_j(n) = \geqslant_{j'}(n')$, i.e. they point to the same OBDD (horizontal reduction due to topological repetitions).

It immediately follows that the set of nodes which refer to the same $OBDD$ can be itself represented using an OBDD of hight u.

Example 6.4 *Consider the graph in Figure 6.3 (see Example 6.3). Using the partition $P = \{A, B, C\}$ we obtain.*

$$Dom_C = \emptyset \quad Dom_B = [000, 001, 010] \quad Dom_A = [001, 011, 100]$$

$$\begin{array}{ll} \geqslant_C(000) = 0 & \\ \geqslant_B(001) = (\neg z_1 \wedge \neg z_2) \vee (z_1 \wedge \neg z_2) & \geqslant_B(010) = (\neg z_1 \wedge z_2) \\ \geqslant_A(011) = (\neg z_1 \wedge \neg z_2) \vee (z_1 \wedge \neg z_2) & \geqslant_A(100) = (\neg z_1 \wedge z_2) \end{array}$$

Hence 001 *and* 011 *share the same OBDD (and similarly* 010 *and* 100*), in particular* 3 *nodes are sufficient. In Figure 6.4 we show the OBDD's relative to $SDom_A$ and $SDom_B$ which can share completely the last two levels.*

Notice we can still use all the optimization techniques that are provided to deal with OBDD's, and, moreover, we are not forced to keep all the OBDD's in central memory.

If $|N| = 2^u - 1$, *out* is the maximum number of outgoing edges from a node, and we partition N in $2^q - 1$ classes with $q \sim u$ (using the first q digits), then we obtain that:

- in each class there are $2^{u-q} - 1 = k$ elements, with $k << |N|$;

- each Dom_j has at most $out * k$ elements;

- the OBDD's of the functions $\geqslant_j(n)$'s have at most $out * k$ levels.

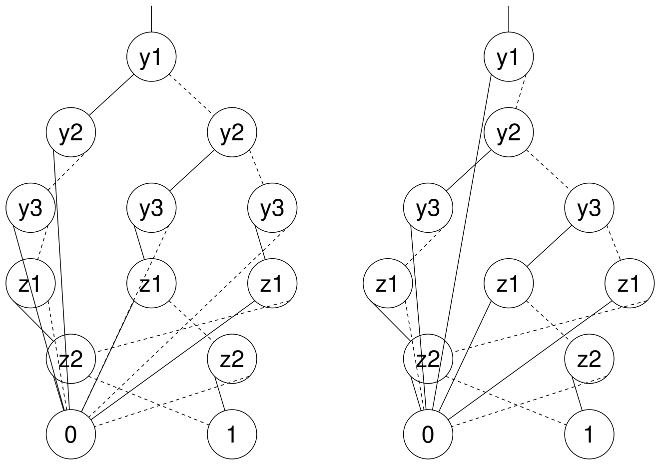


Figure 6.4: Example.

If $out << |N|$, then we obtain that $out * k$ can be considered as a constant w.r.t. $|N| = 2^u - 1$, the OBDD's of $SDom_j$'s have about u levels[2], and a large number of OBDD's of $\gtrdot_j(n)$'s are shared.

Our representation achieve good results in particular in the following cases: there are few outgoing edges from each node; there are topological repetitions. This happens for instance in the case of graphs obtained from programs to be analysed when: a procedure is used more than once; different procedures performs symmetric actions; there are sequences of deterministic iterations. In the graph obtained from the Mutual Exclusion program (see [CGP99] pag. 25), for instance, both horizontal and vertical symmetries can be used to partition the graph.

The operations on the representation we propose are similar to the ones on the classical representation with OBDD's. In this section we briefly discuss how to compute the image of a set of states and the X operation (counter-image).

Let $S \subseteq N$ be a set of states. Assume that we want to compute $\gtrdot(S) = \{m \mid \exists n \in S(n \gtrdot m)\}$. We have that $S = S_1 \cup \ldots \cup S_h$, where each S_i is a subset of a block in P and we obtain that

$$\chi_{\gtrdot(S)}(\vec{y}) \quad = \quad \bigvee_{i=1}^{h} \exists \vec{z}[(\bigvee_{n \in S_i} \gtrdot_i(n)(\vec{z})) \wedge \chi_{SDom_i}(\vec{y}, \vec{z})]$$

If we want to compute $\mathrm{X}(S) = \{m \mid \exists n \in S(m \gtrdot n)\}$ (next operator of CTL), we have to compute

$$\begin{aligned} g_B(\vec{z}) &= \exists \vec{y}(\chi_S(\vec{y}) \wedge \chi_{SDom_B}(\vec{y}, \vec{z})) \\ \chi_{\mathrm{X}(S)}(\vec{x}) &= \bigvee_{B \in P} \bigvee_{\vec{x} \in B} (\exists \vec{z}((g_B(\vec{z}) \wedge \gtrdot_B(\vec{x})(\vec{z})) \leftrightarrow \vec{x})) \end{aligned}$$

Using similar Boolean functions it is possible to move from one representation to the other and vice-versa. In the next section we concentrate on the fact that our representation together with a correct partitioning of the graph can be used to obtain in one single step, starting from a representation of G a symbolic representation of $G/\equiv$.

[2]Exactly $u + \lfloor \log(out * k) \rfloor + 1$ levels.

6.3 A Compact Encoding for the Well-Founded Sets

In order to obtain the OBDD's representation *while* computing G, we will use an idea that can be traced back to the so-called Ackermann encoding of hereditarily finite sets into natural numbers [Ack37, Lev79]. Such an encoding, inductively defined as $A(h) = \Sigma_{h' \in h} 2^{A(h')}$, establishes a bijection between the collections of hereditarily finite sets and the natural numbers. Hence, two sets (possibly represented by different means) will be bisimilar if and only if they are mapped by A into the same natural number. Therefore, the computation of A and the bisimulation relation are naturally carried on *together*. In our approach we essentially redefine A using OBDD's in place of natural numbers and layering the definition on the ranks in order to take advantage of repetitions as illustrated in Section 6.2.

Our OBDD encoding of $G/\equiv$ is defined rank by rank as follows:

1. at each rank i we determine which are the nodes reachable from all nodes at rank i, i.e. the list Dom_i;
2. then, we compute for each node n at rank i the OBDD representing $\gg_i(n)$ (two nodes n and n' at rank i are bisimilar if and only if $\gg_i(n) = \gg_i(n')$);
3. subsequently we collect all the OBDD's obtained at rank i in the list Cod_i (without repetitions) and we assign to each node n at rank i the number $\mathbb{A}(n)$ which is the position of $\gg_i(n)$ in Cod_i;
4. for each node n at rank i its binary encoding is replaced with $\langle i, \mathbb{A}(n)\rangle$.

Hence, two nodes n and n' at rank i are bisimilar if and only if $\mathbb{A}(n) = \mathbb{A}(n')$ and in general two nodes n and n' are bisimilar if and only if $\langle rank(n), \mathbb{A}(n)\rangle = \langle rank(n'), \mathbb{A}(n')\rangle$. For this reason we introduce the *encoding pair* of a node n as the pair $\langle rank(n), \mathbb{A}(n)\rangle$.

Definition 6.5 (OBDD Encoding) *Let $G = \langle N, \gg\rangle$ be an acyclic graph. For each rank i, and each node n of G we define, by induction on the rank, Dom_i, Cod_i, $\gg_i(n)$, and $\mathbb{A}(n)$ as follows:*

$$rank = i \quad \Rightarrow \quad \begin{cases} Dom_i = [\langle rank(m), \mathbb{A}(m)\rangle \mid \exists n(rank(n) = i \land n \gg m)] \\ \gg_i(n) = \bigvee_{n \gg m} \chi_{\{d_i(m)\}} \\ Cod_i = [\gg_i(n) \mid rank(n) = i] \\ \mathbb{A}(n) = k \quad \mathit{iff} \quad Cod_i[k] = \gg_i(n) \end{cases}$$

where Dom_i and Cod_i are ordered lists, $\gg_i(n)$ is an OBDD, and $\mathbb{A}(n)$ is a natural number.

Theorem 6.6 *Let $G = \langle N, \gg\rangle$ be an acyclic graph and $n, n' \in N$ two nodes.*

$$\langle rank(n), \mathbb{A}(n)\rangle = \langle rank(n'), \mathbb{A}(n')\rangle \quad \mathit{iff} \quad n \equiv n'.$$

Proof. ($\Rightarrow$) Consider the relation B defined as nBn' if and only if $\langle rank(n), \mathbb{A}(n)\rangle = \langle rank(n'), \mathbb{A}(n')\rangle$. We prove that B is a bisimulation. If $rank(n) = rank(n') = 0$, then nBn' satisfies the forward and the back condition, since n and n' have no successors. If $rank(n) = rank(n') = i$ and $\mathbb{A}(n) = \mathbb{A}(n')$, then $\gg_i(n) = \gg_i(n')$, hence for each m such that $n \gg m$ and $\langle rank(m), \mathbb{A}(m)\rangle$ is the j^{th} element of Dom_i there exists m' such that $n' \gg m'$ and $\langle rank(m'), \mathbb{A}(m')\rangle$ is the j^{th} element of Dom_i, hence we have that mBm'. Similarly we can prove that for each m' such that $n' \gg m'$ there exists m such that $n \gg m$ and mBm', hence we have that nBn' satisfies the forward and the back conditions. From the fact that B is a bisimulation we have that it is included in $\equiv$, i.e. the thesis.

($\Leftarrow$) By induction on the rank. If $rank(n) = 0$ and $n \equiv n'$, then from Lemma 5.23 we have that $rank(n') = 0$, hence $\mathbb{A}(n) = \mathbb{A}(n') = 0$. If $rank(n) = i$ and $n \equiv n'$, then from Lemma 5.23 we have that $rank(n') = i$, hence we have to prove that $\geqslant_i(n) = \geqslant_i(n')$. If $\chi_{\{d_i(m)\}}$ is a disjoint in $\geqslant_i(n)$, then there exists m such that $n \geqslant m$ and $\langle rank(m), \mathbb{A}(m)\rangle$ is the $d_i(m)^{th}$ element of Dom_i. Since $n \equiv n'$, there exists m' such that $n' \geqslant m'$ and $m \equiv m'$. By inductive hypothesis, since $rank(m) < i$, we obtain that $\langle rank(m), \mathbb{A}(m)\rangle = \langle rank(m'), \mathbb{A}(m')\rangle$, hence $\chi_{\{d_i(m)\}}$ is a disjoint in $\geqslant_i(n')$. Similarly we can prove that if $\chi_{\{d_i(m')\}}$ is a disjoint in $\geqslant_i(n')$, then it is also a disjoint in $\geqslant_i(n)$, from which $\geqslant_i(n) = \geqslant_i(n')$ and hence, $\mathbb{A}(n) = \mathbb{A}(n')$. $\square$

Example 6.7 *Consider the graph G in Figure 6.5*

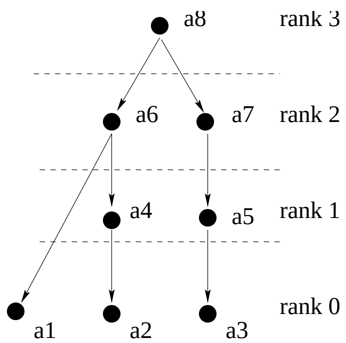


Figure 6.5: Example.

At rank 0 *we obtain:*

$$\begin{array}{l} Dom_0 = \emptyset \\ \geqslant_0(a_1) = \geqslant_0(a_2) = \geqslant(a_3) = \bot \\ \mathbb{A}(a_1) = \mathbb{A}(a_2) = \mathbb{A}(a_3) = 0 \end{array}$$

At rank 1*:*

$$\begin{array}{ll} Dom_1 = [\langle 0,0\rangle] & SDom_1 = \{\langle 0,0,0\rangle\} \\ \geqslant_1(a_4) = \geqslant_1(a_5) = \neg z_1 & \mathbb{A}(a_4) = \mathbb{A}(a_5) = 0 \end{array}$$

At rank 2*:*

$$\begin{array}{ll} Dom_2 = [\langle 0,0\rangle, \langle 1,0\rangle] & SDom_2 = \{\langle 0,0,0\rangle, \langle 1,0,1\rangle\} \\ \geqslant_2(a_6) = \neg z_1 \vee z_1,\ \geqslant_2(a_7) = z_1 & \mathbb{A}(a_6) = 0,\ \mathbb{A}(a_7) = 1 \end{array}$$

At rank 3*:*

$$\begin{array}{ll} Dom_1 = [\langle 2,0\rangle, \langle 2,1\rangle] & SDom_1 = \{\langle 2,0,0\rangle, \langle 2,1,1\rangle\} \\ \geqslant_1(a_8) = \neg z_1 \vee z_1 & \mathbb{A}(a_8) = 0 \end{array}$$

Notice that the encoding we obtain in the acyclic case is an encoding *à la* Ackermann (see [Ack37]) with two more ingredients: since we are encoding only a particular graph, and not all the possible acyclic graphs, we keep the encoding more compact (to this end we use the notion of rank); instead of using the exponential function to compute the encoding, we use OBDD's.

6.4 The Encoding on Non-Well-Founded Sets

In order to generalize our encoding to the cyclic case first of all we need to use the generalization of the notion of rank presented in the previous chapter (cf. Definition 5.27). The notion of rank satisfies the following property (cf. Lemma 5.28):
let G be a graph and $n, n' \in N$ be two nodes of G.

$$n \equiv n' \quad \Rightarrow \quad \mathit{rank}(n) = \mathit{rank}(n').$$

We also recall that it also holds that

$$n \gtrdot m \quad \Rightarrow \quad \mathit{rank}(m) \leq \mathit{rank}(n).$$

In particular, as pointed out in the Chapter 5, in order to determine if two nodes of rank i are bisimilar it is sufficient to know the bisimulation $\equiv$ on the nodes of rank less than i and the edges between the nodes at rank i.

The main idea behind the extension of our way to compute the OBDD encoding in the cyclic case is that of assigning to each node n *two* encodings. This means that we treat $n \gtrdot m$ in an *asymmetric* way with respect to $\gtrdot$. In particular the two encodings are $\gtrdot_i(n)$ and $d(n)$ and we compute $\gtrdot_i(n)$ (lhs of $n \gtrdot m$) using $d(m)$ (rhs of $n \gtrdot m$).

Let us assume that for each $j < i$ we have correctly encoded all the nodes of rank j, i.e. we have assigned to each node n of rank j a number $\mathbb{A}(n)$ in such a way that

$$n \equiv n' \quad \Leftrightarrow \quad \langle \mathit{rank}(n), \mathbb{A}(n)\rangle = \langle \mathit{rank}(n'), \mathbb{A}(n')\rangle.$$

We want to extend the encoding to the nodes at rank i.
First we compute Dom'_i, all the $\gtrdot'_i(n)$'s, Cod'_i, and all the $\mathbb{A}'(n)$'s as in the acyclic case, but without considering the nodes at rank i. This means that $\gtrdot'_i(n)$ and $\mathbb{A}'(n)$ are a first approximation of $\gtrdot_i(n)$ and $\mathbb{A}(n)$ in which we consider only the edges which reach a node m whose rank is less than i. In particular:

$$\mathit{Dom}'_i \quad = \quad [\langle \mathit{rank}(m), \mathbb{A}(m)\rangle \mid \exists n(\mathit{rank}(n) = i \wedge n \gtrdot m \wedge \mathit{rank}(m) < i)],$$

$$\gtrdot'_i(n)(\vec{z}) \quad = \quad \bigvee_{n \gtrdot m \wedge m \in \mathit{Dom}'_i} \chi_{\{d_i(m)\}}(\vec{z}),$$

$$\mathit{Cod}'_i \quad = \quad [\gtrdot'_i(n) \mid \mathit{rank}(n) = i], \text{ and}$$

$$\mathbb{A}'(n) \quad = \quad k \text{ iff } \mathit{Cod}_i[k] = \gtrdot'_i(n).$$

For each node n at rank i consider a variable $\mathtt{W}_\mathtt{n}^{\gtrdot_i}$ (which ranges over Boolean functions) and a variable $\mathtt{d_n}$ (which ranges over natural numbers). If there are r_i nodes at rank i, then we impose

$$\mathtt{d_n} \in \{|\mathit{Dom}'_i| + 1, \ldots, |\mathit{Dom}'_i| + r_i\}.$$

We consider all the Boolean equations

$$\mathtt{W}_\mathtt{n}^{\gtrdot_i}(\vec{z}) \quad = \quad \left(\bigvee_{n \gtrdot m \wedge \mathit{rank}(m) = i} \chi_{\{\mathtt{d_m}\}}(\vec{z}) \right) \vee \gtrdot'_i(n)(\vec{z}).$$

Notice that the second disjoint in the definition of $\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}}$ is a Boolean function, while we cannot explicitly write the first part until we do not know the values of the $\mathtt{d}_\mathtt{n}$'s. Notice also that it is possible to replace the Boolean equation with a numeric equation using the following definition

$$\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} = \sum_{j=|Dom'_i|+1}^{|Dom'_i|+r_i} 2^j * min\left(\sum_{n \geqslant m \wedge rank(m)=i} 1 - \frac{|\mathtt{d}_\mathtt{m} - j|}{max(|\mathtt{d}_\mathtt{m} - j|, 1)}, 1\right) + 2^{d'(n)},$$

with $d'(n) = \mathbb{A}'(n) + |Dom'_i| + r_i + 1$. This means that if $S = \{\geqslant_i(n), d(n) \mid rank(n) = i\}$ is a solution of this system of Boolean equations, then $\geqslant_i(n)$ is the OBDD associated to n and $d(n)$ is the position of n in Dom_i. Moreover we consider all the Boolean equations of the form

$$(\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} \leftrightarrow \mathtt{W}_{\mathtt{n}'}^{\geqslant\mathtt{i}}) \quad \leftrightarrow \quad \mathtt{d}_\mathtt{n} = \mathtt{d}_{\mathtt{n}'}.$$

which can be expressed as numeric equations as:

$$\frac{|\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} - \mathtt{W}_{\mathtt{n}'}^{\geqslant\mathtt{i}}|}{max(|\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} - \mathtt{W}_{\mathtt{n}'}^{\geqslant\mathtt{i}}|, 1)} + \frac{|\mathtt{d}_\mathtt{n} - \mathtt{d}_{\mathtt{n}'}|}{max(|\mathtt{d}_\mathtt{n} - \mathtt{d}_{\mathtt{n}'}|, 1)} = 2 * \frac{|\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} - \mathtt{W}_{\mathtt{n}'}^{\geqslant\mathtt{i}}|}{max(|\mathtt{W}_\mathtt{n}^{\geqslant\mathtt{i}} - \mathtt{W}_{\mathtt{n}'}^{\geqslant\mathtt{i}}|, 1)}.$$

We put as objective function to be minimized

$$max\{\mathtt{d}_\mathtt{n} \mid rank(n) = i\},$$

i.e. we want to maximize the number of equalities between the $\mathtt{d}_\mathtt{n}$'s. Let

$$S = \{\geqslant_i(n), d(n) \mid rank(n) = i\}$$

be a solution of the system of equations which minimize the objective function. We assign to each node n of rank i the code

$$\mathbb{A}(n) = d(n) - (|Dom'_i| + 1).$$

Notice that we always obtain that all the nodes n at rank -1 have $\mathbb{A}(a) = 0$. This is correct, since we are working on unlabelled graphs.

First we have to prove that the system has always a solution. We prove this by proving that from the maximal bisimulation we are able to describe a solution.

Lemma 6.8 *Let G be a graph, let Sys be the system of equations obtained from the nodes at rank i as described above, and let $\equiv$ be the maximum bisimulation over G. Consider an ordering ord of the equivalence classes of $G/\equiv$ at rank i and for each node n at rank i define*

$$\begin{aligned} d_i(n) &= ord([n]) + |Dom'_i| \\ \geqslant_i(n) &= \left(\bigwedge\nolimits_{n \geqslant m \wedge rank(m)=i} \chi_{\{d(m)\}}\right) \vee \geqslant'_i(n). \end{aligned}$$

The set $S = \{\geqslant_i(n), d_i(n) \mid rank(n) = i\}$ is a solution of Sys.

Proof. We recall that we are assuming that for all the nodes at rank less than i we have that

$$n \equiv n' \quad \Leftrightarrow \quad \langle rank(n), \mathbb{A}(n)\rangle = \langle rank(n'), \mathbb{A}(n')\rangle.$$

By induction on the rank.

Let $i = -1$. Since $\equiv$ is the maximum bisimulation all the nodes at rank -1 are bisimilar (see Chapter 5), hence there is only one equivalence class and all the $d(n)$ are equal to 1. We have $\geqslant'_{-1}(n) = \bot$ for all the nodes at rank -1. All the equations of the form $\mathtt{W}_{\mathtt{n}}^{\geqslant_{-1}} = \dots$ are satisfied, since we use them to define the $\geqslant_{-1}(n)$'s in terms of the $d_{-1}(n)$'s. We have to prove that $\geqslant_{-1}(n) = \geqslant_{-1}(n') \leftrightarrow d_{-1}(n) = d_{-1}(n')$. So what we have to prove is that all the $\geqslant_{-1}(n)$'s are equal. If n is a node at rank -1, then there exists at least a node m such that $n \geqslant m$ and $rank(m) = -1$. Hence, if $rank(n) = -1$ we obtain $\geqslant_{-1}(n) = \chi_{\{1\}}$, i.e. all the $\geqslant_{-1}(n)$'s are equal.

Let $i = 0$. Since $\equiv$ is the maximum bisimulation all the nodes at rank 0 are bisimilar (see Chapter 5), hence there is only one equivalence class and all the $d_0(n)$ are equal to 1. All the equations of the form $\mathtt{W}_{\mathtt{n}}^{\geqslant_{-1}} = \dots$ are satisfied, since we use them to define the $\geqslant_0(n)$'s in terms of the $d_0(n)$'s. We have to prove that $\geqslant_0(n) = \geqslant_0(n') \leftrightarrow d_0(n) = d_0(n')$. So what we have to prove is that all the $\geqslant_0(n)$'s are equal. If n is a node at rank 0, then it has no outgoing edges. Hence, if $rank(n) = 0$ we obtain $\geqslant_0(n) = \bot$, i.e. all the $\geqslant_0(n)$'s are equal.

Let $i = k+1$. All the equations of the form $\mathtt{W}_{\mathtt{n}}^{\geqslant_{\mathtt{i}}} = \dots$ are satisfied, since we use them to define the $\geqslant_i(n)$'s in terms of the $d_i(n)$'s. We have to prove that $\geqslant_i(n) = \geqslant_i(n') \leftrightarrow d_i(n) = d_i(n')^*$. If $\geqslant_i(n) = \geqslant_i(n')$, then we have to prove that $d_i(n) = d_i(n')$, i.e. we have to prove that $n \equiv n'$. If $n \geqslant m$ and $rank(m) < i$, then in $\geqslant_i(n)$ there is a disjoint of the form $\chi_{\{d_i(m)\}}$ with $d_i(m) \leq |Dom'_i|$. The same disjoint is in $\geqslant_i(n')$, and hence we obtain that there exists m' such that $m \equiv m'$ and $n' \geqslant m'$. If $n \geqslant m$ and $rank(m) = i$, then in $\geqslant_i(n)$ there is a disjoint of the form $\chi_{\{}d_i(m)\}$ with $d_i(m) > |Dom'_i|$, hence the same disjoint must be in $\geqslant_i(n')$, i.e. (from the definition of $d_i(m)$) there exists m' such that $n' \geqslant m'$ and $m' \equiv m$. Similarly it is possible to prove the back condition. Hence we have $n \equiv n'$, i.e. $d_i(n) = d_i(n')$. If $d_i(n) = d_i(n')$, then $n \equiv n'$. We have to prove that $\geqslant_i(n) = \geqslant_i(n')$. If in $\geqslant_i(n)$ there is a component of the form $\chi_{\{d_i(m)\}}$ with $d_i(m) \leq |Dom'_i|$, then $n \geqslant m$ and $rank(m) < i$, hence there exists m' such that $n' \geqslant m'$ and $m \equiv m'$, i.e. $d_i(m) = d_i(m')$ from which we obtain that in $\geqslant_i(n')$ there is a component of the form $\chi_{\{d_i(m)\}}$. If in $\geqslant_i(n)$ there is a component of the form $\chi_{\{d_i(m)\}}$, with $d_i(m) > |Dom_i|$, then since $n \equiv n'$, there must be m' such that $n' \geqslant m'$, and $m \equiv m'$, from which we obtain that in $\geqslant_i(n')$ there is a component of the form $\chi_{\{d_i(m)\}}$. Similarly it is possible to prove that all the components that are in $\geqslant_i(n')$ are also in $\geqslant_i(n)$, from which we obtain that $\geqslant_i(n) = \geqslant_i(n')$. $\square$

We have to prove that with such an encoding we obtain a representation of $G/\equiv$.

Theorem 6.9 *Let $G = \langle N, \geqslant \rangle$ be a graph and $n, n' \in N$ be two nodes.*

$$\langle rank(n), \mathbb{A}(n) \rangle = \langle rank(n'), \mathbb{A}(n') \rangle \quad \textit{iff} \quad n \equiv n'.$$

Proof. $(\Rightarrow)$ Let B be the relation defined as

$$nBn' \quad \text{iff} \quad \langle rank(n), \mathbb{A}(n) \rangle = \langle rank(n'), \mathbb{A}(n') \rangle.$$

We prove that B is a bisimulation. If $rank(n) = -1$ and nBn', then we have nothing to prove since all the equivalence relations between nodes at rank -1 are bisimulations. If $rank(n) = 0$ and nBn', then we have nothing to prove (all the equivalence relations are bisimulations). Let $rank(n) = i$ and nBn'. If $n \geqslant m$ and $rank(m) < i$, then in $\geqslant_i(n)$ there is a disjoint of the form $\chi_{\{d_i(m)\}}$ with $d_i(m) \leq |Dom'_i|$. The same component occurs in $\geqslant_i(n')$, hence there must exists m' such that $n' \geqslant m'$, $rank(m') < i$ and $\langle rank(m), \mathbb{A}(m) \rangle = \langle rank(m'), \mathbb{A}(m') \rangle$, from which we obtain that $m'Bm$. If $n \geqslant m$ and $rank(m) = i$, then in $\geqslant_i(n)$ there is a component of the form $\chi_{\{d_i(m)\}}$ with $d_i(m) > |Dom'_i|$. Since $\gtrdot_i(n') = \gtrdot_i(n)$ we obtain that in $\gtrdot_i(n')$ there is a component of the form $\chi_{\{d_i(m)\}}$. This implies that there exists m' such that $n' \gtrdot m'$, $rank(m') = i$ and $d_i(m) = d_i(m')$ in the solution of the system. From the fact that $d_i(m) = d_i(m')$ we obtain $\gtrdot_i(m) = \gtrdot_i(m')$, hence mBm'. Similarly it is possible to prove the back condition.

