Institute of Informatics, Warsaw University

A Complete Deductive System for the μ-Calculus

Igor Walukiewicz*

Doctoral Thesis
Written under the supervision of Professor Jerzy Tiuryn
Warsaw June, 1993

(*) This work was partially supported by Polish KBN grant No. 2 1192 91 01

To Parents

Contents

Chapter 1

Overview

It is now common to view computer programs as state transformers, that is actions that can change one state of computer hardware to another. In contrast with classical logics the notion of *change* is intrinsic in *modal* logics. Within modal logic, one can speak about multiple possible *worlds* and relations between them, as, for example, the changes of an environment during time. This property makes the modal logic a valuable tool for description of program behavior that is itself observable through the changes of computer states.

Embedding programs into the syntax of a logic gives us the ability to describe or verify programs inside the logic. The gain is that we do not have to deal with different theories each time we consider a new program. There is of course the price to pay for this, usual problems for predicate modal logics of programs are highly undecidable. For example validity problem for the first order algorithmic or dynamic logics [46, 41] is Π^1_1-complete. On the other hand propositional versions of modal logics do not have this disadvantages even though they were found expressive enough to analyze and check real-life applications.

There are basically two approaches to program verification using propositional modal logics. One, "proof theoretic", advocated by Manna and Pnueli in [28] is to describe a program by a formula of temporal or modal logic ψ. Then showing that the program has a property α is reduced to showing that α follows from ψ, that is to proving that formula ψ implies α.

The other approach, which we call following [4] "model theoretic", gave rise to the discipline called model checking. Here first step in verifying a program is to design a structure reflecting its behavior. The abstraction we have to make is to describe the states of the program using propositional variables. Then we describe the possible connections between different states reflecting possible changes of a state of the system. When we complete this task we obtain what is known in mathematical logic as a Kripke structure, which describes the system in question on the propositional level. Then verifying that program satisfies property α reduces to checking whether α is satisfied in the initial state of this structure.

Both approaches require us to describe the system on the propositional level which is the price to pay for using propositional logic. Fortunately, it occurs that frequently such abstractions of the system are sufficient. This is because while analyzing complex concurrent systems the biggest concern is in mutual interaction between modules which can be considered as finite automata.

Of course modal logics are not the only choice for implementing any of above approaches, one can use classical first or second-order logic to talk about such structures. The problem is that first-order logic is seldom sufficient because it is not expressive enough and second-order logic is usually much too strong with the consequence of being difficult for model-checking or axiomatization.

First-order logic on the infinite sequence is as expressive as very weak temporal logic $\mathcal{T}(O, U)$. This means that a property such as "something holds in all even moments of time" is not expressive in the first-order theory of the infinite sequence [21, 15].

Second-order logic on the other hand stumbles on decidability and completeness. There are two bold exceptions to this rule which are monadic second-order theory of one successor $S1S$ and monadic second order theory of two successors $S2S$. Both logics are very expressive and yet decidable. The first sets target, as far as expressiveness is concerned, for linear temporal logic and the second for modal logics. We say that there are two logics but of course there are also second order theories of $3, 4, \ldots$ successors. All of them and even second order theory of ω successors are easily reducible to $S2S$. On the other hand the difference between $S1S$ and $S2S$ is quite substantial, hence we will distinguish only these two logics forgetting about other theories of successors.

The problem with $S1S$ and $S2S$ is that, although decidable, the complexity of decision procedures is very high. Meyer [31] showed that decidability problem for $S1S$ is not elementary. Siefkes in 1970 presented a finitary axiomatization of $S1S$ [48]. No finitary axiomatization of $S2S$ is known.

The properties of logic we should consider, when we have program verification in mind, are expressiveness, completeness and decidability. The more expressive is the logic, the more properties of the systems we can describe. A complete axiom system allows us to reason about the properties. Decidability and particularly computational complexity of the decision procedure is important for machine aided verification. Finally, there is a question of a model checking, i.e., establishing the truth of a formula in a given state of a (usually very large) structure, which describes a behavior of a complex program.

In this context the μ-calculus is very interesting logic. It is as expressive as $S2S$ but much easier to decide and model check. As we will show here, there is also finitary complete axiomatization of it.

In this chapter we present the μ-calculus in the perspective of other propositional modal logics and monadic second order logics. The issues we are interested in are expressiveness, decidability and existence of complete axiomatizations. Next we introduce some notions from automata theory and Borel games which we will use in this work. Then we describe our proof method in the perspective of other proof methods used for proving completeness for different modal logics. We conclude this chapter with the description of the proposed system and the brief synopsis of the rest of the thesis.

1.1 The μ-calculus

The propositional μ-calculus results from adding fixpoint operator μ to the ordinary modal logic with many modalities. This construct was added to numerous logics see e.g. [47, 20, 40, 43], the logic as we will consider here was presented by Kozen in [24].

For some structure $\mathcal{M}$ the meaning of a formula $\alpha(X)$ of ordinary modal logic is some set of states of $\mathcal{M}$ which depends on the meaning of X. In other words we can consider $\alpha(X)$ as a function from a set of states to a set of states. If X occurs only positively in $\alpha(X)$, i.e., each occurrence of X is under an even number of negations, then the function designated by $\alpha(X)$ is monotone. This means that there is a least fixpoint of this function in the complete lattice of the subsets of the states of the structure. The meaning of the formula $\mu X.\alpha(X)$ will be exactly this least fixpoint of monotone operator $\alpha(X)$.

The syntax of the μ-calculus is based on the three sets:

- *Prop* $= \{p, q, ..\}$, a set of propositional letters,

- $Var = \{X, Y, ..\}$, a set of propositional variables, and
- $Act = \{a, b, ..\}$, a set of actions.

Formulas of the μ-calculus over this three sets can be defined by the following grammar:

$$\begin{aligned} \alpha \quad := \quad & X \mid p \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \\ & \langle a\rangle\alpha \mid [a]\alpha \mid \\ & \mu X.\alpha(X) \mid \nu X.\alpha(X) \end{aligned}$$

In the last two constructs we require that the variable X occurs only under even number of negations in the formula $\alpha(X)$. We will sometimes use $\sigma X.\alpha(X)$ to denote $\mu X.\alpha(X)$ or $\nu X.\alpha(X)$, the symbol *ff* will be an abbreviation for the formula $p \wedge \neg p$ for some propositional letter p.

Formulas are interpreted in Kripke models of the form $\mathcal{M} =< S, R, \rho >$, where:

- S is a nonempty set of states,
- R is a function assigning to each action in Act a binary relation on S,
- ρ is a function assigning to each propositional letter in $Prop$ a set of states.

Intuitively R describes the possible changes of states under different actions and ρ assigns to each propositional letter the set of states where it is true.

The meaning of a formula α in a model $\mathcal{M}$ under an assignment $Val : Var \to \mathcal{P}(S)$ is the set of states where α is true. It will be denoted by $\parallel \alpha \parallel_{Val}^{\mathcal{M}}$ and can be defined inductively by the following clauses (we will omit superscript $\mathcal{M}$ when it causes no ambiguity):

$$\begin{aligned} \parallel X \parallel_{Val} &= Val(X) \\ \parallel p \parallel_{Val} &= \rho(p) \\ \parallel \neg\alpha \parallel_{Val} &= S \setminus \parallel \alpha \parallel_{Val} \\ \parallel \alpha \wedge \beta \parallel_{Val} &= \parallel \alpha \parallel_{Val} \cap \parallel \beta \parallel_{Val} \\ \parallel \langle a\rangle\alpha \parallel_{Val} &= \{s : \exists s'.(s, s') \in R(a) \wedge s' \in \parallel \alpha \parallel_{Val}\} \\ \parallel [a]\alpha \parallel_{Val} &= \{s : \forall s'.(s, s') \in R(a) \Rightarrow s' \in \parallel \alpha \parallel_{Val}\} \\ \parallel \mu X.\alpha(X) \parallel_{Val} &= \bigcap\{S' \subseteq S : \parallel \alpha \parallel_{Val[S'/X]} \subseteq S'\} \\ \parallel \nu X.\alpha(X) \parallel_{Val} &= \bigcup\{S' \subseteq S : S' \subseteq \parallel \alpha \parallel_{Val[S'/X]}\} \end{aligned}$$

The requirement that X occurs only under even number of negations in $\sigma X.\alpha(X)$ guarantees that $\alpha(X)$ is interpreted as a monotone operator. In consequence the meaning of $\mu X.\alpha(X)$ is the least fixpoint of this operator and $\nu X.\alpha(X)$ denotes its greatest fixpoint. We say that a variable X is bounded by μ in a formula $\mu X.\alpha(X)$. By $FL(\psi)$ we denote the set of all free variables of ψ.

Summarizing modal μ-calculus is just a modal logic K with addition of the fixpoint operators. It is easy to observe that least and greatest fixpoints are duals of each other. This means that apart from the usual dualities of propositional classical logic given by DeMorgan laws we have also equivalences:

$$[a]\alpha \equiv \neg\langle a\rangle\neg\alpha \quad \nu X.\alpha(X) \equiv \neg\mu X.\neg\alpha(\neg X)$$

The dualities hint the possible normal form of the μ-calculus formulas which will be of some use in later sections.

Definition 1.1.1 A formula α is called *positive* iff the only negations which occur in α stand before propositional constants and free variables. A variable X in $\mu X.\alpha(X)$ is *guarded* iff every occurrence of X in $\alpha(X)$ is in the scope of some modality operator $\langle\rangle$ or $[]$. We say that a formula is *guarded* iff every bound variable in the formula is guarded.

Proposition 1.1.2 Every formula is equivalent to a positive guarded formula.

Proof
Let φ be any formula, we first show how to obtain an equivalent guarded formula.

Suppose $\varphi = \mu X.\alpha(X)$ and $\alpha(X)$ is a guarded formula. Suppose X is unguarded in some subformula of $\alpha(X)$ of the form $\sigma Y.\beta(Y,X)$ and Y is guarded in $\sigma Y.\beta(Y,X)$. Then one can use the equivalence $\sigma Y.\beta(Y,X) \equiv \beta(\sigma Y.\beta(Y,X),X)$ to obtain a formula with all unguarded occurrences of X outside the fixpoint operator. This way we obtain a formula equivalent to $\alpha(X)$ but with all unguarded occurrences of X not in the scope of a fixpoint operator.

Now using the laws of classical propositional logic we can transform this formula to conjunctive normal form (considering formulas of the form $\langle a\rangle\gamma$ and $[a]\gamma$) as propositional constants. This way we obtain a formula

$$(X \vee \alpha_1(X)) \wedge \ldots \wedge (X \vee \alpha_i(X)) \wedge \beta(X) \tag{1.1}$$

where all occurrences of X in $\alpha_1(X), \ldots, \alpha_i(X), \beta(X)$ are guarded. Variable X occurs only positively in (1.1) because it did so in our original formula. Formula (1.1) is equivalent to

$$(X \vee (\alpha_1(X) \vee \ldots \vee \alpha_i(X))) \wedge \beta(X)$$

We will show that $\mu X.(X \vee \bar{\alpha}(X)) \wedge \beta(X)$ is equivalent to $\mu X.\bar{\alpha}(X) \wedge \beta(X)$. It is obvious that

$$(\mu X.\bar{\alpha}(X) \wedge \beta(X)) \;\Rightarrow\; (\mu X.(X \vee \bar{\alpha}(X)) \wedge \beta(X))$$

Let $\gamma(X)$ stand for $\bar{\alpha}(X) \wedge \beta(X)$. To prove another implication it is enough to observe that $\mu X.\gamma(X)$ is a prefixpoint of $\mu X.(X \vee \bar{\alpha}(X)) \vee \beta(X)$ as the following calculation shows:

$$\begin{aligned} ((\mu X.\gamma(X)) \vee \bar{\alpha}(\mu X.\gamma(X))) \wedge \beta(\mu X.\gamma(X)) &\Rightarrow \\ ((\bar{\alpha}(\mu X.\gamma(X)) \wedge \beta(\mu X.\gamma(X))) \vee \bar{\alpha}(\mu X.\gamma(X))) \wedge \beta(\mu X.\gamma(X)) &\Rightarrow \\ \bar{\alpha}(\mu X.\gamma(X) \wedge \beta(\mu X.\gamma(X)) & \end{aligned}$$

If φ is a guarded formula then we use dualities of the μ-calculus to produce equivalent positive formula. It is easy to see that it will be still guarded formula. □

This proposition shows among others that we could make negation a defined connective. We have decided not to do this although we will frequently restrict to positive guarded formulas.

Despite its appealing simplicity, the μ-calculus is a very expressive logic. It turned out that it is as expressive as $S2S$ [37, 38, 11].

In this context the surprising fact is that checking validity of μ-calculus formulas was shown to be decidable in EXPTIME [7]. Another property connected with decidability is the *small model property.* This property states that if a formula φ is satisfiable then there exists a model for φ, of size bounded by some function of size of φ. The small model property is not a necessary condition for decidability as shows an example of PDLΔ with converse operation [52], but makes the logic more manageable.

The equivalence of the μ-calculus and $S2S$ shows some kind of expressive completeness of the logic. The other result which confirms its quality is that it is closely related to bisimulation equivalences. This makes it a natural logic for process calculi such as CCS [19, 51].

The properties of the μ-calculus made it a valuable tool in verifying real life systems. In the area of model checking μ-calculus is extensively used as it presents a good balance between expressibility and complexity of checking satisfiability of a formula [50, 59]. As we will show in this work, the μ-calculus has also quite an elegant complete axiom system.

1.2 Other Logics of Programs

In this section we will relate μ-calculus to other logics of programs. This will allow us to hint where the problems of finding complete axiomatization for the μ-calculus begin.

1.2.1 Propositional Dynamic Logic

Historically the first propositional modal logic constructed with computer science in mind was propositional dynamic logics (PDL) [12, 13]. This logic extends simple propositional modal logic by allowing regular expressions with tests to appear in the place of modalities.

In PDL a special syntactical category of *programs* is distinguished. Programs are interpreted as binary input-output relations on states. Formulas, as usual, are interpreted as sets of states.

Formulas and programs are defined by mutual recursion

$$\begin{array}{rcl} P & := & a \mid P_1 \cup P_2 \mid P_1; P_2 \mid P^* \mid F? \\ F & := & p \mid \neg F \mid F_1 \wedge F_2 \mid \langle P \rangle F \end{array}$$

Here we used a and p to stand for an atomic action and a propositional letter respectively. Given a Kripke structure $\mathcal{M} =< S, R, \rho >$ we define the meaning of a program and a formula simultaneously:

$$\begin{array}{rcl} \parallel a \parallel_{\mathcal{M}} & = & R(a) \\ \parallel P_1 \cup P_2 \parallel_{\mathcal{M}} & = & \parallel P_1 \parallel_{\mathcal{M}} \cup \parallel P_2 \parallel_{\mathcal{M}} \\ \parallel P_1; P_2 \parallel_{\mathcal{M}} & = & \parallel P_1 \parallel_{\mathcal{M}} \circ \parallel P_2 \parallel_{\mathcal{M}} \\ \parallel P^* \parallel_{\mathcal{M}} & = & \parallel P \parallel_{\mathcal{M}}^* \\ \parallel F? \parallel_{\mathcal{M}} & = & \{(s,s) : s \in \parallel F \parallel_{\mathcal{M}}\} \\ \parallel p \parallel_{\mathcal{M}} & = & \rho(p) \\ \parallel \neg F \parallel_{\mathcal{M}} & = & S \setminus \parallel F \parallel_{\mathcal{M}} \\ \parallel F_1 \wedge F_2 \parallel_{\mathcal{M}} & = & \parallel F_1 \parallel_{\mathcal{M}} \cap \parallel F_2 \parallel_{\mathcal{M}} \\ \parallel \langle P \rangle F \parallel_{\mathcal{M}} & = & \{s : \exists t.(s,t) \in \parallel P \parallel_{\mathcal{M}} \wedge t \in \parallel F \parallel_{\mathcal{M}}\} \end{array}$$

Symbols $\circ$ and * used above denote composition and transitive closure of a relation respectively.

A very pleasant property of PDL, not shared by many other modal logics we will consider, is so called collapsed model property. The property states that if a formula φ is satisfied in a state s of a structure $\mathcal{M}$ then it is satisfied in a structure obtained by identifying states which satisfy the same subformulas of φ. More formally let us define

Definition 1.2.1 The Fisher-Ladner closure of a PDL formula φ, denoted $CL(\varphi)$, is a smallest set of formulas such that $\varphi \in CL(\varphi)$ and for every action a, programs P_1, P_2 and formulas α, β:

- if $\neg\alpha \in CL(\varphi)$ then $\alpha \in CL(\varphi)$,
- if $\alpha \wedge \beta \in CL(\varphi)$ then $\alpha \in CL(\varphi)$ and $\beta \in CL(\varphi)$,
- if $\langle a \rangle\alpha \in CL(\varphi)$ then $\alpha \in CL(\varphi)$,
- if $\langle P_1; P_2 \rangle\alpha \in CL(\varphi)$ then $\langle P_1 \rangle\langle P_2 \rangle\alpha \in CL(\varphi)$,
- if $\langle P_1 \cup P_2 \rangle\alpha \in CL(\varphi)$ then $\langle P_1 \rangle\alpha \in CL(\varphi)$ and $\langle P_2 \rangle\alpha \in CL(\varphi)$,
- if $\langle P^* \rangle\alpha \in CL(\varphi)$ then $\langle P \rangle\langle P^* \rangle\alpha \in CL(\varphi)$ and $\alpha \in CL(\varphi)$,
- if $\langle \beta? \rangle\alpha \in CL(\varphi)$ then $\beta \in CL(\varphi)$ and $\alpha \in CL(\varphi)$,

Definition 1.2.2 For any formula φ and any structure $\mathcal{M} = \langle S, R, \rho \rangle$, we define a *quotient structure* $\mathcal{M}' = \langle S', R', \rho' \rangle$ as follows:

- $\mathcal{M}' = \{[s] : s \in \mathcal{M}\}$ where
$$[s] = \{t \in S : \text{for all } \alpha \in CL(\varphi),\ \mathcal{M}, s \models \alpha \text{ iff } \mathcal{M}, t \models \alpha\}$$
- $([s], [t]) \in R'(a)$ iff there are $s' \in [s]$ and $t' \in [t]$ such that $(s', t') \in R(a)$,
- $\rho'(p) = \{[s] : s \in \rho(p)\}$ ■

Fisher and Ladner [13] showed the following theorem:

Theorem 1.2.3 (Collapsed model property) *Let φ, $\mathcal{M}$ and $\mathcal{M}'$ be as above. Then for every $\alpha \in CL(\varphi)$ and every $s \in S$ we have $\mathcal{M}, s \models \alpha$ iff $\mathcal{M}', [s] \models \alpha$.*

The collapsed structure is relatively small. The size of $CL(\varphi)$ is linear in the size of φ. Hence the size of the structure is bounded by single exponential function in the size of φ. The important consequence of collapsed model property is that it gives us the ability to perform "surgery" on models, i.e., to cut or merge two or more structures leaving some properties unchanged.