($\Leftarrow$) From Lemma 6.8 we have that the maximal bisimulation give us a solution. From each solution we are able to define a bisimulation B such that $max\{d_i(n) \mid rank(n) = i\} + 1$ is the number of classes at rank i (see ($\Rightarrow$)). Hence, it must be that the solution obtained from $\equiv$ minimize $max\{d_i(n) \mid rank(n) = i\}$ at each rank. $\square$

Notice that in the numeric system we use the exponential function $2^{(m)}$ in order to compute $\gtrdot_i(n)$. We could have used any other function f such that:

- $f([m, m|R]) = f([m|R])$ (f does not depend on the number of repetitions);
- $f([m_1, m_2, \ldots, m_h]) = f([m_{i1}, m_{i2}, \ldots, m_{ih}])$, where on the r.h.s. we have a permutation of the l.h.s. (f does not depend on the ordering);
- $f(L1) \neq f(L2)$, if in $L1$ there is an element which is not in $L2$ or viceversa.

Example 6.10 *Let us assume that at rank i we have the subgraph presented in Figure 6.6 and that* $\gtrdot'_i(a) = \gtrdot'_i(c) = f(\vec{z})$ *and* $\gtrdot'_i(b) = \gtrdot'_i(d) = g(\vec{z})$. *In this case a minimal solution*

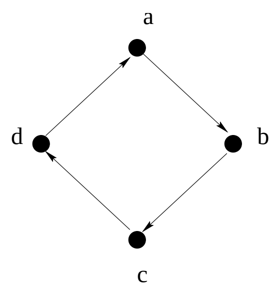


Figure 6.6: Example.

of the system of Boolean equation is given by

$$d(a) = d(c) = |Dom'_i| + 1, d(b) = d(d) = |Dom'_i| + 2,$$

and hence we can assign

$$\mathbb{A}(a) = \mathbb{A}(c) = 0, \mathbb{A}(b) = \mathbb{A}(d) = 1.$$

6.5 Concluding Remarks

In this chapter we started proposing a way to represent graphs via OBDD's *provided* they are reduced with respect to the maximum bisimulation $\equiv$. Our technique is based on the observation that, in a graph quotiented by $\equiv$, each node can be uniquely characterized by the set of its successors. Moreover, the representation we propose can be computed on a graph layered with respect to an equivalence relation. This allows, in general, sharing among the employed OBDD's, thereby guaranteeing savings in space.

The second part of the chapter shows how, in the case the layering is based on a suitable notion of rank, the representation can be used as an Ackermann encoding. Such an encoding allows a computation of the quotient structure *together with* its OBDD's representation. This second part begins with the treatment of acyclic graphs and then provides the representation in the general case as a solution of a system of equations.

Comparing the material of this chapter with the one in the previous chapter we can say that, while in the previous chapter we point our attention on the use of the notion of rank to improve the time performances of algorithms which compute the maximum bisimulation, in this chapter the attention is more on the use of the notion of rank to obtain compact representations of the quotient structure. As far as the complexity issues are concerned, in this chapter we look for *positive* strategies and the complexity of the algorithms which can be designed exploiting the ideas we present depends both on the complexity of the operations on OBDD's and on the power of the numeric instruments used to solve the numeric equations in the non-well-founded case. Optimizations could be obtained by reducing the complexity of the functions involved in the numeric equations.

7

Simulation Reduction as Set Constraint

Simulation relations (see Definition 1.11 in Chapter 1) are binary relations on the nodes of a labelled graphs introduced for the first time by Milner in [Mil71]. Simulations are weaker than bisimulations, since the backward condition is dropped. We showed an example of the difference between the maximum bisimulation and the maximum simulation in the introduction to this part of the thesis. As explained in [HHK95] *in many cases, neither trace equivalence nor bisimilarity, but similarity is the appropriate abstraction for computer-aided verification.* In the case of finite-state systems the similarity quotients can be computed in polynomial time, while this is not the case for trace equivalence quotients. In the case of infinite-state systems, finitely represented using hybrid automaton and other formalisms, the similarity quotients can be computed symbolically and in many cases the quotients are finite (see [HHK95]). Since the conditions in the definition of simulation are weaker than the ones in the definition of bisimulation, simulation provides a better space reduction than bisimulation and it is still adequate for the verification of all the formulae of the branching temporal logic without quantifiers switches [DGG93] (e.g. the formulae of ACTL*, see [GL94]).

In this chapter, first of all, we introduce the notion of simulation. The main difference between simulation and bisimulation is that while the maximum bisimulation $\equiv$ is directly defined as equivalence relation, the maximum simulation equivalence $\equiv_s$ is characterized as $\leq_s \cap \geq_s$, where $\leq_s$ is the maximum simulation relation which is a quasi order. Moreover, in the case of simulation it seems impossible to remove the labels, as it was in the case of bisimulation, and it is not so easy to find a reasonable notion of rank.

However, we provide a set-theoretical encoding for the notion of simulation allowing us to get an incremental computation and to use already existing set-theoretical constraint solvers. We start with an analysis of the difficulties arising, even in the well-founded case, when we try to attribute a set-theoretical meaning to the simulation using memberships to model the edges of the graph. The same difficulties came out if we try to define an encoding *à la* Ackermann similar to the one presented in the previous chapter. The solution we propose here is based on the following ideas: instead of using the membership relation to model the edges of the graph, use it to model the quasi order $\leq_s$; then, as in the previous chapter, in the non-well-founded case we use two codes for each node depending on the side of the relation $\leq_s$ we are looking at. In particular, we encode the simulation problem into a membership/inclusion problem over sets.

A first consequence we obtain is that it is possible to use constraint solving procedures over set-theoretical domains, which can be easily implemented in a constraint logic programming scheme, as algorithms to solve the simulation problem. Moreover, we naturally get to a procedure that works by rank (we use here a different notion of rank) computing the solution *incrementally*.

In some sense the notion of rank we use in the incremental computation is unsatisfactory, since it is possible that during the computation of the simulation at rank i we have to add information to the solution relative to the nodes at rank less than i. However, it allows to work at each step with constraints that are of lower complexity than a global constraint and it is never the case that the information deduced at the previous steps is wrong. Moreover, the incremental computation by rank can be performed using symbolic representations and at the same time a symbolic representation of the quotient structure can be computed *while* computing the simulation relation. Even thought it is possible that we are forced to modify the solution relative to the encoding, it is never the case that we have to modify the quotient structure already built, i.e. the symbolic quotient structure has, w.r.t. the notion of rank, all the good properties it had in the case of bisimulation.

7.1 The Simulation Problem

In this chapter and in the following one we refer to labelled graphs, i.e. graphs with labels on the nodes, whose definition has been given in Chapter 5. Moreover, we assume that the nodes in G are an initial segment of the natural numbers, i.e. $N = \{1, \ldots, k\}$.

Definition 7.1 (Simulation) *Let $G = \langle N, \gtrdot, \Sigma\rangle$ be a labelled graph. A relation $\leq \subseteq N \times N$ is said to be a* simulation *over G iff:*

(l) $n \leq m \Rightarrow [n]_\Sigma = [m]_\Sigma$*;*

(f) $n \leq m \wedge n \gtrdot n' \Rightarrow \exists m'(m \gtrdot m' \wedge n' \leq m')$*.*

In this case we also say that m simulates *n.*
We say that m and n are sim-equivalent *($m \equiv_s n$) if there exist two simulations $\leq_1$ and $\leq_2$, such that $n \leq_1 m$ and $m \leq_2 n$.*

The difference with the definition of bisimulation on labelled graphs is that here there is no backward condition. Notice that a simulation can be neither reflexive nor transitive (e.g. the empty relation is always a simulation), however the reader can easily verify that given an arbitrary simulation its reflexive and transitive closure is always a simulation.

A simulation $\leq_s$ over G is said to be *maximal* if for all the simulations $\leq$ over G it holds $\leq \subseteq \leq_s$.

Lemma 7.2 (Maximal Simulation) *Given a labelled graph $G = \langle N, \gtrdot, \Sigma\rangle$ there always exists a unique maximal simulation $\leq_s$ over G. Moreover $\leq_s$ is a quasi order.*

Proof. We recall that a binary relation is said to be a quasi order if it is reflexive and transitive.

The identity relation I over N is always a simulation relation over G, hence, since N is finite, the set S of the simulation relations over G is finite and not empty.

We prove that if $\leq_1$ and $\leq_2$ are simulation relations over G, then $\leq = \leq_1 \cup \leq_2$ is a simulation relation over G. If $n \leq m$, then $n \leq_1 m$ or $n \leq_2 m$: we can safely assume that $n \leq_1 m$. Since $\leq_1$ is a simulation relation we obtain that $[n]_\Sigma = [m]_\Sigma$ and $\forall n_1 \in N(n \gtrdot n_1 \rightarrow \exists m_1 \in N(m \gtrdot m_1 \wedge n_1 \leq_1 m_1))$. Since $\leq = \leq_1 \cup \leq_2$ we obtain that $[n]_\Sigma = [m]_\Sigma$ and $\forall n_1 \in N(n \gtrdot n_1 \rightarrow \exists m_1 \in N(m \gtrdot m_1 \wedge n_1 \leq m_1))$, which means that $\leq$ is a simulation.

Consider the relation

$$\leq_s = \bigcup_{\leq \in S} \leq .$$

Since S is finite and not empty $\leq_s$ is correctly defined. We have proved that union preserves *be a simulation relation*, hence $\leq_s$ is the unique maximal simulation relation over G.

Since I is a simulation relation $\leq_s$ is reflexive. In order to prove that $\leq_s$ is transitive we prove that if $\leq$ is a simulation relation, then its transitive closure $\leq^+$ is a simulation relation. If $n \leq^+ m$, then there exists $a_1, \ldots, a_k$ in N such that $n \leq a_1 \leq \ldots \leq a_k \leq m$. Hence we obtain that $[n] = [a_1] = \ldots = [a_k] = [m]$. If $n \gtrdot n_1$, then there exists b_1 such that $a_1 \gtrdot b_1$ and $n_1 \leq b_1$. Applying the same argument to a_i, b_i and a_{i+1}, for $i = 1, \ldots, k-1$, and to a_k, b_k and m we obtain a chain of the form $n_1 \leq b_1 \leq \ldots \leq m_1$. Hence, since $\leq^+$ is the transitive closure of $\leq$ we have that $n_1 \leq^+ m_1$, i.e. our thesis. □

Corollary 7.3 *Let $G = \langle N, \gtrdot, \Sigma \rangle$ be a labelled graph and $\leq_s$ be the maximal simulation relation over G. Two nodes $n, m \in N$ are sim-equivalent if and only if $n \leq_s m$ and $m \leq_s n$. The relation $\equiv_s$ (sim-equivalence) is an equivalence relation over N.*

Proof. Let $\leq_s$ be the maximal simulation relation over G. From the definition of sim-equivalence we have that

$$m \equiv_s n \quad \Leftrightarrow \quad n \leq_s m \wedge m \leq_s n.$$

Since $\leq_s$ is a quasi order, $\equiv_s$ is an equivalence relation. □

Given a graph G, we consider the problem of computing $\equiv_s$. All the important algorithms presented in the literature for the computation of the simulation relation ([Blo89], [CPS93], [CS92], [BP95], [HHK95]) obtain the similarity quotient as a by-product of the computation of the entire similarity relation on the set of states N. Their space complexity is then limited from below by $O(|N|^2)$. On the contrary, the algorithm proposed by Bustan and Grumberg gives in output the quotient structure with respect to $\equiv_s$ and the simulation relation among the classes of $M = N/_{\equiv_s}$, *without* computing the entire simulation on N. In the next chapter we will give more details about this algorithm, about the (fast) algorithm presented in [HHK95] and we will propose a new procedure improving on both of them. In this chapter our focus is on the encoding of the simulation problem into a set-constraint problem. This allows both to use already existing constraint solvers to compute the maximum simulation and to exploit a notion of rank to incrementally perform the computation.

A preliminary observation on the minimality of the sim-equivalence quotient is necessary. In [BG00] it has been proved that there always exists a unique smallest labelled graph that is simulation equivalent to G, i.e. there is a unique way to put a minimum number of edges between the elements of $N/\equiv_s$ in order to obtain a labelled graph similar to G. In the case of bisimulation there is a unique reasonable way to put the edges between the equivalence classes of $N/\equiv$, since we had the property ($\flat$) (see Section 5.1), i.e. if a node reaches with an edge an equivalence class, then all the nodes that are equivalent to it reach with an edge the same equivalence class. This is equivalent to say that

$$\forall n, m([n] \cap \lessdot([m]) \neq \emptyset \rightarrow [n] \subseteq \lessdot([m])),$$

which is at the basis of the characterization of the bisimulation problem as (relational) *coarsest partition problem*. In the case of simulation it is possible that there exists a node n which reaches with an edge a node m, while a node which is sim-equivalent to n does not reach any node sim-equivalent to m. When this happens we always have that there exists a node m' such that n reaches m' and $m \leq_s m'$, and we say that m is a *little brother* of m'. Hence, the property which holds for the maximum simulation is that

$$\begin{aligned} &\forall n(\exists n_1, m_1(n_1 \in [n] \wedge n_1 \gtrdot m_1) \rightarrow \\ &\quad \forall n_2(n_2 \in [n] \rightarrow \exists m_2(m_1 \leq_s m_2 \wedge n_2 \gtrdot m_2))), \end{aligned} \tag{$\sharp$}$$

which in particular implies that

$$\forall n([n] \cap \lessdot([m]) \neq \emptyset \rightarrow \exists [m']([n] \subseteq \lessdot([m']))).$$

This property is at the basis of the *generalized stability* condition that we will introduce in the next chapter to provide a characterization of the simulation problem in term of partitioning problem. An immediate consequence is that there is a unique way to put a minimum number of edges among the elements of $N/\equiv_s$ in order to obtain a graph sim-equivalent to G: put an edge from $[n]$ to $[m]$ if and only if it holds $[n] \subseteq \lessdot([m])$ and there is no class $[m']$ such that $[n] \cap \lessdot([m']) \neq \emptyset$ with $m \leq_s m'$ (i.e. m is not a little brother).

Example 7.4 *Consider the two graphs in Figure 7.1. The graph on the right is the quotient w.r.t. the maximum simulation of the graph on the left. In the graph on the left there are no sim-equivalent nodes, hence all the classes in the quotient structure are singletons. In the graph on the left there is a node with label B which is a little brother, hence in the quotient structure we delete an edge.*

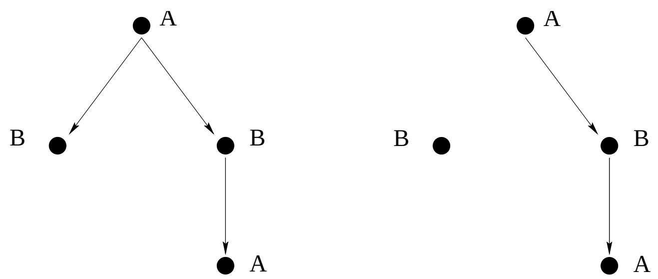


Figure 7.1: Example of little brother.

The construction of the symbolic quotient structure we present in Section 7.7 is based on the elimination of the edges which reach little brothers.

7.2 Simulation, Labels, and Rank

In the previous section we have presented a first important difference between bisimulation and simulation concerning the edges in the quotient structure. In Chapter 5 we proved that in the case of bisimulation working without labels is not a restriction. In the case of simulation it is not possible to remove the labels and this is a consequence of the following result.

Lemma 7.5 *Let $G = \langle N, \gtrdot, \Sigma \rangle$ be a graph such that $\Sigma = \{N\}$, i.e. all the nodes have the same label. Given a node $n \in N$ let $G(n)$ be the subgraph of G of the nodes reachable from n. If $G(n)$ is acyclic, then for all $m \in N$*

$$m \leq_s n \qquad \textit{iff} \qquad G(m) \textit{ is acyclic and } \textit{rank}(m) \leq \textit{rank}(n).$$

If $G(n)$ is cyclic, then for all $m \in N$

$$m \leq_s n.$$

This means that in the case without labels (i.e. only one label) all the nodes which belongs to the non-well-founded part of the graph are sim-equivalent. Consider the graph in Figure 7.2. If we try to remove the labels A and B by adding new nodes and we do not want to use

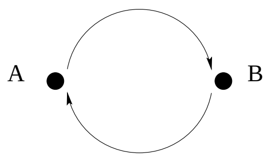


Figure 7.2: Example with Labels.

labels also on the new nodes, then no matter how many new nodes and edges we add the two nodes of the original graph become sim-equivalent, since they are both non-well-founded.

The generalization of the previous lemma to the case with labels has strong implications on the use of the notion of rank in the computation of $\equiv_s$. We have that

$$n \equiv_s m \quad \not\Rightarrow \quad \textit{rank}(n) = \textit{rank}(m).$$

Lemma 7.5 suggests that a reasonable definition of rank to be used in the computation of $\equiv_s$ could be

$$\textit{rank}^*(n) \quad = \quad \begin{cases} \max(\{1 + \textit{rank}(m) \mid c(n) \gtrdot^{scc} c(m)\} & \text{if } n \in WF(G) \\ +\infty & \text{otherwise} \end{cases}$$

i.e. on the well-founded part of the graph we use the usual notion of rank, while all the nodes in the non-well-founded part have rank $+\infty$. Using such a definition we obtain

$$\begin{aligned} n \equiv_s m \quad &\Rightarrow \quad \textit{rank}^*(n) = \textit{rank}^*(m); \\ n \leq_s m \quad &\Rightarrow \quad \textit{rank}^*(n) \leq \textit{rank}^*(m). \end{aligned}$$

Unfortunately, this last property implies that, whenever we use a technique which need to compute $\leq_s$ in order to determine $\equiv_s$, when we process the nodes at rank i we have to keep into account also all the nodes at rank less than i.

A second disadvantage of the use of *rank** is that it is reasonable to believe that at least one half of the nodes have rank $+\infty$, which means that it does not split enough the computation.

For these reasons we take here the opposite approach, i.e. we look for a notion of rank which splits the graph as much as possible admitting to have sim-equivalent nodes with different ranks. However, we require to our notion of rank the property that the information at rank i is enough to compute the similarity relations among the nodes of rank at most i. There are in particular two situations that we want to avoid:

- n and m are two nodes whose rank is at most i, at the end of the i-th iteration we believe that $n \equiv_s m$, while at the end of the $(i+k)$-th iteration we discover that $n \not\equiv_s m$;
- n and m are two nodes whose rank is at most i, at the end of the i-th iteration we believe that $n \not\equiv_s m$, while at the end of the $(i+k)$-th iteration we discover that $n \equiv_s m$;

A necessary and sufficient condition to obtain this property is that:

if $n \gtrdot m$, then the rank of m is not greater than the rank of n.

The *best* notion of rank not depending on the labelling in the above sense, i.e. it satisfies the condition and it splits the graph as much as possible, is the following:

Definition 7.6 (sim-Rank) *Given a graph $G = \langle N, \gtrdot, \Sigma \rangle$, the $rank_s$ of a node is recursively defined as follows:*

$$rank_s(n) \quad = \quad \begin{cases} 0 & \textit{if } n \textit{ is a leaf in } G^{scc} \\ \max\{1 + rank_s(m) \mid \langle c(n), c(m) \rangle \in \gtrdot^{scc}\} & \textit{otherwise} \end{cases}$$

This means that at each rank we consider a set of strongly connected components and the order in which we take these sets of strongly connected components is determined by the usual (well-founded) notion of rank on the graph of the strongly connected components.

7.3 Encoding the Simulation Problem by Sets

Let us, for a moment, ignore the notion of rank and concentrate on the definition of a set-theoretical encoding for the simulation problem.

In Chapter 5 and 6 we showed how the bisimulation problem can be naturally seen as the problem of establishing equality between sets under the anti-foundation axiom (AFA). This is done thinking of nodes in the graph as sets and agreeing that an arc from n to m corresponds to the membership of the set $A(m)$ associated to m, to the set $A(n)$ associated to n:

$$n \gtrdot m \quad \text{if and only if} \quad A(m) \in A(n).$$

Under these assumptions bisimilar nodes are exactly those that have the same associated sets.

If we try to extend the above outlined approach to the task of reducing a graph by eliminating *similar* nodes, a natural idea would be, again, to associate sets to nodes and represent arcs using the membership. The problem with this approach is the following: a moment's thought shows that it can be the case that $m \geq_s n$ without having the set associated to m including the set associated to n. For instance, it can be the case that m' is the only child of m and both m' and n' with $m' \geq_s n'$ are the children of n. In symbols this means that:

$$m \geq_s n \quad \not\Rightarrow \quad A(m) \supseteq A(n).$$

Hence, we can easily build examples in which $m \geq_s n$ and $n \geq_s m$ but either $A(m) \not\supseteq A(n)$ or $A(n) \not\supseteq A(m)$. This implies that the fact that $m \equiv_s n$ cannot be checked considering equality between the associated sets.

Similar problems arise if we try to propose a numerical encoding. For instance, it is immediate to see that it is not possible to assign to each node n a natural number $E(n)$ and to use the relation $\leq$ over numbers to model $\leq_s$: it is possible that $n \not\leq_s m$ and $m \not\leq_s n$, even though, clearly, one between $E(n) \leq E(m)$ and $E(m) \leq E(n)$ must to hold.

The basic idea behind our set-encoding for simulation, is to assign *two* sets $a(n)$ and $A(n)$ to each node n and to use the membership relation to represent $\leq_s$ as follows:

$$a(n) \in A(m) \quad \text{if and only if} \quad n \leq_s m.$$

This means that $a(n)$ is an arbitrary code assigned to n, while $A(m)$ is the set of the a-encoding of the nodes simulated by m. The A-sets can now be used to establish similarity between nodes, however, since $A(m)$ is the collection of the a-encoding relative to nodes the simulated by m, it does not necessarily correspond to the collection of encoding of children of m and the above mentioned consideration does not apply.

More specifically, recalling that, according to our convention, a node $n \in N$ is a natural number, we assign to n the set:

$$a(n) \quad = \quad \{\emptyset\}^n \quad = \quad \overline{n},$$

where $\{\emptyset\}^1 = \{\emptyset\}$, $\{\emptyset\}^2 = \{\{\emptyset\}\}$, and so on.

We determine the A-encoding as solution of a collection of (set-theoretic) constraints keeping into account the above idea as well as the initial labelling on G. Consider, for each node $n \in N$, a set variable $\mathtt{A_n}$ and the constraint $C(n)$ defined as follows:

$$\begin{aligned} C(n) \quad = \quad & \{a(n)\} \subseteq \mathtt{A_n} \subseteq \{a(m) \mid m \in N \wedge [m]_\Sigma = [n]_\Sigma\} \wedge \\ & \bigwedge_{[m]_\Sigma = [n]_\Sigma} (a(m) \in \mathtt{A_n} \leftrightarrow \{a(r) \mid m \gg r\} \subseteq \bigcup_{n \gg s} \mathtt{A_s}). \end{aligned}$$

Finally, let Sys_G be the constraint obtained as conjunction of all the constraints $C(n)$'s:

$$Sys_G \quad = \quad \bigwedge_{n \in N} C(n).$$

We prove the correctness of our approach by showing that:

1. each solution Sol of Sys_G determines a simulation $\leq_{Sol}$;
2. the maximal simulation relation $\leq_s$ is determined by a solution SM of Sys_G, i.e $\leq_s = \leq_{SM}$;
3. if $\leq_{Sol1} \subseteq \leq_{Sol2}$, then $Sol1 \sqsubseteq Sol2$, where $\sqsubseteq$ is a partial order naturally induced over the set of solutions of Sys_G (cf. Definition 7.9).

Lemma 7.7 *Let $G = \langle N, \gg, \Sigma \rangle$ be a graph and Sys_G be the system of constraint obtained from G as described above. Consider a solution*[1] *$Sol = \{A(n) \mid n \in N\}$ of Sys_G and let*

$$n \leq_{Sol} m \quad \textit{iff} \quad a(n) \in A(m).$$

The relation $\leq_{Sol}$ is a simulation over G.

Proof. We have to prove that:

[1] The existence of at least one solution is a consequence of Lemma 7.8.

1. $n \leq_{Sol} m \Rightarrow [n]_\Sigma = [m]_\Sigma$;
2. $n \leq_{Sol} m \wedge n \gtrdot n' \Rightarrow \exists m'(m \gtrdot m' \wedge n' \leq_{Sol} m')$.

As for (1), if $n \leq_{Sol} m$, then $a(n) \in A(m)$, hence, since Sol is a solution of Sys_G, we have that in $C(m)$ we impose $[n]_\Sigma = [m]_\Sigma$.

To see (2), notice that if $n \leq_{Sol} m$, then $a(n) \in A(m)$. Hence, since Sol is a solution of Sys_G, we have that in $C(m)$ we imposed

$$\{a(r) \mid n \gtrdot r\} \quad \subseteq \quad \bigcup_{m \gtrdot s} A(s)).$$

The above inclusion means that there exists m' such that $m \gtrdot m'$ and

$$a(n') \quad \in \quad A(m'),$$

and from this last, using the definition of $\leq_{Sol}$, we obtain:

$$n' \quad \leq_{Sol} \quad m'.$$

□

The following lemma is a sort of converse of the above and shows that the maximum simulation $\leq_s$ over G determines a solution of Sys_G, i.e. Sys_G has always at least one solution.

Lemma 7.8 *Let $G = \langle N, \gtrdot, \Sigma\rangle$ be a graph and $\leq_s$ be the maximum simulation relation over G. Let*

$$SM \quad = \quad \{A(n) = \{a(m) \mid m \leq_s n\} \mid n \in N\}.$$

SM is a solution of Sys_G. Moreover,

$$\leq_{SM} = \leq_s,$$

where $\leq_{SM}$ is the simulation defined from SM as in Lemma 7.7.