On the other hand Theorem 1.2.3 points out an essential weakness of PDL. This logic can express only local properties of the state or, to put it differently, input-output relation of programs. In consequence some important global properties of the structure (like well foundedness) are not expressible in PDL. In other words, total correctness is not expressible in PDL.

There are finitary complete axiomatizations of PDL. Sometimes the completeness we have in mind is called weak completeness because it is only concerned with proving validity of a formula.

Strong completeness refers to the axiomatization of semantic consequence relation, $\Gamma \models \varphi$ for any set of formulas Γ and formula φ. We say that an axiomatization is weakly complete iff it is possible to prove $\vdash \varphi$ for any valid formula φ. In classical first order logic there is no difference between weak and strong completeness because the logic is compact.

PDL is not compact, i.e., there is a set of formulas Γ and a formula φ s.t. $\Gamma \models \varphi$ but for any finite subset $\Sigma \subseteq \Gamma$, we have $\Sigma \not\models \varphi$. This implies that there cannot be a finitary strongly complete axiomatization of PDL although there are infinitary ones [18, 32]. In what follows by completeness we always understand weak completeness.

Some simple program constructs, including nondeterministic choice, are easily definable in PDL:

- **if** γ **then** P_1 **else** P_2 **fi** as $(\gamma?; P_1) \cup (\neg\gamma?; P_2)$
- **while** γ **do** P **od** as $(\gamma?; P)^*; \neg\gamma?$
- **either** P_1 **or** P_2 **ro** as $P_1 \cup P_2$

Decidability of PDL is relatively easy in the sense of complexity. It is EXPTIME complete [12, 13, 42], hence at the present state of knowledge similar to the decidability of the classical propositional logic. It is worth to recall here, that the decidability of the μ-calculus is also complete in EXPTIME even thought it is much more expressive logic.

The fact that decidability of both logics is equally difficult is even more surprising in the light of the fact that there is a simple syntactic translation of PDL into the μ-calculus. Constructs not present in the μ-calculus can be translated using the following clauses:

- $[\![\langle P_1; P_2\rangle\alpha]\!]^\circ = [\![\langle P_1\rangle\langle P_2\rangle\alpha]\!]^\circ$,
- $[\![\langle P_1 \cup P_2\rangle\alpha]\!]^\circ = [\![\langle P_1\rangle\alpha]\!]^\circ \vee [\![\langle P_2\rangle\alpha]\!]^\circ$,
- $[\![\langle P^*\rangle\alpha]\!]^\circ = \mu X.([\![\alpha]\!]^\circ \vee [\![\langle P\rangle X]\!]^\circ)$ where X is a fresh variable,
- $[\![\langle \gamma?\rangle\alpha]\!]^\circ = [\![\gamma]\!]^\circ \wedge [\![\alpha]\!]^\circ$,

Notice that a translated formula can be exponentially larger then the original one (but not if we present a formula as a DAG). The translation becomes linear if we use multiple fixpoint μ-calculus as described in [58].

1.2.2 Total correctness

Because total correctness is an important property of program behavior there were many suggestions how to introduce it to modal logics of programs. We will present here the logic called PDLΔ considered by Streett [52]. The other approaches worth mentioning are Pratt's continuous μ-calculus [43], propositional algorithmic logic (PAL) [32] and process logic (PL) [17].

PDLΔ is an extension of PDL with a new operator Δ. For any program P the intended meaning of ΔP is that infinite repetition of P is possible, i.e.,

$$\mathcal{M}, s \models \Delta P \text{ iff } \exists s_0, s_1, \ldots \text{ s.t. } s_0 = s \text{ and } \forall i \geq 0, (s_i, s_{i+1}) \in \parallel P \parallel_{\mathcal{M}}$$

With this operator the statement that all executions of program P^* terminate in a state satisfying α can be written as a formula $\neg\Delta P \wedge [P^*]\alpha$. Formula $[P^*]\alpha$ alone states only partial correctness, i.e. that every terminating execution of P^* ends in a state satisfying α. Total correctness clauses for program constructs other than iteration are straightforward.

Finite model property and decidability of PDLΔ follow from the fact that PDLΔ can be coded into the μ-calculus by adding the following clause to the translation of PDL:

$$[\![\Delta P]\!]^\circ = \nu X.[\![\langle P\rangle X]\!]^\circ$$

Similarly PAL and PL have simple syntactic translations into the μ-calculus.

An interesting fact about PDLΔ is that although it has the small model property, it does not have the collapsed model property as the following example shows.

Let us define the structures $\mathcal{M}_n = \langle\{0,\dots,n\}, R_n, \rho_n\rangle$, where $R_n(a) = \{(i,i+1) : i = 0,\dots,i-1\}$ and ρ_n is arbitrary. Let us consider the formula $\neg\Delta a$ which certainly is true in the state 0 of every one of this structures.

Suppose now that we have somehow defined a Fisher-Ladner closure of a PDLΔ formula to be a finite set of formulas. For any n the size of $\mathcal{M}_n$ after collapsing cannot be greater then some fixed number which depends only on the size of the closure and not on n. Hence for sufficiently large n there will be a loop in a collapsed structure obtained from $\mathcal{M}_n$. This implies that $\neg\Delta a$ will not be true in the equivalence class containing state 0.

This example shows that lack of the collapsed model property is not specific to PDLΔ but rather connected to the ability to express total correctness in the logic.

This increased expressibility seems to introduce new level of difficulty in finding complete finitary axiom systems. Completeness of PDL can be shown in numerous different ways [14, 39, 26, 23, 27]. All of them seem to use the collapsed model property in more or less direct way. Since for PDLΔ, and hence also for the μ-calculus, the collapsed model property is not true, there is no hope to extend those methods to show the completeness of the μ-calculus.

An infinitary axiomatization of PAL was given in [32]. No complete finitary axiomatization of PDLΔ, PAL or PL is known. We believe that it should be possible to obtain axiomatizations of these logics using methods presented in this thesis.

Even though PDLΔ is strictly more expressive then PDL, it is still less expressive then the μ-calculus. There is a very elegant example due to Damian Niwiński. He proved that the formula $\nu X.[a]X \wedge [b]X$ is not expressible in PDLΔ [36]. This example shows that there is something more in the μ-calculus then just termination arguments.

1.2.3 Temporal Logics

There are numerous systems of propositional temporal logics as well as a big variety of predicate temporal logics. The difference between modal and temporal logic is that formulas of temporal logics express properties of runs, i.e. sequences of states. This is influenced by the different idea which lies behind temporal logics. Modal logics of programs mention programs explicitly. On the other hand temporal logics are designed to reason directly about execution sequences of the programs, i.e. sequences of states which are possible executions of one assumed program.

Temporal logics basically come in two kinds: linear time and branching time. First allows to reason about one execution at time, the second can talk about all executions simultaneously. Most linear time propositional temporal logics are reducible to the μ-calculus in the straightforward way [6]. We briefly sketch here the definition of branching time temporal logic CTL [2].

Formulas of CTL are divided into two categories of state and path formulas which can be defined by mutual induction as follows:

- atomic letter is a state formula,
- if φ, φ' are state formulas then so are $\varphi \wedge \varphi', \neg\varphi$,
- if ψ is a path formula then $\mathbf{E}\psi$ and $\mathbf{A}\psi$ are state formulas.
- if φ, φ' are state formulas then $\mathbf{X}\varphi, \varphi\mathbf{U}\varphi'$ are path formulas.

A formula of CTL is interpreted in a Kripke structure $\mathcal{M} =< S, R, \rho >$ but because we don't have atomic actions, R now is just a binary relation on states which is total, i.e. there is at least one arc from any state. We will use the convention that π denotes an infinite path in $\mathcal{M}$, i.e. a sequence $(s_1, s_2, \ldots)$, and π^i the suffix path $(s_i, s_{i+1}, \ldots)$. Because there are two categories of formulas we will use $\mathcal{M}, s \models \varphi$ to mean that a state formula φ is satisfied in state s and similarly $\mathcal{M}, \pi \models \psi$ for a path formula ψ. The inductive definition of satisfiability can be presented as follows:

- $\mathcal{M}, s \models q$ iff $s \in \rho(q)$,
- $\mathcal{M}, s \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \models \varphi_1$ and $\mathcal{M}, s \models \varphi_2$
  $\mathcal{M}, s \models \neg\varphi$ iff not $\mathcal{M}, s \models \varphi$,
- $\mathcal{M}, s \models \mathbf{E}\psi$ iff there exists a path $\pi = (s, s_2, \ldots)$ in $\mathcal{M}$ and $\mathcal{M}, \pi \models \psi$,
  $\mathcal{M}, s \models \mathbf{A}\psi$ iff for all paths $\pi = (s, s_2, \ldots)$ in $\mathcal{M}$ and $\mathcal{M}, \pi \models \psi$,
- $\mathcal{M}, \pi \models \varphi_1\mathbf{U}\varphi_2$ iff there is i s.t. $\mathcal{M}, s_i \models \varphi_2$ and for all $j < i$ it holds that $\mathcal{M}, s_j \models \varphi_1$.
- $\mathcal{M}, \pi \models \mathbf{X}\varphi$ iff $\mathcal{M}, \pi^1 \models \varphi$.

When the restrictions on the use of path quantifiers $\mathbf{E}, \mathbf{A}$ are dropped we obtain the logic CTL* which is strictly more expressive then CTL. When we add, to CTL* Büchi automata on infinite words as temporal operators, we obtain ECTL*. See [4] for overview of numerous temporal logics.

CTL does not have the collapsed model property. Let us take the formula $\mathbf{A}(true\mathbf{U}\alpha)$ which states that on every path a state satisfying α can be eventually reached. Using the argument very similar to that for PDLΔ one can show that CTL cannot have the collapsed model property if only we define closure of a formula to be a finite set.

Translation of branching time temporal logic into the μ-calculus is more complicated then for linear time temporal logics, especially when one considers such powerful branching temporal logics as CTL* [9] and ECTL* [57, 54]. Existing translations of this logics are quite complex and refer to automata theory or properties of models. More "syntactical" translations [3] rely heavily on tableaux method. In consequence the translation is not compositional and quite complex as well.

The other way to show that the μ-calculus is at least as expressive, or even strictly more expressive than CTL* or ECTL*, is semantical [60]. One can show that the sets definable by those logics are also definable in the μ-calculus. Indeed, μ-calculus is strictly more expressive [60].

There is a complete axiomatization of CTL [10]. There are no known axiomatizations of CTL*. An interesting fact is that decidability of CTL* is complete for double exponential time. Hence even though less expressive, the logic is more succinct than the μ-calculus.

1.3 Automata on infinite objects

Automata theory proved to be a valuable tool for modal logics of programs. It is especially fruitful for proving decidability of numerous logics and classifying their expressibility. For example the satisfiability problem for a formula of linear temporal logic is easily reducible to the emptiness problem of some finite automaton on infinite strings. Similarly, for branching time temporal logics and for modal logics, checking satisfiability of a formula reduces to testing emptiness of a certain finite automaton on infinite trees.

Following this examples we will essentially use automata theoretic results and methods in our completeness proof. In this section we will introduce some basic definitions and a bit of automata theory we will need.

A *Büchi automaton* over an alphabet Σ is a quadruple $\mathcal{A} = (Q, q_0, \Delta, F)$

where:

- Q is a finite set of states,
- $q_0 \in Q$ is an initial state,
- $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- $F \subseteq Q$ the set of final states

A *run* of $\mathcal{A}$ on a an infinite word $v_1 v_2 \ldots$ over Σ is an infinite sequence of states $s_1, s_2, \ldots$ such that $s_1 = q_0$ and $(s_i, v_i, s_{i+1}) \in \Delta$, for each $i \geq 0$.

A run is called *accepting* iff there is a state from F which occurs infinitely often on the run. The *language accepted* by the automaton $\mathcal{A}$ is the set of those infinite words over Σ for which there is an accepting run of $\mathcal{A}$. The *emptiness problem* for the automata is the question whether the language accepted by a given automaton is empty or not. Emptiness problem for Büchi automata was shown NlogSpace complete [56].

Deterministic Büchi automata are weaker then their nondeterministic counterparts. There are other kinds of automata the deterministic and nondeterministic versions of which have the same expressive power. One of them is Rabin automaton on strings.

The only difference between Rabin and Büchi automata is in acceptance conditions. In Rabin automata, instead of one set of final states there is a set of pairs of states $\Omega = \{(L_1, U_1), \ldots, (L_n, U_n)\}$. A run is accepting, subject to the conditions Ω, iff there is $i \in \{1, \ldots, n\}$ s.t. no state from L_i occurs infinitely often on the run and some state from U_i occurs infinitely often on the run.

Büchi automata were used to show decidability of the monadic second order theory of one successor S1S [1]. The second order theory of two successors proved to be much more difficult and finally was shown decidable by Rabin [44]. In his proof a new kind of automata, later called Rabin automata on trees was used.

Rabin automaton on trees over an alphabet Σ is can be presented as a quadruple $\mathcal{A} = \langle Q, q_0, \Delta, \Omega \rangle$ where Q is a set of states, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q \times Q$ a transition relation and Ω the set of pairs as in Rabin automaton on words. A run of $\mathcal{A}$ on a binary tree t is a function $r : t \to Q$ s.t. $r(\varepsilon) = q_0$ and $(r(w), t(w), r(w0), r(w1)) \in \Delta$. The run is successful iff every path of the tree satisfies Rabin conditions Ω. A tree t is accepted by an automaton $\mathcal{A}$ iff there is a successful run of $\mathcal{A}$ on t.

Proofs of decidability for $S1S$ as well as for $S2S$ followed the same pattern. First some kind of automata was selected and the emptiness problem for this kind of automata was shown decidable. Next, the reduction of the satisfiability of the formula to the emptiness problem of the chosen kind of automata was shown. Both Büchi and Rabin gave compositional, syntax directed translations from a given formula φ to an automaton $\mathcal{A}_\varphi$ s.t. $\mathcal{A}_\varphi$ accepted some input iff φ was satisfiable.

The technique of reducing decidability problem to the emptiness problem was found very fruitful for modal logics of programs. Streett, using this method, showed decidability of PDLΔ with the converse operator. In 1984 Streett and Emerson [53] showed in this way that the μ-calculus is elementary decidable.

One very important difference between the proofs for $S1S$ or $S2S$ and the proofs for modal logics is that in the later case the translations of formulas into automata are not compositional and syntax driven. Instead, to derive an automaton a tableau method was used. We will introduce the tableau method in the next chapter where we extend the results from [53].

Once one knows that the logic is decidable, the natural question to ask is what is the lower bound on the complexity of a decision procedure and how to construct an optimal algorithm. Somewhat surprisingly the reductions used to show decidability led to essentially optimal decision procedures. To put it more strongly, the only known optimal decision procedures for several logics of programs, μ-calculus included, are derived from the translations to the automata.

Obtaining optimal decision procedures was a very complex task. The problem was, that in such procedures either determinization or complementation of automata was used. This influenced increased interest in optimizing determinization and complementation procedures. It was also important to have optimal procedure for testing emptiness of the Rabin automata.

First step in designing optimal procedure for testing emptiness of Rabin automata was the following theorem proved in [5]

Theorem 1.3.1 (Emerson) *Suppose $\mathcal{A}$ is a Rabin automaton over a single letter alphabet. If $\mathcal{A}$ accepts some tree then there is a graph $\mathcal{G}$ with states of $\mathcal{A}$ as nodes which unwinds to an accepting run of $\mathcal{A}$.*

Although it was shown already by Rabin in 1972 that if a Rabin automaton accepts some tree then it accepts a regular tree, the above theorem gives particularly elegant description of such a regular tree and the bound on its size.

The theorem was used in [5] to show that the emptiness problem for Rabin automata is in NP [5, 55]. Then Emerson and Jutla [7] showed that the problem is indeed NP-complete. In the same paper they show that there is a deterministic algorithm testing emptiness of an automaton which works in time $\mathcal{O}((mn)^{3n})$, where m is the number of states and n the number of pairs in the acceptance condition.

The quest for optimal complementation of Büchi automata led through determinization of the automata. As it was mentioned, nondeterministic Büchi automata are strictly stronger then their deterministic counterparts. Hence determinization procedure for Büchi automata must give a different kind of automata, such as deterministic Rabin or Muller automaton in the result. The first such procedure was proposed by McNaughton [30]. Optimal and particularly elegant procedure was given by Safra [45]. Optimal complementation procedure for Büchi automata was presented in [49]. Optimal simultaneous complementation and determinization procedure was given by Emerson and Jutla [8].

1.4 Games

We need to mention one other significant problem of automata theory. It is complementation of Rabin automaton on trees. The first complementation construction was the heart of the Rabin's proof of the decidability of $S2S$. The proof was extremely complicated. In 1970 Rabin himself put as an open problem finding a simpler complementation construction. Particularly elegant solution was given by Muchnik [34]. Gurevich and Harington [16] show the result by proving that certain infinite games are determined and what is more important there are finite forgetful winning strategies in such a game. The task of accepting or rejecting a tree is then presented as a game of this kind. An automaton accepting a complement of the language is constructed from a strategy for a player which tries to reject an input tree. This metaphor of game proved to be very fruitful. In 1992 Klarlund [22] showed essentially optimal complementation construction for Rabin automata using this ideas.

The games we will need in this thesis will have simpler form then those used in the results mentioned above. The result we will use is Martin's Theorem [29] about determinization of Borel games. We will briefly describe the basic concepts of such games for details see e.g. [33].

Let Y be a set of finite sequences such that every prefix of an element of Y belongs to Y and such that every element is a proper prefix of some element of Y. Such an Y will be called *arena* for the game. Let $\mathcal{F}(Y)$ be the collection of all infinite sequences $y_0y_1\ldots$ whose finite initial segments belong to Y. For each $A \subseteq \mathcal{F}(Y)$ one can define a game as follows.

The game is played between two players, *I* and *II*. Player *I* starts by picking $y_0 \in Y$, then player *II* responds by picking y_1 s.t. $y_0y_1 \in Y$ etc. Player *I* wins iff the chosen sequence $y_0y_1\ldots$ belongs to A. This game is a game of perfect information in the sense that a player can see all previous moves while making next move. A strategy $\mathcal{S}$ for player *I* is a function whose domain is the set of elements of Y of even length such that always $y_0\ldots y_{2n-1}\mathcal{S}(y_0\ldots y_{2n-1}) \in Y$. Player *I* plays according to the strategy $\mathcal{S}$ iff for all n, $y_{2n} = \mathcal{S}(y_0\ldots y_{2n-1})$. Similarly we can define the notions of a strategy and a play according to a strategy for player *II*.

We say that the game is determined iff there is a wining strategy for one of the players. Of course, the existence of the winning strategy strongly depends on the acceptance set A.