Proof. From the fact that

$$m \leq_s n \Rightarrow [m]_\Sigma = [n]_\Sigma$$

and

$$n \leq_s n$$

we have that

$$\{a(n)\} \subseteq A(n) \subseteq \{a(m) \mid m \in N \wedge [m]_\Sigma = [n]_\Sigma\}.$$

Now we have to prove that if $[m]_\Sigma = [n]_\Sigma$, then

$$a(m) \in A(n) \quad \leftrightarrow \quad \{a(r) \mid m \gtrdot r\} \subseteq \bigcup_{n \gtrdot s} A(s).$$

From $m \leq_s n$ we obtain that

$$\forall r(m \gg r \to \exists s(n \gg s \wedge r \leq_s s)),$$

hence

$$\forall r(m \gg r \to \exists s(n \gg s \wedge a(r) \in A(s))),$$

from which

$$\forall r(m \gg r \to \exists s(n \gg s \wedge \{a(r)\} \subseteq A(s))),$$

which implies

$$\{a(r) \mid m \gg r\} \subseteq \bigcup_{n \gg s} A(s).$$

Conversely, if $\{a(r) \mid m \gg r\} \subseteq \bigcup_{n \gg s} A(s)$ we, similarly, obtain that

$$\forall r(m \gg r \to \exists s(n \gg s \wedge r \leq_s s)),$$

from which, since $\leq_s$ is the maximum simulation, we can deduce that

$$m \quad \leq_s \quad n.$$

Finally, from the definition of SM and $\leq_{SM}$ it follows that $\leq_s = \leq_{SM}$. □

The following definition introduces an ordering among the A-encoding (i.e. the solutions of the system Sys_G defined above) that will turn out to be the key to choose the encoding associated to the maximal simulation.

Definition 7.9 *Let $G = \langle N, \gg, \Sigma \rangle$ be a graph and $\mathcal{S}$ be the set of solutions of Sys_G. Consider $Sol_1 = \{A_1(n) \mid n \in N\}$ and $Sol_2 = \{A_2(n) \mid n \in N\}$ two elements of $\mathcal{S}$, we define the partial order $\sqsubseteq$ over $\mathcal{S}$ as follows:*

$$Sol_1 \sqsubseteq Sol_2 \quad \leftrightarrow \quad \forall n \in N(A_1(n) \subseteq A_2(n))).$$

The following lemma shows that the inclusion relation among simulation relations corresponds to the above ordering between A-encoding, thereby ensuring that the maximal A-encoding does correspond to the maximal simulation relation.

Lemma 7.10 *If $\leq_{Sol_1} \subseteq \leq_{Sol_2}$, then $Sol_1 \sqsubseteq Sol_2$.*

Proof. We have to prove that $\forall n \in N$

$$A_1(n) \quad \subseteq \quad A_2(n).$$

Assume, by contradiction, that there exists n such that $A_1(n) \not\subseteq A_2(n)$. Since in Sys we impose $\mathtt{A_n} \subseteq \{a(m) \mid m \in N \wedge [m]_\Sigma = [n]_\Sigma\}$, there must exists m such that $a(m) \in A_1(n)$ and $a(m) \notin A_2(n)$. Hence, $m \leq_{Sol_1} n$ and $m \not\leq_{Sol_2} n$. This contradicts the fact that $\leq_{Sol_1} \subseteq \leq_{Sol_2}$. □

Theorem 7.11 *Let G be a graph. There exists a $\sqsubseteq$-maximum solution of Sys_G. Moreover, if SM is the $\sqsubseteq$-maximum solution of Sys_G, then $\leq_{SM}$ is the maximum simulation over G.*

Proof. Let SM be the solution of Sys_G we obtain from the maximum simulation $\leq_s$ as described in Lemma 7.8: we prove that SM is the $\sqsubseteq$-maximum solution of Sys_G. Let Sol be a solution of Sys_G from the fact that $\leq_{Sol} \subseteq \leq_{SM} = \leq_s$ we obtain, using Lemma 7.10, that $Sol \sqsubseteq SM$. Hence, from Lemma 7.8 we immediately have also the second part of the thesis. $\square$

Example 7.12 *From the graph G in Figure 7.3 we obtain:*

$$\begin{gathered}
\{a(1)\} \subseteq \mathtt{A}_1 \subseteq \{a(1), a(3), a(5)\} \wedge (a(1) \in \mathtt{A}_1 \leftrightarrow \emptyset \subseteq \emptyset) \wedge \\
(a(3) \in \mathtt{A}_1 \leftrightarrow \{a(2), a(4)\} \subseteq \emptyset) \\
\dots \\
\{a(6)\} \subseteq \mathtt{A}_6 \subseteq \{a(2), a(4), a(6)\} \wedge \\
(a(2) \in \mathtt{A}_6 \leftrightarrow \{a(1)\} \subseteq \mathtt{A}_3) \wedge (a(4) \in \mathtt{A}_6 \leftrightarrow \{a(5)\} \subseteq \mathtt{A}_3) \wedge (a(6) \in \mathtt{A}_6 \leftrightarrow \{a(3)\} \subseteq \mathtt{A}_3)
\end{gathered}$$

The maximum solution of Sys_G is

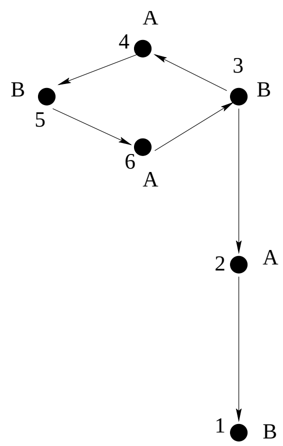


Figure 7.3: Example.

$$\begin{aligned}
&A(1) = \{a(1)\} \quad A(2) = \{a(2)\} \\
&A(3) = A(5) = \{a(1), a(3), a(5)\} \\
&A(4) = A(6) = \{a(4), a(6)\}.
\end{aligned}$$

Now that we have defined a set-theoretical encoding we can try to exploit the notion of rank introduced in the previous section to optimize the computation like we did in Chapters 5 and 6 in the case of bisimulation. The observation which follows is our starting point in the introduction of the notion of rank as far as the computation is concerned.

The pair of codes associated to the nodes using the above outlined technique is, in general, redundant in the following sense: different nodes have always different a-encoding, while similar nodes, for example, could obviously share the same a-encoding. Clearly, a simplification (decrease of the a's) based on the knowledge of the simulation relation would not make sense since it is possible only *a-posteriori*. However, as we will see in the next section, an elimination of redundancies becomes viable when performed after assigning a rank to the nodes.

A nice consequence we obtain when removing the redundancy in the a-encoding is that:

$$\begin{array}{rcl} a(n) = a(m) & \Leftrightarrow & n \equiv_s m \\ A(n) \subseteq A(m) & \Leftrightarrow & n \leq_s m \end{array}$$

that is the a-encoding represents the relation $\equiv_s$, while the A-encoding represents the relation $\leq_s$.

7.4 Determining the Encoding by Ranks

In order to make use of the possibility of reducing the number of different a's using the information conveyed by the A's, in this section we describe a strategy based on a *layering* of the input graph roughly corresponding to the layering induced by a notion of *rank* among sets. In Section 7.2 we explain which are the difficulties which arises in the use of a notion of rank in the computation of the maximum simulation $\equiv_s$. The difficulties we describe are not relative to a particular method of computation, but are in general shared by all the methods in which $\equiv_s$ is computed passing through the computation of $\leq_s$. To the best of our knowledge, all the algorithms proposed for the computation of $\equiv_s$ first compute $\leq_s$: in some cases the whole relation $\leq_s$ is computed (see [HHK95]), in other cases the relation $\leq_s$ is determined among the equivalence classes of $\equiv_s$ (see [BG00]). Also the method we proposed in the previous section to compute $\equiv_s$, which is based on a set-theoretical encoding, passes through the computation of $\leq_s$. Moreover, in Section 7.2 we discuss which are the possible choices for a reasonable notion of rank and we motivate the fact that our choice is to use a notion of rank which splits as much as possible the graph, i.e. rank_s. In this section we describe how to combine the use of rank_s with our set-theoretical encoding in order to get an incremental computation. Notice that all the results we obtain using rank_s remain true using a notion of rank such that if a node m is reachable from a node n, then its rank is not greater than the rank of n. In particular, we in Section 7.2 we say that rank_s is the notion of rank which splits most the graph independently from the labels: it is possible that two strongly connected components belong to the same rank, while using the labels they can be put at different ranks.

Using the definition of rank_s presented in Section 7.2 the computation of the a's and the A's can be carried out rank by rank as in the following example [2].

Example 7.13 *Consider the graph in Figure 7*

First considering only the nodes at rank_s 0 *we obtain*

$$a(1) = \overline{1} \quad a(2) = \overline{2}$$

and

$$\begin{array}{c} \{a(1)\} \subseteq \mathtt{A}_1 \subseteq \{a(1), a(2)\} \wedge (a(1) \in \mathtt{A}_1 \leftrightarrow \emptyset \subseteq \emptyset) \wedge (a(2) \in \mathtt{A}_1 \leftrightarrow \emptyset \subseteq \emptyset) \wedge \\ \{a(1)\} \subseteq \mathtt{A}_2 \subseteq \{a(1), a(2)\} \wedge (a(1) \in \mathtt{A}_2 \leftrightarrow \emptyset \subseteq \emptyset) \wedge (a(2) \in \mathtt{A}_2 \leftrightarrow \emptyset \subseteq \emptyset) \end{array}$$

whose maximum solution is

$$A(1) = A(2) = \{a(1), a(2)\}.$$

This means that 1 *and* 2 *are similar, hence we can re-assign to them the same a-encoding:*

$$a(1) = a(2) = \overline{1}.$$

[2] It is clear that in this chapter we refer to rank_s, also when in the sentences we write rank.

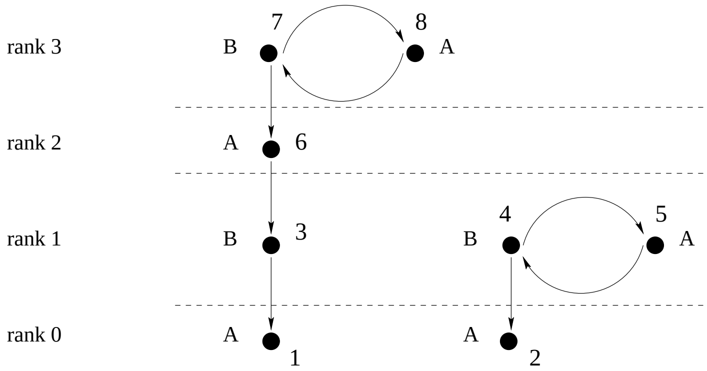


Figure 7.4: Example.

As second step we consider the nodes at $rank_s$ 1 *that must be considered together with the nodes at* $rank_s$ 0. *The following are the corresponding values of the* a*'s:*

$$a(1) = a(2) = \overline{1} \quad a(3) = \overline{2} \quad a(4) = \overline{3} \quad a(5) = \overline{4}$$

from which we obtain the constraint

$$\{a(1)\} \subseteq \mathtt{A}_1 \subseteq \{a(1), a(5)\} \wedge (a(5) \in \mathtt{A}_1 \leftrightarrow \{a(4)\} \subseteq \emptyset) \wedge \dots$$

whose solution is

$$\begin{array}{ll} A(1) = A(2) = \{a(1)\} & A(3) = \{a(3)\} \\ A(4) = \{a(3), a(4)\} & A(5) = \{a(5), a(1)\}. \end{array}$$

Since there are no two nodes n, m *such that* $a(n) \neq a(m)$ *and* $A(n) = A(m)$*, we do not have to re-assign the* a*-encodings.*

In the third step we consider all the nodes at ranks $0, 1, 2$*:*

$$a(1) = a(2) = \overline{1} \quad a(3) = \overline{2} \quad a(4) = \overline{3} \quad a(5) = \overline{4} \quad a(6) = \overline{5},$$

and

$$\{a(1)\} \subseteq \mathtt{A}_1 \subseteq \{a(1), a(6)\} \wedge (a(6) \in \mathtt{A}_1 \leftrightarrow \{a(3)\} \subseteq \emptyset) \wedge \dots$$

whose solution is

$$\begin{array}{lll} A(1) = A(2) = \{a(1)\} & A(3) = \{a(3)\} & A(4) = \{a(3), a(4)\} \\ A(5) = \{a(5), a(1), a(6)\} & A(6) = \{a(6), a(1)\}. & \end{array}$$

Notice that here we have added $a(6)$ *to* $A(5)$.

Similarly during the fourth step we consider all the nodes at ranks $0, 1, 2, 3$*:*

$$\begin{array}{llll} a(1) = a(2) = \overline{1} & a(3) = \overline{2} & a(4) = \overline{3} & \\ a(5) = \overline{4} & a(6) = \overline{5} & a(7) = \overline{6} & a(8) = \overline{7}, \end{array}$$

and

$$\dots \wedge (a(7) \in \mathtt{A}_3 \leftrightarrow \{a(6), a(8)\} \subseteq \mathtt{A}_1) \wedge \dots$$

whose solution is

$$\begin{array}{ll} A(1) = A(2) = \{a(1)\} & A(3) = \{a(3)\} \\ A(4) = A(7) = \{a(3), a(4), a(7)\} & \\ A(5) = A(8) = \{a(5), a(1), a(6), a(8)\} & A(6) = \{a(6), a(1)\}. \end{array}$$

Hence, we can re-assign the a-encoding as follows:

$$a(1) = a(2) = \overline{1} \quad a(3) = \overline{2} \quad a(4) = a(7) = \overline{3} \quad a(5) = a(8) = \overline{4} \quad a(6) = \overline{5},$$

and obtain

$$\begin{array}{ll} A(1) = A(2) = \{\overline{1}\} & A(3) = \{\overline{2}\} \\ A(4) = A(7) = \{\overline{2}, \overline{3}\} & \\ A(5) = A(8) = \{\overline{4}, \overline{1}, \overline{5}\} & A(6) = \{\overline{5}, \overline{1}\}. \end{array}$$

As illustrated in the above example, the computation proceeds by computing the a's and the A's (with the technique outlined in Section 7.3) and then by simplifying the choices for the a's using the information conveyed by the A's. This simplification turns out to be useful in subsequent steps (greater ranks) of the computation. The fact that the A-values computed must be, in general, re-calculated in subsequent steps, is a consequence of the fact that, again in general, there are similar nodes at different ranks (see, for example, nodes 5 and 8 in Figure 7).

An important feature of the technique presented above, is that even though the A-values must be re-calculated at each step, their value can only increase. In fact, the A-values to be modified are those that must be updated by adding a-values of nodes at the current rank which turn out to be smaller w.r.t. $\leq_s$ (e.g., while processing nodes at $rank_s$ 2, since $6 \leq_s 5$ the set $A(5)$ must get $a(6)$). As explained in Section 7.2 this is not only a feature of the set-theoretical encoding we use, but it is a more general consequence of the properties of $rank_s$.

We describe now an algorithm based on the ideas presented and naturally implementing the above incremental strategy.

Assuming that at step i we have considered the nodes at $rank_s$ $0, 1, 2, \ldots, i-1$ obtaining the partial solution

$$a(1) = \overline{1} \quad \ldots \quad a(k) = \overline{k'} \tag{7.1}$$

$$A(1) = \{a(1), \ldots\} \quad \ldots \quad A(k) = \{a(k), \ldots, \},$$

with

$$a(h) = a(j) \Leftarrow A(h) = A(j)$$

(i.e. we have re-assigned the a-encodings) and

$$max\{a(1), \ldots, a(k)\} = \overline{m},$$

then at step $i+1$ we have to:

- consider all the nodes at $rank_s$ $0, 1, 2, \ldots, i$;
- assign to the (say) s nodes at $rank_s$ i an a-encoding

$$a(k+1) = \overline{m+1} \quad \ldots \quad a(k+s) = \overline{m+s}; \tag{7.2}$$

- if h is a node at $rank_s$ $0, 1, 2, \ldots, i-1$, then consider the constraint

$$A(h) \subseteq \mathtt{A_h} \subseteq A(h) \cup \{a(j) \mid rank_s(j) = i+1 \wedge [j]_\Sigma = [h]_\Sigma\} \wedge$$
$$\bigwedge_{rank_s(j)=i+1 \wedge [j]_\Sigma = [h]_\Sigma} (a(j) \in \mathtt{A_h} \leftrightarrow \{a(t) \mid j \gg t\} \subseteq \bigcup_{h \gg v} \mathtt{A_v});$$

- if h is a node at $rank_s$ i, then consider the constraint

$$\{a(h)\} \subseteq \mathtt{A_h} \subseteq \{a(j) \mid rank_s(j) \leq i+1 \wedge [j]_\Sigma = [h]_\Sigma\} \wedge$$
$$\bigwedge_{rank_s(j)\leq i+1 \wedge [j]_\Sigma=[h]_\Sigma} (a(j) \in \mathtt{A_h} \leftrightarrow \{a(t) \mid j \gg t\} \subseteq \bigcup_{h \gg v} \mathtt{A_v});$$

- if
$$A(1) = \{a(1), \ldots\} \quad \ldots \quad A(k+s) = \{a(k+s), \ldots\}$$
is the $\sqsubseteq$-maximum solution of the conjunction of these constraints, and there exist h, j such that
$$A(h) = A(j) \wedge a(h) < a(j),$$
then re-assign to j the a-encoding $a(h)$, i.e. put
$$a(j) = a(h)$$
and replace it also in the solution;

- repeat the last step until it is possible to apply it.

As already explained in Section 7.4, the computation is incremental in the sense that once we put $a(j)$ in $A(h)$ we never remove it. As a matter of fact, during the step $i+1$, relatively to the nodes h of $rank_s$ less than i, we only try to add to $A(h)$ the a-encodings of the nodes of $rank_s$ i. The fact that the computation is incremental is a consequence of the fact that if j is a node at $rank_s$ i, then all the nodes reachable from j have $rank_s$ at most i, and these are the only nodes that may cause $a(j)$ to be added to $A(h)$. Moreover, relatively to the nodes h at $rank_s$ i we start by putting in $A(h)$ at least $a(h)$, since it is always true that a node simulates itself.

Example 7.14 *Consider the graph in Figure 7.5.*

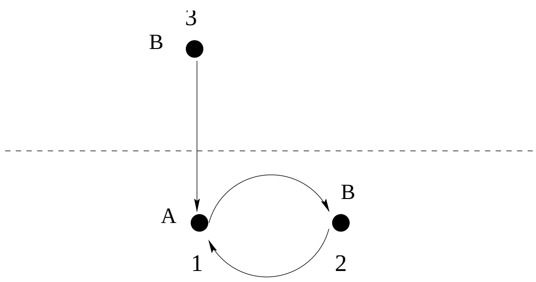


Figure 7.5: Example.

During the first step we consider the nodes at $rank_s$ 0. We put
$$a(1) = \overline{1} \quad a(2) = \overline{2}$$
and we consider the constraint
$$\{\overline{1}\} \subseteq \mathtt{A_1} \subseteq \{\overline{1}\} \wedge \{\overline{2}\} \subseteq \mathtt{A_2} \subseteq \{\overline{2}\}$$

whose maximum solution is

$$A(1) = \{\overline{1}\} \quad A(2) = \{\overline{2}\}.$$

During the second step we consider the nodes at $rank_s$ 0 *and* 1. *We put*

$$a(1) = \overline{1} \quad a(2) = \overline{2} \quad a(3) = \overline{3}$$

and we consider the constraint

$$\begin{gathered} \{\overline{1}\} \subseteq \mathtt{A}_1 \subseteq \{\overline{1}\} \wedge \\ \{\overline{2}\} \subseteq \mathtt{A}_2 \subseteq \{\overline{2}, \overline{3}\} \wedge (\overline{3} \in \mathtt{A}_2 \leftrightarrow \{\overline{1}\} \subseteq \mathtt{A}_1) \wedge \\ \{\overline{3}\} \subseteq \mathtt{A}_3 \subseteq \{\overline{3}, \overline{2}\} \wedge (\overline{2} \in \mathtt{A}_3 \leftrightarrow \{\overline{1}\} \subseteq \mathtt{A}_1) \end{gathered}$$

whose maximum solution is

$$A(1) = \{\overline{1}\} \quad A(2) = A(3) = \{\overline{2}, \overline{3}\}$$

Since $A(2) = A(3)$ *and* $a(2) < a(3)$ *we put* $a(3) = \overline{2}$ *obtaining*

$$a(1) = \overline{1} \quad a(2) = a(3) = \overline{2}$$

$$A(1) = \{\overline{1}\} \quad A(2) = A(3) = \{\overline{2}\}.$$

7.5 Incremental Computation using $CLP(\mathcal{SET})$

Given a graph G, it is extremely natural to use a constraint solver like the interpreter {log}, which is based on the constraint logic programming language $CLP(\mathcal{SET})$ (see [DPPR00]), in order to determine the $\sqsubseteq$-maximum solution of Sys_G. There are other constraint solvers which can be used for this purpose but we prefer to refer to {log}, since this is the one we are more familiar with. The version of the interpreter {log} we refer to in this chapter is available at `http://www.dimi.uniud.it/~piazza/LOG` and can be *consulted* using SICStus Prolog. We show in this section how to use {log} in the incremental computation of the solution of Sys_G.

First we discuss the (limited) fragment of $CLP(\mathcal{SET})$ needed to solve our constraints.

A $CLP(\mathcal{SET})$ *-term* is a term of the following language

$$t ::= \{\} \mid n \mid X \mid \{t_1 \mid t_2\},$$

where $\{\}$ is the empty set, $n \in \mathbb{N}$, X is a variable (from now on all capital letters will represent variables), $\{t_1 \mid t_2\}$ stands for $\{t_1\} \cup t_2$ (i.e. the elements of $\{t_1 \mid t_2\}$ are t_1 and all the elements of t_2). Moreover, it is possible to write the term $\{t_1 \mid \{t_2 \mid \ldots \{t_n \mid \{\}\} \ldots\}\}$ as

$$\{t_1, t_2, \ldots, t_n\},$$

and $t_1, t_2, \ldots, t_n$ are all its elements.

A $CLP(\mathcal{SET})$*-constraint* is composed by conjunctions (written &) and disjunctions (written `or`) of literals of the form[3]

$$\begin{gathered} t_1 = t_2 \quad t_1\, \mathtt{neq}\, t_2 \quad t_1\, \mathtt{in}\, t_2 \qquad t_1\, \mathtt{nin}\, t_2 \\ \mathtt{un}(t_1, t_2, t_3) \quad \mathtt{nun}(t_1, t_2, t_3) \\ \mathtt{sub}(t_1, t_2) \quad \mathtt{nsub}(t_1, t_2) \quad \mathtt{betw}(t_1, t_2, t_3) \end{gathered}$$

[3]In the standard version of $CLP(\mathcal{SET})$ presented in [DPPR00] and in the standard interpreter {log} available at `http://prmat.math.unipr.it/~gianfr/setlog.Home.html` there are no predicates `sub`, `nsub` and `betw`, which have easily been added in the version of the interpreter used here: they are only shorthand for standard $CLP(\mathcal{SET})$ -constraints.

whose semantics can be intuitively described as follows

$$\begin{array}{lcl}
t_1 = t_2 & \Leftrightarrow & \forall t_3 (t_3 \in t_1 \leftrightarrow t_3 \in t_2) \\
t_1\, \mathtt{neq}\, t_2 & \Leftrightarrow & t_1 \neq t_2 \\
t_1\, \mathtt{in}\, t_2 & \Leftrightarrow & t_1 \in t_2 \\
t_1\, \mathtt{nin}\, t_2 & \Leftrightarrow & t_1 \notin t_2 \\
\mathtt{un}(t_1, t_2, t_3) & \Leftrightarrow & t_1 \cup t_2 = t_3 \\
\mathtt{nun}(t_1, t_2, t_3) & \Leftrightarrow & t_1 \cup t_2 \neq t_3 \\
\mathtt{sub}(t_1, t_2) & \Leftrightarrow & t_1 \subseteq t_2 \\
\mathtt{nsub}(t_1, t_2) & \Leftrightarrow & t_1 \not\subseteq t_2 \\
\mathtt{betw}(t_1, t_2, t_3) & \Leftrightarrow & t_1 \subseteq t_2 \subseteq t_3
\end{array}$$

Example 7.15 *The formula*

$$\overline{1} \in \mathtt{A}_2 \leftrightarrow \emptyset \subseteq \emptyset$$

can be written as $CLP(\mathcal{SET})$ constraint using the following syntax

$$(1\, \mathtt{in}\, A_2\ \&\ \mathtt{sub}(\{\}, \{\}))\ \mathtt{or}\ (1\, \mathtt{nin}\, A_2\ \&\ \mathtt{nsub}(\{\}, \{\})).$$

The interpreter {log} gives us all the solutions of the $CLP(\mathcal{SET})$ -constraint. In this case the collection of solutions is given parametrically as

$$\mathtt{A}_2 = \{1 \mid \mathtt{N}\}$$

which means that 1 is an element of $\mathtt{A}_2$ and in $\mathtt{A}_2$ there can be arbitrarily many other elements represented by (the set variable) $\mathtt{N}$.

Now we show an example of the incremental computation of the simulation using the language $CLP(\mathcal{SET})$ and the interpreter {log}.

Example 7.16 *Consider the graph in Figure 7.6.*

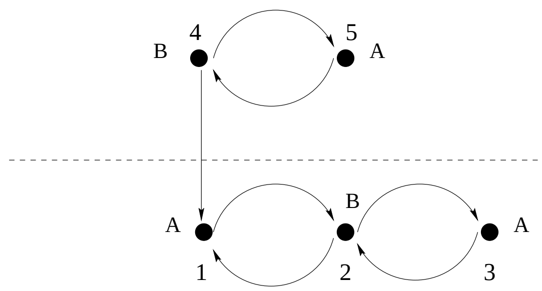


Figure 7.6: Example.

First we have to consider all the nodes at $rank_s$ 0. The a-encoding is

$$a(1) = \overline{1} \quad a(2) = \overline{2} \quad a(3) = \overline{3}$$

and the corresponding constraint is

$$\begin{array}{c}
\{\overline{1}\} \subseteq \mathtt{A}_1 \subseteq \{\overline{1}, \overline{3}\} \wedge (\overline{3} \in \mathtt{A}_1 \leftrightarrow \{\overline{2}\} \subseteq \mathtt{A}_2) \wedge \\
\{\overline{2}\} \subseteq \mathtt{A}_2 \subseteq \{\overline{2}\} \wedge \\
\{\overline{3}\} \subseteq \mathtt{A}_3 \subseteq \{\overline{1}, \overline{3}\} \wedge (\overline{1} \in \mathtt{A}_3 \leftrightarrow \{\overline{2}\} \subseteq \mathtt{A}_2),
\end{array}$$

which written as $CLP(\mathcal{SET})$ -constraint is

$$\begin{gathered}\mathtt{betw}(\{1\},\mathtt{A}_1,\{1,3\})\ \&\ ((3\ \mathtt{in}\ \mathtt{A}_1\ \&\ \mathtt{sub}(\{2\},\mathtt{A}_2))\ \mathtt{or}\ (3\ \mathtt{nin}\ \mathtt{A}_1\ \&\ \mathtt{nsub}(\{2\},\mathtt{A}_2)))\ \&\\ \mathtt{betw}(\{2\},\mathtt{A}_2,\{2\})\ \&\\ \mathtt{betw}(\{3\},\mathtt{A}_3,\{1,3\})\ \&\ ((1\ \mathtt{in}\ \mathtt{A}_3\ \&\ \mathtt{sub}(\{2\},\mathtt{A}_2))\ \mathtt{or}\ (1\ \mathtt{nin}\ \mathtt{A}_3\ \&\ \mathtt{nsub}(\{2\},\mathtt{A}_2))).\end{gathered}$$

In this case {log} gives us only the solution

$$\begin{aligned}\mathtt{A}_1 &= \{1,3\},\\ \mathtt{A}_2 &= \{2\},\\ \mathtt{A}_3 &= \{3,1\}.\end{aligned}$$

Since $\mathtt{A}_1 = \mathtt{A}_3$ *and* $a(1) < a(3)$ *we re-assign*

$$a(3) = a(1) = \overline{1},$$

obtaining

$$\begin{gathered}a(1) = a(3) = \overline{1} \quad a(2) = \overline{2}\\ A(1) = A(3) = \{\overline{1}\} \quad A(2) = \{\overline{2}\}.\end{gathered}$$

At the second step we start with the a-encoding

$$a(1) = a(3) = \overline{1} \quad a(2) = \overline{2} \quad a(4) = \overline{3} \quad a(5) = \overline{4},$$

and the constraint

$$\begin{gathered}\{\overline{1}\} \subseteq \mathtt{A}_1 \subseteq \{\overline{1},\overline{4}\} \wedge (\overline{4} \in \mathtt{A}_1 \leftrightarrow \{\overline{3}\} \subseteq \mathtt{A}_2) \wedge\\ \{\overline{2}\} \subseteq \mathtt{A}_2 \subseteq \{\overline{2},\overline{3}\} \wedge (\overline{3} \in \mathtt{A}_2 \leftrightarrow \{\overline{4},\overline{1}\} \subseteq \mathtt{A}_1) \wedge\\ \{\overline{3}\} \subseteq \mathtt{A}_3 \subseteq \{\overline{3},\overline{2}\} \wedge (\overline{2} \in \mathtt{A}_3 \leftrightarrow \{\overline{1}\} \subseteq \mathtt{A}_1 \cup \mathtt{A}_4) \wedge\\ \{\overline{4}\} \subseteq \mathtt{A}_4 \subseteq \{\overline{4},\overline{1}\} \wedge (\overline{1} \in \mathtt{A}_4 \leftrightarrow \{\overline{2}\} \subseteq \mathtt{A}_3),\end{gathered}$$

which written as $CLP(\mathcal{SET})$ -constraint becomes

$$\begin{gathered}\mathtt{betw}(\{1\},\mathtt{A}_1,\{1,4\})\ \&\ ((4\ \mathtt{in}\ \mathtt{A}_1\ \&\ \mathtt{sub}(\{3\},\mathtt{A}_2))\ \mathtt{or}\ (4\ \mathtt{nin}\ \mathtt{A}_1\ \&\ \mathtt{nsub}(\{3\},\mathtt{A}_2)))\ \&\\ \mathtt{betw}(\{2\},\mathtt{A}_2,\{2,3\})\ \&\\ ((3\ \mathtt{in}\ \mathtt{A}_2\ \&\ \mathtt{sub}(\{4,1\},\mathtt{A}_1))\ \mathtt{or}\ (3\ \mathtt{nin}\ \mathtt{A}_2\ \&\ \mathtt{nsub}(\{4,1\},\mathtt{A}_1)))\ \&\\ \mathtt{betw}(\{3\},\mathtt{A}_3,\{3,2\})\ \&\ \mathtt{un}(\mathtt{A}_1,\mathtt{A}_4,\mathtt{X})\ \&\\ ((2\ \mathtt{in}\ \mathtt{A}_3\ \&\ \mathtt{sub}(\{1\},\mathtt{X}))\ \mathtt{or}\ (2\ \mathtt{nin}\ \mathtt{A}_3\ \&\ \mathtt{nsub}(\{1\},\mathtt{X})))\ \&\\ \mathtt{betw}(\{4\},\mathtt{A}_4,\{4,1\})\ \&\ ((1\ \mathtt{in}\ \mathtt{A}_4\ \&\ \mathtt{sub}(\{2\},\mathtt{A}_3))\ \mathtt{or}\ (1\ \mathtt{nin}\ \mathtt{A}_4\ \&\ \mathtt{nsub}(\{2\},\mathtt{A}_3))).\end{gathered}$$

In this case {log} gives us the two solutions

$$\begin{aligned}\mathtt{A}_1 &= \{1\}, & \mathtt{A}_1 &= \{1,4\},\\ \mathtt{A}_2 &= \{2\}, & \mathtt{A}_2 &= \{2,3\},\\ \mathtt{A}_3 &= \{3,2\}, & \mathtt{A}_3 &= \{3,2\},\\ \mathtt{A}_4 &= \{4,1\}. & \mathtt{A}_4 &= \{4,1\}.\end{aligned}$$

The second solution is the maximum one and using it to re-assign the a-encoding we obtain

$$\begin{gathered}a(1) = a(3) = a(5) = \overline{1} \quad a(2) = a(4) = \overline{2},\\ A(1) = A(3) = A(5) = \{\overline{1}\} \quad A(2) = A(4) = \{\overline{2}\}.\end{gathered}$$

Notice that, even though the A's are singletons here, this is not true in general (cf. Examples 7.12 and 7.13).

7.6 Complexity Issues

In this section we analyse the worst case complexity of the constraint problem encoding the simulation problem assuming that all the nodes in the graph are at the same rank.

First of all we recall from [DPPR00] that it is possible to encode NP-complete problems in $CLP(\mathcal{SET})$-constraints: given an instance of the 3-SAT problem in k propositional variables it is possible to write a $CLP(\mathcal{SET})$ -constraint in k variables which has a solution if and only if the answer to the instance of 3-SAT is positive. This implies that, in general, the interpreter {log} can take more than polynomial time to produce a solution for a constraint.

Given a graph $G = \langle N, \gtrdot, \Sigma\rangle$ we obtain a constraint Sys_G in $|N|$ variables (the variables $\mathtt{A_n}$'s). In the case of the constraints of the form Sys_G the interpreter {log}, as it is currently implemented, uses a sort of *generate and test* strategy: it starts initializing all the $\mathtt{A_n}$'s to N, it tests if all the literals in Sys_G are satisfied, if this is not the case it removes one element from one of the $\mathtt{A_n}$'s, and it tests if now all the literals in Sys_G are satisfied, and so on, until the solution is found. Since the number of possible assignments to the variables $\mathtt{A_n}$'s is $(2^{(|N|-1)})^{|N|}$, in the worst case we have a non-polynomial complexity.

However, it is possible to optimize the interpreter {log} for the constraints of the form Sys_G. Using counters and data-structures similar to the ones used in the algorithm proposed in [HHK95], it would be possible to force {log} to solve the constraints of the form Sys_G in time $O(|N||\gtrdot|)$, but this seems to be an unnatural solution which does not directly come from the set theoretical encoding we propose.

Let us analyse some possible optimizations suggested from the set encoding, which allow to design a constraint-solver for the constraints of the form Sys_G and which works in polynomial time. We can imagine that the constraint-solver works in this way:

- the $\mathtt{A_n}$'s are all initialized to N;
- each time that $Succ(m) = \{a(r) \mid m \gtrdot r\} \not\subseteq \bigcup_{n \gtrdot s} \mathtt{A_s} = \mathtt{B_n}$ it removes $a(m)$ from $\mathtt{A_n}$.

In the worst case the constraint-solver performs $|N|^2$ steps. The cost of one step for a single $C(n)$ is given by:

- the cost to compute $\mathtt{B_n}$;
- for each m the cost to perform the check $S(m) \subseteq \mathtt{B_n}$

Using lists to represent the $S(m)$'s and arrays of length $|N|$ for the $\mathtt{A_n}$'s and the $\mathtt{B_n}$'s we obtain that the total cost to find the maximal solution of Sys_G is $O(|N|^3|\gtrdot|)$. Using lists without repetitions to represent the $S(m)$'s, the $\mathtt{A_n}$'s, and the $\mathtt{B_n}$'s we obtain that the cost is $O(|N|^4|\gtrdot|)$.

7.7 Simulation Reduction and Symbolic Computation

The simulation's set-encoding proposed in the previous sections can be naturally combined with the so-called symbolic representation of Kripke structures based on the use of OBDD's (see Chapter 6). In particular OBDD's can be used both to perform the incremental computation and to obtain a compact representation of the quotiented structure.

In order to use OBDD's in the incremental computation our idea is to:

- consider the binary representation of the a-encoding;

- determine the OBDD representations of the A's associated to the nodes of the graph;
- use the OBDD representations of the A's in order to refine $\{a(n) \mid n \in N\}$ (the set of a-encoding assigned to the graph's vertexes).

Let us assume that at step i we have the partial solution as described in Equation 7.1 in Section 7.5 with the a-encoding represented in binary notation and the A's represented as OBDD's. We proceed assigning at each node h at $rank_s$ i a new binary number $\underline{a(h)}$ as described in Equation 7.2. We compute the $A(h)$'s relative to all the nodes at $rank_s$ at most i using the OBDD:

$$\bigvee_{a(j) \in A(h)} \underline{a(j)}$$

to represent $A(h)$. Such OBDD representation of the A-encoding turns out to be useful in the process of simplifying our choices for the a's: the a-encoding of two nodes h and j whose $rank_s$ is at most i must be the same if and only if the OBDD's for h and j are equal. Once refined the set $\{\underline{a(h)} \mid rank_s(h) \leq i\}$ we can give a more compact symbolic representation (i.e. associated to a boolean function with less variables) of the A's at $rank_s$ at most i.

At the end of step i, when we already have refined the set $\{\underline{a(h)} \mid rank_s(h) \leq i\}$ of binary numbers, if h is a node at $rank_s$ i such that $a(h) > \overline{m}$ (see Section 7.5, Equation 7.2), then we consider the Boolean function

$$\geqslant(h) \quad = \quad \bigvee_{v \in Big(h)} \underline{a(v)}$$

where $Big(h)$ the set of the successors of h which are not simulated by any other successor of h is formally defined as

$$Big(h) \quad = \quad \{v \mid h \geqslant v \wedge a(v) \notin \bigcup_{h \geqslant v' \wedge v' \neq v} A(v')\}.$$

The Boolean function $\geqslant(h)$ can be represented as OBDD in the standard way or using the layered representation based on Dom_i presented in Chapter 6.

The OBDD representation of the quotient structure $G/\equiv_s$ is given by the collection of all the OBDD's of the form $\geqslant(h)$ that we compute. It is important to notice that the computation of the quotient structure is fully incremental in the sense that once we have computed $\geqslant(h)$ we never recompute it. This is due to the facts that: if a node j is a successor of a node h, then $rank_s(j) \leq rank_s(h)$; if a node h has $rank_s$ i, then after the end of the i-th iteration $a(h)$ is never modified. Moreover, we avoid to compute $\geqslant(h)$ whenever h is similar to a node j and we have already computed $\geqslant(j)$.

7.8 Concluding Remarks

A novel method to compute the simulation relation, based on the encoding of the problem into a collection of set-theoretic constraint, is presented. The method allows for a natural and simple implementation in $CLP(\mathcal{SET})$ in which the solution of the constraint is incrementally determined.

As for the technique outlined in Section 7.5, it is easy to imagine that the whole incremental search for the final solution could be automated. To this end it is necessary to integrate our procedure with an algorithm for the computation of the rank that at each step builds the constraint, passes it to {log} and, using the maximum solution, builds the constraint for the next step. An implementation of this kind of interface is undergoing.

The main open problem we would like to mention, is the search for a characterization of $\equiv_s$ which does not explicitly use $\leq_s$. This could lead to the definition of an algorithm in which the computation of $\equiv_s$ does not pass through the computation of $\leq_s$. In this a notion rank which like *rank** has the property that

$$n \equiv_s m \quad \Rightarrow \quad rank^*(n) = rank^*(m)$$

and which splits more than *rank** the graph, using for instance the information coming from the labels, could lead to the same results we obtain in the case of bisimulation.

Finally, notice that the approach outlined in this chapter is easily paired with a symbolic representation of the labelled structures and its subsets (cf. Section 7.7). In particular, it allows to combine the determination of the OBDD-representation *together* with the simulation reduction. This positive by-product was already obtained for the bisimulation relation with a similar approach in Chapter 6. The fact that we have positive results on an OBDD representation build by rank both in the case of bisimulation and simulation has encouraged us to start to study the development of an OBDD package working with the *rank*-OBDD representation we propose.

8

Simulation as Coarsest Partition Problem

In this chapter, for a moment, we *abandon* the set-theoretical semantics and we work in a general algebraic environment in order to address time and space complexity issues in the computation of the maximum simulation $\equiv_s$. On the one hand, the considerations we present here are not linked to the set-theoretical point of view which is the *leit-motiv* of this thesis, on the other hand they are suggested by the study of the best (time-complexity) algorithms to compute bisimulation, whose importance in non-well-founded set theories is indisputable.

One of our motivations in the study of simulation from an algebraic point of view cames from the considerations at the end of the previous chapter: if we were able to find a characterization of $\equiv_s$ which does not depends on $\leq_s$ we could exploit in a significant way a (suitable) notion of rank in the computation of the maximum simulation. Unfortunately, in this sense, the results we describe in this chapter are a *unsatisfactory*, i.e. our algebraic characterization make use of $\leq_s$. However, it is not a complete failure: the characterization allows us to improve the time/space complexity of the best algorithms for the simulation problem. Moreover, it shed some light on the relationships between bisimulation and simulation in terms of partitioning problems.

To motivate our efforts for an algebraic characterization of $\equiv_s$, we start with a comparison with the situation in the case of bisimulation. If the naturalness of the concept of bisimulation explains its large usage—especially in connection with circular structures—, from a computational point of view the main reasons for its fortune and for its best solution lie in the possibility of re-formulating a bisimulation problem in purely (elementary) algebraic terms:

> a bisimulation is the coarsest partition finer than an input one and *stable* with respect to the relation $\gtrdot$ of the graph (a partition is stable with respect to $\gtrdot$ iff any of its classes is either sent by $\gtrdot$ entirely within or entirely outside any other of its classes).

Several polynomial-time algorithms for computing similarity quotients have been proposed: the ones presented in [Blo89], [CPS93], and [CS92] achieve time complexities of the orders $O(|N|^6|\gtrdot|)$, $O(|N|^4|\gtrdot|)$, and $O(|\gtrdot|^2)$, respectively. The best time complexity for algorithms computing the maximal simulation is the one presented in [HHK95] and [BP95]: $O(|N||\gtrdot|)$.

All of the algorithms just mentioned ([Blo89], [CPS93], [CS92], [BP95], [HHK95]) obtain the similarity quotient as a by-product of the computation of the similarity relation on the set of states N. Their space complexity is then limited from below by $O(|N|^2)$.

Recently Bustan and Grumberg proposed an algorithm which gives in output the quotient structure with respect to $\equiv_s$ and the simulation relation among the classes of $M = N/_{\equiv_s}$ without computing the entire simulation on N. The space requirements (often more critical, especially in the field of verification) of this algorithm depend on the size of M and are lower than the ones of the algorithms in [Blo89], [CPS93], [CS92], [BP95], and [HHK95]. In more detail, the so-called *Partitioning Algorithm* described in [BG00] uses only $|M|^2 + |N| \log(|M|)$ space. The time complexity of the *Partitioning Algorithm* is instead rather heavy: it is $O(|M|^4(|\gtrdot| + |M|^2) + |M|^2|N|(|N| + |M|^2))$.

A simulation relation can be seen as a bisimulation in which the condition on the children is weaker than the condition on the parents: in order to have $m \equiv_s n$ is sufficient that a back-and-forth condition involving $\leq_s$ is satisfied. In the case of $\equiv$, instead, in order to have $m \equiv n$ any child of m (resp. n) must be in the same relation $\equiv$ with some of n's (resp. m's) children. This observation is, ultimately, the reason why on the one hand, the computational steps for determining a simulation must compute (successive approximations of) both $\equiv_s$ and $\leq_s$ and, on the other hand, it is not easy to rephrase the notion of simulation in purely algebraic terms as for bisimulation.

In this chapter we show that such a rephrasing is in fact possible and that the simulation relation corresponds to the coarsest relation stable w.r.t. a suitable condition. Moreover, on this ground, we propose an algorithm for determining M and the simulation among its classes that improves (significantly) on either space or time complexity with respect to the best known algorithms for the same problem.

The key idea in our approach is that of using, in a sequence of approximation steps determining M, a graph (the $\exists\forall$-structure, similar to the structures used in [DGG97]) whose nodes are equivalence classes and whose edges are capable to convey all the necessary information present in the original structure G. Such information is captured in a very natural way: since in the approximations nodes are in fact classes of nodes, we introduce two kind of edges between classes α and α', corresponding to the case in which either *there is* an edge between an element in α and one in α', or to the case in which *all* nodes in α are sent in α' by $\gtrdot$. The stability condition (to be proved) equivalent to the notion of simulation is the following: a class β simulates a class α only if whenever there is an $\exists$-edge between α and γ, then there exists a $\forall$-edge between β and δ simulating γ (cf. Figure 8.2).

The use of the above structure and the formulation of the simulation problem as a coarsest stable partition problem allows us to define an algorithm computing the simulation relation as follows:

1. the algorithm computes a series of pairs of the form $\langle \Sigma_i, P_i \rangle$, where Σ_i is a partition over N coarser than M and P_i is a relation on Σ_i which includes (as set of pairs) the simulation relation among the classes of M;

2. [HHK95] can be used on a suitable structure (the $\exists\forall$-structure) to move from an approximation to the next one, without wasting space (moreover, at any step the partition Σ_i is modified);

3. the algorithm stops when stability is reached.

The space complexity of our algorithm is $O(|M|^2 + |N| \log |M|)$, as that of [BG00] with which our algorithm shares the general structure. However, due to the use of [HHK95] as subroutine, our time complexity turns out to be $O(|M|^2| \gtrdot |)$, much better than $O(|M|^4(| \gtrdot | + |M|^2) + |M|^2|N|(|N| + |M|^2))$ which is the time complexity of the algorithm in [BG00].

8.1 Simulation as Coarsest Partition Problem

We recall here a few basic definitions which are very important in the rest of this chapter.

Definition 8.1 *Let S be a set and $Q \subseteq S \times S$ a binary relation over S:*

- *Q is said to be a* quasi order *over S if and only if Q is reflexive and transitive;*
- *Q is said to be a* partial order *over S if and only if Q is reflexive, antisymmetric, and transitive;*
- *Q is said to be* acyclic *if and only if its transitive closure is antisymmetric.*

We will use Q^+ to refer to the transitive closure of Q and Q^* to refer to the reflexive and transitive closure of Q.

Notice that if a relation is acyclic, then it is antisymmetric, while the converse does not hold (unless it is transitive).

In Section 7.2 we observe that in the case of simulation labels are fundamental: it is not possible to completely remove the labels. However, we start here by considering that it is possible to encode the information in Σ (the labels) by adding new nodes to the labelled graph. This encoding turns out to be useful in order to compare the problem of computing the similarity quotient with the *generalized partitioning problem* we will present later in this section.

Our encoding allows us to start with a partition in which all the nodes of the original labelled graph are in the same class (i.e., they have the same label).

Definition 8.2 (S-unlabelling) *Let $G = \langle N, \gtrdot, \Sigma \rangle$ be a labelled graph. Consider the following sets:*

$$\begin{array}{rcl} L & = & \{\ell_\alpha \mid \alpha \in \Sigma\} \\ N' & = & N \cup L \\ \gtrdot' & = & \gtrdot \cup \{\langle n, \ell_\alpha \rangle \mid n \in \alpha \in \Sigma\} \\ \Sigma' & = & \{N\} \cup \{\{\ell_\alpha\} \mid \ell_\alpha \in L\}. \end{array}$$

We call the labelled graph $G' = \langle N', \gtrdot', \Sigma' \rangle$ the s-unlabelling *of G.*

In the definition we use *s-unlabelling* (s stands for simulation) to stress the fact that this is a different unlabelling w.r.t. the one we present in Chapter 5. While in Chapter 5 we were able to completely remove the labels, here we are only able to *push them down* on the new leaves we add.

In Figure 8.1 we give an example of a graph together with its s-unlabelling.

Definition 8.3 *Let G be a labelled graph and G' be its s-unlabelling. If $\leq^{G'}$ is a simulation over G', then we define its restriction to G $\leq^G$ as follows*

$$\leq^G = r(\leq^{G'}) = \leq^{G'} \restriction N.$$

If $\leq^G$ is a simulation over G, then we define (its extension to G')

$$\leq^{G'} = e(\leq^G) = \leq^G \cup \{\langle \ell_\alpha, \ell_\alpha \rangle \mid \ell_\alpha \in L\}.$$

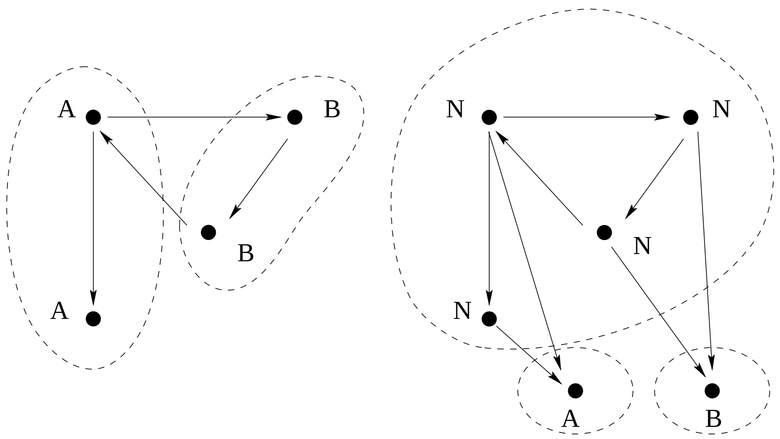


Figure 8.1: S-unlabelling Example.

Lemma 8.4 *Let G be a labelled graph and let G' be its s-unlabelling.*
If $\leq^{G'}$ is a simulation over G', then its restriction to N, $\leq^G = r(\leq^{G'})$, is a simulation over G.
If $\leq^G$ is a simulation over G, then its extension to N', $\leq^{G'} = e(\leq^G)$ is a simulation over G'. Moreover it holds that:

1. *if $\leq_1^{G'} \subseteq \leq_2^{G'}$, then $r(\leq_1^{G'}) \subseteq r(\leq_2^{G'})$;*
2. *if $\leq_1^{G} \subseteq \leq_2^{G}$, then $e(\leq_1^{G}) \subseteq e(\leq_2^{G})$;*
3. *$r(e(\leq^G)) = \leq^G$;*
4. *$e(r(\leq^{G'})) = \leq^{G'}$.*

Proof. Let $r(\leq^{G'}) = \leq^G$. We have to prove that $\leq^G$ is a simulation over G. If $n \leq^G m$ and $n \in \alpha \in \Sigma$, then since $\leq^{G'}$ is a simulation over G', $n \gtrdot' \ell_\alpha$, and since $[\ell_\alpha]_{\Sigma'} = \{\ell_\alpha\}$, it must be $m \gtrdot' \ell_\alpha$, i.e. $m \in \alpha$. If $n \leq^G m$ and $n \gtrdot s$, then since $\leq^{G'}$ is a simulation over G', there must exist $t \in N'$ such that $m \gtrdot' t$ and $s \leq^{G'} t$. Since $s \in N$ and in Σ' the set N is a class, from $s \leq^{G'} t$, we obtain that $t \in N$. Hence we have obtained that there exists t such that $m \gtrdot t$ and $s \leq^G t$.

Let $e(\leq^G) = \leq^{G'}$. We have to prove that $\leq^{G'}$ is a simulation over G. If $n \leq^{G'} m$ and $n \in L$, then we have that $m = n$, hence it is a simulation. If $n \leq^{G'} m$ and $n \in N$, then we have that $m \in N$. If $n \gtrdot' s$ and $s \in L$, then from the fact that $\leq^G$ is a simulation, we obtain that $m \gtrdot' s$ and $s \leq^{G'} s$. If $n \gtrdot' s$ and $s \in N$, then let t be the element of N such that $m \gtrdot t$ and $s \leq^G t$, we have that $m \gtrdot' t$ and $s \leq^{G'} t$. Hence we have obtained the thesis in all the possible cases.

The rest of the thesis follows immediately from the definitions of r and e. $\square$

The above lemma establishes a link among the simulations on a graph and the simulations on its s-unlabelling which is necessary to prove that from the maximum simulation on the s-unlabelling we can immediately obtain the maximum simulation on the original graph.

Theorem 8.5 *Let G be a labelled graph and let G' be its s-unlabelling. Let $\leq_s^{G'}$ be the maximal simulation over G'. Its restriction to N,*

$$\leq_s^{G'} \restriction N = r(\leq_s^{G'}),$$

is the maximal simulation over G. Let $\leq_s^G$ be the maximal simulation over G. Its extension to N' defined as

$$\leq_s^G \cup \{\langle \ell_\alpha, \ell_\alpha\rangle \mid \ell_\alpha \in L\} = e(\leq_s^G)$$

is the maximal simulation over G'.

Proof. It is an immediate consequence of Lemma 8.4. □

The graphs that are obtained as s-unlabelling will turn out to be useful to establish a connection between simulations and partition problems.

We are now ready to introduce the *Generalized (Stable) Coarsest Partition Problem*, GCPP, which is the central notion in our approach. We call it *generalized* because we are not only going to deal with partitions to be refined (as in the classical coarsest partition problems [PTB85, PT87, Hop71]), but with pairs in which we have a partition and a relation over the partition. The equivalence of the similarity quotient problem and the GCPP will be proved at the end of this section.

Definition 8.6 (Partition-pair) *Let $G = \langle N, \to\rangle$ be a finite graph. A* partition-pair *over G is a pair $\langle \Sigma, P\rangle$ in which Σ is a partition over N, and $P \subseteq \Sigma^2$ is a reflexive and acyclic relation over Σ.*

Notice that a labelled graph $G = \langle N, \to, \Sigma\rangle$ can be seen as a graph $G' = \langle N, \to\rangle$ together with the partition-pair $\langle \Sigma, I\rangle$, where I is the identity relation over Σ.