On $\mathcal{F}(Y)$ we can give a topology by taking as a base all subsets of $\mathcal{F}(Y)$ of the form $\{x : p \text{ is a prefix of } x\}$ for $p \in Y$. A set $A \subseteq \mathcal{F}(Y)$ is Borel iff it belongs to the σ-algebra generated by the open subsets of $\mathcal{F}(Y)$, i.e., it is constructed from open sets by taking infinite sums and complementations.

Martin [29] proved that all Borel games are determined, i.e., if A is Borel than there exists a wining strategy for one of the players in a game described above. We will not need this result in its full strength, our games will lie very low in the Borel hierarchy.

1.5 Completeness problem in propositional modal logics

As one might expect the difficulty of designing and proving completeness of axiomatizations increases with the increase of expressibility of the logic. In this section we will try to present different approaches to proving completeness for different propositional modal logics.

The simplest modal logic is the system K where there are only atomic actions as modalities. In this logic almost the same methods as for classical propositional logic apply. We will present one of them, of which our completeness proof for the μ-calculus is an extension.

We will consider only positive guarded formulas of modal logic and *sequents* of form $\Gamma \vdash$. The interpretation of such a sequent is standard, i.e., that conjunction of formulas in Γ is not satisfiable. One sided sequent calculi tend to have less rules then two sided versions. We use sequents $\Gamma\vdash$ instead of more common $\vdash\Gamma$ because we think they better illustrate the point we want to make. The sequent $\Gamma\vdash$ is called an *axiom* iff there is a propositional letter p such that $p, \neg p \in \Gamma$.

For a simple modal logic the following system is sufficient:

$$(and) \qquad \frac{\alpha, \beta, \Gamma\vdash}{\alpha \wedge \beta, \Gamma\vdash}$$

$$(or) \qquad \frac{\alpha, \Gamma\vdash \quad \beta, \Gamma\vdash}{\alpha \vee \beta, \Gamma\vdash}$$

$$(\langle\rangle) \qquad \frac{\alpha, \{\beta : [a]\beta \in \Gamma\}\vdash}{\langle a\rangle\alpha, \Gamma\vdash}$$

The *proof* of a sequent $\Gamma\vdash$ in this system is a diagram obtained using the rules above with the root labeled by $\Gamma\vdash$ and all leaves labeled by axioms.

It is easy to check that all the rules are sound, i.e., if all assumptions of the rule are valid then the conclusion is valid. The completeness proof is not more difficult than that for classical propositional logic which is essentially due to the fact that any tableau is a finite tree.

Also finite model theorem can be easily deduced using this system. We can consider the rules above as a rules for constructing more general *tableaux*. Instead of the rule $(\langle\rangle)$ we take the rule:

$$(all\langle\rangle) \qquad \frac{\{\alpha, \{\beta : [a]\beta \in \Gamma\}\vdash : \langle a\rangle\alpha \in \Gamma\}}{\Gamma\vdash}$$

This rule has as many assumptions as the number of formulas of the form $\langle a\rangle\alpha$ in Γ. When there are no such formulas at all then we cannot apply this rule. Rule $(all\langle\rangle)$ can be seen as a very weak logical rule but it is probably better not to look at it this way. It is rather different type of rule, because it is enough to prove only one of the assumptions of the rule to prove the conclusion, while for other rules we have to prove all the assumptions. This distinction will be elaborated below.

Given a sequent $\Gamma\vdash$ of positive formulas, we construct a tableau for it in this tableau system. One important difference with the classical propositional logic is that we would like tableaux to be maximal in a sense that the rule $(all\langle\rangle)$ is used only to the sequents $\Sigma\vdash$ such that each formula in Σ is either a propositional constant, its negation or a formula of the form $\langle b\rangle\beta$ or $[b]\beta$ for some action b and a formula β.

The tableau constructed will obviously be finite. Some of the leaves will be axioms and others will be unreducible sequents which are not axioms. Axiom sequent is clearly valid because some propositional constant and its negation occurs in it. If $\Sigma \vdash$ is an unreducible sequent which is not an axiom then it is easy to find a state where it is not true. This is because only propositional constants, its negations, and formulas of the form $[a]\alpha$ for some a and α can occur in Σ.

The general property of tableaux for $\Gamma \vdash_{\mathcal{D}}$ is that one can find either a proof of $\Gamma \vdash$ in a tableau or we can read a finite model for the formula $\bigwedge \Gamma$ from it. To see this we apply the following simple marking procedure. We mark all the leaves labeled with axiom sequents with 1 and all other leaves with 0. Then if in a node of the tableau the rule (and) was used we mark it by the same number as its only son. If (or) rule was used in a node and both sons are labeled by 1 then we mark the node by 1 otherwise we mark it by 0. For each node where $(all\langle\rangle)$ rule was used we mark it with 0 if all sons of it are marked with 0, otherwise we mark it with 1. It should be obvious that if the root of the tableau is marked with 1 then there is a proof of $\Gamma \vdash$ in the tableau.

If the root of the tableau is labeled by 0 then we can find a model for $\bigwedge \Gamma$ in the tableau. Let us define the structure $\mathcal{M} = \langle S, R, \rho \rangle$ where:

- S is a set of all leaves in the tableau and all nodes of the tableau in which $(all\langle\rangle)$ rule was used.
- $(s, t) \in R(a)$ iff there is a son of s, r obtained by reduction of the action a and on the path from r to t the rule $(all\langle\rangle)$ was not used.
- $s \in \rho(p)$ iff s is labeled by $\Gamma \vdash$ and $p \in \Gamma$.

It is easy to prove by induction on the formula α that $\mathcal{M}, s \models \alpha$ if there is a node t labeled by $\alpha, \Gamma \vdash$ and on the path from t to s the rule $(all\langle\rangle)$ was not used.

Situation gets more difficult when we consider PDL. There is no problem in writing rules for all the connectives except *. If we include the rule:

$$(reg) \quad \frac{\alpha, \Gamma \vdash \quad \langle P \rangle \langle P^* \rangle \alpha, \Gamma \vdash}{\langle P^* \rangle \alpha, \Gamma \vdash}$$

then we would lose the property that a tableau of a sequent is always finite, hence our previous method of proving completeness does not apply and we must look for other ideas.

One solution is to use Henkin method. This in general comes from the idea that instead of showing that every valid formula is provable one shows that every consistent formula, i.e. such that its negation is not provable, has a model. To show this one usually constructs a syntactic model from a given formula. This in turn is done using a filtration method well known form classical and simple modal logics.

Completeness of PDL can be shown in numerous different ways [14, 39, 26, 23, 27]. All of them seem to use Henkin method and the collapsed model property in more or less direct way.

Very interesting in this light is the proof of completeness for CTL [9]. This logic does not have collapsed model property as was remarked in section 1.2.3. In the completeness proof for CTL, Emerson and Halpern showed that, although there is no collapsed model property, the simple syntactic structure, constructed in the way similar to that for PDL, can be transformed into a model of a starting formula.

It is not known how to extend this technique to CTL* or ECTL*. The problem is that, with increased expressibility of the logics, transformations of models became more and more difficult. Probably this lack of techniques for model manipulation is the main problem in finding complete axiom systems for more expressive logics of programs.

In conclusion, Henkin method does not seem very promising for proving completeness of the μ-calculus because of the difficulty in syntactical model constructions for such an expressible logic. Other possibility would be to try to follow the proof of Siefkes for $S1S$ and to "internalize" the decidability proof, that is to perform automata-theoretic constructions on the logical level. This looks like a possible choice because Rabin automata closely correspond to the μ-calculus. This idea of proving completeness was used among others by Kozen [25] to show the completeness of his axiomatization of Kleene algebras. An application of similar ideas to ECTL* can also be found in [3]. A disadvantage of this approach is that the automata-theoretic constructions we would have to internalize are quite complex.

In our proof we will follow neither this method nor Henkin's method. We will rather return to the ideas of the completeness proof for simple modal logic as presented at the beginning of this section and use bits of other methods on the way.

It is remarkable that although collapsing and similar techniques do not work for such logics as CTL*, ECTL*, PDLΔ, PAL, PL and μ-calculus, there is another technique which allows to prove the small model property and elementary decidability of any of this logics. All these results are obtained by employing automata-theoretic techniques. The method, although very strong, has one disadvantage when considered as a step to the completeness proof. The connection between a formula and a small model for it, which was evident for PDL because of collapsed model property, becomes very vague in case of this more expressible logics.

The first step of our proof is to examine this vague connection between a formula and a small model for it by revising the small model construction for the μ-calculus [53] and other expressive modal logics. In this results the so called tableau method is employed. This technique can be viewed as a refinement of the model construction for the simple modal logic we presented at the beginning of this section. For a logic in question one uses the reduction rules similar to that we used for modal logic. In case of PDLΔ one might add regeneration rule (reg). For other logics similar simple rules are added. Of course the consequence of employing such rules is that a tableau of a formula may be infinite. It occurs that one can give such a condition on infinite paths of a tableau that if there is "good" tableau $\mathcal{D}_\varphi$, for a formula φ, then a structure satisfying φ can be constructed from it. This structure is usually defined almost exactly as we did in the case of simple modal logic.

Next, one establishes that every satisfiable formula has a tableau of this special kind. This way the problem of checking satisfiability of a formula reduces to the problem of checking whether there is a tableau of a special kind for the formula.

At this moment automata theory comes with some solutions. Usually the condition on infinite paths of a tableau can be expressed either as Büchi or Rabin condition or negation of one of them. If it is the case then we can construct for a given formula φ an automaton $\mathcal{A}_\varphi$ accepting the set of "good" tableaus of φ. In this construction determinization or complementation procedure is usually used. This way we know that a formula φ is satisfiable iff the language accepted by $\mathcal{A}_\varphi$ is not empty. It remains to use an algorithm for testing emptiness for an appropriate kind of automata to have the decidability procedure. One can also use Theorem 1.3.1 to obtain a small model property.

This technique is clearly a refinement of the one used for modal logic. Only one part of this technique is employed, i.e. nothing is said about what happens when a formula is not satisfiable. In case of modal logic the algorithm for finding a model was dual to the task of proving that the formula is not satisfiable. It seems that this duality was left unexplored for stronger logics. We investigate it in chapter 2 in case of the μ-calculus. It occurs that when a formula φ is not satisfiable then there is a tableau for $\varphi\vdash$ which looks almost as a proof, except that it can have infinite paths which are subject to some Büchi condition. We call such a tableau a *refutation* of φ.

In some sense the status of refutation is similar to that of proof. It is a syntactical object and there are also only syntactical conditions for deciding whether something is a refutation or not. As we will see there are regular refutations, i.e., finite graphs which unwind to refutations. For any unsatisfiable formula there is a regular refutation of size bounded by the exponential function in the size of the formula. There is an algorithm which for a given formula φ constructs either a refutation of γ or a model for γ. The algorithm runs in single exponential time. Hence, if we are interested in showing validity, or equivalently, unsatisfiability of the formula, then refutations may be as good choice as proofs.

On the other hand refutations have also some disadvantages. They are too closely connected with automata hence they do not present an entirely new insight into the logic. The conditions defining refutations, even though syntactic, are global, i.e. refer to the whole tree, while proofs are only subject to local conditions, i.e., correctness of applying rules. In consequence the standard proof theoretic methods do not apply to refutations.

Even though not as good as proofs, refutations are an important step in the completeness proof. The property that the set of refutations of a formula is recognized by a Rabin automaton allows us to use automata theoretic results which in turn allow us to obtain refutations of the "right" shape. This can be compared with constructing special syntactic models in Henkin method.

The next step is to show that transitions in this "right" refutations can be internalized inside the logic. That is, there is a coding of nodes of "right" refutations such that if $m_1, \ldots m_i$ are all sons of n then the sequent $F_f(n) \vdash F_f(m_1), \ldots, F_f(m_i)$ is provable in the system, where we used $F_f(n), F_f(m_1), \ldots, F_f(m_i)$ to denote codings of the corresponding nodes. Finally, using this coding we "read" the proof of $\varphi \vdash$ from the "right" refutation of φ.

1.6 Proposed system

We propose the following axiom system for the μ-calculus. We present it here in the form similar to that of Kozen [24]. The other, sequent style presentation of the system can be found in section 3.1.

In this presentation the judgment has the form $\varphi = \psi$ with the meaning that the formulas φ and ψ are equivalent. Inequality $\varphi \leq \psi$ is considered as an abbreviation of $\varphi \vee \psi = \psi$.

Apart from the axioms and rules of equational logic (including substitution of equals by equals), i.e. cut rule, there are the following axioms and rules:

$(K1)$ axioms for Boolean algebra

$$(K2) \qquad \langle a\rangle\varphi \vee \langle a\rangle\psi = \langle a\rangle(\varphi \vee \psi)$$

$$(K3) \qquad \langle a\rangle\varphi \wedge [a]\psi \leq \langle a\rangle(\varphi \wedge \psi)$$

$$(K4) \qquad \langle a\rangle \mathit{ff} = \mathit{ff}$$

$$(K5) \qquad \alpha(\mu X.\alpha(X)) \leq \mu X.\alpha(X)$$

$$(I) \qquad \frac{\varphi(\mathit{ff}) \leq \psi \quad \varphi(\alpha(\mu X.Z \wedge \alpha(X))) \leq \psi \vee \langle P\rangle\varphi(\mu X.Z \wedge \alpha(X))}{\varphi(\mu X.\alpha(X)) \leq \langle P^*\rangle\psi}$$

In the last rule the variable Z is a new propositional variable not occurring free in $\varphi(\mu X.\alpha(X))$ or ψ. The letter P stands for a PDL program, and $\langle P\rangle\alpha$ stands for an obvious translation of this formula into the μ-calculus.

In the above notation $\varphi(\mu X.\alpha(X))$ stands for the result of the substitution $\varphi[\mu X.\alpha(X)/[]]$ where $[]$ is a special variable. The substitution $\varphi[\alpha/X]$ is legal only if no free variable of α becomes bound in $\varphi[\alpha/X]$.

The rule (I) is an induction rule. It allows to reason about fixpoint formulas in terms of their approximations. From the small model property we know that if $\varphi(\mu X.\alpha(X)) \leq \langle P^*\rangle\psi$ is not true then there exists a small, i.e., finite model where it is not true. This means that there exists a natural number n s.t. $\varphi(\alpha^n(\mathit{ff})) \leq \langle P^*\rangle\psi$ is not true. In other words to show that some μ-formula implies some formula it is enough to show that all its finite approximations imply this formula. This observation leads to the natural induction rule:

$$\frac{\varphi(\mathit{ff}) \leq \psi \quad \varphi(\alpha^{n+1}(\mathit{ff})) \leq \psi \vee \langle P\rangle\varphi(\alpha^n(\mathit{ff}))}{\varphi(\mu X.\alpha(X)) \leq \langle P^*\rangle\psi}$$

The reason why we consider μ-formula in a context is that, as it turns out, we need to perform induction on the several occurrences of the same μ-formula simultaneously.

This rule as it is written is not a syntactically correct rule of the μ-calculus. The reason is that n acts here as a variable ranging over natural numbers. The easiest way to remedy this drawback is to place a fresh propositional variable in place of $\alpha^n(\mathit{ff})$ which will result in the rule:

$$\frac{\varphi(\mathit{ff}) \leq \psi \quad \varphi(\alpha(Z)) \leq \psi \vee \langle P\rangle\varphi(Z)}{\varphi(\mu X.\alpha(X)) \leq \langle P^*\rangle\psi}$$

This rule unfortunately appears to be too weak, which is because we have strengthened the assumption to much. The reason is that the value of Z can be arbitrary while the value of $\alpha^n(\mathit{ff})$ ranged only over approximations of the μ-formula.

There is no way to express that something is exactly an approximation of the μ-formula but we can use the following observation:

If $Val(Z) = \parallel \alpha^n(\mathit{ff}) \parallel_{Val}$ then $\parallel \mu X.Z \wedge \alpha(X) \parallel_{Val} = \parallel \alpha^n(\mathit{ff}) \parallel_{Val}$.

This means that if the value of Z is some approximation of the μ-formula then the value of $\mu X.Z \wedge \alpha(X)$ is also this approximation and when the value of Z is something else, the value of $\mu X.Z \wedge \alpha(X)$ is "truncated" to approximate the meaning of $\mu X.\alpha(X)$. Our induction rule (ind) is the result of putting $\mu X.Z \wedge \alpha(X)$ in place of Z in the last of the above rules.

The only difference between the above system and the system proposed by Kozen is that the later, instead of (I) contains the rule:

$$(K6) \quad \frac{\alpha(\varphi) \leq \varphi}{\mu X.\alpha(X) \leq \varphi}$$

We will show in section 3.2 that the rule (I) is not derivable from rules $(K1), \ldots, (K6)$ but the rule $(K6)$ is derivable in our system.

1.7 Synopsis

Our first goal will be to obtain a syntactic characterization of validity. We start by reexamination of syntactic characterization of satisfiability as presented in [53]. Because we would like to base other constructions on such a characterization, we are interested in as simple characterization as possible. We find that the concept of definition constants introduced in the context of model checking [50] is especially useful for expressing some of the ideas. We define a set of rules for tableau construction and show that a formula is satisfiable iff there is a special tableau which we call a *pre-model* of the formula.

Having a characterization of satisfiability by means of infinite trees a natural question to ask is what happens when a formula is not satisfiable. In case of simple modal logic the answer is: a proof of unsatisfiability of a formula. Here we cannot hope for so much but we can at least follow the same pattern of investigation.

We construct a more general *tableau* for a formula. Because there can be infinite paths in such a tableau, a simple marking technique is useless in this case. Instead, we use Martin's determinacy theorem of infinite Borel games [29]. It occurs that a two player game on a tableau for a formula φ can be defined in such a way that a strategy for one player can be presented as a pre-model of φ. The strategy for the second player can be presented as a tree which we call a *refutation* of φ. From determinacy of this game a characterization of unsatisfiable sequents by refutations is obtained. In the last section of Chapter 2 we show that any unsatisfiable formula has a refutation which is an unwinding of a graph of size exponential in the size of the formula. There is also an algorithm which finds a pre-model or a refutation for a formula in exponential time in the size of the formula.

In chapter 3 we propose an axiom system for the μ-calculus. It can be briefly described as a system for simple modal logic with addition of two rules for fixpoint formulas. The first rule states that the interpretation of a formula $\mu X.\alpha(X)$ is a fixpoint. The second, is the induction rule. It allows us to reason about fixpoint formulas using their approximations. In that chapter we also compare the proposed system with the system presented by Kozen in [24]. We show that all the rules of Kozen's system are derivable in our system but our induction rule is not derivable in the Kozen's system.

Chapter 4 is devoted to the completeness proof. We take any unsatisfiable positive guarded formula φ_0 and show how to construct a proof of the sequent $\varphi_0 \vdash$ in our system.

From our characterization of validity we know that there is a refutation of φ_0. Because the set of refutations of a formula is recognizable by a Rabin automaton, there is a regular refutation. This refutation looks almost like a proof except that it can have some loops.