Given two partitions Π and Σ, such that Π is finer than Σ, and a relation P over Σ, we use the notation $P(\Pi)$ to refer to the relation *induced* on Π by P, i.e.:

$$\forall \alpha\beta \in \Pi((\alpha, \beta) \in P(\Pi) \leftrightarrow \exists \alpha'\beta'((\alpha \subseteq \alpha' \wedge \beta \subseteq \beta' \wedge (\alpha', \beta') \in P)).$$

Denoting by $\mathcal{P}(G)$ the set of partition-pairs over G we now introduce the partial order we need to be able to define the notion *coarsest partition-pair.*

Definition 8.7 *Let $\langle \Sigma, P\rangle, \langle \Pi, Q\rangle \in \mathcal{P}(G)$:*

$$\langle \Pi, Q\rangle \sqsubseteq \langle \Sigma, P\rangle \quad \textit{iff} \quad \Pi \textit{ is finer than } \Sigma \textit{ and } Q \subseteq P(\Pi).$$

The following definition introduces structures that in the algorithm, will suggest us the use of a subroutine based on the simulation algorithm by Henzinger, Henzinger, and Kopke in [HHK95]. The use of such a subroutine guarantees a weak form of stability (see Definition 8.19) in the intermediate stages and is one of the keys in the improved time performance of our algorithm w.r.t. the one by Bustan and Grumberg [BG00].

Definition 8.8 ($\to_\exists$, $\to_\forall$ and Quotient Structures) *Let $G = \langle N, \to\rangle$ be a graph, and Π a partition of N. The relations $\to_\exists, \to_\forall \subseteq \Pi \times \Pi$ are defined as follows:*

$$\alpha \to_\exists \beta \quad \textit{iff} \quad \exists n \exists m (n \in \alpha \wedge m \in \beta \wedge n \to m);$$

$$\alpha \to_\forall \beta \quad \textit{iff} \quad \forall n (n \in \alpha \to \exists m (m \in \beta \wedge n \to m)).$$

The $\exists$-quotient structure *over Π is the graph $\Pi_\exists = \langle \Pi, \to_\exists\rangle$. The* $\forall$-quotient structure *over Π is the graph $\Pi_\forall = \langle \Pi, \to_\forall\rangle$. The* $\exists\forall$-quotient structure *over Π is the structure $\Pi_{\exists\forall} = \langle \Pi, \to_\exists, \to_\forall\rangle$.*

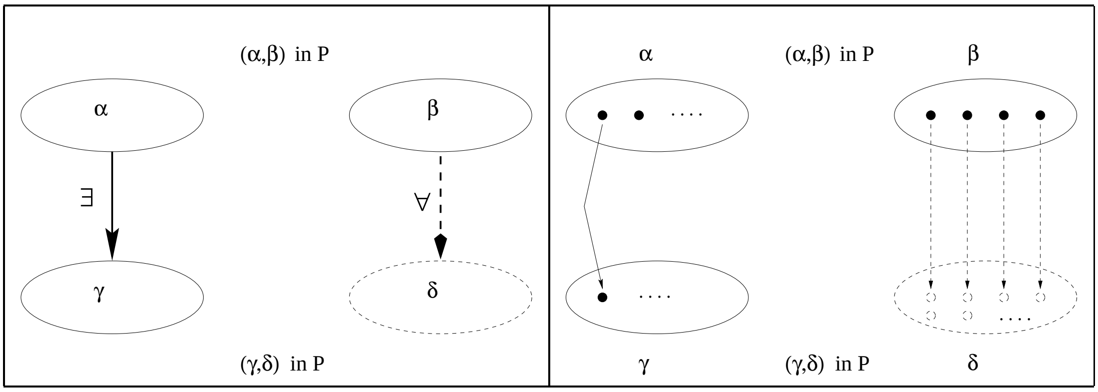


Figure 8.2: The Stability Condition.

Notice that it is always $\geqslant_\forall \subseteq \geqslant_\exists$. In other worlds $\alpha \geqslant_\forall \beta$ is a short hand for $\alpha \subseteq \geqslant^{-1}\beta$, while $\alpha \geqslant_\exists \beta$ stands for $\alpha \cap \geqslant^{-1}(\beta) \neq \emptyset$. Similar notations (called *relation transformers* and *abstract transition relations*) were introduced in [DGG97] in order to combine Model Checking and Abstract Interpretation.

We introduce here a definition, strongly connected to the definition of the quotient structures, that we will use later in the description of our algorithm (see Section 8.3).

Definition 8.9 (Induced Structure) *Given a graph, $G = \langle N, \geqslant \rangle$, and two partitions of N, Σ and Π with Π finer than Σ, the* $\exists\forall$-induced quotient structure *over Π is the structure $\Sigma_{\exists\forall}(\Pi) = \langle \Pi, \geqslant^{\Sigma}_{\exists}(\Pi), \geqslant^{\Sigma}_{\forall}(\Pi)\rangle\rangle$, where:*

$$\begin{array}{lll} \alpha \geqslant^{\Sigma}_{\exists} (\Pi)\beta & \textit{iff} & \alpha \cap \geqslant^{-1}(\beta) \neq \emptyset \\ \alpha \geqslant^{\Sigma}_{\forall} (\Pi)\beta & \textit{iff} & \alpha \geqslant^{\Sigma}_{\exists} (\Pi)\beta \wedge \alpha \subseteq \geqslant^{-1}(\beta') \wedge \beta \subseteq \beta' \in \Sigma \end{array}$$

with $\alpha, \beta \in \Pi$.

We are now ready to introduce the fundamental notion in the generalized coarsest partition problems, the notion of stability of a partition-pair w.r.t a relation.

Definition 8.10 (Stability) *Given a graph $G = \langle N, \geqslant \rangle$ we say that a partition-pair $\langle \Sigma, P \rangle$ is* stable *w.r.t. the relation $\geqslant$ if and only if*

$$\forall \alpha, \beta, \gamma \in \Sigma((\alpha, \beta) \in P \wedge \alpha \geqslant_\exists \gamma \rightarrow \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \beta \geqslant_\forall \delta)).$$

The condition in the definition of stability is equivalent to:

$$\forall \alpha, \beta, \gamma \in \Sigma((\alpha, \beta) \in P \wedge \alpha \cap \geqslant^{-1}(\gamma) \neq \emptyset \rightarrow \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \beta \subseteq \geqslant^{-1}(\delta))).$$

The stability condition is exactly the condition which holds between the classes of $N/\equiv_s$: if $\alpha, \beta \in N/\equiv_s$ with $\alpha \leq_s \beta$ (i.e. all the elements of α are simulated by all the elements of β), and an element a in α reaches an element c in γ, then all the elements b of β must reach at least one element d which simulates c. Taking, in particular, all the maximal (w.r.t. $\leq_s$) elements d simulating c reached by elements in β, we have that all the elements in β reach a class δ which simulates c and, hence, which simulates γ.

In Figure 8.2 we give a graphical representation of the notion of stability using both the characterizations.

We will use the notion of stability of a partition-pair w.r.t. a relation in the definition of generalized coarsest partition problem, in the same way in which the notion of stability of a partition w.r.t. a relation is used in the definition of coarsest partition problem.

Definition 8.11 (Generalized Coarsest Partition) *Given a graph $G = \langle N, \gtrdot \rangle$ and a partition-pair $\langle \Sigma, P \rangle$ over G, the* generalized coarsest partition problem *consists in finding a partition-pair $\langle M, \preceq \rangle$ such that:*

(a) $\langle M, \preceq \rangle \sqsubseteq \langle \Sigma, P^+ \rangle$, where P^+ is the transitive closure of P;

(b) $\langle M, \preceq \rangle$ is stable w.r.t. $\gtrdot$;

(c) $\langle M, \preceq \rangle$ is $\sqsubseteq$-maximal satisfying (a) and (b).

If $\langle M, \preceq \rangle$ is a partition-pair which satisfies (a) and (b) we say that it is a stable refinement *of $\langle \Sigma, P \rangle$.*

Notice that in the above definition $\langle M, \preceq \rangle$ is a refinement of $\langle \Sigma, P^+ \rangle$, while it can be the case that it is not a refinement of $\langle \Sigma, P \rangle$. In particular this happens always in case $\langle \Sigma, P \rangle$ is not stable while $\langle \Sigma, P^+ \rangle$ is stable. In this case $\langle \Sigma, P^+ \rangle$ itself is a solution of the GCPP. In general, the solution of the GCPP can always be found by a suitable sequence of splits (of classes) and adequate completion of the relation in order to take the newly generated classes into account.

Remark 8.12 *Notice that it is important that $\preceq$ is reflexive (it is reflexive, since $\langle M, \preceq \rangle$ is a partition-pair). This is necessary in order to prove that there is always a unique maximal solution. Consider the graph $G = \langle N, \gtrdot \rangle$ with $N = \{n_1, n_2\}$ and $\gtrdot = \{(n_1, n_2)\}$, the partition-pair $\langle \Sigma, P \rangle$ with $\Sigma = \{N\}$ and $P = \{(N, N)\}$. It holds that both $\langle \Sigma, \emptyset \rangle$ and $\langle \Pi, Q \rangle$ with $\Pi = \{\{n_1\}, \{n_2\}\}$ and $Q = \{(\{n_1\}, \{n_1\}), (\{n_2\}, \{n_2\}), (\{n_2\}, \{n_1\})\}$ are maximal solution of the partition problem, but the first is not a partition-pair. Similarly the acyclicity condition is important because otherwise we could coarsen the partition merging all classes which form cycles.*

We have to prove that given a GCPP it has always a unique solution and we have to spell out the connection between the similarity quotient problems and the GCPP's. To this purpose we establish a connection between GCPP's and similarity quotient problems over labelled graphs that are obtained as s-unlabellings of others labelled graphs. The generalized s-unlabelling of $G = \langle N, \gtrdot \rangle$ and $\langle \Sigma, I \rangle$, where I is the identity relation over Σ, is exactly the s-unlabelling of $\langle N, \gtrdot, \Sigma \rangle$ (see Definition 8.2). In this sense the *generalized s-unlabelling* is a generalization of the s-unlabelling in the case that we have to deal with a partition-pair and not only with a partition.

Definition 8.13 (Generalized S-unlabelling) *Let $G = \langle N, \gtrdot \rangle$ be a graph and $\langle \Sigma, P \rangle$ be a partition-pair over G. Consider the labelled graph $G' = \langle N', \gtrdot', \Sigma' \rangle$ defined as:*

$$\begin{array}{rcl} L & = & \{\ell_\alpha \mid \alpha \in \Sigma\} \\ N' & = & N \uplus L \\ \gtrdot' & = & \gtrdot \cup \{\langle n, \ell_\alpha \rangle \mid n \in \beta \in \Sigma \wedge (\alpha, \beta) \in P^+\} \\ \Sigma' & = & \{N\} \cup \{\{\ell_\alpha\} \mid \ell_\alpha \in L\}, \end{array}$$

where $\uplus$ is the disjoint union. We call the labelled graph G' the generalized s-unlabelling *of G and $\langle \Sigma, P \rangle$.*

Here we have to use P^+ in the definition of $\gtrdot'$ in order to produce a simulation from a stable refinement, since α-nodes can simulate β-nodes whenever $(\alpha, \beta) \in P^+$. In Figure 8.3 we present a partition-pair together with its generalized s-unlabelling.

Now we prove the strong connection between stable refinements of $\langle \Sigma, P \rangle$ and simulations over G'.

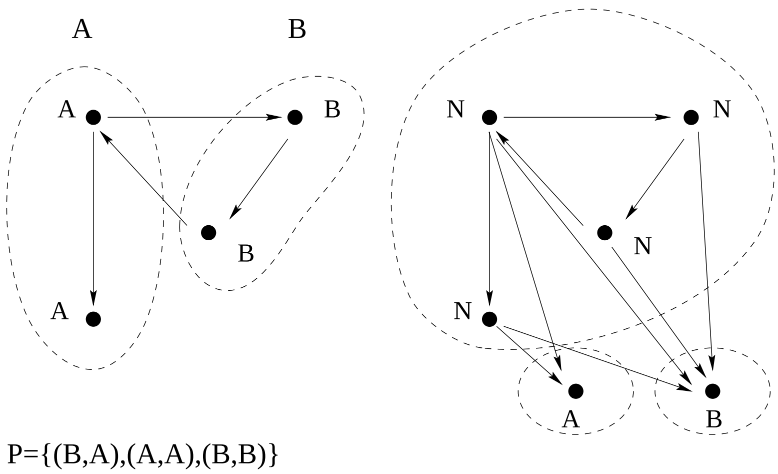


Figure 8.3: Generalized S-unlabelling Example.

Definition 8.14 *Let $G = \langle N, \gtrdot \rangle$ be a graph, $\langle \Sigma, P \rangle$ a partition-pair over G, and G' the generalized s-unlabelling of G and $\langle \Sigma, P \rangle$. Consider the set S_{GP} of the stable (w.r.t. $\gtrdot$) refinements of $\langle \Sigma, P \rangle$ and the set S_{sim} of the simulations over G'. We define two functions f_1 and f_2 as follows*[1]*:*

$$\begin{gathered} f_1 : S_{GP} \mapsto S_{sim} \\ f_1(\langle \Pi, Q \rangle) = \leq_{\langle \Pi, Q \rangle} \\ \forall n, m \in N(n \leq_{\langle \Pi, Q \rangle} m \textit{ iff } n \in \alpha \in \Pi \wedge m \in \beta \in \Pi \wedge (\alpha, \beta) \in Q) \\ \forall \ell_\alpha \in L(\ell_\alpha \leq_{\langle \Pi, Q \rangle} \ell_\alpha) \end{gathered}$$

$$\begin{gathered} f_2 : S_{sim} \mapsto S_{GP} \\ f_2(\leq) = \langle \Pi_s, Q_s \rangle \\ \forall n \in N([n] = \{m \mid n \leq^* m \leq^* n\} \in \Pi_s) \\ (\alpha, \beta) \in Q_s \textit{ iff } \exists n \in \alpha \exists m \in \beta (n \leq^* m) \end{gathered}$$

Notice that the condition $\exists n \in \alpha \exists m \in \beta (n \leq^* m)$ is equivalent to $\forall n \in \alpha \forall m \in \beta (n \leq^* m)$, since we are using the transitive closure of the simulation $\leq$. Notice also that it is necessary to prove that the codomains of f_1 and f_2 are correctly defined. We prove this fact and some properties of f_1 and f_2 in the following Lemma.

Lemma 8.15 *The function f_1 and f_2 are such that:*

1. *if $\langle \Pi, Q \rangle \in S_{GP}$, then $f_1(\langle \Pi, Q \rangle) \in S_{sim}$;*

2. *if $\leq \in S_{sim}$, then $f_2(\leq) \in S_{GP}$;*

3. *if $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$, then $f_1(\langle \Pi_1, Q_1 \rangle) \subseteq f_1(\langle \Pi_2, Q_2 \rangle)$;*

4. *if $\leq^1 \subseteq \leq^2$, then $f_2(\leq^1)) \sqsubseteq f_2(\leq^2)$;*

5. *$\leq \subseteq f_1(f_2(\leq))$;*

6. *$\langle \Pi, Q \rangle \sqsubseteq f_2(f_1(\langle \Pi, Q \rangle))$.*

[1] $\leq^*$ is the reflexive and transitive closure of $\leq$.

Proof. First we prove (1), i.e. we prove that if $\langle \Pi, Q\rangle$ belongs to S_{GP}, then $\leq_{\langle \Pi,Q\rangle}$ is a simulation over G'.
If $n \leq_{\langle \Pi,Q\rangle} m$, then only two cases are possible:

- n and m are elements of N;
- $n = m$ and n is an element of L.

Hence in both cases we have that they belong to the same class in Σ'.
The second case is trivial. In the first case if s is such that $n \geqslant' s$, then it can either be $s \in L$ or $s \in N$.
If $s \in L$, from $n \geqslant' s$ we have that $n \in \alpha \in \Sigma$, $s = \ell_\gamma$ and $(\gamma, \alpha) \in P^+$. From the fact that $n \leq_{\langle \Pi,Q\rangle} m$ we have that $n \in \alpha' \in \Pi$ and $m \in \beta' \in \Pi$ and $(\alpha', \beta') \in Q$. Hence we have that $m \in \beta \in \Sigma$ and $(\alpha, \beta) \in P^+$. From $(\gamma, \alpha) \in P^+$ and $(\alpha, \beta) \in P^+$ we obtain $(\gamma, \beta) \in P^+$, hence $m \geqslant' s$, and $s \leq_{\langle \Pi,Q\rangle} s$.
If $s \in N$, then we have that $n \in \alpha' \in \Pi$, $s \in \gamma' \in \Pi$, $m \in \beta' \in \Pi$, $\alpha' \geqslant_\exists \gamma'$, and $(\alpha', \beta') \in Q$ (since $n \leq_{\langle \Pi,Q\rangle} m$). Hence, since $\langle \Pi, Q\rangle$ is stable, there exists $\delta' \in \Pi$ such that $(\gamma', \delta') \in Q$ and $\beta' \geqslant_\forall \delta'$. Let $t \in \delta'$ be such that $m \geqslant t$. Since $s \leq_{\langle \Pi,Q\rangle} t$, the thesis follows.

It is immediate to prove that if $\leq$ is a simulation over G', then $\langle \Pi_s, Q_s\rangle$ is a partition-pair, i.e. Q_s is reflexive and acyclic.

Now we prove (2), i.e. if $\leq$ is a simulation over G', then $\langle \Pi_s, Q_s\rangle$ is a stable refinement of $\langle \Sigma, P\rangle$.
Observe that Q_s is acyclic.
Let α', β', γ' in Π_s be such that $\alpha' \geqslant_\exists \gamma'$ and $(\alpha', \beta') \in Q_s$. We have to prove that there exists $\delta' \in \Pi_s$ such that $(\gamma', \delta') \in Q_s$ and $\beta' \geqslant_\forall \delta'$.
From $\alpha' \geqslant_\exists \gamma'$ we obtain that there exist $n \in \alpha'$ and $s \in \gamma'$ such that $n \geqslant' s$. From $(\alpha', \beta') \in Q_s$ we obtain that $n \leq^* m$ for all $n \in \alpha'$ and $m \in \beta'$. Hence, since $\leq^*$ is a simulation, we have that for all $m \in \beta'$ there exists t_m such that $m \geqslant' t_m$ and $s \leq^* t_m$ (hence $t_m \in N$). Let $m_1, \ldots, m_b$ be an ordering of the elements of β'.

We inductively define as follows the lists $L^1, .., L^n, ..$ of elements related throught $\geqslant'$ with $m_1, \ldots, m_b$:

- $L^1 = t_1^1, \ldots, t_b^1$
  where t_1^1 is such that $m_1 \geqslant' t_1^1$ and $s \leq^* t_1^1$ and, for $j = 2, \ldots, b$, t_j^1 is such that $m_j \geqslant' t_j^1$ and $t_{j-1}^1 \leq^* t_j^1$. Notice that L^1 is correctly defined (i.e $t_1^1, \ldots, t_b^1$ always exist) since $\leq^*$ is a simulation, $n \leq^* m \wedge n \geqslant' s$ and, for $j = 2, \ldots, b$, $m_{j-1} \leq^* m_j$.
- $L^{h+1} = t_1^{h+1}, \ldots, t_b^{h+1}$
  where t_1^{h+1} is such that $m_1 \geqslant' t_1^{h+1}$ and $t_b^h \leq^* t_1^{h+1}$ and, for $j = 2, \ldots, b$, t_j^{h+1} is such that $m_j \geqslant' t_j^{h+1}$ and $t_{j-1}^{h+1} \leq^* t_j^{h+1}$.
  L^{h+1} is correctly defined (i.e $t_1^{h+1}, \ldots, t_b^{h+1}$ always exist) since $m_1, \ldots, m_b$ is an ordering of the elements of the class β' and $\leq^*$ is a simulation.

By concatenation of the lists $L^1, L^2, \ldots$ we obtain an infinite chain of elements

$$t_1^1, \ldots, t_b^1, t_1^2, \ldots, t_b^2, .., t_1^n, \ldots, t_b^n, ..$$

such that

1. for all $k > 0$ and for $j = 1, \ldots, b$ $m_j^k \geqslant' t_j^k$
2. for all $k > 0$ and for $j = 2, \ldots, b$, $t_{j-1}^k \leq^* t_j^k$

3. $s \leq^* t_1^1$ and for all $k > 0$ $t_b^{k-1} \leq^* t_1^k$

Since the set of the successors of m_1 is a finite set we have that there exist v, u such that $t_1^v = t_1^u = t_1^*$ and $v < u$. Hence we obtain a prefix of the chain of the form $t_1^1 \ldots \leq^* t_1^* \leq^* \ldots \leq^* t_1^*$ with $s \leq^* t_1^1$. This means that all the elements of this succession between the two occurrences of t_1^* belongs to the same class $\delta' \in \Pi$. Moreover all the elements β' $(m_1, \ldots, m_b)$ have a successor in δ' and obviously $(\gamma', \delta') \in Q_s$, i.e. the thesis.
Now we prove (3).
Let $\leq_1 = f_1(\langle \Pi_1, Q_1 \rangle)$ and $\leq_2 = f_1(\langle \Pi_2, Q_2 \rangle)$. We assume that $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$ and we have to prove that $\leq_1 \subseteq \leq_2$.
If $n \leq_1 m$, then $n \in \alpha_1 \in \Pi_1$, $m \in \beta_1 \in \Pi_1$ and $(\alpha_1, \beta_1) \in Q_1$, hence $n \in \alpha_2 \in \Pi_2$, $m \in \beta_2 \in \Pi_2$ and $(\alpha_2, \beta_2) \in Q_2$, from which $n \leq_1 m$.

We prove (4).
Let $\langle \Pi_1, Q_1 \rangle = f_2(\leq_1)$ and $\langle \Pi_2, Q_2 \rangle = f_2(\leq_2)$. We assume that $\leq_1 \subseteq \leq_2$ and we prove that $\langle \Pi_1, Q_1 \rangle \sqsubseteq \langle \Pi_2, Q_2 \rangle$.
From $\leq_1 \subseteq \leq_2$ we have $\leq_1^* \subseteq \leq_2^*$.
If $n, m \in \alpha_1 \in \Pi_1$, then $n \leq_1^* m \leq_1^* n$, hence $n \leq_2^* m \leq_2^* n$, from which $n, m \in \alpha_2 \in \Pi_2$, i.e. Π_1 is finer than Π_2.
If $(\alpha_1, \beta_1) \in Q_1$, then since Π_1 is finer than Π_2 there exists $\alpha_2, \beta_2 \in \Pi_2$ such that $\alpha_1 \subseteq \alpha_2$ and $\beta_1 \subseteq \beta_2$. We have to prove that $(\alpha_2, \beta_2) \in Q_2$. Let $s \in \alpha_1$ and $t \in \beta_1$ be such that $s \leq_1^* t$. It holds that $s \leq_2 Tt$, $s \in \alpha_2$ and $t \in \beta_2$, hence $(\alpha_2, \beta_2) \in Q_2$.

Let us prove (5).
If $n \leq m$, then $n \in \alpha \in \Pi_s$, $m \in \beta \in \Pi_s$ and $(\alpha, \beta) \in Q_s$. Hence we obtain $n \leq_{\langle \Pi_s, Q_s \rangle} m$.

It only remains to prove (6).
If $n, m \in \alpha \in \Pi$, then, since Q is reflexive, it holds $n \leq_{\langle \Pi, Q \rangle} m \leq_{\langle \Pi, Q \rangle} n$, hence $n, m \in \alpha' \in \Pi_{\langle \Pi, Q \rangle}$, i.e. Π is finer than $\Pi_{\langle \Pi, Q \rangle}$. Let $(\alpha, \beta) \in Q$, consider $\alpha', \beta' \in \Pi_{\langle \Pi, Q \rangle}$ such that $\alpha \subseteq \alpha'$ and $\beta \subseteq \beta'$. Let $s \in \alpha$ and $t \in \beta$. It holds that $s \leq_{\langle \Pi, Q \rangle} t$, $s \in \alpha'$, and $t \in \beta'$, hence $(\alpha', \beta') \in Q_{\langle \Pi, Q \rangle}$. □

Theorem 8.16 (Existence and Uniqueness) *Let $G = \langle N, \gtrdot \rangle$ be a graph and $\langle \Sigma, P \rangle$ be a partition-pair over G, the GCPP over G and $\langle \Sigma, P \rangle$ has always a unique solution.*

Proof. Let $\leq_s$ be the maximal simulation over G'. From Lemma 8.15(2) we have that $\langle \Pi_m, Q_m \rangle = f_2(\leq_s)$ is a stable refinement of $\langle \Sigma, P \rangle$. We prove that if $\langle \Pi, Q \rangle$ is another stable refinement, then $\langle \Pi, Q \rangle \sqsubseteq \langle \Pi_m, Q_m \rangle$. From Lemma 8.15(6) we know that $\langle \Pi, Q \rangle \sqsubseteq f_2(f_1(\langle \Pi, Q \rangle))$. Since $f_1(\langle \Pi, Q \rangle)$ is a simulation over G' we also know that $f_1(\langle \Pi, Q \rangle) \subseteq \leq_s^m$. Hence, using Lemma 8.15(4) we obtain that $\langle \Pi, Q \rangle \sqsubseteq f_2(f_1(\langle \Pi, Q \rangle)) \sqsubseteq f_2(\leq_s^m) = \langle \Pi_m, Q_m \rangle$. $\langle \Pi_m, Q_m \rangle = f_2(\leq_s)$ is then a solution of the GCPP over G and $\langle \Sigma, P \rangle$; by contradiction suppose that this solution is not unique. If $\langle \Pi', Q' \rangle \neq \langle \Pi_m, Q_m \rangle$ is another solution of the given problem then $\langle \Pi_m, Q_m \rangle \sqsubseteq \langle \Pi', Q' \rangle$. Using Definition 8.14 of f_1 (which maps stable refinments $\langle \Sigma, P \rangle$ into simulation over G') we immediately deduce $f_1(\langle \Pi', Q' \rangle) \neq f_1(\langle \Pi_m, Q_m \rangle)$ from $\langle \Pi', Q' \rangle \neq \langle \Pi_m, Q_m \rangle$. By Lemma 8.15(3) and 8.15(5) we then obtain the absurd $\leq_s = f_1(f_2(\leq_s)) \subset f_1(\langle \Pi', Q' \rangle)$. □

Corollary 8.17 *If $\langle M, \preceq \rangle$ is the solution of the GCPP over G and $\langle \Sigma, P \rangle$, then $\preceq$ is a partial order over M.*

Proof. We have to prove that $\preceq$ is transitive. This is an immediate consequence of the fact that $\langle M, \preceq \rangle = f_2(\leq_s)$, where $\leq_s$ is the maximal simulation over G' (generalized s-unlabelling). □

We have proved that each generalized coarsest partition problem has a unique solution which can be determined by solving a similarity quotient problem. We are interested also in the opposite direction, i.e. we want to prove that a similarity quotient problem can be solved using a generalized coarsest partition problem. That is what we prove in the following theorem.