Using the ideas of Safra's determinization construction [45] we construct a special Rabin automaton $\mathcal{TA}_{\varphi_0}$ on trees over one letter alphabet, which runs correspond closely to the refutations of φ_0. We modify Safra's construction because we will then encode the states of the automaton as formulas of the μ-calculus, hence the form of the states will be very important to us.

The next step is to use Theorem 1.3.1 which guarantees existence of a special graph $\mathcal{G}_{\varphi_0}$ with states of $\mathcal{TA}_{\varphi_0}$ as nodes, which unwinds to an accepting run of $\mathcal{TA}_{\varphi_0}$. In section 4.2 we investigate the properties of this graph. We show that in $\mathcal{G}_{\varphi_0}$, special nodes, which we call *loop nodes*, can be distinguished. This nodes can be seen as "witnesses" that $\mathcal{G}_{\varphi_0}$ is accepted by the automaton. On every cycle there is exactly one loop node which "confirms" that the unwinding of the cycle is accepted by $\mathcal{TA}_{\varphi_0}$. Another important thing is that there is a natural partial order on loop nodes.

After observing these properties of $\mathcal{G}_{\varphi_0}$ we encode its nodes as formulas. The coding $F_f(s)$ of a state s depends on the function f designating the part of the structure of s to be coded. The coding is such that each transition in $\mathcal{G}_{\varphi_0}$ will be derived in our system. That is, if $t_1, \ldots, t_k$ are all nodes to which there is an edge from s in $\mathcal{G}_{\varphi_0}$ then the sequent $F_f(s) \vdash F_f(t_1), \ldots, F_f(t_n)$ is provable.

Finally we perform the special unwinding of $\mathcal{G}_{\varphi_0}$ into a finite tree with back edges $\mathcal{T}_{\varphi_0}$. This unwinding is not essential but it simplifies the proof construction. The construction of the proof of $\varphi_0 \vdash$ will be directed by the structure of $\mathcal{T}_{\varphi_0}$.

Chapter 2

Characterization

In this chapter we present a characterization of the validity of the μ-calculus formulas by means of infinite regular trees labeled with sequents. The results of this chapter were obtained jointly with Damian Niwiński [35].

We propose two systems of tableau rules. For the first system we show that a formula γ is satisfiable iff there exists a successful tableau for γ. We call it a *pre-model* of γ. Successful tableaux constructed in the second system will be infinite trees labeled with sequents, subject to some Büchi conditions on infinite paths. What's more, the rules of this second system are sound when considered as logical rules and for a tableau to be successful all leaves must be labeled with axioms. Successful tableaux of this kind will be called *refutations*.

Interestingly, it turns out that a pre-model of γ and a refutation of γ can be naturally presented as a wining strategy for one of the players in a certain infinite game. The game is determined, by the Martin's Theorem. This way we show that a formula γ is not satisfiable iff there is a refutation for γ. This characterization of the validity will be the starting point of our completeness proof.

In the last section we consider the possible size of a refutation and conclude that if a formula γ is not satisfiable then there exists regular refutation for γ of the size not bigger then $exp(|\gamma|)$ (single exponential function in the size of γ). Hence finding a refutation occurs to be as easy as testing satisfiability. What's more, there is an algorithm which in time $exp(|\gamma|)$ gives us either a refutation of γ, if γ is not satisfiable, or a model for γ, if there is one.

In this chapter we will consider only positive guarded formulas. By

Proposition 1.1.2 it is not a restriction when semantics is concerned.

2.1 Tableaux

In this section we present a system of rules for constructing a tableau for a formula. Tableaux will serve as arenas for a game we will describe later. We will also define two substructures of a tableau: quasi-model and quasi-refutation. It is convenient to introduce the concept of a definition list [50] which will name the fixpoint subformulas of a given formula in order of their nesting.

We extend vocabulary of the μ-calculus by a countable set *Dcons* of fresh symbols that will be referred to as *definition constants* and usually denoted $U, V, \ldots$ These new symbols are now allowed to appear positively in formulas, like propositional variables.

A *definition list* is a finite sequence of equations :

$$\mathcal{D} = ((U_1 = \sigma_1 X.\alpha_1(X)), \ldots, (U_n = \sigma_n X.\alpha_n(X))$$

where $U_1, \ldots, U_n \in \mathit{DCons}$ and $\sigma_i X.\alpha_i(X)$ is a formula such that all definition constants appearing in α_i are among $U_1, \ldots, U_{i-1}$. We assume that $U_i \neq U_j$ and $\alpha_i \neq \alpha_j$, for $i \neq j$. If $i < j$ then U_i is said to be *older* than U_j (U_j younger then U_i).

We assign a definition list to a formula γ by means of the contraction operation $\rrparenthesis \alpha \llparenthesis$ which is defined recursively as follows:

1. $\rrparenthesis p \llparenthesis = \rrparenthesis \neg p \llparenthesis = \rrparenthesis X \llparenthesis = \rrparenthesis U \llparenthesis = \emptyset$
2. $\rrparenthesis \langle a \rangle \alpha \llparenthesis = \rrparenthesis [a]\alpha \llparenthesis = \rrparenthesis \alpha \llparenthesis$
3. $\rrparenthesis \alpha \wedge \beta \llparenthesis = \rrparenthesis \alpha \vee \beta \llparenthesis = \rrparenthesis \alpha \llparenthesis \circ \rrparenthesis \beta \llparenthesis$
4. $\rrparenthesis \mu X.\alpha(X) \llparenthesis = (U = \mu X.\alpha(X)), \rrparenthesis \alpha(U) \llparenthesis$ where U is new.
5. $\rrparenthesis \nu X.\alpha(X) \llparenthesis = (U = \nu X.\alpha(X)), \rrparenthesis \alpha(U) \llparenthesis$ where U is new.

The operation $\rrparenthesis \alpha \llparenthesis \circ \rrparenthesis \beta \llparenthesis$ is defined as follows. First we make sure that the definition constants used in $\rrparenthesis \alpha \llparenthesis$ are disjoint form those used in $\rrparenthesis \beta \llparenthesis$. Then if it happens that $(U = \gamma) \in \rrparenthesis \alpha \llparenthesis$ and $(V = \gamma) \in \rrparenthesis \beta \llparenthesis$, we delete the definition from list $\rrparenthesis \beta \llparenthesis$ and replace V with U in $\rrparenthesis \beta \llparenthesis$. This may cause other formulas to be doubly defined and we deal with them in the same way.

We will say that U is a μ-constant if $(U = \mu X.\beta(X)) \in \mathcal{D}$, if $(U = \nu X.\beta(X)) \in \mathcal{D}$, U will be called a ν-constant. Observe that every constant occurring in $\mathcal{D}$ is either μ or ν constant.

For a formula α and a definition list $\mathcal{D}$ containing all definition constants occurring in α we define the *expansion operation* $\langle\!\langle\alpha\rangle\!\rangle_{\mathcal{D}}$, which subsequently replaces definition constants appearing in the formula by the right hand sides of the defining equations

$$\langle\!\langle\alpha\rangle\!\rangle_{\mathcal{D}} = \alpha[\alpha_n/U_n]...[\alpha_1/U_1] \quad \text{where } \mathcal{D} = (U_1 = \alpha_1), .., (U_n = \alpha_n)$$

A *tableau sequent* is a pair $(\Gamma, \mathcal{D})$, where $\mathcal{D}$ is a definition list and Γ is a finite set of formulas such that the only constants that occur in them are those from $\mathcal{D}$. We will denote $(\Gamma, \mathcal{D})$ by $\Gamma \vdash_{\mathcal{D}}$.

A *tableau axiom* is a sequent $\Gamma \vdash_{\mathcal{D}}$ such that some formula and its negation occurs in Γ.

We extend expansion operations to tableau sequents in a natural way:

$$\langle\!\langle\Gamma \vdash_{\mathcal{D}} \rangle\!\rangle_{\mathcal{D}} = \{\langle\!\langle\gamma\rangle\!\rangle_{\mathcal{D}} : \gamma \in \Gamma\}$$

In what follows we will frequently drop a prefix “tableau” if it is clear from the context. The other, two sided, type of sequents will be introduced in the chapter presenting axiomatization.

Below we present the set of rules for constructing tableaux. This rules can be considered as logical rules when read upside-down. We write them with premises below the line because it is more appropriate for tableaux construction. This style also puts emphasis on the fact that this rules are used to construct tableaux not proofs.

Definition 2.1.1 Let $\mathcal{S}$ be the following set of tableau rules :

$$(and) \qquad \frac{\alpha \wedge \beta, \Gamma \vdash_{\mathcal{D}}}{\alpha, \beta, \Gamma \vdash_{\mathcal{D}}}$$

$$(or) \qquad \frac{\alpha \vee \beta, \Gamma \vdash_{\mathcal{D}}}{\alpha, \Gamma \vdash_{\mathcal{D}} \quad \beta, \Gamma \vdash_{\mathcal{D}}}$$

$$(cons) \qquad \frac{U, \Gamma \vdash_{\mathcal{D}}}{\alpha(U), \Gamma \vdash_{\mathcal{D}}} \quad \text{whenever } (U = \sigma X.\alpha(X)) \in \mathcal{D}$$

$$(\mu) \qquad \frac{\mu X.\alpha(X), \Gamma \vdash_{\mathcal{D}}}{U, \Gamma \vdash_{\mathcal{D}}} \quad \text{whenever } (U = \mu X.\alpha(X)) \in \mathcal{D}$$

$$(\nu) \qquad \frac{\nu X.\alpha(X), \Gamma \vdash_{\mathcal{D}}}{U, \Gamma \vdash_{\mathcal{D}}} \quad \text{whenever } (U = \nu X.\alpha(X)) \in \mathcal{D}$$

$$(all\langle\rangle) \qquad \frac{\Gamma \vdash_{\mathcal{D}}}{\{\alpha, \{\beta : [a]\beta \in \Gamma\} \vdash_{\mathcal{D}} \; : \langle a \rangle \alpha \in \Gamma\}}$$

where in the last rule each formula in Γ is a propositional constant, a variable, a negation of one of them or a formula of the form $\langle b \rangle \beta$ or $[b]\beta$ for some action b and a formula β.

Observe that each rule, except (or) or $(all\langle\rangle)$, has exactly one premise. The rule (or) has two premises and the number of premises in the rule $(all\langle\rangle)$ is equal to the number of formulas of the form $\langle a \rangle \alpha$ in Γ and may be 0.

The system $\mathcal{S}_{mod}$ is obtained from $\mathcal{S}$ by replacing the rule (or) by two rules (or_{left}) and (or_{right}) defined in the obvious way.

The system $\mathcal{S}_{ref}$ is obtained from $\mathcal{S}$ by replacing the rule $(all\langle\rangle)$ by the rule

$$(\langle\rangle) \qquad \frac{\langle a \rangle \alpha, \Gamma \vdash_{\mathcal{D}}}{\alpha, \{\beta : [a]\beta \in \Gamma\} \vdash_{\mathcal{D}}}$$

with the same restrictions on formulas in Γ as in the case of $(all\langle\rangle)$ rule.

Observe that if we consider a sequent $\Gamma \vdash_{\mathcal{D}}$ as a formula $\bigwedge \langle\!|\Gamma|\!\rangle_{\mathcal{D}} \Rightarrow \mathit{ff}$ then the rules of the system $\mathcal{S}_{ref}$ become sound logical rules.

Definition 2.1.2 Given a positive guarded formula γ, a *tableau* for γ is any labeled tree $\langle K, L \rangle$, where K is a tree and L a labeling function, such that

1. the root of K is labeled with $\gamma \vdash_{\mathcal{D}}$ where $\mathcal{D} = |\!\rangle \gamma \langle\!|$,
2. if $L(n)$ is a tableau axiom then n is a leaf of K,

3. if $L(n)$ is not an axiom then the sons of n in K are created and labeled according to the rules of the system $\mathcal{S}$.

A *quasi-model* of γ is defined in a similar way to tableau, except the system $\mathcal{S}_{mod}$ is used instead of $\mathcal{S}$ and we impose the additional requirement that *no* leaf is labeled by a tableau axiom.

A *quasi-refutation* of γ is defined in a similar way to tableau, except the system $\mathcal{S}_{ref}$ is used instead of $\mathcal{S}$ and we impose the additional requirement that *every* leaf is labeled by a tableau axiom. ∎

Remark: Observe that each quasi-model, as well as a quasi-refutation can be obtained from a tableau by cutting off some nodes.

Definition 2.1.3 Let $\mathcal{T} = \langle K, L\rangle$ be a tableau for a positive guarded formula γ and $\mathcal{D} = \llbracket\gamma\rrbracket$. Let $\mathcal{P} = (v_1, v_2, \ldots)$ be a path in the tree K, i.e., each v_{i+1} is a son of v_i. A *trace* $\mathcal{T}r$ on the path $\mathcal{P}$ is any sequence of formulas $\{\alpha_i\}_{i\in I}$ such that $\alpha_i \in L(v_i)$ and α_{i+1} is either:

- α_i if formula α_i was not reduced by the rule applied in v_i, or otherwise

- α_{i+1} is one of the formulas obtained by applying the rule to α_i.

This last notion should be clear if the rule is other then $(\langle\rangle)$. If the rule in question is $(\langle\rangle)$ and it reduces a formula $\langle a\rangle\alpha$ then $\alpha_{i+1} = \beta$ iff $\alpha_i = [a]\beta$ or $\alpha_{i+1} = \alpha$ if $\alpha_i = \langle a\rangle\alpha$; in other cases α_i is the last element of the trace.

Definition 2.1.4 A constant U *regenerates* on the trace $\mathcal{T}r$ if for some i, $a_i = U$ and $a_{i+1} = \alpha(U)$ where $(U = \sigma X.\alpha(X)) \in \mathcal{D}$

The trace $\mathcal{T}r$ is called a *ν-trace* iff it is finite and does not end with a tableau axiom, or if the oldest constant in the definition list $\mathcal{D}$ which is regenerated infinitely often on $\mathcal{T}r$ is a ν-constant. Otherwise the trace is called a *μ-trace.* ∎

Observe that a μ-trace is either a finite trace ending in a tableau axiom or an infinite trace on which the oldest constant regenerated infinitely often is some μ-constant.

The above definition applies as well to quasi-models and to quasi-refutations.

Definition 2.1.5

— A quasi-model $\mathcal{PM}$ is called *pre-model* iff any trace on any path of $\mathcal{PM}$ is a ν-trace.

— A quasi-refutation of γ is called a *refutation* of γ iff on every path of it there exists a μ-trace.

We will show in the next sections that a formula is satisfiable iff there is a pre-model for it and that a formula is unsatisfiable iff there is a refutation for it. The condition laid on pre-models is due to the observation that if for some structure $\mathcal{M}$ and state s, $\mathcal{M}, s \models \mu X.\alpha(X)$ then the smallest ordinal τ s.t. $\mathcal{M}, s \models \alpha^\tau(\mathit{ff})$ must be a successor ordinal. Hence $\mathcal{M}, s \models \alpha(\alpha^\sigma(\mathit{ff}))$ for some $\sigma < \tau$. That is in the process of "investigating" the "reasons" why the μ-formula is satisfied we have managed to reduce the index of the formula. Because ordinals are well ordered it means that we will need to do regeneration of the μ-formula only finitely many times. The condition on refutations is dual and obtained from analysis of a game we are going to describe in the next section.

2.2 Games

In this section we show that any formula γ has either a pre-model or a refutation.

Let $\mathcal{T}$ be a tableau for γ. We define an infinite game for two players to be played on $\mathcal{T}$. Intuitively, player *I* will try to show that γ is satisfiable and player *II* that it is not. Our two players play the game as follows.

- game will start in the root of $\mathcal{T}$,
- in any (or) node, i.e. node where (or) rule was applied, player *I* chooses one of the sons,
- in any $(all\langle\rangle)$ node, player *II* chooses one of the sons,
- in other nodes which are not leaves automatically the only son is chosen.

The result of such a game is either a finite or an infinite path of the tableau $\mathcal{T}$. The path can be finite only when it ends in a leaf which can be labeled either by axiom or by unreducible sequent but not an axiom. In the former case player *II* wins and in the latter case player *I* is the winner. If the resulting path is infinite, then player *II* wins iff we can find a μ-trace on the path.

Now, it is easy to see that our game can be presented in the form that satisfies the conditions of the Martin's theorem on determinacy of infinite Borel games [29], and hence it is determined. A winning strategy of either player can be naturally presented as a tree. More precisely, a winning strategy for player *I* may be identified with a pre-model of γ, while a winning strategy for player *II* can be identified with a refutation of γ. Hence, we have the following:

Proposition 2.2.1 For each formula γ there exists a pre-model of γ or a refutation of γ in any tableau for γ.

2.3 Characterization

In this section we prove that γ is satisfiable iff there exists a pre-model for it. From the results of the previous section, it will follow that γ is true iff there exists refutation for $\neg\gamma$.

It will be convenient to use a characterization of the extremal fixpoints in terms of possibly transfinite induction. We introduce two new constructs $\mu^\tau X.\alpha(X)$ and $\nu^\tau X.\alpha(X)$, where τ is any ordinal, with the following semantics:

— $\| \mu^0 X.\alpha(X) \|_{Val} = \emptyset$, $\| \nu^0 X.\alpha(X) \|_{Val} = S$,

— $\| \sigma^{\tau+1} X.\alpha(X) \|_{Val} = \| \alpha(X) \|_{Val[\|\sigma^\tau X.\alpha(X)\|_{Val}/X]}$ (σ means μ or ν),

— $\| \mu^\tau X.\alpha(X) \|_{Val} = \bigcup_{\tau'<\tau} \| \mu^{\tau'} X.\alpha(X) \|_{Val}$, for τ limit ordinal,

— $\| \nu^\tau X.\alpha(X) \|_{Val} = \bigcap_{\tau'<\tau} \| \nu^{\tau'} X.\alpha(X) \|_{Val}$, for τ limit ordinal.

Then we have:

$$\begin{aligned} \| \mu X.\alpha(X) \|_{Val} &= \bigcup_\tau \| \mu^\tau X.\alpha(X) \|_{Val} \\ \| \nu X.\alpha(X) \|_{Val} &= \bigcap_\tau \| \nu^\tau X.\alpha(X) \|_{Val} \end{aligned}$$

Now we introduce the notion of a signature similar to that considered by Streett and Emerson [53].

Definition 2.3.1 Let us take a formula β, a definition list $\mathcal{D}$ containing all definition constants occurring in β, and a state s of a model $\mathcal{M}$ such that $\mathcal{M}, s \models \langle\!\langle \beta \rangle\!\rangle_{\mathcal{D}}$. We define a *signature* of β in s, $Sig_{\mathcal{D}}(\beta, s)$, as the least, in lexicographical ordering, sequence of ordinals $(\tau_1, \ldots, \tau_{d^\mu})$ such that $\mathcal{M}, s \models \langle\!\langle \beta \rangle\!\rangle_{\mathcal{D}'}$, where $\mathcal{D}'$ is a definition list constructed from $\mathcal{D}$ by replacing the i-th $\mu-$constant definition $(U_i = \mu X.\alpha_i(X)) \in \mathcal{D}$ by $(U_i = \mu^{\tau_i} X.\alpha_i(X))$ for each $i = 1, \ldots, d^\mu$

It can be shown that signatures behave nicely with respect to formula reduction namely:

Lemma 2.3.2 For any state s of a model $\mathcal{M}$:

— If $\mathcal{M}, s \models \langle\!\langle \alpha \wedge \beta \rangle\!\rangle_{\mathcal{D}}$ then $Sig_{\mathcal{D}}(\alpha \wedge \beta, s) = \max(Sig_{\mathcal{D}}(\alpha, s), Sig_{\mathcal{D}}(\beta, s))$.