Theorem 8.18 (Simulation as GCPP) *Let $G = \langle N, \to, \Sigma\rangle$ be a labelled graph and let the pair $\langle \Pi, Q\rangle$ be the solution of the GCPP over $G^- = \langle N, \to\rangle$ and $\langle \Sigma, I\rangle$, where I is the identity relation over Σ. Π is the simulation quotient of G, and $f_1(\langle \Pi, Q\rangle)$ is the maximal simulation over G.*

Proof. As we already noticed, since the relation over Σ is the identity relation, we have that the generalized s-unlabelling G' of G^- is exactly the s-unlabelling of G. Hence from Lemma 8.15 and Theorem 8.5 we immediately obtain the thesis. □

Hence, in order to solve the problem of determining the simulation quotient of a labelled graph $G = \langle N, \to, \Sigma\rangle$ we can equivalently solve the generalized coarsest partition problem over $\langle N, \to\rangle$ and $\langle \Sigma, I\rangle$. If $\langle M, \preceq\rangle$ is the solution of the GCPP, then the relation $\leq_{\langle M, \preceq\rangle}$ defined as

$$\forall n, m \in N(n \leq_{\langle M,\preceq\rangle} m \text{ iff } [n] \preceq [m])$$

is the maximal simulation over G, and M is the partition which corresponds to the sim-equivalence $\equiv_s$ (c.f. Definition 7.1).

8.2 Computing a Solution to the GCPP

We now introduce an operator σ mapping partition-pairs into partition-pairs, which will turn out to be the engine of our algorithm. The results in this section will allow us to conclude that a procedure which compute σ can be used to solve GCPP's and, hence, to compute similarity quotients: it is only necessary to iterate the computation of σ at most $|M|$ times.

In particular, the operator σ is defined in such a way that it refines the partition-pair $\langle \Sigma, P\rangle$ obtaining a partition-pair $\langle \Pi, Q\rangle$ which is *more stable* than $\langle \Sigma, P\rangle$ and is never finer than the solution of the GCPP over $\langle \Sigma, P\rangle$. In the first condition of σ we impose to split the classes of Σ which do not respect the stability condition w.r.t. themselves: if a class α is such that $\alpha \to_\exists \gamma$ and it does not exists a class δ such that $(\gamma, \delta) \in P$ and $\alpha \to_\forall \delta$, then the pair (α, α) does not respect the stability condition, hence we must split α. The first condition is used to build Π, and then the second and the third conditions are in σ used to define Q using the Π already obtained. Intuitively, the second and the third conditions allow to obtain Q from P by starting from $P(\Pi)$ and removing the minimum number of pairs which contradict stability.

Definition 8.19 (Operator σ) *Let $G = \langle N, \to\rangle$ be a graph and $\langle \Sigma, P\rangle$ be a partition-pair over G, the partition-pair $\langle \Pi, Q\rangle = \sigma(\langle \Sigma, P\rangle)$ is defined as:*

(1σ) Π *is the coarsest partition finer than* Σ *such that*

$$(a) \quad \forall \alpha \in \Pi\, \forall \gamma \in \Sigma(\alpha \to_\exists \gamma \to \exists \delta \in \Sigma((\gamma, \delta) \in P \wedge \alpha \to_\forall \delta));$$

(2σ) Q *is the maximal relation over* Π *such that* $Q \subseteq P(\Pi)$ *and if* $(\alpha, \beta) \in Q$*, then:*

$$(b) \quad \forall \gamma \in \Sigma(\alpha \to_\forall \gamma \to \exists \gamma' \in \Sigma((\gamma, \gamma') \in P \wedge \beta \to_\exists \gamma')) \quad \textit{and}$$
$$(c) \quad \forall \gamma \in \Pi(\alpha \to_\forall \gamma \to \exists \gamma' \in \Pi((\gamma, \gamma') \in Q \wedge \beta \to_\exists \gamma')).$$

By abuse of notation we use $\gg_\exists$ and $\gg_\forall$ also when the classes belong to different partitions.

Condition (a) in (1σ) imposes to opportunely split the classes of the partition Σ: these splits are forced by the fact that we are looking for a partition-pair *stable* w.r.t. the relation of the given graph and we know that in each partition-pair $\langle \Sigma, P\rangle$ the second component is reflexive (i.e. $\forall \alpha \in \Sigma(\alpha,\alpha) \in P$). Using condition (b) in (2σ), together with condition (a) in (1σ) and exploiting the fact that P is acyclic, it is possible to prove that Q is acyclic: the acyclicity of P ensures that a cycle could arise only among classes of Π which are all contained in a unique class of Σ, then using condition (a) in (1σ) and (b) in (2σ) we obtain a contradiction. Condition (c) is fundamental, together with condition (a), in order to obtain the result in Theorem 8.25: if it holds that $\alpha E_\forall \gamma$, then *no matter how we split* α one of the subclasses generated from α has a chance (in the solution of GCPP) to be in relation with at least one of the subclasses generated from β only if $\beta E_\exists \gamma'$ and γ is in relation with γ'. The following results guarantee the correctness of σ and, hence, of our approach.

Lemma 8.20 (Existence) *Let $G = \langle N, \gg\rangle$ be a graph and $\langle \Sigma, P\rangle$ be a partition-pair over G. There exists at least a partition-pair $\langle \Pi, Q\rangle$ which satisfies the conditions in Definition 8.19, i.e. σ is always defined.*

Proof. Consider the partition Π_1 defined as $\Pi_1 = \{\{n\} \mid n \in N\}$. Clearly Π_1 is finer than Σ. Π_1 satisfies all the conditions in (1σ), since P is reflexive (even though it can easily be the case that Π_1 is not the coarsest).
Let Π be a partition such that:

- Π is finer than Σ;
- Π_1 is finer than Π;
- Π satisfies all the conditions in (1σ);
- if Π is finer than Π_2 and Π_2 is finer than Σ, then Π_2 does not satisfy the conditions in (1σ).

Notice that there exists at least a partition Π which satisfies all these conditions because there are finitely many partitions on N.

Consider $Q_1 = \{(\alpha,\alpha) \mid \alpha \in \Pi\}$ over Π.
Clearly Q_1 satisfies all the conditions in (2σ), but it can be the case that it is not maximal. Let Q be a relation over Π such that:

- $Q_1 \subseteq Q \subseteq P(\Pi)$;
- Q_1 is acyclic;
- Q satisfies all the conditions in (2σ);
- if Q_2 is such that $Q \subseteq Q_2 \subseteq P(\Pi)$, then Q_2 does not satisfy the conditions in $(2)\sigma$.

There exists at least a relation Q which satisfies all these conditions because there are only a finite number of relations. $\square$

Theorem 8.21 (Uniqueness) *Let $G = \langle N, \gg\rangle$ be a graph and $\langle \Sigma, P\rangle$ be a partition-pair over G. There exists a unique maximal pair $\langle \Pi, Q\rangle$ which satisfies the conditions in Definition 8.19, i.e. σ is always uniquely defined.*

Proof. The previous lemma states that there always exists a maximal pair $\langle \Pi, Q \rangle$ which satisfies the conditions in Definition 8.19; we have to prove that this maximal pair is unique. Let Sort be a topological sorting of the elements in Σ with respect to P.
Consider Π defined in Figure 8.4. Notice that Π does not depend on the topological sorting

$\Pi := \Sigma;$
while $\mathsf{Sort} \neq \emptyset$ do
    $\gamma := \mathit{dequeue}(\mathsf{Sort});$
    for each $\alpha \in \Pi$ do
        if $\alpha \gg_{\exists} \gamma \wedge \forall \delta((\gamma, \delta) \in P \rightarrow \neg \alpha \gg_{\forall} \delta)$ then
            replace α with $\alpha \cap \gg^{-1}(\gamma)$ and $\alpha \setminus \gg^{-1}(\gamma)$ in Π

Figure 8.4: Existence.

we have chosen.
It is clear that Π satisfies (1σ).
Let us assume by contradiction that there exists Π_1 such that Π_1 satisfies (1σ) and Π_1 is not finer than Π. This means that there exist $n, m \in N$, $\alpha \in \Pi_1$ and $\alpha_n, \alpha_m \in \Pi$ such that $n, m \in \alpha$, $n \in \alpha_n$, $m \in \alpha_m$ and $\alpha_n \neq \alpha_m$. Consider the class α' which was obtained during the construction of Π and which was the smallest class obtained during the construction such that $\alpha \subseteq \alpha'$. Let γ be the class of Σ which first split α' into two parts. Notice that for all δ such that $(\gamma, \delta) \in P$ and $\delta \neq \gamma$, it does not hold $\alpha' \gg_{\exists} \delta$ (since all these δ's have already been extracted from Sort). Hence it holds that for all δ such that $(\gamma, \delta) \in P$ and $\delta \neq \gamma$ $\neg \alpha \gg_{\forall} \delta$, moreover, since α is not a subset of one of the two part of α', $\neg \alpha \gg_{\forall} \gamma$ and $\alpha \gg_{\exists} \gamma$. This is in contradiction with the fact that Π_1 satisfies (1σ).

Similarly it is possible to prove that there exists a unique maximal relation Q over Π which satisfies (2σ).

It is possible to obtain Q from $P(\Pi)$ by removing all the pairs which do not fulfill the condition, until a fix point is reached. It is immediate to prove that such a fix point is reflexive. In fact, being $\langle \Sigma, P \rangle$ a partition-pair over G, we have that P and hence $P(\Pi)$ are reflexive. As each pair belonging to $\{(\alpha, \alpha) \mid \alpha \in \Pi\}$ respect both conditions (b) and (c) in (2σ), we can conclude that Q is reflexive.

As far as the acyclicity of Q is concerned consider the relation Q_b on Π obtained from $P(\Pi)$ by removing those pairs which do not fulfill condition (b) in (2σ); we will prove that Q_b is acyclic. Being Q contained in Q_b we will obtain, as a byproduct, the acyclicity of Q.

Assume, by contradiction, that Q_b is cyclic i.e that there exist $\alpha_1, \alpha_2, .., \alpha_n \in \Pi$ with $\alpha_i \neq \alpha_j$ for $1 \leq i, j \leq n, i \neq j$ such that $(\alpha_1, \alpha_2) \in Q_b \wedge .. \wedge ..(\alpha_{n-1}, \alpha_n) \in Q_b \wedge (\alpha_n, \alpha_1) \in Q_b$. Since P is acyclic and $Q \subseteq P(\Pi)$ there must exists a class $\alpha \in \Sigma$ such that $\alpha_1, \ldots, \alpha_n \subseteq \alpha$. Let α be the class in Σ such that $\alpha_i \subseteq \alpha$ for $1 \leq i \leq n$. We have just proved that Π, i.e the coarsest partition finer than Σ which fulfill condition (a) in (2σ), can be obtained using the procedure in Figure 8.4, where Sort is a topological sort of the elements in Σ with respect to P. As, for $1 \leq i, j \leq n \wedge i \neq j$, we have $\alpha_i \neq \alpha_j$ and $\alpha_i \subseteq \alpha \in \Sigma$, by the above pseudo-code we deduce that there exist $\gamma \in \Sigma$, a permutation of $\alpha_1, .., \alpha_n$, say $\alpha'_1, .., \alpha'_n$, and an index $1 \leq k \leq n-1$ such that

$$\alpha'_1 \gg_{\forall} \gamma \wedge .. \wedge \alpha'_k \gg_{\forall} \gamma \wedge \nexists \delta \in \Sigma((\gamma, \delta) \in P \wedge (\alpha'_{k+1} \gg_{\exists} \delta \vee .. \vee \alpha'_n \gg_{\exists} \delta)) \tag{8.1}$$

Let's use the letters M and N to refer respectively to $\alpha'_1 \cup .. \cup \alpha'_k$ and $\alpha'_{k+1} \cup .. \cup \alpha'_n$. As $\alpha'_1, .., \alpha'_n$ is a permutation of $\alpha_1, .., \alpha_n$ and we have supposed that $(\alpha_1, \alpha_2) \in Q_b \wedge .. \wedge$

..$(\alpha_{n-1},\alpha_n) \in Q_b \wedge (\alpha_n,\alpha_1) \in Q_b$ we have that there exist two indexes $1 \leq t, s \leq n$ such that $s = t+1 \vee (t = n \wedge s = 1)$ and $\alpha_t \in M \wedge \alpha_s \in N$. From $\alpha_t \in M$ we obtain that $\alpha_t \geqslant_\forall \gamma$; since $(\alpha_t,\alpha_s) \in Q_b$ and Q_b respect condition (b) in (2σ) we have that $\alpha_s \geqslant_\exists \delta \wedge (\gamma,\delta) \in P$ and $\alpha_s \in N$ which contradicts the condition in Equation (8.1). □

Now that we have obtained that given a partition-pair $\langle \Sigma, P\rangle$ there exists a unique solution of $\sigma(\langle \Sigma, P\rangle)$, we want to connect fix points of the operator σ with solutions of GCPP's. We start with some lemma.

Lemma 8.22 *Let $\langle M, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Sigma, P\rangle$. If $\langle M, \preceq\rangle \sqsubseteq \langle \Pi, Q\rangle$, then $\langle M, \preceq\rangle \sqsubseteq \sigma(\langle \Pi, Q\rangle)$.*

Proof. From Corollary 8.17 we have that $\preceq$ is transitive.
We have to prove that if $\langle M, \preceq\rangle \sqsubseteq \langle \Pi, Q\rangle$, then $\langle M, \preceq\rangle \sqsubseteq \sigma(\langle \Pi, Q\rangle) = \langle \Pi', Q'\rangle$ i.e that

1. M is a partition finer than Π'
2. The partial order (on M) $\preceq$ is included in the relation induced on M by Q' (which is an acyclic and reflessive relation on Π')

Consider 1.: if we prove that $\langle M, \preceq\rangle$ satisfies $(1\sigma)(a)$ (over $\langle \Pi, Q\rangle$) we can then conclude, by theorem 8.21, that M is comparable with with Π' and finer then Π' i.e our thesis. So let's start proving that $\langle M, \preceq\rangle$ satisfies $(1\sigma)(a)$ (over $\langle \Pi, Q\rangle$). If $\beta \in M$ and $\alpha \in \Pi$ are such that $\beta \geqslant_\exists \alpha$, then there exists $\alpha' \in M$ such that $\alpha' \subseteq \alpha$ and $\beta \geqslant_\exists \alpha'$. Since $\langle M, \preceq\rangle$ is the solution of a GCPP it holds that there exists $\delta' \in M$ such that $(\alpha',\delta') \in \preceq$ and $\beta \geqslant_\forall \delta'$. Hence since $\preceq \subseteq Q(M)$ it holds that $\delta' \subseteq \delta \in \Pi$, $(\alpha,\delta) \in Q$, and $\beta \geqslant_\forall \delta$, i.e. (1σ) is satisfied.

Consider 2.: we have to prove that the partial order (on M) $\preceq$ is included in the relation induced on M by Q', where $\langle \Pi', Q'\rangle = \sigma(\langle \Pi, Q\rangle)$. Notice that as a by-product of 1. we have that the partition M is finer than Π' which is finer than Π. We will use the notation α_P, β_P.. to denote the elements of a partition P. Suppose by contradiction that there exists $\overline{\alpha}_M, \overline{\beta}_M, \overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'}$ such that

$$\overline{\alpha}_M \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_M \subseteq \overline{\beta}_{\Pi'} \text{ and } (\overline{\alpha}_M \preceq \overline{\beta}_M \ \wedge \ (\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'}) \notin Q').$$

Without loss of generality we can assume that the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is the first one which is not in Q' and is such that there exist $\overline{\alpha}_M \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_M \subseteq \overline{\beta}_{\Pi'}$ with $\overline{\alpha}_M \preceq \overline{\beta}_M$. We have to consider 3 cases:

1. the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is not in the relation induced on Π' by Q;
2. the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ has been removed from the relation induced on Π' by Q in order to satisfy condition (b) in (2σ);
3. the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ has been removed from Q' in order to satisfy condition (c) in (2σ).

In case (a), consider the classes of Π, $\overline{\alpha}_\Pi$ and $\overline{\beta}_\Pi$ such that $\overline{\alpha}_\Pi \supseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_\Pi \supseteq \overline{\beta}_{\Pi'}$; we immediately deduce that $\overline{\alpha}_M \supseteq \overline{\alpha}_\Pi, \overline{\beta}_M \supseteq \overline{\beta}_\Pi$ and $(\overline{\alpha}_M \preceq \overline{\beta}_M \ \wedge \ (\overline{\alpha}_\Pi, \overline{\beta}_\Pi) \notin Q,)$ which contradicts our assumption that $\langle M, \preceq\rangle \sqsubseteq \langle \Pi, Q\rangle$.

In case (b), recall that since $(\langle M, \preceq\rangle)$ is the solution of the GCPP it holds that $(\langle M, \preceq\rangle)$ is stable i.e

$$\forall \alpha_M, \beta_M, \gamma_M (\alpha_M \preceq \beta_M \wedge \alpha_M \geqslant_\exists \gamma_M \rightarrow \exists \delta_M (\gamma_M \preceq \delta_M \wedge \beta_M \geqslant_\forall \delta_M)).$$

If the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is assumed not to bee in Q' because of condition (b) in (2σ), then there exists a class of Π, γ_Π, such that:

$$\overline{\alpha}_{\Pi'} \gg_\forall \gamma_\Pi \wedge \neg\exists\delta_\Pi((\gamma_\Pi, \delta_\Pi) \in Q \wedge \overline{\beta}_{\Pi'} \gg_\exists \delta_\Pi) \tag{8.2}$$

From $\overline{\alpha}_{\Pi'} \gg_\forall \gamma_\Pi$ we obtain that that for every class of M, α_M which is included[2] in $\alpha_{\Pi'}$ there exists a class of M included in γ_Π, γ_M, such that $\alpha_M \gg_\exists \gamma_M$. Hence we have that there is a class of M, say γ^*_M, included in γ_Π and such that $\overline{\alpha}_M \gg_\exists \gamma^*_M$. From the second conjunct in assertion (8.2), $\gamma^*_M \subseteq \gamma_\Pi$, $\overline{\beta}_M \subseteq \overline{\beta}_{\Pi'}$ and from our assumption that $\preceq$ is included in the relation on M induced by Q we deduce that

$$\neg\exists\delta_M(\gamma^*_M \preceq \delta_M \wedge \overline{\beta}_M \gg_\exists \delta_M)$$

and hence that

$$\overline{\alpha}_M \preceq \overline{\beta}_M \wedge \overline{\alpha}_M \gg_\exists \gamma^*_M \wedge \neg\exists\delta_M(\gamma^*_M \preceq \delta_M \wedge \overline{\beta}_M \gg_\forall \delta_M)$$

i.e $\langle M, \preceq\rangle$ is not stable, which is a contradiction.

Finally in case (c), recall that we assumed that the pair $(\overline{\alpha}_{\Pi'}, \overline{\beta}_{\Pi'})$ is the first one which is not in Q' and is such that there exist $\overline{\alpha}_M \subseteq \overline{\alpha}_{\Pi'}, \overline{\beta}_M \subseteq \overline{\beta}_{\Pi'}$ with $\overline{\alpha}_M \preceq \overline{\beta}_M$.
Exploiting this assumption and using the same reasoning-schema used in (b) we deduce the absurd that $\langle M, \preceq\rangle$ is not stable. □

Lemma 8.23 *Let $G = \langle N, \gg\rangle$ be a graph and $\langle \Pi, Q\rangle$ be a partition-pair over G. Let $\langle M, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Pi, Q\rangle$. If $\sigma(\langle \Pi, Q\rangle) = \langle \Pi, Q\rangle$, then $\langle \Pi, Q^+\rangle = \langle M, \preceq\rangle$.*

Proof. We know that, by definition, $\langle M, \preceq\rangle \sqsubseteq \langle \Pi, Q^+\rangle$. Hence we only have to prove that Q^+ is stable w.r.t G. If $\alpha, \beta, \gamma \in \Pi$ are such that $(\alpha, \beta) \in Q^+$ and $\alpha \gg_\exists \gamma$, then from the condition in (1σ) applied to α and γ we have that there exists $\gamma'_1 \in \Pi$ such that $(\gamma, \gamma'_1) \in Q$ and $\alpha \gg_\forall \gamma'_1$. Let $\beta_1, \ldots, \beta_n$ be such that $(\alpha, \beta_1) \in Q$, $(\beta_i, \beta_{i+1}) \in Q$ and $\beta_n = \beta$. Using the second condition of (2σ) on (α, β_1) and γ'_1 we obtain that there exists γ''_1 such that $(\gamma'_1, \gamma''_1) \in Q$ and $\beta_1 \gg_\exists \gamma''_1$. Now, applying (1σ) to β_1 and γ''_1 we obtain that there exists γ'''_1 such that $(\gamma''_1, \gamma'''_1) \in Q$ and $\beta_1 \gg_\forall \gamma'''_1$. Similarly by induction on n we obtain a chain of classes of Q of the form $\gamma, \gamma'_1, \gamma''_1, \gamma'''_1, \ldots, \gamma'_n, \gamma''_n, \gamma'''_n$ such that $\beta \gg_\forall \gamma'''_n$. Hence $(\gamma, \gamma'''_n) \in Q^+$ and $\beta \gg_\forall \gamma'''_n$, i.e. our thesis. □

Lemma 8.24 *Let $G = \langle N, \gg\rangle$ be a graph and $\langle \Pi, Q\rangle \sqsubseteq \langle \Sigma, P\rangle$ be two partition-pairs. Let $\langle M, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Sigma, P\rangle$. If $\langle M, \preceq\rangle \sqsubseteq \langle \Pi, Q^+\rangle$, then $\langle M, \preceq\rangle$ is also the solution of the GCPP over G and $\langle \Pi, Q\rangle$.*

Proof. Immediate from the definition of GCPP. □

We are now ready to state the first important result in this section, i.e. σ can be used to solve GCPP's.

Lemma 8.25 (Fix-Point) *Let $G = \langle N, \gg\rangle$ be a graph and $\langle \Sigma, P\rangle$ a partition-pair over G with P transitive. Let $\langle M, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Sigma, P\rangle$. If n is such that $\sigma^n(\langle \Sigma, P\rangle) = \sigma^{n+1}(\langle \Sigma, P\rangle)$, then $\sigma^n(\langle \Sigma, P\rangle) = \langle M, \preceq\rangle$.*

[2] From 1. we have that the partition M is finer than Π' which is finer then Π.

Proof. Since P is transitive we have that $\langle M, \preceq\rangle \sqsubseteq \langle \Sigma, P\rangle$. Hence applying Lemma 8.22 we obtain that $\langle M, \preceq\rangle \sqsubseteq \sigma(\langle \Sigma, P\rangle)$. Assume that we have proved that $\langle M, \preceq\rangle \sqsubseteq \sigma^i(\langle \Sigma, P\rangle)$ applying Lemma 8.22 we obtain that $\langle M, \preceq\rangle \sqsubseteq \sigma^{i+1}(\langle \Sigma, P\rangle)$.
Hence we have obtained that for all $i \in \mathbb{N}$ it holds that

$$\langle M, \preceq\rangle \sqsubseteq \sigma^i(\langle \Sigma, P\rangle) \sqsubseteq \langle \Sigma, P\rangle.$$

Moreover from the fact that P is transitive we obtain that

$$\sigma^i(\langle \Sigma, P\rangle)^+ \sqsubseteq \langle \Sigma, P\rangle,$$

where $\sigma^i(\langle \Sigma, P\rangle)^+$ is the transitive closure of the relation in $\sigma^i(\langle \Sigma, P\rangle)$. Hence we have that

$$\langle M, \preceq\rangle \sqsubseteq \sigma^i(\langle \Sigma, P\rangle)^+,$$

so applying Lemma 8.24 we obtain that for all i it holds that $\langle M, \preceq\rangle$ is also the solution of the GCPP over $\sigma^i(\langle \Sigma, P\rangle)$.
Applying Lemma 8.23 to $\sigma^n(\langle \Sigma, P\rangle)$ we obtain that $\langle M, \preceq\rangle = \sigma^n(\langle \Sigma, P\rangle)^+$ and we already have that $\langle M, \preceq\rangle \sqsubseteq \sigma^n(\langle \Sigma, P\rangle)$, hence it must be $\langle M, \preceq\rangle = \sigma^n(\langle \Sigma, P\rangle)$. □

The meaning of this theorem is that if $\langle \Sigma, P\rangle$ is a partition-pair over a graph G such that P is transitive, then there exists i such that $\sigma^i(\langle \Sigma, P\rangle) = \sigma^{i+1}(\langle \Sigma, P\rangle)$ and $\sigma^i(\langle \Sigma, P\rangle)$ is the solution of the GCPP over G and $\langle \Sigma, P\rangle$.

Now we have to face with two questions:

- what happens if we want to solve a GCPP in which P is not transitive?
- how large is the index i in the worst case?

The first question is easy. Given a graph G and a partition-pair $\langle \Sigma, P\rangle$ the solution of the GPPC over G and $\langle \Sigma, P\rangle$ is equal to the solution of the GPPC over G and $\langle \Sigma, P^+\rangle$, i.e. we do not loose in generality if we consider only GCPP in which P is transitive.

The answer to the second question allow us to point out another important property of the operator σ. When we iteratively apply the operator σ until we reach a fix point, at each iteration we refine a partition and we remove pairs from a relation. What we prove in the following theorem is that is never the case that there exists an iteration in which we do not refine the partition, but we remove pairs from the relation. In a certain sense this means that the two conditions we have given in (2σ) to remove pairs are *optimal.*

Theorem 8.26 (Complexity) *Let $G = \langle N, \Rightarrow\rangle$ be a graph and $\langle \Sigma, P\rangle$ a partition-pair over G with P transitive. Let $\langle M, \preceq\rangle$ be the solution of the GCPP over G and $\langle \Sigma, P\rangle$. If i is such that $\sigma^i(\langle \Sigma, P\rangle) = \langle \Sigma_i, \mathsf{P}_i\rangle$ and $\sigma^{i+1}(\langle \Sigma, P\rangle) = \langle \Sigma_i, \mathsf{P}_{i+1}\rangle$, then $\mathsf{P}_{i+1} = \mathsf{P}_i$ and $\langle \Sigma_i, \mathsf{P}_i\rangle = \langle M, \preceq\rangle$.*

Proof. Let $\langle M, \preceq\rangle$ be the solution of the GCPP on G and $\langle \Sigma, P\rangle$. Using the same reasoning-schema of Lemma 8.25 we obtain that $\langle M, \preceq\rangle \sqsubseteq \langle \Sigma_i, \mathsf{P}_{i+1}\rangle \sqsubseteq \langle \Sigma_i, \mathsf{P}_i\rangle$. If we prove that $\langle \Sigma_i, \mathsf{P}_i^+\rangle$ is stable we can conclude by Lemma 8.15 that $\langle M, \preceq\rangle \sqsupseteq \langle \Sigma_i, \mathsf{P}_i^+\rangle$ and hence $\langle M, \preceq\rangle = \langle \Sigma_i, \mathsf{P}_i\rangle$ and $\mathsf{P}_{i+1} = \mathsf{P}_i$.
Let $\alpha, \beta, \gamma \in \Sigma_i$ be such that $(\alpha, \beta) \in \mathsf{P}_i^+$ and $\alpha \Rightarrow_\exists \gamma$. From the fact that $(\alpha, \beta) \in \mathsf{P}_i^+$ we have that there exists $(\alpha, \beta_1), \ldots, (\beta_n, \beta) \in \mathsf{P}_i$. Since $\alpha \Rightarrow_\exists \gamma$ and $\Sigma_{i+1} = \Sigma_i$, we have that there exists γ' such that $(\gamma, \gamma') \in \mathsf{P}_i$ and $\alpha \Rightarrow_\forall \gamma'$. Since P_i is obtained from $\sigma^i(\langle \Sigma, P\rangle)$ we obtain that there exists δ_1' such that $(\gamma', \delta_1') \in \mathsf{P}_i$ and $\beta_1 \Rightarrow_\exists \delta_1'$. Using again the fact that

$\Sigma_{i+1} = \Sigma_i$ we obtain that there exists δ_1 such that $(\delta'_1, \delta_1) \in \mathsf{P}_1$ and $\beta_1 \gg_\forall \delta_1$. By induction on n we arrive to the conclusion that there exists δ such that $(\gamma, \delta) \in \mathsf{P}_i^+$ and $\beta \gg_\forall \delta$. □

Hence from this Theorem, since σ never adds pairs to the relations, we can conclude that if P is transitive and $\Sigma_{i+1} = \Sigma_i$, then $\mathsf{P}_{i+1} = \mathsf{P}_i$, this last is transitive, and $\langle \Sigma_i, \mathsf{P}_i \rangle = \sigma^i(\langle \Sigma, P \rangle)$ is the solution of the GCPP on G and $\langle \Sigma, P \rangle$. In the case that P is not transitive we obtain the same result starting from P^+ instead of P. This gives us an upper bound to the index i which is at most $|\Sigma_i|$, this last is in the worst case $O(|N|)$.

Corollary 8.27 *The solution $\langle M, \preceq \rangle$ of the GCPP over a graph G and a partition-pair $\langle \Sigma, P \rangle$ can be computed computing at most $|M|$ times the operator σ.*

Proof. From Theorem 8.26 we obtain that

$$\langle M, \preceq \rangle = \sigma^{|M|}(\langle \Sigma, P \rangle).$$

□

The characterizations obtained in this section allow us to conclude that if we are able to define a procedure which given a partition-pair $\langle \Sigma, P \rangle$ computes $\sigma(\langle \Sigma, P \rangle)$, then we can use it to solve GCPP's and hence to compute similarity quotients. In particular we recall that given a similarity quotient problem over a labelled graph $G = \langle N, \gg, \Sigma \rangle$ in order to solve it it is sufficient solve the GCPP over G and $\langle \Sigma, I \rangle$ (notice that I is trivially transitive).

Moreover, the last corollary ensures that it will be necessary to iterate the procedure which computes σ at most $|M|$ times. This is a first improvement on time complexity w.r.t. the algorithm presented in [BG00]: in a certain sense [BG00] computes an operator which refines the partition-pair less than σ, and hence it is possible that the computation has to be iterated up to $|M|^2$ times.

In the next section we present the procedure which computes σ.

8.3 Partitioning Algorithm

In this section we propose an algorithm which solves the generalized coarsest partition problem we have presented in the previous section. The Partitioning Algorithm takes as input a graph $G = \langle N, \gg \rangle$ and a partition-pair $\langle \Sigma, P \rangle$, with P transitive, calls the two functions Refine and Update until a fix point is reached, and returns the partition-pair $\langle M, \preceq \rangle$ which is the solution of the GCPP over G and $\langle \Sigma, P \rangle$. In order to solve the GCPP over G and $\langle \Sigma, P \rangle$, with P not transitive, it is sufficient to first compute P^+ and the cost of this operation is $O(\Sigma^3)$, which, hence, does not affect the global cost of our algorithm. The function Refine takes as input a partition-pair $\langle \Sigma_i, P_i \rangle$ and it returns the partition Σ_{i+1} which is the coarsest that satisfies the condition in (1σ) of Definition 8.19. The function Update takes as input a partition-pair $\langle \Sigma_i, P_i \rangle$ and a the refinement Σ_{i+1} and it produces the acyclic and reflexive relation over Σ_{i+1} which is the largest that satisfies the conditions in (2σ) of Definition 8.19. In particular, at the end of each iteration of the while-loop in the Partitioning Algorithm we have that $\langle \Sigma_{i+1}, P_{i+1} \rangle = \sigma(\langle \Sigma_i, P_i \rangle)$. It is immediate to see that the Refine function works exactly as described in the proof of Theorem 8.21, and hence it produces the partition which is the first element of $\sigma(\Sigma_i, \mathsf{P}_i)$.

Corollary 8.28 *If $\sigma(\langle \Sigma_i, \mathsf{P}_i \rangle) = \langle \Pi, Q \rangle$, then $\Pi = \Sigma_{i+1}$.*

```
Partitioning Algorithm(⟨N, ≽⟩, ⟨Σ, P⟩)
change := ⊤;
i := 0;
while change do
        change := ⊥;
        Σ_{i+1} := Refine(Σ_i, P_i, change);
        P_{i+1} := Update(Σ_i, P_i, Σ_{i+1});
        i := i + 1;
```

Figure 8.5: The Partitioning Algorithm.

```
Refine(Σ_i, P_i, change)
Σ_{i+1} := Σ_i;
for each α ∈ Σ_{i+1} do Stable(α) := ∅;
for each γ ∈ Σ_i do Row(γ) := {γ' | (γ, γ') ∈ P_i)};
Let Sort be a topological sorting of ⟨Σ_i, P_i⟩;
while Sort ≠ ∅ do
        γ := dequeue(Sort);
        A := ∅;
        for each α ∈ Σ_{i+1}, α ≽_∃ γ, Stable(α) ∩ Row(γ) = ∅ do
                α_1 := α ∩ ≽^{-1}(γ);
                α_2 := α \ α_1;
                if α_2 ≠ ∅ then change := ⊤;
                Σ_{i+1} := Σ_{i+1} \ {α};
                A := A ∪ {α_1, α_2};
                Stable(α_1) := Stable(α) ∪ {γ};
                Stable(α_2) := Stable(α);
        Σ_{i+1} := Σ_{i+1} ∪ A;
        Sort := Sort \ {γ};
return Σ_{i+1}
```

Figure 8.6: The Refine Function.

$\mathsf{Update}(\Sigma_i, \mathsf{P}_i, \Sigma_{i+1})$
$\mathsf{Ind}_{i+1} := \{(\alpha_1, \beta_1) \mid \alpha_1, \beta_1 \in \Sigma_{i+1}, \alpha_1 \subseteq \alpha, \beta_1 \subseteq \beta(\alpha, \beta) \in \mathsf{P}_i\};$ $\Sigma_{i\exists\forall}(\Sigma_{i+1}) := \langle \Sigma_{i+1}, \gg_{\exists}^{\Sigma_i}(\Sigma_{i+1}), \gg_{\exists}^{\Sigma_i}(\Sigma_{i+1}) \rangle;$ $\mathsf{Ref}_{i+1} := \mathsf{New\_HHK}(\Sigma_{i\exists\forall}(\Sigma_{i+1}), \mathsf{Ind}_{i+1}, \bot);$ $\Sigma_{(i+1)\exists\forall} := \langle \Sigma_{i+1}, \gg_{\exists}, \gg_{\forall} \rangle;$ $\mathsf{P}_{i+1} := \mathsf{New\_HHK}(\Sigma_{(i+1)\exists\forall}, \mathsf{Ref}_{i+1}, \top);$ return P_{i+1}

Figure 8.7: The Update Function.

The deletion of *wrong* pairs is performed by **Update** through two calls to the function **New_HHK**, which is a version of [HHK95] adapted to our purposes here. The function **Update** removes pairs from $\mathsf{P}_i(\Sigma_{i+1}) = \mathsf{Ind}_{i+1}$ in order to obtain the relation P_{i+1} which satisfies condition (2σ) of Definition 8.19. In particular Ref_{i+1} (obtained after the first call to **New_HHK** only) satisfies the first of the two conditions but not necessarily the second one.

Notice that the space complexity of the calls to **New_HHK** remains limited since they are made on quotient stuctures. This function is based on the use of the two structures $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ (cf. Definition 8.9) and $\Sigma_{i+1\exists\forall}$ (cf. Definition 8.8), and on the following equivalent formulation of condition (2σ).

Theorem 8.29 *Let $G = \langle N, \gg \rangle$ be a graph, $\langle \Sigma, P \rangle$ be a partition-pair and Π be a partition finer than Σ. Q satisfies (2σ) of Definition 8.19 if and only if Q is the maximal relation over Π such that $Q \subseteq P(\Pi)$ and if $(\alpha, \beta) \in Q$, then:*

$$\begin{array}{l}\forall \gamma \in \Pi(\alpha \gg_{\forall}^{\Sigma}(\Pi)\gamma \rightarrow \exists \gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \wedge \beta \gg_{\exists}^{\Sigma}(\Pi)\gamma')) \\ \forall \gamma \in \Pi(\alpha \gg_{\forall} \gamma \rightarrow \exists \gamma' \in \Pi((\gamma, \gamma') \in Q \wedge \beta \gg_{\exists} \gamma')),\end{array}$$

where $\gg_{\forall}^{\Sigma}(\Pi)$ and $\gg_{\forall}^{\Sigma}(\Pi)$ are the edges of the $\exists\forall$-induced quotient structure, while $\gg_{\exists}$ and $\gg_{\forall}$ are the edges of the $\exists\forall$ quotient structure.

Proof. In order to have our thesis we only need to prove that

$$\forall \gamma \in \Pi(\alpha \gg_{\forall}^{\Sigma}(\Pi)\gamma \rightarrow \exists \gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \wedge \beta \gg_{\exists}^{\Sigma}(\Pi)\gamma')) \Leftrightarrow$$

$$\forall \gamma \in \Sigma(\alpha \gg_{\forall} \gamma \rightarrow \exists \gamma' \in \Sigma((\gamma, \gamma') \in P \wedge \beta \gg_{\exists} \gamma))$$

($\Rightarrow$) Assume by contradiction that

$$\forall \gamma \in \Pi(\alpha \gg_{\forall}^{\Sigma}(\Pi)\gamma \rightarrow \exists \gamma' \in \Pi((\gamma, \gamma') \in P(\Pi) \wedge \beta \gg_{\exists}^{\Sigma}(\Pi)\gamma'))\wedge$$

$$\exists \gamma \in \Sigma(\alpha \gg_{\forall} \gamma \wedge \forall \gamma' \in \Sigma((\gamma, \gamma') \in P \rightarrow \neg \beta \gg_{\exists} \gamma)) \tag{8.3}$$

Let $\gamma \in \Sigma$ be the class such that $(\alpha \gg_{\forall} \gamma \wedge \forall \gamma' \in \Sigma((\gamma, \gamma') \in P \rightarrow \neg \beta \gg_{\exists} \gamma))$. As $\alpha \gg_{\forall} \gamma$ we can find a class of Π, γ^1 such that $\alpha \gg_{\exists} \gamma^1 \wedge \gamma^1 \subseteq \gamma$. By construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ we have that $\alpha \gg_{\forall}^{\Sigma}(\Pi)\gamma^1$ and hence (by equation 8.3) that there exists a class in Π, say δ^1, such that $\beta \gg_{\exists}^{\Sigma}(\Pi)\delta^1 \wedge (\gamma^1, \delta^1) \in P(\Pi)$.
Let δ be the class of Σ such that $\delta^1 \subseteq \delta$. As $\beta \gg_{\exists}^{\Sigma}(\Pi)\delta^1 \wedge (\gamma^1, \delta^1) \in P(\Pi)$ we deduce, by definition of $P(\Pi)$ and by construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$, that $\beta \gg_{\exists} \delta \wedge (\gamma, \delta) \in P$ which contradicts our assumption.

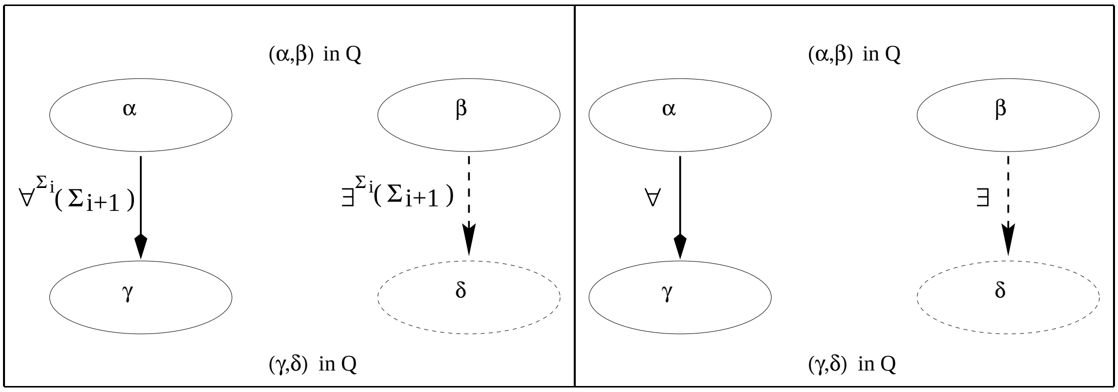


Figure 8.8: The conditions in (2σ) on the quotient structures.

($\Leftarrow$) Assume by contradiction that

$$\forall\gamma \in \Sigma(\alpha \geqslant_\forall \gamma \to \exists\gamma' \in \Sigma((\gamma,\gamma') \in P \wedge \beta \geqslant_\exists \gamma))\wedge$$

$$\exists\gamma \in \Pi(\alpha \geqslant^\Sigma_\forall (\Pi)\gamma \wedge \forall\gamma' \in \Pi((\gamma,\gamma') \in P(\Pi) \to \neg\beta \geqslant^\Sigma_\exists (\Pi)\gamma')) \tag{8.4}$$

Let γ^1 be the class of Π such that $\alpha \geqslant^\Sigma_\forall (\Pi)\gamma^1 \wedge \forall\delta^1 \in \Pi((\gamma^1,\delta^1) \in P(\Pi) \to \neg\beta \geqslant^\Sigma_\exists (\Pi)\delta^1)$. Consider $\gamma \in \Sigma$ such that $\gamma^1 \subseteq \gamma$. By construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$ (as $\alpha \geqslant^\Sigma_\forall (\Pi)\gamma^1$) we have that $\alpha \geqslant_\forall \gamma$ and hence, by equation (8.4), that $\exists\delta \in \Sigma$ such that $(\gamma,\delta) \in P \wedge \beta \geqslant_\exists \delta$. Let $\delta^1 \in \Pi$ be such that $\delta^1 \subseteq \delta$. As $\beta \geqslant_\exists \delta \wedge (\gamma,\delta) \in P$ we deduce, by definition of $P(\Pi)$ and by construction of $\Sigma_{i\exists\forall}(\Sigma_{i+1})$, that $\beta \geqslant^\Sigma_\exists (\Pi)\delta^1 \wedge (\gamma^1,\delta^1) \in P$ which contradicts our assumption. $\square$

In Figure 8.8 we present the conditions described in Theorem 8.29 on the $\exists\forall$-induced quotient structure and on the $\exists\forall$-quotient structure. As a consequence of Lemma 8.25 these conditions are weaker than the stability condition.

The computation performed by Update correspond to determine the largest relation included in P_i and satisfying conditions (2σ), thereby getting us closer to stability. Such a computation is proved correct as a fix-point computation of an operator τ defined as follows:

Definition 8.30 *Let $D = \langle S, R_1, R_2\rangle$ be such that $R_2 \subseteq R_1 \subseteq S \times S$, and $K \subseteq S \times S$ a relation over S. We define $\tau_D(K)$ as:*

$$\tau_D(K) = K \setminus \{(b,c) \mid \exists a(bR_2a \wedge \forall d(cR_1d \to (a,d) \notin K))\}.$$

We use $Fix(\tau_D)(K)$ to denote the greatest fix point of τ_D smaller than K.

Corollary 8.31 *Let $G = \langle N, \geqslant\rangle$ be a graph, and $\langle\Sigma, P\rangle$ be a partition-pair. If $\sigma(\langle\Sigma,P\rangle) = \langle\Pi, Q\rangle$, then:*

$$Q = Fix(\tau_{\Pi_{\exists\forall}})(\tau_{\Sigma_{\exists\forall}(\Pi)}(P(\Pi))).$$

Proof. It is an immediate consequence of the definition of τ and of Theorem 8.29. $\square$

Now establish a connection between the operator τ and the function $\mathsf{New\_HHK}$ presented in Figure 8.9 .

$\mathsf{New\_HHK}(\langle S, R_1, R_2\rangle, K, U)$

$P := K$;
for each $a \in S$ do
  $sim(a) := \{e \mid (a,e) \in K\}$;
  $rem(a) := S \setminus pre_1(sim(a))$;
while $\{a \mid rem(a) \neq \emptyset\} \neq \emptyset$ do
  let $a \in \{a \mid rem(a) \neq \emptyset\}$
  for each $b \in pre_2(a), c \in rem(a), c \in sim(b)$ do
    $sim(b) := sim(b) \setminus \{c\}$;
    $P := P \setminus \{(b,c)\}$;
    if U then for each $c_1 \in pre_1(c)$ do
      if $post_1(c_1) \cap sim(b) = \emptyset$ then $rem(b) := rem(b) \cup \{c_1\}$;
  $rem(a) := \emptyset$;
return P

Figure 8.9: The New_HHK Function.

Lemma 8.32 *It holds that*

$$\begin{array}{rcl} \mathsf{New\_HHK}(D, K, \perp) & = & \tau_D(K) \\ \mathsf{New\_HHK}(D, K, \top) & = & Fix(\tau_D)(K) \end{array}$$

Proof. Let $P_1 = \mathsf{New\_HHK}(D, K, \perp)$. Let $(b,c) \notin P_1$, we have to prove that $(b,c) \notin \tau_D(K)$. If $(b,c) \notin K$, then it is trivial. If $(b,c) \in K$, from $(b,c) \notin P_1$ we have that there exists a such that $b \in pre_2(a), c \in rem(a)$ and $c \in sim(b)$. From $b \in pre_2(a)$ we have that bR_2a. From $c \in rem(a)$, since we are in the case in which $U = \perp$, we have that $c \in S \setminus pre_1(\{e \mid (a,e) \in K\})$ (notice that $rem(a)$ is never modified and when it is build the definition of $sim(a)$ is $\{e \mid (a,e) \in K\}$). This means that $\forall e((a,e) \in K \to \neg cR_1e)$, i.e. $\forall d(cR_1d \to (a,d) \notin K)$. Hence we have obtained bR_2a and $\forall d(cR_1d \to (a,d) \notin K)$, which implies that (b,c) is not in $\tau_D(K)$. Now we assume that $(b,c) \notin \tau_D(K)$ and we prove that $(b,c) \notin P_1$. If $(b,c) \notin K$ is trivial. If $(b,c) \in K$, then from the definition of τ_D, we have that there exists a such that bR_2a and $\forall d(cR_1d \to (a,d) \notin K)$. This implies that $b \in pre_2(a)$ and $c \in rem(a)$. From the fact that $(b,c) \in K$ we also have in the initialization it holds $c \in sim(b)$. We can assume that a is the first such that $rem(a) \neq \emptyset$ and $c \in rem(a), b \in pre_2(a)$. This means that it is still true also $c \in sim(b)$. Hence at this step (b,c) is removed from P_1, i.e. the thesis.

Let $P_2 = \mathsf{New\_HHK}(D, K, \top)$. Let $(b,c) \notin P_2$ we have to prove that

$$(b,c) \notin Fix(\tau_D)(K).$$

The case in which $(b,c) \notin K$ is trivial. Let k be the number of extraction we have done from $LS = \{a \mid rem(a) \neq \emptyset\}$ at the iteration in which (b,c) is removed from P_2. We proceed by induction on k. If $k = 1$, then a is the first element we extract from LS, hence $rem(a) := S \setminus pre_1(\{e \mid (a,e) \in K\})$, i.e. $rem(a) = \{c \mid \forall d(cR_1d \to (a,d) \notin K\}$. Since we remove (b,c) from P_2 this means that bR_2a, $c \in rem(a)$ and $c \in sim(b)$, and using the fact that $rem(a) = \{c \mid \forall d(cR_1d \to (a,d) \notin K\}$, this implies that $(b,c) \notin \tau_D(K)$, hence $(b,c) \notin Fix(\tau_D)(K)$. Let us assume we have proved the thesis in the case $k \leq h$, and (b,c) is such that it is removed from P_2 at the $h+1$-th extraction from LS. This means that there exists a such that bR_2a and $c \in rem(a)$. If c was in $rem(a)$ in the initialization then as in the base case we can prove that $(b,c) \notin \tau_D(K)$. If c has been added to $rem(a)$

after the initialization, then there exists $h' \leq h$ such that at the h'-th extraction from LS we have the following situation: a pair (a,d) has been removed from P_2, $c \in pre_1(d)$ and $post_1(c) \cap sim(a) = \emptyset$. It is immediate to prove that the following invariant holds:

$$\forall a \in S(sim(a) = \{e \mid (a,e) \in P\}).$$

Hence we have that the situation at the h'-th can be rewritten as (a,d) has been removed from P_2, $c \in pre_1(d)$ and $\forall d'(cR_1d' \to (a,d') \notin P_2)$. By inductive hypothesis we have that $(a,d) \notin Fix(\tau_D)(K)$ and the same for each d' such that cR_1d', i.e. there exists i such that $(a,d) \notin \tau_D^i(K)$ and $\forall d'(cR_1d' \to (a,d') \notin \tau_D^i(K))$. We have obtained that bR_2a and $\forall d'(cR_1d' \to (a,d') \notin \tau_D^i(K))$, hence $(b,c) \notin \tau_D^{i+1}(K)$. We now prove that if $(b,c) \notin Fix(\tau_D(K))$, then $(b,c) \notin P_2$. If $(b,c) \notin K$, then is trivial. Let k be the number such that $(b,c) \in \tau_D^{k-1}(K)$ and $(b,c) \notin \tau_D^k(K)$. We proceed by induction on k and we prove that $(b,c) \notin \tau_D^k(K)$ implies $(b,c) \notin P_2$. If $k=1$, then this is equivalent to say that there exists a such that bR_2a and $\forall d(cR_1d \to (a,d) \notin K)$. This implies that initially $c \in rem(a)$. Since we are assuming that $(b,c) \in K$ we also have that initially $c \in sim(b)$. Hence when a is extract for the first time from LS (since S is finite we are sure that if $sim(a) \neq \emptyset$, then there exists an iteration in which a is extract from LS) if (b,c) has not yet been removed from P_2 we have that bR_2a, $c \in rem(a)$, and $c \in sim(b)$. Hence at this point (b,c) is removed from P_2. Let us assume we have proved the thesis in the case $k \leq h$ and we prove the thesis for $k+1$. Our hypothesis is that $(b,c) \in \tau_D^k(K)$ and $(b,c) \notin \tau_D^{k+1}(K)$. This means that there exists a such that bR_2a and $\forall d(cR_1d \to (a,d) \notin \tau_D^k(K))$. Applying the inductive hypothesis we obtain that $\forall d(cR_1d \to (a,d) \notin P_2)$. When we remove from P_2 the last pair (a,d') such that cR_1d' we have that $c \in prec_1(d')$ and $post_1(c) \cap sim(a) = \emptyset$. This implies that c is added to $rem(a)$. Let us consider the first iteration after the one in which we add c to $rem(a)$ in which a is extracted from LS. We have that bR_2a, $c \in rem(a)$, and, assuming that (b,c) has not yet been removed from P_2, $c \in sim(b)$, hence at this point (b,c) is removed from P_2. □

We are now ready to prove the invariants relative to the functions Update and Refine.

Theorem 8.33 (Refine-Update invariants) *It holds that*

$$\langle \Sigma_{i+1}, \mathsf{P}_{i+1} \rangle = \sigma(\langle \Sigma_i, \mathsf{P}_i \rangle).$$

Proof. This is an immediate consequence of Corollary 8.28, of Corollary 8.31, and of Lemma 8.32. □

As a consequence of Theorem 8.26 and of Corollary 8.27, since the Partitioning Algorithm terminates whenever $\Sigma_{i+1} = \Sigma_i$, we can conclude that the Partitioning Algorithm computes the solution $\langle M, \preceq \rangle$ of the GCPP over G and $\langle \Sigma, P \rangle$, with P transitive, performing at most $|M|$ iterations of the while-loop.

8.4 Complexity Results

Theorem 8.34 (Time Complexity) *Given a graph $G = \langle N, \gtrdot \rangle$ and a partition-pair $\langle \Sigma, P \rangle$, with P transitive, the algorithm* Partitioning Algorithm *computes the solution $\langle M, \preceq \rangle$ of the GCPP over them in time $O(|M|^2 | \gtrdot |)$.*

Proof. From Corollary 8.27 we have that at most $|M|$ iterations of the while-loop are performed.
We prove that both Refine and Update have cost $O(|\Sigma_i| | \gtrdot |)$.

The initialization phase (the instructions before the while-loop) in Refine takes time $O(|\Sigma_i|^2)$. During each iteration of the while-loop an element α is taken out of Sort and it is never reinserted in Sort. The for-loop is executed on α and β only if $\beta \gtrdot_\exists \alpha$. There are at most $|\gtrdot^{-1}(\alpha)|$ classes β's such that $\beta \gtrdot_\exists \alpha$. The check $\mathsf{Stable}(\beta) \cap \mathsf{Row}(\alpha) = \emptyset$ costs $O(|\Sigma_i|)$ and the remaining operations in the for-loop can be performed in constant time. Hence, the global cost of the while-loop is $O(\gtrdot^{-1}(\alpha_1)|\Sigma_i| + \ldots + \gtrdot^{-1}(\alpha_{|\Sigma_i|})|\Sigma_i|) = O(|\gtrdot||\Sigma_i|)$. So the cost of Refine is $O(|\Sigma_i||\gtrdot|)$.