— If $\mathcal{M}, s \models \langle\!\langle \alpha \vee \beta \rangle\!\rangle_{\mathcal{D}}$ then $Sig_{\mathcal{D}}(\alpha \vee \beta, s) = Sig_{\mathcal{D}}(\alpha, s)$ or $Sig_{\mathcal{D}}(\alpha \vee \beta, s) = Sig_{\mathcal{D}}(\beta, s)$.

— If $\mathcal{M}, s \models \langle\!\langle \langle a \rangle \alpha \rangle\!\rangle_{\mathcal{D}}$ then $Sig_{\mathcal{D}}(\langle a \rangle \alpha, s) = Sig_{\mathcal{D}}(\alpha, t)$ for some state t such that $(s, t) \in R^{\mathcal{M}}(a)$.

— If $\mathcal{M}, s \models \langle\!\langle [a] \alpha \rangle\!\rangle_{\mathcal{D}}$ then $Sig_{\mathcal{D}}([a]\alpha, s) \geq Sig_{\mathcal{D}}(\alpha, t)$ for every state t such that $(s, t) \in R^{\mathcal{M}}(a)$.

— If $\mathcal{M}, s \models \langle\!\langle \nu X.\alpha(X) \rangle\!\rangle_{\mathcal{D}}$ and $(V = \nu X.\alpha(X)) \in \mathcal{D}$ then $Sig_{\mathcal{D}}(\nu X.\alpha(X), s) = Sig_{\mathcal{D}}(V, s)$.

— If $\mathcal{M}, s \models \langle\!\langle \mu X.\alpha(X) \rangle\!\rangle_{\mathcal{D}}$ and $(U_i = \mu X.\alpha(X)) \in \mathcal{D}$ is the i-th μ-constant in $\mathcal{D}$ then the prefixes of length $i - 1$ of $Sig_{\mathcal{D}}(\mu X.\alpha(X), s)$ and $Sig_{\mathcal{D}}(U_i, s)$ are equal.

— If $\mathcal{M}, s \models \langle\!\langle W \rangle\!\rangle_{\mathcal{D}}$ and $(W = \sigma X.\alpha(X)) \in \mathcal{D}$ then $Sig_{\mathcal{D}}(W, s) = Sig_{\mathcal{D}}(\alpha(W), s)$ if W is a ν constant. If W is the i-th μ-constant in $\mathcal{D}$ then the second signature is smaller and they differ at position i or smaller.

Proof
We will consider only the last case. Suppose $\mathcal{M}, s \models \langle\!\langle U_i \rangle\!\rangle_{\mathcal{D}}$, where U_i is the i-th definition constant from $\mathcal{D}$. Let $Sig_{\mathcal{D}}(U_i, s) = (\tau_1, \ldots, \tau_n)$ and $\mathcal{D}'$ be a definition list obtained from $\mathcal{D}$ by replacing j-th μ-constant definition $(U_j = \mu X.\alpha_j(X)) \in \mathcal{D}$ by $(U_j = \mu^{\tau_j} X.\alpha_j(X))$ for every $j = 1, \ldots, n$. Please note that only definition constants older then U_i can appear in $\alpha_i(X)$. Let us denote $\beta(X) = \langle\!\langle \alpha_i(X) \rangle\!\rangle_{\mathcal{D}'}$. From the definition of the signature we have

$\mathcal{M}, s \models \mu^{\tau_i} X.\beta(X)$. Observe that τ_i must be a successor ordinal hence $\mathcal{M}, s \models \beta(\mu^{\tau_i - 1} X.\beta(X))$ which implies the thesis of the lemma $\square$

Proposition 2.3.3 If a positive guarded sentence γ is satisfiable then any tableau for γ contains a pre-model for γ as its subtree.

Proof
Let us take a sentence γ and a model $\mathcal{M} =< S^{\mathcal{M}}, R^{\mathcal{M}}, \rho^{\mathcal{M}} >$ in which γ is satisfiable. Let $\mathcal{D} = \rrbracket\gamma\llbracket = (W_1 = \gamma_1), \ldots, (W_d = \gamma_d)$. Let $(U_1 = \mu X.\alpha_1(X)), \ldots, (U_{d^\mu} = \mu X.\alpha_{d^\mu}(X))$ be a subsequence of μ definitions in $\mathcal{D}$.

Given a tableau $\mathcal{T}$ for γ we will construct a pre-model $\mathcal{PM} = \langle K, L\rangle$ as a subtree of $\mathcal{T}$. Starting from the root of $\mathcal{T}$ we will subsequently select the nodes of $\mathcal{T}$ that will be included in $\mathcal{PM}$. With every node n, of $\mathcal{PM}$ under construction, we will associate a state s_n of $\mathcal{M}$ such that $\mathcal{M}, s_n \models \langle\!\langle L(n)\rangle\!\rangle$.

The root of $\mathcal{T}$ becomes the root of $\mathcal{PM}$ and for the associated state we choose any state of $\mathcal{M}$ in which the formula γ is satisfied.

Suppose that we have already selected a node n of $\mathcal{PM}$ with an associated state s_n. We will show how to proceed from this point depending on what rule was used in node n of $\mathcal{T}$:

1. If the (or) rule was applied to $L(n) = \alpha \vee \beta, \Gamma \vdash_{\mathcal{D}}$ then select the left son of n if $Sig_{\mathcal{D}}(\alpha, s_n) \leq Sig_{\mathcal{D}}(\alpha \vee \beta, s_n)$ and the right son otherwise. Associate the state s_n with the chosen node.

2. If the $(all\langle\rangle)$ rule was applied then for any son n' of n, there is a formula of the form $\langle a\rangle\alpha$ which reduction resulted in the label of n'. For the node n' choose a state t such that $(s_n, t) \in R(a)$ and $Sig_{\mathcal{D}}(\langle a\rangle\alpha, s_n) \geq Sig_{\mathcal{D}}(\alpha, t)$.

3. For all other rules, just take the only son of n in $\mathcal{T}$ as the next node of $\mathcal{PM}$ and associate the state s_n with it.

We show that $\mathcal{PM}$ constructed in this way is indeed a pre-model for γ.

It is easy to see that a leaf of $\mathcal{PM}$ cannot be labeled by a tableau axiom because for every leaf n we have that $\langle\!\langle L(n)\rangle\!\rangle$ is satisfiable and obviously $p \wedge \neg p$ cannot be satisfied.

It remains to show that any infinite trace $T = \{\alpha_n\}_{n \in P}$ on any path P is a ν-trace. Suppose that we can find a trace T such that the oldest constant regenerated infinitely often on it is some i-th μ-constant U_i. Clearly, after some point n_0, U_i must be the oldest constant which will regenerate on the trace.

Observe that Lemma 2.3.2 implies that from the point n_0, prefix of length i of signatures of formulas in T never increases. The only way the prefix could increase without regeneration of a definition constant older than U_i is an application of the rule (μ) with a constant older than U_i. But then the next reduction on the trace would be a regeneration of the constant older than U_i.

Lemma 2.3.2 also says that the prefix actually decreases after each regeneration of U_i. Since we have assumed that U_i is regenerated infinitely often we obtain a contradiction, because the lexicographical ordering on sequences of bounded length over a well ordering is also a well ordering. This shows that every trace of $\mathcal{PM}$ is a ν-trace, hence $\mathcal{PM}$ is a pre-model. $\square$

Now we will show implication in the other direction, i.e., given a pre-model for γ, how to construct a structure where γ is satisfied.

Definition 2.3.4 Given a pre-model $\mathcal{PM}$, the *canonical structure* for $\mathcal{PM}$ is a structure $\mathcal{M} = \langle S^{\mathcal{M}}, R^{\mathcal{M}}, \rho^{\mathcal{M}} \rangle$ such that:

1. $S^{\mathcal{M}}$ is the set of all nodes of $\mathcal{PM}$ which are either leaves or to which $(all\langle\rangle)$ rule was applied. For any node n of $\mathcal{PM}$ we will denote by s_n the closest descendant of n belonging to $S^{\mathcal{M}}$.

2. $(s, s') \in R^{\mathcal{M}}(a)$ iff there is a son n of s with $s_n = s'$, such that $L(n)$ was obtained from $L(s)$ by reducing a formula of the form $\langle a\rangle\alpha$.

3. $\rho^{\mathcal{M}}(p) = \{s : \text{p occurs in the sequent Ł}(s)\}$.

Proposition 2.3.5 If there exists a pre-model $\mathcal{PM}$ for a positive guarded sentence γ then γ is satisfiable in the canonical structure for $\mathcal{PM}$.

Proof
Let $\mathcal{PM} = \langle K, L\rangle$ be a pre-model of a sentence γ. Let $\mathcal{M} =< S^{\mathcal{M}}, R^{\mathcal{M}}, \rho^{\mathcal{M}} >$ be the canonical structure for $\mathcal{PM}$. Let $\mathcal{D} = \rrbracket\gamma\llbracket = (W_1 = \gamma_1), \ldots, (W_d = \gamma_g)$ be the definition list and let $(V_1 = \nu X.\beta_1(X)), \ldots, (V_{d^\nu} = \nu X.\beta_{d^\nu}(X))$ be a sublist of ν-definitions from $\mathcal{D}$.

Suppose that $\mathcal{M}, s_{n_0} \not\models \gamma$, where n_0 is the root of $\mathcal{PM}$. From now on we will proceed in a similar way to the previous proof.

First, for a formula α and state s s.t. $\mathcal{M}, s \not\models \alpha$ we define a ν-signature, $Sig^{\nu}_{\mathcal{D}}(\alpha, s)$ to be the smallest sequence of ordinals $(\tau_1, \ldots, \tau_{d^\nu})$ such that when we take definition list $\mathcal{D}'$ obtained from $\mathcal{D}$ by replacing $\nu X.\beta_i(X)$ by $\nu^{\tau_i} X.\beta_i(X)$, then $\mathcal{M}, s \not\models \langle\!\lbrack \alpha \rbrack\!\rangle_{\mathcal{D}'}$ still holds. For ν-signatures we can prove an analog of Lemma 2.3.2 but with interchanging: μ with ν, $\langle\rangle$ with $[]$ and conjunction with disjunction.

Next we show that from the assumption $\mathcal{M}, s_{n_0} \not\models \gamma$, it follows that we can construct a μ-trace on some path P of $\mathcal{PM}$, which contradicts the assumption that $\mathcal{PM}$ is a pre-model of γ.

The first element of a trace $T = \{\alpha_n\}_{n\in P}$ will be of course $\alpha_{n_0} = \gamma$. Now suppose that we have constructed T up to the element $\alpha_m \in L(m)$, such that $\mathcal{M}, s_m \not\models \langle\!\lbrack \alpha_m \rbrack\!\rangle$. We proceed as follows:

— if a rule different from $(all\langle\rangle)$ was applied to $L(m)$ then there is only one son m' of m and:

- if α_m wasn't reduced by this rule then $\alpha_{m'} = \alpha_m$,
- if $\alpha_m = \varphi \wedge \psi$ was reduced then choose $\alpha_{m'} = \varphi$ if $Sig^{\nu}_{\mathcal{D}}(\varphi \wedge \psi, s_m) \geq Sig^{\nu}_{\mathcal{D}}(\varphi, s_m)$, else choose $\alpha_{m'} = \psi$ as the next element of the sequence,
- if $\alpha_m = \varphi \vee \psi$ then choose $\alpha_{m'}$ to be the formula which occurs in $L(m')$,
- in other subcases just take the resulting formula as the next element of the trace,

— if $(all\langle\rangle)$ rule was applied then:

- if $\alpha_m = \langle a\rangle\varphi$ then there is a son m' of m the label of which was obtained by reducing this formula. Take $\alpha_{m'} = \varphi$
- if $\alpha_m = [a]\varphi$ then, because $\sigma_m \not\models \alpha_m$, there exists a state t such that $(s_m, t) \in R^{\mathcal{M}}(a)$ and $Sig^{\nu}_{\mathcal{D}}(\varphi, t) \leq Sig^{\nu}_{\mathcal{D}}([a]\varphi, s)$. Observe that from the definition of our structure this means that there is some son m' of m, such that $\varphi \in L(m')$ and $s_{m'} = t$. So let us take $\alpha_{m'} = \varphi$.

If the constructed trace were finite then from the definition of the canonical structure and restrictions on the application of the rule $(all\langle\rangle)$ it follows that the last element of the trace α_m, must be some propositional constant p, its negation, or m can be a leaf of $\mathcal{PM}$ and α_m a formula of the form $[a]\psi$. Then from the definition of the canonical structure we have that $\mathcal{M}, s_m \models \alpha_m$, contradiction.

Using an argument about signatures similar to the one from Proposition 2.3.5, we can show that the oldest constant regenerated infinitely often on T must be a μ-constant. But $\mathcal{PM}$ is a pre-model hence all its traces are ν-traces, contradiction. □

It should be clear that Propositions 2.3.3 and 2.3.5 hold not only for positive guarded sentences but for all positive guarded formulas. Putting them together we obtain:

Theorem 2.3.6 There exists pre-model of a positive guarded formula γ iff γ is satisfiable.

Finally using our results from previous section we obtain also a characterization of valid formulas.

Theorem 2.3.7 Positive guarded formula $\neg\gamma$ is valid iff there exits a refutation for γ.

Proof
Consider any tableau $\mathcal{T}$ for γ. Since there is no model for γ, we cannot find a pre-model in $\mathcal{T}$ hence from Proposition 2.2.1 there is a refutation in $\mathcal{T}$.

In other direction if $\mathcal{P}$ is a refutation for γ then we know that in the tableau $\mathcal{T}$, of which $\mathcal{P}$ is a subtree, we cannot find a pre-model. Hence from Proposition 2.3.3 it follows that γ is not satisfiable and $\neg\gamma$ is valid. □

2.4 Complexity

In this section we give bounds on the size of a refutation of a formula and consider the computational complexity of the problem for finding a refutation.

Emerson and Jutla [8] gave the exact bound of the decidability problem for the μ-calculus, by showing that the satisfiability problem for the μ-calculus is decidable in deterministic exponential time. The completeness of the problem for this complexity class follows from the lower bound for PDL due to Fischer and Ladner(1979) [13]. Streett and Emerson [53] also show a *small model theorem* for the propositional μ-calculus which, combined with the later results of Emerson and Jutla [8] tells that *if a sentence has a model then it has a finite model of the exponential size in the size of the sentence.* Considering the proof method used in [8], we can restate this result as follows.

Theorem 2.4.1 (Emerson and Jutla) *There is an algorithm which decides if a μ-calculus sentence γ is satisfiable in time exp$(|\gamma|)$. If this is the case, the algorithm constructs a model of the size exp$(|\gamma|)$ in time exp$(|\gamma|)$.*

The key fact needed for this theorem is a result on complexity of Rabin tree automata that is also shown in [8].

Theorem 2.4.2 (Emerson and Jutla) *There is an algorithm which, given a Rabin automaton with n states and m pairs decides if the automaton accepts any tree in time $\mathcal{O}(n^m)$. If it is the case, the algorithm constructs in time $\mathcal{O}(n^m)$ a regular tree of the size $\mathcal{O}(n)$ accepted by the automaton.*

This last result can be also used for obtaining the upper bound for the size (and the constructing time) of the refutations.

Let a μ-calculus sentence γ be given. Since the rules of the system S are nondeterministic, there may be many tableaux of γ. Clearly, we can construct a tableau $\mathcal{T}$ of γ, which is a *regular tree* of size $\exp(|\gamma|)$, and the construction can be performed in time $\exp(|\gamma|)$. Let Σ be the alphabet consisting of the sequents labeling the nodes of $\mathcal{T}$. Observe that $|\Sigma| = \exp(|\gamma|)$. We construct a Büchi automaton on infinite words over Σ which nondeterministically chooses a trace from a path in $\mathcal{T}$ and accepts iff this is a μ-trace. It is easy to see that $\mathcal{O}(|\gamma|)$ states are sufficient for this job. Applying the Safra determinization construction [45], we get a Rabin automaton on ω-words with $\exp(|\gamma|)$ states and $\mathcal{O}(|\gamma|)$ pairs, and finally, following the construction of [53], we obtain a Rabin tree automaton with $\exp(|\gamma|)$ states and $\mathcal{O}(|\gamma|)$ pairs, which accepts precisely the refutations of γ. We can therefore state the following.

Theorem 2.4.3 There is an algorithm which, given a μ-calculus sentence γ, constructs a model of size $\exp(|\gamma|)$ or a refutation of size $\exp(|\gamma|)$. The algorithm runs in time $\exp(|\gamma|)$.

Chapter 3

Axiomatization

This chapter is divided into two sections. In the first we present a finitary axiom system for the μ-calculus. This system will be shown complete in the next chapter. In the second section we will relate the presented system to the Kozen's system proposed in the paper introducing μ-calculus [24].

3.1 The system

In this section we present a finitary deduction system for the μ-calculus. We have chosen sequent calculus presentation because it corresponds nicely with the tableau rules used to obtain the characterization of the validity.

A *sequent* is a pair of finite sets of formulas and it will be denoted $\Gamma \vdash \Delta$. If $\Gamma \cap \Delta \neq \emptyset$ then the sequent $\Gamma \vdash \Delta$ is called *axiom*.

In what follows it will be sometimes convenient to use PDL syntax. We will sometimes write $\langle P \rangle \alpha$ or $[P]\alpha$, where α is a formula of μ-calculus and P is a PDL program, i.e., a regular expression over the set of actions. The meaning of such a formula is given by the usual translation of PDL formulas into μ-calculus as presented in the section 1.2.1.