The initialization operations in the Update function have cost $O(|\Sigma_i|^2 + |\gtrdot|) = O(|\gtrdot|)$. The fact that the cost of the function New_HHK is $O(|\Sigma_i||\gtrdot|)$ is a consequence of what has been proved in [HHK95]: the first call to New_HHK corresponds to a call to the procedure presented in [HHK95] stopping the procedure after one iteration; the second call to New_HHK corresponds to a call to the procedure presented in [HHK95] in which there are two types of edges, $\forall$-edges and $\exists$-edges, and this fact does not affect the complexity result. Hence, the time complexity of Update is $O(|\Sigma_i||\gtrdot|)$ □

Theorem 8.35 (Space Complexity) *Given a graph $G = \langle N, \gtrdot\rangle$ and a partition-pair $\langle \Sigma, P\rangle$, with P transitive, the algorithm* Partitioning Algorithm *computes the solution $\langle M, \preceq\rangle$ of the GCPP over them in space $O(|M|^2 + |N|\log(|M|))$.*

Proof. During each iteration of the algorithm it is necessary to consider:

- the relation $\gtrdot$;
- the relation P_i: at most $O(|\Sigma_i|^2)$ space;
- the relation which maps each node in N into the class of Σ_i to which it belongs: space $O(|N|\log|\Sigma_i|)$;
- a counters table necessary to the function New_HHK in order to work in time $O(|\Sigma_i||\gtrdot|)$ (see [HHK95]): this takes space $O(|\Sigma_i|^2)$). We observe that it is not really necessary to keep the relation $\gtrdot$ in memory: we use it only when it is necessary to provide the set of successors and predecessors of a given node. Hence the space complexity is $O(|M|^2 + |N|\log(|M|))$.

□

8.5 Concluding Remarks

We think that the circle of ideas presented here can be useful in the study of fast simulation algorithm developed to operate in situations in which strong space constraint are present. Moreover, the use of the $\forall\exists$-structure and the definition of coarsest partition problems on them, seem to be a methodology with some potential in all those situations in which a fix-point in the lattice of equivalence relations is to be computed.

We plain to work on a symbolic implementation of our algorithm which is naturally suggested by the fact that our algorithm always works on sets of nodes (the classes of the partitions).

An attempt to combine the algorithm we present here with the use of a suitable notion of rank (as it has been done in Chapter 7) as well as the possibility of defining positive strategies for the computation of simulation (as it has been done in Chapters 5 and 6 in the case of bisimulation) is under study.

9

Conclusions

In this dissertation we considered decidability and algorithmic problems in the context of non-well-founded set theories. Sets provide a general and flexible framework in which it is possible to study and solve problems coming from different areas of computer science. The authors of [CLR90] in the introduction to the third part of their book wrote:

> Sets are as fundamental to computer science as they are to mathematics. Whereas mathematical sets are unchanging, the sets manipulated by algorithms can grow, shrink, or otherwise change over time. We call such sets **dynamic**...Not surprisingly, the best way to implement a dynamic set depends upon the operations that must be supported.

We tried to provide general reasoning instruments useful in different contexts: the definition of tableau methods to solve decidability problems coming from the translation of Modal Logic in set theory shown in the first part; the use of the notion of rank in the computation of bisimulation and simulation in the second part. Hence, the results presented in this thesis should be particularly useful in the development of theorem provers for different theories and of model checkers for systems verification. As far as the first part of the thesis is concerned, the methods we introduce (quotients, saturated models, tableaux, and snapshots) contribute to uniformly present and generalize decidability results of Modal Logic which, using different approaches, usually require *ad-hoc* techniques. For this reason our next step will be to investigate the existence of a termination condition for the tableau procedure presented in Chapter 4: such a termination condition would allow to characterize and uniformly decide a large class of modal logics.

As far as the second part of the thesis is concerned we plan to work in two directions. We intend to improve our results by implementing and testing all the algorithms we presented. In particular, we are interested in testing our algorithms in applications coming from the area of security of protocols (see [FPR02]). At the moment we are interfacing the C++ implementation of our bisimulation algorithm (see Chapter 5) with the model checker XEVE ([Bou98]). The second direction concerns the application of our techniques to the infinite case (see [PLY99]). At the moment all the existing techniques for the infinite case are based on negative strategies, while it seems reasonable that a generalization of the Ackermann encoding we introduced in Chapter 6 could lead to a positive strategy for the infinite case.

Bibliography

[Abr96] J. R. Abrial. *The B-Book: Assigning Programs to Meaning.* Cambridge University Press, 1996.

[Ack37] W. Ackermann. Die widerspruchsfreiheit der allgemeinen mengenlhere. *Mathematische Annalen*, 114, 1937.

[Acz88] P. Aczel. *Non-well-founded sets*, volume 14 of *CSLI Lecture Notes.* Stanford University Press, 1988.

[AD97] P. Arenas-Sánchez and A. Dovier. A minimality study for set unification. *Journal of Functional and Logic Programming*, 1997(7):1–49, 1997.

[ADR99] D. Aliffi, A. Dovier, and G. Rossi. From set to hyperset unification. *Journal of Functional and Logic Programming*, 10:1–48, 1999.

[AG91] S. Abiteboul and S. Grumbach. A rule-based language with functions and sets. *ACM Transactions on Database Systems*, 16(1):1–30, 1991.

[Ake78] S. B. Akers. Binary decision diagrams. *IEEE Transaction on Computers*, 27(6):509–516, 1978.

[ASSB94] A. Aziz, V. Sighal, G. Swamy, and R. Brayton. Minimizing interacting finite state machines: a compositional approach to language containment. In *Proc. of Int. Conference on Computer Design (ICCD'94)*, pages 255–263. IEEE Computer Society Press, 1994.

[Bar75] J. Barwise. *Admissible Sets and Structures.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1975.

[Bar86] J. Barwise. Situations, sets and the axiom of foundation. In Wilkie and Wilmers, editors, *Proc. of Logic Colloquium '84*, Studies in Logic, pages 21–36. North Holland, Amsterdam, 1986.

[BB88] F. Baader and W. Büttner. Unification in commutative and idempotent monoids. *Theoretical Computer Science*, 56:345–352, 1988.

[BDMP97] J. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti. Modal deduction in second-order logic and set theory - I. *Journal of Logic and Computation*, 7(2):251–265, 1997.

[BDMP98] J. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti. Modal deduction in second-order logic and set theory - II. *Studia Logica*, 60(3):387–420, 1998.

[BdS92] A. Bouali and R. de Simone. Symbolic bisimulation minimization. In G. von Bochmann and D. K. Probst, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'92)*, volume 663 of *Lecture Notes in Computer Science*, pages 96–108. Springer-Verlag, Berlin, 1992.

[Ben76] J. van Benthem. *Modal Correspondence Theory.* PhD thesis, Universiteit van Amsterdam, Instituut voor Logica en Grondslagenonderzoek van Exacte Wetenschappen, 1976.

[Ben83] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, 1983.

[BFH90] A. Bouajjani, J.C. Fernandez, and N. Halbwachs. Minimal model generation. In E. Clarke and R. Kurshan, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'90)*, volume 531 of *Lecture Notes in Computer Science*, pages 197–203. Springer-Verlag, Berlin, 1990.

[BG00] D. Bustan and O. Grumberg. Simulation based minimization. In D. A. McAllester, editor, *Proc. of Int. Conference on Automated Deduction (CADE'00)*, volume 1831 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, Berlin, 2000.

[BGG97] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1997.

[Blo89] B. Bloom. *Ready Simulation, Bisimulation, and the Semantics of CCS-like Languages.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1989.

[BM91] J. Barwise and L. Moss. Hypersets. *The Math. Intelligencer*, 13(4):31–41, 1991.

[BM96] J. Barwise and L. Moss. *Vicious Circles. On the Mathematics of non-well-founded phenomena*, volume 60 of *CSLI Lecture Notes.* Stanford University Press, 1996.

[BNST91] C. Beeri, S. Naqvi, O. Shmueli, and S. Tsur. Set constructors in a logic database language. *Journal of Logic Programming*, 10(3):181–232, 1991.

[Bou98] A. Bouali. XEVE, an ESTEREL verification environment. In A. J. Hu and M. Y. Vardi, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *Lecture Notes in Computer Science*, pages 500–504. Springer-Verlag, Berlin, 1998.

[BP95] B. Bloom and R. Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *Science of Computer Programming*, 24(3):189–220, 1995.

[BRV01] P. Blackburn, M. De Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2001.

[Bry85] R. E. Bryant. Symbolic manipulation of boolean functions using a graphical representation. In *Proc. of 22nd Design Automation Conference*, 1985.

[Bry86] R. E. Bryant. Graph based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, C-35(8):677–691, 1986.

[Büt86] W. Büttner. Unification in the data structure sets. In J.K. Siekmann, editor, *Proc. of Int. Conference on Automated Deduction (CADE'86)*, volume 230 of *Lecture Notes in Computer Science*, pages 470–488. Springer-Verlag, Berlin, 1986.

[Can87] D. Cantone. *A decision procedure for a class of unquantified formulae of set theory involving the powerset and singleton operators.* PhD thesis, New York University, 1987.

[Can97] D. Cantone. A fast saturation strategy for set-theoretic tableaux. In D. Galmiche, editor, *Proc. of TABLEAUX'97*, volume 1227 of *Lecture Notes in Artificial Intelligence*, pages 122–137. Springer-Verlag, Berlin, 1997.

[CC95] D. Cantone and R. Ruggeri Cannata. Deciding set-theoretic formulae with the predicate finite by a tableau calculus. *Matematiche*, 50(2):99–118, 1995.

[CDQT98] A. Cortesi, A. Dovier, E. Quintarelli, and L. Tanca. Operational and abstract semantics of a query language for semi-structured information. In P. Fraternali, U. Geske, C. Ruiz, and D. Seipel, editors, *Proc. of 6th Int'l Workshop on Deductive Databases and Logic Programming (DDLP'98)*, pages 127–139. GMD Report 22, 1998.

[CE82] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using brancing time temporal logic. In *Proc. of Workshop on Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, Berlin, 1982.

[CF95] D. Cantone and A. Ferro. Techniques of computable set theory with applications to proof verification. *Comm. Pure App. Math.*, 48(9–10):1–45, 1995. Special Issue in honor of J.T. Schwartz.

[CFO89] D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory Vol. 1.* Int. Series of Monographs on Computer Science. Oxford University Press, 1989.

[CFS85] D. Cantone, A. Ferro, and J. T. Schwartz. Decision procedures for elementary sublanguages of set theory VI. Multilevel syllogistic extended by the powerset operator. *Comm. Pure App. Math.*, 38(1):549–571, 1985.

[CFS87] D. Cantone, A. Ferro, and J. T. Schwartz. Decision procedures for elementary sublanguages of set theory V. Multilevel syllogistic extended by the general union operator. *Journal of Computer and Systems Sciences*, 34(1):1–18, 1987.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking.* The MIT Press, Cambridge, Mass., 1999.

[Chu36] A. Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1:101–102, 1936.

[CL96] Y. Caseau and F. Laburthe. Introduction to CLAIRE programming language. Technical Report 4, Institute LIENS, 1996. Available at `http://www.ens.fr/ laburthe/claire.html`.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Mass., 1990.

[Col90] A. Colmerauer. An introduction to Prolog III. *Communications of the ACM*, 33(7):69–90, 1990.

[COP01] D. Cantone, E. G. Omodeo, and A. Policriti. *Set Theory for Computing. From Decision Procedures to Declarative Programming with Sets.* Monographs in Computer Science. Springer-Verlag, Berlin, 2001.

[COU01] D. Cantone, E. G. Omodeo, and P. Ursino. Formative processes with applications to the decision problem in set theory: I. Powerset and singleton operators. *Information and Computation*, 2001. To appear.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, 1993.

[CS92] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In K. G. Larsen and A. Skou, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'91)*, volume 575 of *Lecture Notes in Computer Science*, pages 48–58. Springer-Verlag, Berlin, 1992.

[CU97] D. Cantone and P. Ursino. A unifying approach to computable set theory. In *Proc. of Logic Colloquium*, 1997.

[CvHC95] A. Cortesi, P. van Henteryck, and B. Le Charlier. Type analysis of prolog using type graphs. *Journal of Logic Programming*, 22(3):179–208, 1995.

[CZ99] D. Cantone and C. Zarba. A tableau-based decision procedure for a fragment of set theory involving a restricted form of quantification. In N. Murray, editor, *Proc. of TABLEAUX'99*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 97–112. Springer-Verlag, Berlin, 1999.

[CZ00a] D. Cantone and C. Zarba. *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Computer Science*, chapter A new fast tableau-based procedure for an unquantified fragment of set theory, pages 126–136. Springer-Verlag, Berlin, 2000.

[CZ00b] D. Cantone and C. Zarba. A tableau calculus for integrating first-order reasoning with elementary set theory reasoning. In R. Dyckhoff, editor, *Proc. of TABLEAUX'00*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 143–159. Springer-Verlag, Berlin, 2000.

[D'A98] G. D'Agostino. *Modal Logic and non-well-founded Set Theory: translation, bisimulation, interpolation.* PhD thesis, Institute for Logic, Language and Computation - Universiteit van Amsterdam, 1998.

[Dam96] D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking.* PhD thesis, Eindhoven University of Technology, 1996.

[DGG93] D. Dams, R. Gerth, and O. Grumberg. Generation of reduced models for checking fragments of CTL. In C. Courcoubetis, editor, *Proc. of Int. Conference on Computer Aided Verification (CAV'93)*, volume 697 of *Lecture Notes in Computer Science*, pages 479–490. Springer-Verlag, Berlin, 1993.

[DGG97] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, 1997.

[DGHJP99] M. D'Agostino, D. Gabbay, R. Hähnle, and editors J. Posegga. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[DKR82] D. Dolev, M. M. Klawe, and M. Rodeh. An $o(n \log n)$ unidirectional distributed algorithm for extrema finding in a circle. *Journal of Algorithms*, 3(3):245–260, 1982.

[DMO92] R. Dionne, E. Mays, and F. J. Oles. A non-well-founded approach to terminological cycles. In P. Rosenbloom and P. Szolovits, editors, *Proc. of 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 761–766. The MIT Press, Cambridge, Mass., 1992.

[DMO93] R. Dionne, E. Mays, and F. J. Oles. The equivalence of model-theoretic and structural subsumption in description logics. In *Proc. of Int. Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 710–716. Morgan Kaufmann, 1993.

[DMP95] G. D'Agostino, A. Montanari, and A. Policriti. A set-theoretic translation method for polymodal logics. *Journal of Automated Reasoning*, 15(3):317–337, 1995.

[DMP96] G. D'Agostino, A. Montanari, and A. Policriti. Set theoretic decidability results for modal theorem proving. In *Proc. of Italian Conference on Theoretical Computer Science (ICTCS'95)*, pages 326–342. World Scientific Publishing Co, 1996.

[DOP99] A. Dovier, E. G. Omodeo, and A. Policriti. Solvable set/hyperset contexts: II. A goal-driven unification algorithm for the blended case. *Applicable Algebra in Engineering, Communication and Computing*, 9(4):293–332, 1999.

[DOPR96] A. Dovier, E. G. Omodeo, E. Pontelli, and G. Rossi. A language for programming in logic with finite sets. *Journal of Logic Programming*, 28(1):1–44, 1996.

[Dov96] A. Dovier. *Computable Set Theory and Logic Programming*. PhD thesis, Università di Pisa, Dipartimento di Informatica, 1996.

[DPP01] A. Dovier, C. Piazza, and A. Policriti. A fast bisimulation algorithm. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of Int. Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 79–90. Springer-Verlag, Berlin, 2001.

[DPPR99] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. ACI1 constraints. In G. Gupta and D. De Schreye, editors, *Proc. of Int. Conference on Logic Programming (ICLP'99)*, pages 573–587. The MIT Press, Cambridge, Mass., 1999.

[DPPR00] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. Sets and constraint logic programming. *ACM Transactions on Programming Languages and Systems*, 22(5):861–931, 2000.

[DPR98] A. Dovier, E. Pontelli, and G. Rossi. Set unification revisited. NMSU-CSTR-9817, Dept. of Computer Science, New Mexico State University, USA, 1998.

[EC82] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[EH85] E. A. Emerson and J. Y. Halpem. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Science*, 30(1):1–24, 1985.

[EITM65] Y. Ershov, I.Lavrov, A. Taimanov, and M.Taitslin. Elementary theories. *Russian Math. Surveys*, 20:35–105, 1965. English Translation.

[FH83] M. Forti and F. Honsell. Set theory with free construction principles. *Annali Scuola Normale Superiore di Pisa, Cl. Sc.*, IV(10):493–522, 1983.

[FHL94] M. Forti, F. Honsell, and M. Lenisa. Processes and hyperuniverses. In I. Privara, editor, *Proc. of Mathematical Foundations of Computer Science (MFCS'94)*, volume 841 of *Lecture Notes in Computer Science*, pages 352–372, 1994.

[Fit93] M. C. Fitting. Basic modal logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 1: Logical Foundations*, pages 365–448. Clarendon Press, 1993.

[Fit96] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, Berlin, 2nd edition, 1996.

[For95] T. E. Forster. *Set Theory With a Universal Set*, volume 31 of *Oxford Logic Guides*. Oxford University Press, ii edition, 1995.

[FOS80] A. Ferro, E. G. Omodeo, and J. T. Schwartz. Decision procedures for elementary sublanguages of set theory I. Multi-level syllogistic and some extensions. *Comm. Pure App. Math.*, 33:599–608, 1980.

[FPR02] R. Focardi, C. Piazza, and S. Rossi. Proofs methods for bisimulation based information flow security. In A. Cortesi, editor, *Proc. of Frontiers of Verification, Model Checking and Abstract Interpretation (VMCAI'02)*, volume ?? of *Lecture Notes in Computer Science*, page ?? Springer-Verlag, Berlin, 2002.

[FR79] J. Ferrante and C. Rackoff. *The Computation Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1979.

[Fri00] Sy D. Friedman. *Fine Structure and Class Forcing*, volume 3 of *de Gruyter Series in Logic and Its Applications*. de Gruyter, 2000.

[FV99] K. Fisler and M. Y. Vardi. Bisimulation and model checking. In L. Pierre and T. Kropf, editors, *Proc. of Correct Hardware Design and Verification Methods (CHARME'99)*, volume 1703 of *Lecture Notes in Computer Science*, pages 338–341. Springer-Verlag, Berlin, 1999.

[Ger97] C. Gervet. Interval propagation to reason about sets: Definition and implementation of a practical language. *Constraints*, 1:191–246, 1997.

[GL94] O. Grumberg and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.

[Gol87] R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes*. Chicago University Press, 1987.

[HC84] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen, 1984.

[Her71] J. Herbrand. *Herbrand, Logical Writings*. Harvard University Press, 1971.

[HHK95] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of 36th IEEE Symposium on Foundations of Computer Science, (FOCS'1995)*, pages 453–462. IEEE Computer Society Press, 1995.

[Hil71] D. Hilbert. Probleme der grundlegung der mathematik. In K. Reidemeister, editor, *Hilbert Gedenkband*, pages 9–19. Springer-Verlag, Berlin, 1971.

[Hil77] D. Hilbert. *From Frege to Gödel*, chapter On the infinite, pages 367–392. Source books in the history of the sciences. Harvard University Press, 3th edition, 1977.

[HL89] P. Hill and J. Loyd. *The Gödel Programming Language*. The MIT Press, Cambridge, Mass., 1989.

[Hol91] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[Hol97] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[Hop71] J. E. Hopcroft. An $nlogn$ algorithm for minimizing states in a finite automaton. In Kohavi and Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[HP99] G. J. Holzmann and A. Puri. A minimized automaton representation of reachable states. *Software Tools for Technology Transfer*, 2(3):270–278, 1999.

[Jec01] J. Jech. *Set Theory*. Springer-Verlag, Berlin, 2001. Perspectives in Mathematical Logic.

[Jen69] R. B. Jensen. On the consistency of a slight modification of Quine's new foundations. *Synthese*, 19:250–263, 1969.

[JP89] B. Jayaraman and D. A. Plaisted. Programming with equations, subsets and relations. In E. Lusk and R. Overbeek, editors, *Proc. of Workshop on Types in Logic Programming (NACLP'89)*, pages 1051–1068. The MIT Press, Cambridge, Mass., 1989.

[KMP77] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[KS90] P. C. Kannellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[Kun80] K. Kunen. *Set Theory. An Introduction to Independence Proofs.* Studies in Logic. North Holland, Amsterdam, 1980.

[Lee59] C. Y. Lee. Binary decision programs. *Bell System Technical Journal*, 38(4):985–999, 1959.

[Lev79] A. Levy. *Basic Set Theory.* Springer-Verlag, Berlin, 1979. Perspectives in Mathematical Logic.

[LL91] B. Legeard and E. Legros. CLPS: A set constraints logic programming language. Technical report, Laboratoire d'Automatique de Besançon, Institut de Productique, Besançon, France, February 1991.

[Llo99] J. W. Lloyd. Programming in an integrated functional and logic language. *Journal of Functional and Logic Programming*, 3:1–49, 1999.

[LS76] M. Livesey and J. Siekmann. Unification of sets and multisets. Technical report, Institut für Informatik I, Universität Karlsruhe, 1976.

[LSY97] A. Lisitsa, P. Sazonov, and V. Yu. Bounded hyperset theory and web-like data bases. In Gottlob, Leitsch, and Mundici, editors, *Proc. of Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium*, volume 1289 of *Lecture Notes in Computer Science*, pages 172–185. Springer-Verlag, Berlin, 1997.

[LY92] D. Lee and M. Yannakakis. Online minimization of transition systems. In *Proc. of 24th ACM Symposium on Theory of Computing (STOC'92)*, pages 264–274. ACM Press, 1992.

[McM93] K. L. McMillan. *Symbolic model checking: an approach to the state explosion problem.* Kluwer Academic Publishers, 1993.

[Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proc. of the 2th Int. Joint Conference on Artificial Intelligence (IJCAI'71)*, pages 481–489. Morgan Kaufmann, 1971.

[Mil80] R. Milner. A calculus of communicating systems. In G. Goos and J. Hartmanis, editors, *Lecture Notes in Computer Science*, volume 92. Springer-Verlag, Berlin, 1980.

[Mir17] D. Mirimanoff. Les antinomies de Russell et de Burali-Forti, et le problème fondemental de la théorie des ensembles. *Enseignement Mathématique*, 19:37–52, 1917.

[MP81] Z. Manna and A. Pnueli. *The Correctness Problem in Computer Science*, chapter Verification of Concurrent Programs: The Temporal Framework, pages 215–273. Academic Press, London, 1981.

[MS99] K. Marriott and P. J. Stuckey. *Programming With Constraints: An Introduction.* The MIT Press, Cambridge, Mass., 1999.

[Ohl91] H. J. Ohlbach. Semantics-based translation methods for modal logics. *Journal of Logic and Computation*, 1(6):691–746, 1991.

[Ohl93] H. J. Ohlbach. Optimized translation of multi modal logic into predicate logic. In A. Voronkov, editor, *Proc. of 4th Int. Conference on Logic Programming and Automated Reasoning (LPAR'93)*, volume 698 of *Lecture Notes in Computer Science*, pages 253–264. Springer-Verlag, Berlin, 1993.

[OP95] E. G. Omodeo and A. Policriti. Solvable set/hyperset contexts: I. Some decision procedures for the pure, finite case. *Comm. Pure App. Math.*, 48(9–10):1123–1155, 1995. Special Issue in honor of J.T. Schwartz.

[OSH96] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logics. In H. Wansing, editor, *Proof Theory and Modal Logic - Applied Logic Series 2*, volume 2, chapter 14, pages 253–291. Kluwer Academic Publishers, 1996.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. of 5th Int. Conference on Theorical Computer Science*, volume 104 of *lncs*, pages 167–183. Springer-Verlag, Berlin, 1981.

[PLY99] G. Pappas, G. Lafferriere, and S. Yovine. A new class of decidable hybrid systems. In *Proc. of Hybrid Systems Computation and Control (HSCC'99)*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer-Verlag, Berlin, 1999.

[PP91] F. Parlamento and A. Policriti. Expressing infinity without foundation. *Journal of Symbolic Logic*, 56(4):1230–1235, 1991.

[PP00] C. Piazza and A. Policriti. Towards tableau-based decision procedures for non-well-founded fragments of set theory. In R. Dyckhoff, editor, *Proc. of TABLEAUX'00*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 368–382. Springer-Verlag, Berlin, 2000.

[PP01a] C. Piazza and A. Policriti. Ackermann encoding, bisimulations, and OBDD's. In M. Leuschel, A. Podelski, C.R. Ramakrishnan, and U. Ultes-Nitsche, editors, *Proc. of 2nd Int. Workshop on Verification and Computational Logic (VCL'01)*, volume DSSE-TR-2001-3 of *Southampton University Technical Report*, pages 43–53. ACM Digital Library, 2001.

[PP01b] C. Piazza and A. Policriti. Deciding modal logics using tableaux and set theory. In L. M. Pereira and P. Quaresma, editors, *Proc. of Joint Conference on Declarative Programming (APPIA-GULP-PRODE'01)*, Evora University Technical Report, pages 261–276, 2001.

[PPR93] F. Parlamento, A. Policriti, and K. P. S. B. Rao. Witnessing differences without redundancies. Technical Report 23/93, Università di Udine, 1993.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[PTB85] R. Paige, R. E. Tarjan, and R. Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40:67–84, 1985.

[QS82] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. of 5th Int. Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–371. Springer-Verlag, Berlin, 1982.

[Qui67] W. v. O. Quine. *Set theory and its logic.* Belknap Press, 1967.

[Rab77] M. O. Rabin. Elements of recursion theory. In J. Barwise, editor, *Handbook of Mathematical Logic.* North Holland, Amsterdam, 1977.

[Rah98] F. Rahim. Property-dependent modular model checking application to vhdl with computational results. In *Proc. of Int. Workshop HLDVT*, 1998.

[Ser99] Y. Serdyuk. Partial evaluation in a set-theoretic query language for WWW. In B. Jayaraman and G. Rossi, editors, *Proc of Workshop on Declarative Programming with Sets (DPS'99)*, volume 200, pages 83–90. Università di Parma, 1999.

[SES94] L. Sterling and editors E. Shapiro. *The Art of Prolog: Advanced Programming Techniques.* Series in Logic Programming. The MIT Press, Cambridge, Mass., 1994.

[Sho67] J. R. Shoenfield. *Mathematical Logic.* Addison-Wesley, 1967.

[Smo95] G. Smolka. The OZ programming model. In J. van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 324–343. Springer-Verlag, Berlin, 1995.

[Spi92] J. M. Spivey. *The Z Notation: a reference Manual, 2^{nd} edition.* International Series in Computer Science. Prentice Hall, 1992.

[Sto99] F. Stolzenburg. An algorithm for general set unification and its complexity. *Journal of Automated Reasoning*, 22(1):45–63, 1999.

[Sza69] M. E. Szabo. *The Collected Papers of Gerhard Gentzen.* North Holland, Amsterdam, 1969.

[Tar72] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Tur37] A. Turing. On computable numbers, with an application to the entscheidungsproblem. In *Proc. of the London Math. Society*, volume 42, pages 230–265, 1937.

[Wan93] H. Wang. *Popular Lectures on Mathematical Logic*. Dover books in Science and Mathematics. Dover Publications, Inc., 1993.