Our system consists of two groups of rules and some additional axioms. First we take the rules of the simple propositional modal logic.

$$(\neg) \qquad \frac{\Gamma \vdash \Delta, \alpha}{\Gamma, \neg\alpha \vdash \Delta} \quad \frac{\Gamma, \alpha \vdash \Delta}{\Gamma \vdash \neg\alpha, \Delta}$$

$$(\wedge) \qquad \frac{\alpha, \beta, \Gamma \vdash \Delta}{\alpha \wedge \beta, \Gamma \vdash \Delta} \quad \frac{\Gamma \vdash \alpha, \Delta \quad \Gamma \vdash \beta, \Delta}{\Gamma \vdash \alpha \wedge \beta, \Delta}$$

$$(\vee) \qquad \frac{\alpha, \Gamma \vdash \Delta \quad \beta, \Gamma \vdash \Delta}{\alpha \vee \beta, \Gamma \vdash \Delta} \quad \frac{\Gamma \vdash \alpha, \beta, \Delta}{\Gamma \vdash \alpha \vee \beta, \Delta}$$

$$(\langle\rangle) \qquad \frac{\alpha, \{\beta : [a]\beta \in \Gamma\} \vdash \{\gamma : \langle a\rangle\gamma \in \Delta\}}{\langle a\rangle\alpha, \Gamma \vdash \Delta}$$

$$(cut) \qquad \frac{\Gamma \vdash \Delta, \gamma \quad \Sigma, \gamma \vdash \Omega}{\Gamma, \Sigma \vdash \Delta, \Omega}$$

Then we add the rules concerning fixpoints

$$(\mu) \qquad \frac{\Gamma \vdash \alpha(\mu X.\alpha(X)), \Delta}{\Gamma \vdash \mu X.\alpha(X), \Delta}$$

$$(ind) \qquad \frac{\varphi(f\!f) \vdash \Delta \quad \varphi(\alpha(\mu X.Z \wedge \alpha(X))) \vdash \Delta, \langle P\rangle\varphi(\mu X.Z \wedge \alpha(X))}{\varphi(\mu X.\alpha(X)) \vdash \langle P^*\rangle\Delta}$$

$$Z \notin FV(\varphi(\mu X.\alpha(X)), \Delta)$$

In the last rule P is a PDL program, $\langle P\rangle\Delta = \{\langle P\rangle\delta : \delta \in \Delta\}$ and Z is a new propositional variable not occurring free in $\varphi(\mu X.\alpha(X)) \vdash \Delta$.

In the above notation $\varphi(\mu X.\alpha(X))$ stands for the result of substitution $\varphi[\mu X.\alpha(X)/[]]$ where $[]$ is a special variable. The substitution $\varphi[\alpha/X]$ is legal only if no free variable of α becomes bound in $\varphi[\alpha/X]$.

Finally because we had chosen to include constructions $[a]\alpha$ and $\nu X.\alpha(X)$ into the language we have to define them using other connectives by adding the following axioms:

$$([]L) \qquad [a]\alpha \vdash \neg\langle a\rangle\neg\alpha \qquad ([]R) \qquad \langle a\rangle\alpha \vdash \neg[a]\neg\alpha$$

$$(\nu L) \quad \nu X.\alpha(X) \vdash \neg\mu X.\neg\alpha(\neg X) \quad (\nu R) \quad \mu X.\alpha(X) \vdash \neg\nu X.\neg\alpha(\neg X)$$

Definition 3.1.1 A finite tree constructed with the use of the above rules will be called *diagram*. A *proof* will be a diagram whose all leaves are labeled with axioms i.e., instances of the axioms above or sequents with the same formula on both sides. We consider *ff* as an abbreviation of $p \wedge \neg p$ for some propositional constant p.

Observe the difference between diagrams and tableaux. The first are always finite while the second not necessary so. They are also constructed using different, although similar, sets of rules.

The rule (*ind*) is a stronger version of the induction rule:

$$\frac{\varphi(\mathit{ff}) \vdash \Delta \quad \varphi(\alpha(Z)) \vdash \Delta, \langle P \rangle \varphi(Z)}{\varphi(\mu X.\alpha(X)) \vdash \langle P^* \rangle \Delta} \tag{3.1}$$

where Z is a new variable. Rule (*ind*) is stronger because the assumptions of (*ind*) rule are weaker then those of (3.1). It turns out that the rule (3.1) is too weak to make our completeness proof work. The reason is that putting variable Z in place of $\mu X.\alpha(X)$ we "forget" that Z is an approximation of $\mu X.\alpha(X)$. On the other hand, as the next lemma shows $\mu X.Z \wedge \alpha(X)$ behaves very much the same as the variable Z alone.

The presence of modalities $\langle P \rangle$ and $\langle P^* \rangle$, in the (*ind*) rule should be also explained. This construction works like existential quantifier on states. To see this let us look at a formula of the form $\langle P^* \rangle \alpha$, where $P = a_1 \cup \ldots \cup a_n$ and $a_1, \ldots, a_n$ are all actions occurring in α. The meaning of $\langle P^* \rangle \alpha$ is roughly the same as the statement that there exists a reachable state satisfying α.

Lemma 3.1.2 For any structure $\mathcal{M}$ and valuation Val such that $Val(Z) = \| \alpha^k(\mathit{ff}) \|_{Val}$ for some $k \in \mathcal{N}$ and $Z \notin FV(\alpha(\mathit{ff}))$,

$$\| \mu X.Z \wedge \alpha(X) \|_{Val} = Val(Z)$$

Proof
Let $Val(Z) = \| \alpha^k(\mathit{ff}) \|_{Val}$ then

$$\| \mu X.Z \wedge \alpha(X) \|_{Val} = \| \mu X.\alpha^k(\mathit{ff}) \wedge \alpha(X) \|_{Val} = \| \alpha^k(\mathit{ff}) \|_{Val}$$

$\square$

The next proposition states soundness of the system. There are at least three possible notions of soundness of a rule in a modal logic:

- The strongest will be to require that if the assumptions of a rule are satisfied in a state of a structure then the conclusion must be satisfied in this state. It is easy to see that rules (*neg*), (*and*), (*or*), (*cut*) and (μ) are sound in this sense.

- The other would be to require that, for any structure $\mathcal{M}$, if all assumptions of the rule are valid in a structure (i.e. satisfied in every state) then the conclusion must be valid in this structure. The rule ($\langle\rangle$) is sound in this sense. We will show in the next section that all the rules of the Kozen's axiomatization of the μ-calculus are also sound in this sense.

- The weakest of the three is the condition that if all assumptions of the rule are valid then the conclusion is valid. The rule (*ind*) is sound in this sense and as we will show in the next section it is not sound in the sense of any of the two preceding notions of soundness. Of course this type of soundness is all that we need to require from the axiomatization of the validity relation. In the following by soundness we will always understand this weak notion of soundness.

Proposition 3.1.3 All rules of our system are sound and all axioms are valid

Proof
Validity of the axioms is standard. Also standard is validity of all the rules except (*ind*) which soundness we prove below.

Let us assume conversely that the rule (*ind*) is not sound. Then from the finite model property we have a finite structure $\mathcal{M}$ such that the sequents:

$$\varphi(\mathit{ff}) \vdash \Delta \quad \varphi(\alpha(\mu X.Z \wedge \alpha(X))) \vdash \Delta, \langle P\rangle\varphi(\mu X.Z \wedge \alpha(X))$$

are valid in $\mathcal{M}$ and $\varphi(\mu X.\alpha(X)) \vdash \langle P^*\rangle\Delta$ is not. We assume that the variable Z does not appear free in $\varphi(\mu X.\alpha(X))$ or Δ. Let k be the smallest integer s.t. there exists a state s of $\mathcal{M}$ and valuation Val s.t. $\mathcal{M}, s, Val \models \varphi(\alpha^k(\mathit{ff}))$ and for all $\delta \in \Delta$, $\mathcal{M}, s, Val \not\models \langle P^*\rangle\delta$.

We have two cases depending on whether $k = 0$ or not.

— If $k = 0$ then $\mathcal{M}, s, Val \models \varphi(\mathit{ff})$ hence from the assumption that $\varphi(\mathit{ff}) \vdash \Delta$ is valid in $\mathcal{M}$ we have that $\mathcal{M}, s, Val \models \delta$ for some $\delta \in \Delta$, contradiction.

— If $k > 0$ then let Val' be identical to Val except that for the variable Z, $Val'(Z) = \| \alpha^{k-1}(\mathit{ff}) \|_{Val}$. Hence $\mathcal{M}, s, Val' \models \varphi(\alpha(Z))$ and from lemma 3.1.2 it follows that $\mathcal{M}, s, Val' \models \varphi(\alpha(\mu X.Z \wedge \alpha(X)))$. From the validity of the second premise we have that $\mathcal{M}, t, Val' \models \varphi(\mu X.Z \wedge \alpha(X))$ for some state t reachable from s by P or $\mathcal{M}, s, Val \models \delta$ for some $\delta \in \Delta$. The second case is impossible by our assumption.

If $\mathcal{M}, t, Val' \models \varphi(\mu X.Z \wedge \alpha(X))$ then by lemma 3.1.2 $\mathcal{M}, t, Val' \models \varphi(Z)$, hence $\mathcal{M}, t, Val \models \varphi(\alpha^{k-1}(\mathit{ff}))$. Because k was chosen the smallest possible there must exist $\delta \in \Delta$ s.t. $\mathcal{M}, t, Val \models \langle P^* \rangle \delta$. But then because t is reachable from s by program P we have also that $\mathcal{M}, s, Val \models \langle P^* \rangle \delta$. Contradiction.

$\square$

This is maybe the place to comment on the inclusion of the cut rule in our system. Since the rule (ind) has the specific shape of reducing connective which can be "hidden" deep into the formula, the addition of the cut rule seems to be a reasonable way to make the system usable. The completeness proof will give us an algorithm for constructing a proof of a valid formula and, as we will see this proof will have some kind of subformula property even though cuts will be used in it. The conclusion is that cuts in the proof can be used in a very restricted way and one can substitute them by appropriate rules which may be added to the system. Addition of the cut rule makes the system simpler and since we have an algorithm for automatic construction of a proof of a valid sequent we prefer the presented formulation.

The next step is to show the following useful fact which will allow substituting equals for equals:

Fact 3.1.4 For any formula $\psi(X_1, \dots, X_k)$ with $X_1, \dots, X_k$ occurring positively in ψ and provable sequents $\delta_i \vdash \gamma_i$ for each $i = 1, \dots, k$, the sequent $\psi(\delta_1, \dots, \delta_k) \vdash \psi(\gamma_1, \dots, \gamma_k)$ is provable.

This will follow from

Lemma 3.1.5 Let $\vec{X}$ and $\vec{Y}$ denote the sequences of variables $X_1, \dots, X_n$ and $Y_1, \dots, Y_n$ respectively. Let $\varphi(\vec{X})$ be a formula with free variables $\vec{X}$ each occurring only positively or negatively in it, let $a_1, \dots, a_i$ be all actions occurring in $\varphi(\vec{X})$ and let $P = a_1 \cup \dots \cup a_i$. For any finite set of formulas Δ there is a diagram of $\varphi(\vec{X}) \vdash \varphi(\vec{Y}), \langle P^* \rangle \Delta$ in which the only leaves which are not axioms are of the form $X_i \vdash Y_i, \langle P^* \rangle \Delta$, if the variable X_i occurs only positively, and $Y_i \vdash X_i, \langle P^* \rangle \Delta$, if the variable X_i occurs only negatively in $\varphi(X_1, \dots, X_n)$.

Proof
Proof proceeds by induction on the structure of φ.

— if φ is one of the free variables or some propositional constant then the lemma is obvious.

— if $\varphi = \neg\psi$ then the application of the derivable rule

$$\frac{\psi(\vec{Y}) \vdash \psi(\vec{X}), \langle P^*\rangle\Delta}{\neg\psi(\vec{X}) \vdash \neg\psi(\vec{Y}), \langle P^*\rangle\Delta}$$

gives us the desired conclusion.

— if $\varphi = \psi_1 \wedge \psi_2$ then we use the derivable rule

$$\frac{\psi_1(\vec{X}) \vdash \psi_1(\vec{Y}), \langle P^*\rangle\Delta \quad \psi_2(\vec{X}) \vdash \psi_2(\vec{Y}), \langle P^*\rangle\Delta}{\psi_1(\vec{X}) \wedge \psi_2(\vec{X}) \vdash \psi_1(\vec{Y}) \wedge \psi_2(\vec{Y}), \langle P^*\rangle\Delta}$$

— if $\varphi = \langle a\rangle\psi$ then the rule to use is $(\langle\rangle)$

$$\frac{\psi(\vec{X}) \vdash \psi(\vec{Y}), \langle P^*\rangle\Delta}{\langle a\rangle\psi(\vec{X}) \vdash \langle a\rangle\psi(\vec{Y}), \langle P^*\rangle\Delta}$$

— if $\varphi = \mu V.\beta(V)$ then we use rule (ind) to the sequent

$$\mu V.\beta(V, \vec{X}), \nu V.\neg\beta(\neg V, \vec{Y}) \vdash \langle P^*\rangle\Delta \tag{3.2}$$

from which the sequent

$$\mu V.\beta(V, \vec{X}) \vdash \mu V.\beta(V, \vec{Y}), \langle P^*\rangle\Delta$$

is obtainable by the (cut) rule. The sequents:

$$ff, \nu V.\neg\beta(\neg V, \vec{Y}) \vdash \langle P^*\rangle\Delta$$

$$\beta(\mu V.Z \wedge \beta(V, \vec{X}), \vec{X}), \nu V.\neg\beta(\neg V, \vec{Y}) \vdash \\ \langle P^*\rangle((\mu V.Z \wedge \beta(V, \vec{X})) \wedge \nu V.\neg\beta(\neg V, \vec{Y})), \langle P^*\rangle\Delta$$

are assumptions of the instance of (ind) rule in which (3.2) is the conclusion. Clearly $ff, \nu V.\beta(V, \vec{Y}) \vdash \langle P^*\rangle\Delta$ is provable and from the induction hypothesis we have an appropriate diagram for

$$\beta(\mu V.Z \wedge \beta(V, \vec{X}), \vec{X}) \vdash \\ \beta(\mu V.\beta(V, \vec{Y}), \vec{Y}), \langle P^*\rangle((\mu V.Z \wedge \beta(V, \vec{X})) \wedge \nu V.\neg\beta(\neg V, \vec{Y})), \langle P^*\rangle\Delta \tag{3.3}$$

The only leaves in that diagram which are not labeled by axioms are

$$\mu V.Z \wedge \beta(V, \vec{X}) \vdash \mu V.\beta(V, \vec{Y}), \langle P^*\rangle((\mu V.Z \wedge \beta(V, \vec{X})) \wedge \nu V.\neg\beta(\neg V, \vec{Y})), \langle P^*\rangle\Delta$$

This sequent is clearly provable. Hence sequent 3.3 is provable. Observe that sequent 3.3 is equivalent to the second assumption of the instance of (ind) rule.

— in cases when $\varphi = \psi_1 \vee \psi_2$, $\varphi = [a]\psi$ or $\varphi = \nu V.\beta(V)$ we can just use the fact that these connectives are defined by the connectives already considered.

$\square$

Proposition 3.1.6 Any formula is provably equivalent to some positive guarded formula

Proof
From Fact 3.1.4 it follows that it is enough to consider a formula of the form $\mu X.\alpha(X)$ where $\alpha(X)$ is already positive guarded formula. Then, following the steps of the proof of proposition 1.1.2 we obtain a formula $\mu X.(X \vee \bar{\alpha}(X)) \wedge \beta(X)$ which is provably equivalent to $\mu X.\alpha(X)$ and where all occurrences of X in $\bar{\alpha}(X)$ and $\beta(X)$ are guarded.

It is enough to show that $\mu X.(X \vee \bar{\alpha}(X)) \wedge \beta(X)$ is provably equivalent to $\mu X.\bar{\alpha}(X) \wedge \beta(X)$. This can done in quite straightforward way using Lemma 3.1.5.

$\square$

3.2 Discussion

Here we would like to consider some properties of the presented axiomatization and relate it to the Kozen's axiom system from the paper introducing the μ-calculus [24]. Let us call this system $K\mu$ here.

The system $K\mu$ uses a different form of judgment. It has the form of equality $\varphi = \psi$ with the meaning that the formulas φ and ψ are semantically equivalent. Inequality $\varphi \leq \psi$ is considered as an abbreviation of $\varphi \vee \psi = \psi$.

Apart from the axioms and rules of equational logic (including substitution of equals by equals) there are the following axioms and rules:

$$
\begin{array}{ll}
(K1) & \text{axioms for Boolean algebra} \\[1em]
(K2) & \langle a\rangle\varphi \vee \langle a\rangle\psi = \langle a\rangle(\varphi \vee \psi) \\[1em]
(K3) & \langle a\rangle\varphi \wedge [a]\psi \leq \langle a\rangle(\varphi \wedge \psi) \\[1em]
(K4) & \langle a\rangle\mathit{ff} = \mathit{ff} \\[1em]
(K5) & \alpha(\mu X.\alpha(X)) \leq \mu X.\alpha(X) \\[1em]
(K6) & \dfrac{\alpha(\varphi) \leq \varphi}{\mu X.\alpha(X) \leq \varphi}
\end{array}
$$

Our sequent $\Gamma \vdash \Delta$ corresponds to inequality $\bigwedge \Gamma \leq \bigvee \Delta$ in this notation. We will call a rule *derivable* in the system $K\mu$ iff there is a way to derive the conclusion of the rule, assuming all the premises of the rule.

Proposition 3.2.1 Any rule derivable in the system $K\mu$ must have the property that for any structure $\mathcal{M}$:

> if all the premises of the rule are true in $\mathcal{M}$ then the conclusion is true in $\mathcal{M}$

The proposition follows directly from the observation that all the rules of $K\mu$ have this property. We will show that the rule (*ind*) of our system does not have this property hence it cannot be derived in $K\mu$.

Proposition 3.2.2 There is a structure $\mathcal{M}$ and an instance of the (*ind*) rule such that the premises are true in $\mathcal{M}$ but the conclusion is not.

Proof
Consider a structure $\mathcal{M} = \langle S, R, \rho\rangle$ such that:

- $S = \mathcal{N} \cup \{\infty\}$
- $R(a) = \{(i+1, i) : i \in \mathcal{N}\} \cup \{(\infty, i) : i \in \mathcal{N}\}$
- $\rho(p) = \{\infty\}$

Here a is some action, p some propositional constant and $\mathcal{N}$ the set of natural numbers.

Consider a sequent:

$$p \wedge [a]\mu X.[a]X \vdash \tag{3.4}$$

which is not valid in $\mathcal{M}$ because $\mathcal{M}, \infty \models p \wedge [a]\mu X.[a]X$. If we were to prove the sequent 3.4 we could use rule (ind) directly and have the premises:

$$p, [a]\mathit{ff} \vdash \tag{3.5}$$

$$p, [a]([a]\mu X.Z \wedge [a]X) \vdash p \wedge [a]\mu X.Z \wedge [a]X \tag{3.6}$$

Clearly sequent 3.5 is true in $\mathcal{M}$. To see why sequent 3.6 is true in $\mathcal{M}$ suppose that for some valuation Val we have $\mathcal{M}, \infty, Val \models p \wedge [a][a](\mu X.Z \wedge [a]X)$ then clearly $\mathcal{N} \subseteq Val(Z)$ hence also by lemma 3.1.2 $\mathcal{N} \subseteq \| \mu X.Z \wedge \alpha(X) \|_{Val}$ which means that $\mathcal{M}, \infty, Val \models p \wedge [a](\mu X.Z \wedge [a]X)$.

□

The above proof also shows why we have called the rule, induction rule. Of course Proposition 3.2.2 does not imply incompleteness of the system $K\mu$ even though the rules of $K\mu$ are derivable in our system.

Proposition 3.2.3 All axioms of the system $K\mu$ are provable in our system. All rules of $K\mu$ are derivable in our system

Proof
Provability of the axioms is very easy, so is also deriveability of the rules other then $(K6)$ and substitution of equals by equals. This second rule is derivable by the fact 3.1.4.

For the rule $(K6)$ let us assume $\alpha(\varphi) \vdash \varphi$ and we will show how to prove $\neg\varphi \wedge \mu X.\alpha(X) \vdash$. Let $P = a_1 \cup \ldots \cup a_i$ where $\{a_1, \ldots, a_i\}$ is the set of all actions occurring in $\alpha(\varphi)$.

To obtain $\neg\varphi \wedge \mu X.\alpha(X) \vdash$ we can use induction rule if we prove

$$\mathit{ff} \wedge \neg\varphi \vdash \quad \text{and} \quad \neg\varphi \wedge \alpha(\mu X.Z \wedge \alpha(X)) \vdash \langle P^* \rangle \neg\varphi \wedge \mu X.Z \wedge \alpha(X)$$

The first sequent is clearly provable. Using assumption that $\alpha(\varphi) \vdash \varphi$ we will have a proof of the second sequent if we only prove:

$$\neg\alpha(\varphi) \wedge \alpha(\mu X.Z \wedge \alpha(X)) \vdash \langle P^* \rangle \neg\varphi \wedge \mu X.Z \wedge \alpha(X)$$

This in turn is equivalent to proving

$$\alpha(\mu X.Z \wedge \alpha(X)) \vdash \alpha(\varphi), \langle P^* \rangle \neg\varphi \wedge \mu X.Z \wedge \alpha(X)$$

By Lemma 3.1.5 there is a diagram for this sequent in which the only leaves which are not axioms are of the form

$$\mu X.Z \wedge \alpha(X) \vdash \varphi, \langle P^* \rangle \neg\varphi \wedge \mu X.Z \wedge \alpha(X)$$

But such sequent is clearly provable.

$\square$

Chapter 4

Proof of Completeness

Let us fix an unsatisfiable positive guarded formula φ_0 of the μ-calculus. In this chapter we will show how to construct a proof of the sequent $\varphi_0 \vdash$ in our system. We can consider only positive guarded formulas because by Proposition 3.1.6 any formula is provably equivalent to some positive guarded formula.

From Theorem 2.3.7 we know that there is a refutation of φ_0. Because the set of refutations of a formula is recognizable by a Rabin automaton, there is a regular refutation. This refutation looks almost like a proof except that it can have some loops.

The difficulty we face is as follows. If we take an infinite path of such a refutation then we know that there is a μ-trace on it, i.e. there is an occurrence of a μ-formula which regenerates i.o. and never disappears. It was already shown by Kozen [24] how to use some derivable rules of his system in such a clever way that it is possible to "cut" such a path, i.e., arrive at the axiom sequent after a finite number of steps. We can of course apply such a cutting strategy to each path of a refutation. The problem is that we will obtain a set of finite paths, each of them finishing with an axiom, but usually it would be impossible to compose them back to a tree.

There is a similar problem in automata theory, with constructing a tree automaton recognizing trees such that each path is accepted by some nondeterministic Büchi automaton. The solution there is to determinize the Büchi automaton first. We will follow this idea here.

An elegant solution for the problem of determinizing Büchi automaton was presented by Safra [45]. Rather then using his construction directly we will modify it a little bit. We do this because we will then code every state of the automaton as a formula of the μ-calculus hence the form of the state will be very important to us.

This way we construct appropriate Rabin automaton over one letter alphabet $\mathcal{TA}_{\varphi_0}$ whose runs closely correspond to the refutations of φ_0. We can then use Theorem 1.3.1 which allows us to conclude that there is a small graph $\mathcal{G}_{\varphi_0}$, with states of $\mathcal{TA}_{\varphi_0}$ as nodes, which unwinds to an accepting run of $\mathcal{TA}_{\varphi_0}$. In this graph special nodes which we call *loop nodes* can be distinguished. This nodes can be seen as "witnesses" that unwinding of $\mathcal{G}_{\varphi_0}$ is accepted by the automaton. On every cycle there is exactly one loop node which "confirms" that the unwinding of the cycle is accepted by $\mathcal{A}_{\varphi_0}$. Another important thing is that there is a natural partial order on loop nodes.

What we would like to do now is to sketch the method we use to transform a refutation of φ_0 into a proof of the sequent $\varphi_0 \vdash$. This method will be different from the one used by Kozen and more suitable to use with our induction rule. The example will allow us to show why deterministic strategy is essential. It should also give some intuitions about coding states of $\mathcal{G}_{\varphi_0}$ into the formulas of the μ-calculus, which will be the next step of the proof.

Let $\mathcal{D}_{\varphi_0}$ be a definition list for the formula φ_0. Let us suppose that we would like to transform some refutation of φ_0 into a proof of the sequent $\varphi_0 \vdash$. Let us look closer at some path $\mathcal{P} = \{\Sigma_i \vdash_{\mathcal{D}_{\varphi_0}}\}_{i \in P}$ of this refutation. Each Σ_i is a set of formulas, probably with definition constants from $\mathcal{D}_{\varphi_0}$. If we look at the sequence of expanded tableau sequents $\langle\!\langle \mathcal{P} \rangle\!\rangle = \{\langle\!\langle \Sigma_i \rangle\!\rangle_{\mathcal{D}_{\varphi_0}} \vdash_{\mathcal{D}}\}_{i \in P}$ then it occurs that this sequence can be seen as a path of a proof because each sequent, accept the first, is one of assumptions in some logical rule of which the preceding sequent is a conclusion. If $\mathcal{P}$ is finite then the last sequent in the sequence is moreover an axiom. This means that a finite path of a refutation can be transformed into a finite path which can be seen as a path of a proof.

If we use this expansion to the whole refutation, we will obtain a tree labeled with sequents which is almost a proof except that there can be infinite paths. The whole problem lies in "cutting" this infinite paths.

Suppose $\mathcal{P} = \{\mathcal{D}_{\varphi_0} \vdash_{\Sigma_i}\}_{i \in P}$ is infinite then of course also the path $\langle\!\langle \mathcal{P} \rangle\!\rangle = \{\langle\!\langle \Sigma_i \rangle\!\rangle_{\mathcal{D}_{\varphi_0}} \vdash_{\mathcal{D}}\}_{i \in P}$ is infinite. The condition lied on infinite paths of refutations says that on $\mathcal{P}$ there must be a μ-trace, i.e. an infinite trace on which the oldest regenerated constant is some μ-constant, say $(U = \mu X.\alpha(X)) \in \mathcal{D}_{\varphi_0}$. Because there can be only finitely many different sequents, U must be regenerated infintely often in the same sequent, say $U, \Sigma \vdash_{\mathcal{D}_{\varphi_0}}$. This situation is schematically presented below (the formulas from the trace are put in boxes and on the right the path with expanded sequents is presented).

$$
\begin{array}{r|r}
\varphi_0 \vdash_{\mathcal{D}_{\varphi_0}} & \varphi_0 \vdash \\
\boxed{\mathrm{U}}, \Sigma \vdash_{\mathcal{D}_{\varphi_0}} & \boxed{\mu X.\alpha(X)}, \langle\!|\Sigma|\!\rangle \vdash \\
\boxed{\alpha(U)}, \Sigma \vdash_{\mathcal{D}_{\varphi_0}} & \boxed{\alpha(\mu X.\alpha(X))}, \langle\!|\Sigma|\!\rangle \vdash \\
\vdots & \vdots \\
\boxed{U}, \Sigma \vdash_{\mathcal{D}_{\varphi_0}} & \boxed{\mu X.\alpha(X)}, \langle\!|\Sigma|\!\rangle \vdash \\
\vdots & \vdots
\end{array}
$$

The path on the right is constructed according to logical rules but unfortunately it is infinite. Let $a_1, \ldots, a_n$ be all actions appearing in the formula φ_0 and let P stand for a PDL program $(a_1 \cup \ldots \cup a_n)^*$. First observation is that if we take any formula β and put a formula $\langle P \rangle \beta$ on right side of the sequents then the obtained path

$$
\begin{array}{r}
\boxed{\mu X.\alpha(X)}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\boxed{\alpha(\mu X.\alpha(X))}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\vdots \\
\boxed{\mu X.\alpha(X)}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\vdots
\end{array}
$$

will be still localy correct path in the sese that each sequent is obtained from the other by some number of logical rules.

The second obsrevation is that if we take a fresh propositional variable, Z, and repalce U with $\alpha(\mu X.Z \wedge \alpha(X))$ (instead of $\mu X.\alpha(X)$) in the first sequent of the path, we can get the following, still locally correct, path.

$$
\begin{array}{r}
\boxed{\alpha(\mu X.Z \wedge \alpha(X))}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\boxed{\alpha(\mu X.Z \wedge \alpha(X))}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\vdots \\
\boxed{\mu X.Z \wedge \alpha(X)}, \langle\!|\Sigma|\!\rangle \vdash \langle P \rangle \beta \\
\vdots
\end{array}
$$

Observe that the first step becomes just identity but it has some influence. We can look at the second sequent as obtained by replacing U with $\mu X.Z \wedge \alpha(X)$. This is the reason why we have $\mu X.Z \wedge \alpha(X)$ in place of U in the last sequent. It is also important to know that U does not disapear on the trace. If U disapeared on the trace then also additionall information would disapear with it and all we could get would be $\mu X.\alpha(X)$ in the last sequent. As we will see appearing and dissapearing of definition constants will concern us during the whole proof.

Putting this observations together we proceed in a folowing way. In the first place where regeneration of U was preformed, i.e. in the seqent $\boxed{\mu X.\alpha(X)}, \langle\!\langle\Sigma\rangle\!\rangle \vdash$, we apply the rule (*ind*) and obtain two premisses

$$\boxed{\mathit{ff}}, \langle\!\langle\Sigma\rangle\!\rangle \vdash \quad \boxed{\alpha(\mu X.Z \wedge \alpha(X))}, \langle\!\langle\Sigma\rangle\!\rangle \vdash \langle P\rangle(\langle\!\langle\Sigma\rangle\!\rangle \wedge \mu X.Z \wedge \alpha(X))$$

The first one is clearly provable and with the second we just travel allong the path of the refutation obaining the following path of a proof

$$\begin{array}{r}
\boxed{\alpha(\mu X.Z \wedge \alpha(X))}, \langle\!\langle\Sigma\rangle\!\rangle \vdash \langle P\rangle(\langle\!\langle\Sigma\rangle\!\rangle \wedge \mu X.Z \wedge \alpha(X)) \\
\boxed{\alpha(\mu X.Z \wedge \alpha(X))}, \langle\!\langle\Sigma\rangle\!\rangle \vdash \langle P\rangle(\langle\!\langle\Sigma\rangle\!\rangle \wedge \mu X.Z \wedge \alpha(X)) \\
\vdots \\
\boxed{\mu X.Z \wedge \alpha(X)}, \langle\!\langle\Sigma\rangle\!\rangle \vdash \langle P\rangle(\langle\!\langle\Sigma\rangle\!\rangle \wedge \mu X.Z \wedge \alpha(X))
\end{array}$$

Observe that the last sequent is easily provable.

This way we have transformed an infinte path of a refutation into a finite path of a proof tree. Unfortunatelly with this approach as it is now we can run into at least one problem.

Important point is that while describing the cutting procedure, we have taken a path of a refutation *first* and only then constructed a part of a proof. To cut the path we needed to know which constant regenerates infintely often and does not disapear on it. If we took different path, there could be other constant and cutting strategy might be different. We would arrive with parts of the proof, each for different path of the refutation, which would not neccessary compose back to a tree. In other words we need a deterministic cutting strategy, i.e., such a strategy which would depend only on ancestors of a node and not on the whole path.

From the graph $\mathcal{G}_{\varphi_0}$ we can read a deterministic strategy for dealing with every path of the refutation, obtained by unwinding of $\mathcal{G}_{\varphi_0}$ into an infinite tree. The strategy will work almost as described in the example above.

The one very important difference is that it occurs that we are forced to use induction rule in nodes where no regeneration of the μ-formula occurs. That is we have to use induction rule on several occurences of the μ-formula simultanously. This is the reason why the induction rule allows to consider μ-formulas in the context.

The first step in construction of a deterministic strategy is to develop means to code nodes of $\mathcal{G}_{\varphi_0}$, which are the states $\mathcal{TA}_{\varphi_0}$, as formulas. The coding $F_f(s)$ of a state s will depend on the function f designating the part of the structure of s to be coded.

The coding will be such that each transition in $\mathcal{G}_{\varphi_0}$ will be derived in our system. That is if $t_1, \ldots, t_k$ are all nodes to which there is an edge from s in $\mathcal{G}_{\varphi_0}$ then the sequent $F_f(s) \vdash F_f(t_1), \ldots, F_f(t_n)$ will be provable.

In the last section of this chapter we construct a proof of $\varphi_0 \vdash$. The proof construction is guided not by the graph $\mathcal{G}_{\varphi_0}$ itself, but by its unwinding to the finite tree with "back edges" $\mathcal{T}_{\varphi_0}$. This simplifies construction a little because there is a direct correspondence between $\mathcal{T}_{\varphi_0}$ and the shape of a proof obtained for $\varphi_0 \vdash$.

The formula φ_0 is fixed throughout this chapter. Additionally let $\mathcal{D}_{\varphi_0}$ denote a definition list $\rrbracket \varphi_0 \llbracket = (W_1 = \sigma X.\gamma_1(X)) \ldots (W_d = \sigma X.\gamma_d(X))$. We will consider only definition constants from $\mathcal{D}_{\varphi_0}$ and when we will say that a definition constant is older (younger) then the other we will mean that it is older (younger) with respect to the definition list $\mathcal{D}_{\varphi_0}$.

4.1 Automaton

The goal of this section is to construct a special Rabin automaton $\mathcal{TA}_{\varphi_0}$ on trees over one letter alphabet, which accepting runs closely correspond to the refutations of φ_0. We do this by constructing a special deterministic Rabin automaton $\mathcal{A}_{\varphi_0}$ on paths, which recognizes exactly the paths of a tableau for φ_0 with μ-trace on them.

Instead of determinizing some obvious Büchi automaton we will construct $\mathcal{A}_{\varphi_0}$ from the scratch. Since we will quickly arrive at trees labeled with states which are itself trees, we will always call nodes of states *vertices* and nodes of trees of states just *nodes*.

Consider an infinite path $\mathcal{P}$ of a tableau for φ_0. We would like to check whether there is a μ-trace on $\mathcal{P}$ or not. We can do this using simple non-deterministic Büchi automaton. This automaton will go along the path and pick one formula from each tableau sequent in such a way that it will form a trace. Then acceptance conditions will be such that this chosen trace will be accepted iff it is μ-trace. Obviously this automaton is nondeterministic because of the tableau rule (*and*). If we have chosen formula $\alpha \wedge \beta$ in a sequent $\alpha \wedge \beta, \Gamma \vdash_{\mathcal{D}}$ then after application of the rule (*and*) we obtain a sequent $\alpha, \beta, \Gamma \vdash_{\mathcal{D}}$ and we may choose either formula α or formula β as the next formula in a trace.

Our goal is to construct an automaton on trees recognizing refutations of φ_0. We cannot directly convert a nondeterministic automaton $\mathcal{A}$ on paths, into an automaton on trees accepting trees whose all paths are accepted by $\mathcal{A}$. The reason is that although we can run $\mathcal{A}$ on each path separately, its accepting runs not necessary may compose back to a tree to give a run of the tree automaton. They would certainly compose if the path automaton were deterministic.

An elegant determinization construction was given by Safra [45]. Later we will need to code states of obtained automaton into formulas of the μ-calculus. This means that the form of a states is important to us and the succinct construction of Safra seems to be a good starting point.

Let us look more closely at the nondeterministic Büchi automaton we have described above. We will first introduce two notions to facilitate its description.

Definition 4.1.1 For any formula φ, possibly with definition constants, we define the FL-closure of φ, $FL(\varphi)$ as the set of all formulas which can occur in a sequent $\Sigma \vdash_{\mathcal{D}_{\varphi_0}}$ obtainable from $\varphi \vdash_{\mathcal{D}_{\varphi_0}}$ by application of some number of the tableau rules.

For any definition constant $(U = \sigma X.\alpha(X)) \in \mathcal{D}_{\varphi_0}$, let $Cl(U)$ be the set of all formulas which can occur in a sequent $\Sigma \vdash_{\mathcal{D}_{\varphi_0}}$ obtainable from $\varphi \vdash_{\mathcal{D}_{\varphi_0}}$ by application of some number of the tableau rules accept regeneration of constants older then U. ∎

The set of states of our nondeterministic Büchi automaton will contain the set

$$\{(\psi, \Gamma) : \psi \in FL(\varphi_0), \Gamma \subseteq FL(\varphi_0), \psi \in \Gamma\}$$

and sets

$$\{(U, \psi, \Gamma) : \psi \in Cl(U), \Gamma \subseteq FL(\varphi_0), \psi \in \Gamma\}$$

for any μ-constant, U in $\mathcal{D}$. We also add an artificial start start state ε. After reading the next input, $\Sigma \vdash_{\mathcal{D}}$ from a path $\mathcal{P}$, the automaton will be in a state (ψ, Σ) or (U, ψ, Σ). Intuitively the meaning of such a state is that automaton has chosen formula ψ as a next formula of the trace under construction. If the state is a triple (U, ψ, Σ) then it also means that the automaton has decided that on the trace it constructs U will be the oldest constant regenerated infinitely often. We will denote the set of all states by Q. Observe that Q is finite.

The transition function of the automaton, δ, should be rather obvious. From a state (ψ, Σ) on input $\Sigma' \vdash_{\mathcal{D}}$ it can go to a state (ψ', Σ') or (U, ψ', Σ'), where U is some μ-constant and ψ' is a formula obtained from ψ if a reduction of ψ took place, or $\psi' = \psi$ otherwise. The transition function behaves similarly if a state is a triple (U, ψ, Σ) but in this case the next possible state must be of the form (U, ψ', Σ') that is it can be only a triple with the same definition constant. Observe that in this case there is no possible next state if ψ is a definition constant older then U and it is regenerated in the input sequent. Form the start state ε there is only one transition, on the input $\varphi_0 \vdash_{\mathcal{D}}$ it goes to the state $(\varphi_0, \{\varphi_0\})$.

The accepting set of the automaton is the set of states $F = \{(U, U, \Sigma) : \Sigma \in FL(\varphi_0)\}$. It is easy to see that this automaton accepts exactly those paths of a tableau for φ_0 which have a μ-trace on them.

As we mentioned earlier we would like to have a deterministic automaton and we are going to apply Safra's determinization construction [45]. Let us look what will be the result of this procedure.

The states of the obtained automaton will be labeled trees, i.e., tuples $T = \langle N, r, p, \prec, nl \rangle$ where:

- N is a set of vertices,
- $r \in N$ the root of the tree,
- $p : (N \setminus \{r\}) \to N$ a parenthood function,
- $\prec$ is a sibling ordering, i.e., left-to-right ordering of nodes with a common father,
- $nl(v)$, for any vertex $v \in N$, is a non-empty set of states of the nondeterministic automaton. It is called a *node label* of v.

Additionally each vertex can be colored white or green and the following conditions hold:

1. The union of the node labels of the sons of any vertex is a subset of the label of the father.

2. Any two sons of the same vertex have disjoint node labels.

Being in a state s and given a next input sequent $\Sigma \vdash_{\mathcal{D}}$ the next state is obtained by performing the following sequence of actions:

1. Set color of all vertices to white.
2. For every vertex v of s replace $nl(v)$ by $\bigcup_{p \in nl(v)} \delta(p, \Sigma \vdash_{\mathcal{D}})$.
3. If for some vertex v and constant U, a state (U, U, Σ) is in $nl(v)$ then create a rightmost son of v with the label $\{(U, U, \Sigma)\}$.
4. For every vertex v, if some $p \in nl(v)$ belongs to a vertex to the left of v in the tree s then delete p from $nl(v)$.
5. Remove all vertices with empty labels.
6. For every vertex v whose label is equal to the union of the labels of its sons, remove all descendants of v and light v green.

The accepting condition is that some vertex lights green infinitely often on the run and disappears only finitely many times on the run.

From the results of Safra it follows that described deterministic automaton accepts exactly the paths of a tableau for φ_0 with μ-trace on them. To see why it is the case it is probably convenient to think about the vertex of a state s as representing a property that for each state (of the nondeterministic automaton) occurring in the label of the vertex there is a run to it (of the nondeterministic automaton) on which accepting states appear some number of times. This number is exactly the depth of the vertex plus the number of times the vertex lighted green since it disappeared last time.

Let us look at the states of the above deterministic automaton. Of course it will be always the case that after reading a sequent $\Sigma \vdash_{\mathcal{D}}$ the automaton will be in a state s such that each pair (ψ, Ω) or triple (U, ψ, Ω) occurring in it will have the property that $\Omega = \Sigma$. It will be also the case that for every $\psi \in \Sigma$ there will be a pair (ψ, Σ) in the label of the root. This means that the last element of a state is redundant – it can be reconstructed from the state. In the label of any son of the root of s there will be only triples, and all triples in one label will have the same first component. Hence keeping U in each triple (U, ψ, Σ) of a label is also somehow redundant. We can label the whole vertex with U instead. Another redundancy is that for a fixed ψ there can be many states (U, ψ, Σ) in s for different U. What's more if

(U, ψ, Σ) appears in some label of s then also (U', ψ, Σ) will appear in s, for any μ-constant U' older then U. This triples will appear in different subtrees of s. All this redundancies make coding of the states into formulas more difficult.

Below we will describe the construction which will give us the desired automaton $\mathcal{A}_{\varphi_0}$, whose states are trees labeled just with formulas. Each formula will appear on at most one path of a state. This means that for any formula φ in the state there will be the lowest vertex containing this formula. The price to pay is additional complexity of the states resulting from adding a new labeling of vertices with definition constants from $\mathcal{D}_{\varphi_0}$.

States of the automaton $\mathcal{A}_{\varphi_0}$ will be labeled ordered trees, i.e., tuples $T = \langle N, r, p, \prec, nl, el \rangle$ where:

- N is a set of vertices,
- $r \in N$ the root of the tree,
- $p : (N \setminus \{r\}) \to N$ a parenthood function,
- $\prec$ is a sibling ordering, i.e., partial ordering relation which linearly orders nodes with a common father,
- $nl(v)$, for any vertex $v \in N$, is a non-empty subset of $FL(\varphi_0)$ called a *node label* of v.
- $el(v)$, for any vertex $v \in N \setminus \{r\}$, is a definition constant from $\mathcal{D}_{\varphi_0}$ called an *edge label* of v.

Additionally each vertex can be colored white or green and the following conditions hold:

1. The union of the node labels of the sons of any vertex is a subset (not necessarily proper) of the label of the father.
2. For any vertex v the union of the node labels of those sons of v which edge labels are equal $el(v)$ is a proper subset of $nl(v)$.
3. Any two sons of the same vertex have disjoint node labels.
4. If v is a son of w then $el(v)$ is not older than $el(w)$ with respect to the definition list $\mathcal{D}_{\varphi_0}$.
5. If $el(v)$ is a ν-constant then v has no ν-sons, i.e., sons with the edge label being a ν-constant.

6. For any two sons, v and w, of the same vertex, if $el(v)$ is older than $el(w)$ with respect to $\mathcal{D}_{\varphi_0}$, then $v \prec w$.

Ordering $\prec$ can be extended to an ordering between not necessarily ancestral vertices. We say that v is to the *left* of w iff some (not proper) ancestor of v is smaller in $\prec$ ordering then some (not proper) ancestor of w. If v is a son of w and $el(v) = W$ then we say that v is a *W-son* of w. Conditions 1 and 3 guarantee that for any formula ψ occurring in a state s there is the lowest vertex v such that $\psi \in nl(v)$. We will call such a vertex the *ψ-vertex of s*. Node and edge labels of a node v of a state s will be denoted as $nl_s(v)$ and $el_s(v)$ respectively.

The initial state of the automaton, s_0 will be a tree consisting only of the root labeled $\{\varphi_0\}$. We do not introduce any artificial start state, as we have done in the previous automaton, in order to simplify the description. Instead we will assume that a path of a tableau starts not from the root but from the next node. If we added artificial start state ε, then all the runs of the automaton would have the same first two elements ε, s_0.

It might be worth to compare the set of states obtained by this construction with that obtained by Safra's construction. Vertices are now labeled with formulas and not with pairs or triples, but we have added new labeling. Because of introduction of this labeling states have now more additional properties. Property 1 is weaker than corresponding property in Safra's construction. We need additional properties 2, 4 and 5 to guarantee that states are finite trees. The property 3 did not change and new property 6 describes the behavior of the edge labeling with respect to the encoding.

In this new form of states, a vertex v, such that $el(v)$ is a μ-constant U, represents the property that there exists a trace to it on which, from some point, U is the oldest regenerated constant. This constant was regenerated as many times as the vertex lighted green plus the number of ancestors with the same edge label. Similarly we can interpret vertices labeled with ν-constants except that we are not interested in the number of regenerations in this case. There is also some similarity between edge labels of ancestors of a vertex with last appearance records introduced in [16].

Next we describe the deterministic transition function of the automaton. It will be always the case that after reading a sequent $\Sigma \vdash_{\mathcal{D}_{\varphi_0}}$ the automaton will enter a state with the root labeled by Σ. Suppose the automaton is in a state s after reading $\Sigma \vdash_{\mathcal{D}_{\varphi_0}}$ and the next input is $\Sigma' \vdash_{\mathcal{D}}$. The next state is obtained by applying, to the tree s, the following sequence of actions:

1. Set color of all vertices to white.

2. Look at the next sequent $\Sigma' \vdash_{\mathcal{D}_{\varphi_0}}$ on the path and locally transform the labels of some vertices, depending on the rule applied to obtain this sequent:

— for all rules other then (*cons*) just replace the reduced formula with the resulting formulas, appearing in Σ', in all vertices of s. This means that for all rules other then ($\langle\rangle$) the reduced formula is replaced by at most two resulting formulas and other formulas remain intact. In case of the ($\langle\rangle$) rule:

$$\frac{\langle a\rangle\alpha, \Sigma \vdash_{\mathcal{D}}}{\alpha, \{\beta : [a]\beta \in \Sigma\} \vdash_{\mathcal{D}_{\varphi_0}}}$$

we first delete all formulas not of the form $[a]\beta$, for some β, except the formula $\langle a\rangle\alpha$. Then we replace $\langle a\rangle\alpha$ with α and $[a]\beta$ with β in each node label of s

— if we apply regeneration rule (*cons*):

$$\frac{U, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{\alpha(U), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

then let v be the lowest vertex of s such that $U \in nl(v)$ and $el(v)$ is not younger (older or equal with respect to $\mathcal{D}_{\varphi_0}$) than U. If there is no such vertex then let v be the root of s. First replace U with $\alpha(U)$ in the node labels of v and all ancestors of v. Next delete U from node labels of all proper descendants of v. Additionally, if U or $el(v)$ is not a ν-constant, create a son w of v, with labels $el(w) = U$ and $nl(w) = \{\alpha(U)\}$. Finally make w $\prec$-bigger than all its brothers with edge labels not younger than U and smaller from all other brothers.

3. For any vertex v, if a formula φ occurs already in a vertex to the left of v then delete φ from the node label of v.

4. Delete all vertices with empty labels.

5. For any vertex v such that $el(v) = U$ is a μ-constant and $nl(v)$ is equal to the sum of the labels of its U-sons, light v green and delete all descendants of v from the tree.

Automaton $\mathcal{A}_{\varphi_0}$ accepts a run iff there is a vertex which disappears only finitely many times and lights green infinitely many times on the run.

One word must be said about "vertex management". In clause 2 of the definition of the transition function we are at some point required to add a new vertex to the state, i.e., a one not occurring in it already. Clearly we could not have an infinite supply of vertices because the number of states of the automaton must be finite. The solution is to "reuse" vertices, i.e., when a vertex is deleted we put it into a common pool and when a new vertex is needed just take any vertex from the pool. If we put initially into the pool more vertices then the size of the largest possible state then we would be sure that there is always something in the pool when needed. Hence our first step is to find the bound on the size of the states of $\mathcal{A}_{\varphi_0}$.

Lemma 4.1.2 For any formula φ the size of $FL(\varphi)$ is linear in the size of φ.

Lemma 4.1.3 The size of a state tree is bounded by $n * m + 1$ where n is the number of formulas in $FL(\varphi_0)$ and m the number of definition constants in $\mathcal{D}_{\varphi_0}$. The number of states is finite.

Proof
Let s be any state of $\mathcal{G}_{\varphi_0}$. With every vertex v of s, except the root, we can assign a pair $(el(v), \varphi)$, where $\varphi \in nl(v)$ is a formula not occurring in any $el(v)$-son of v. This shows the upper bound on the number of states because the assignment is injective.

The second part of the lemma follows directly from the first if we apply the strategy for "reusing vertices" described above. □

Hence what we have described is a Rabin automaton on strings. Before proving correctness of the construction let us focus on one more specialized property of the $\mathcal{A}_{\varphi_0}$ which we will need in the future.

Lemma 4.1.4 For any μ-constant $U_i = \mu X.\alpha_i(X)$ in $\mathcal{D}_{\varphi_0}$ and vertex v of a state s, reachable from the start state of the automaton $\mathcal{A}_{\varphi_0}$, s.t. $el(v) = U_i$, the formula $\mu X.\alpha_i(X)$ does not belong to $nl(v)$.

Proof
Let us take a μ-constant U_i as above and define the set of formulas Cl as a smallest set such that:

- $\alpha(U_i)$ belongs to Cl,

- if $\psi \in Cl$ then all subformulas of ψ belong to Cl,
- if $\sigma X.\beta(X) \in Cl$ and there is a definition constant W such that $W = \sigma X.\beta(X)$ is in $\mathcal{D}_{\varphi_0}$ then $\beta(W)$ belongs to Cl.

Observe that $\mu X.\alpha_i(X) \notin Cl$ because all formulas in Cl are shorter. For any state s and its vertex v s.t. $el(v) = U_i$ we show by induction on the number of steps needed to reach state s, that $nl_s(v) \subseteq Cl$.

The base step is when the vertex v is created in a state s. It's label is then exactly $\{\alpha(U_i)\}$. The induction step is straightforward.

□

Finally we show correctness of the construction

Proposition 4.1.5 The automaton accepts a path $\mathcal{P}$ of a tableau for φ_0 iff there exists an infinite μ-trace on $\mathcal{P}$.

The proof of the proposition is divided into left to right and right to left implications.

Lemma 4.1.6 If the automaton $\mathcal{A}_{\varphi_0}$ accepts a path $\mathcal{P}$ of a tableau for φ_0 then there exists an infinite μ-trace on the path.

Proof
Consider an accepting run of the automaton $S_0, S_1, \ldots$ We will denote by nl_i, el_i the node labeling and the edge labeling of the state S_i respectively. Let v be a vertex which lights green i.o. in this run and disappears only finitely many times. Consider two positions i, j, after v disappears last time, such that v lights green in S_i and next time it lights green in S_j.

First we would like to show that for every formula in $nl_j(v)$ there exists a trace from some formula in $nl_i(v)$ such that the oldest constant regenerated on the trace is $el_i(v) = el_j(v)$. Clearly $el_i(v)$ is a μ-constant.
To do this we show that for any S_k, between S_i and S_j, and any formula $\varphi \in nl_k(v)$:

- if $\varphi \in nl_k(w)$ for some son w of v, then there is a part of a trace to φ from some formula in $nl_i(v)$ such that the highest regeneration on it is that of $el_k(w)$
- otherwise there is a part of a trace to φ from some formula in $nl_i(v)$ without any regeneration at all.

The proof is by induction on the distance from S_i

- Base step when $k = i$ is trivial.
- All steps except of regenerations are rather straightforward.
- If the last step was regeneration of a constant V then we have two cases. The first is when $V \notin nl_k(v)$ or V is older than $el_k(V)$. This case is easy because the only thing which can happen is that some formula can be removed from the labels of all vertices in the subtree of v.

  In the other case when $V \in nl_k(v)$ and $el_k(v)$ is not younger than V we have two possibilities

  - If there is a son w of v, such that $V \in nl_k(w)$ and $el_k(w)$ is not younger than V then no new son of v is created and V is replaced by an appropriate $\alpha(V)$ in the label of w. This is sound as there is a trace to $\alpha(V)$ with oldest regeneration being that of $el_k(w)$ because there was one to V by induction hypothesis.
  - If, on the other hand, V is older than $el_k(w)$ or there is no w at all then from the induction hypothesis there is a trace to V on which no constant older than V was regenerated. Hence there is a trace to $\alpha(V)$ where V is the oldest regenerated constant. According to the definition of the transition function a new son of v, w' is created with the node label $\{\alpha(V)\}$ and edge label V. Additionally V is deleted from the label of w. These are exactly the steps which must be done to make the induction thesis satisfied.

Observe that vertices with edges labeled with ν-constants were needed in the last step of the above induction.

Now consider a graph with the set of nodes

$$\{(\varphi, k) : \varphi \in nl_k(v), v \text{ lights green in } S_k\}$$

and an edge from (φ_1, k_1) to (φ_2, k_2) when there is a trace from φ_1 in S_{k_1} to φ_2 in S_{k_2} on which $el(v)$ is the oldest regenerated constant. Please remember that v is our node which lights green infinitely often on the run. From what we have shown above it follows that at for any (φ, k) there is an edge leading to it from some (ψ, k'). This is because v lights green only when the sum of the labels of its $el(v)$-sons is equal $nl(v)$. This means that at least a part of this graph is an infinite connected directed acyclic graph. The degree of every vertex of it is of course finite hence there must exist an infinite path in this graph. This path is a μ-trace we were looking for.

□

Lemma 4.1.7 If there is a μ-trace on a path $\mathcal{P}$ of a tableau for φ_0 then the automaton $\mathcal{A}_{\varphi_0}$ accepts $\mathcal{P}$.

Proof
Let us consider a path $\mathcal{P}$ with a μ-trace and the run of the automaton, $S_0, S_1, \ldots$ on it. The trace on $\mathcal{P}$ is a sequence of formulas $\{\alpha_i\}_{i\in I}$ s.t. α_i occurs in the i-th tableau sequent of $\mathcal{P}$. On the other hand the automaton $\mathcal{A}_{\varphi_0}$ has the property that after reading a tableau sequent $\Sigma \vdash_{\mathcal{D}_{\varphi_0}}$, the root of its current state is labeled by Σ. Hence α_i is always present in the root of the state S_i.

From some moment j_0, the oldest constant regenerated on the trace is some μ-constant U. We can assume, without loss of generality, that α_{j_0} occurs in a son of the root with edge label, not younger than U. Indeed, otherwise we just have to wait, in the worst case, for the next regeneration of U. From that moment formulas from the trace can only move to the left according to clause 3 of the description of the transition function. But because they can do it only finitely many times, after some time the trace must settle in some son of the root, say v_0, forever.

If v_0 lights green i.o. then we are done. If not then let j_1 be the last moment v_0 lights green. Not later than the next regeneration of U the formula from the trace will appear in some W_1-son of v_0 where W_1 is not younger than U. After some moves to the left it must settle in some v_1 forever. If v_1 lights green i.o. we are done if not we repeat the reasoning and find a son of v_1, v_2 an so on. Observe that we cannot go this way forever because each state is a tree of bounded size. □

Together Lemmas 4.1.7 and 4.1.6 prove Proposition 4.1.5 thereby showing correctness of the construction.

The automaton $\mathcal{A}_{\varphi_0}$ expands to a nondeterministic automaton on trees over one letter alphabet $\mathcal{TA}_{\varphi_0}$, such that its accepting runs closely correspond to the refutations of φ_0. Automaton $\mathcal{TA}_{\varphi_0}$ nondeterministically constructs a quasi-refutation for φ_0 and runs $\mathcal{A}_{\varphi_0}$ on each path. There is no need to remember tableau separately because sequents which are read by

$\mathcal{A}_{\varphi_0}$ are remembered in the roots of the corresponding states. Hence states of $\mathcal{TA}_{\varphi_0}$ are exactly the states of $\mathcal{A}_{\varphi_0}$. The roots of an accepting run of $\mathcal{TA}_{\varphi_0}$ give us a refutation of φ_0. Other way around given a refutation of φ_0 we can run the path automaton $\mathcal{A}_{\varphi_0}$ on each path of the refutation separately and because it is deterministic we will obtain a tree which is an accepting run of $\mathcal{TA}_{\varphi_0}$.

In [5] the following theorem is (implicitly) stated

Theorem 4.1.8 (Emerson) *Suppose $\mathcal{A}$ is a Rabin automaton over a single letter alphabet. If $\mathcal{A}$ accepts some tree then there is a graph $\mathcal{G}$ with states of $\mathcal{A}$ as nodes which unwinds to an accepting run of $\mathcal{A}$.*

Because we know that there is a refutation of φ_0, from this theorem we conclude that there is a graph $\mathcal{G}_{\varphi_0}$, with states of $\mathcal{A}_{\varphi_0}$ as nodes, which unwinds to an accepting run of $\mathcal{TA}_{\varphi_0}$. In the next sections we will show how to construct a proof of $\varphi_0\vdash$ from this graph.

4.2 Constructions on the graph

In this section we investigate the properties of the graph $\mathcal{G}_{\varphi_0}$. We define several concepts which in turn will allow us to define the unwinding of $\mathcal{G}_{\varphi_0}$ to a finite tree with "back edges" $\mathcal{T}_{\varphi_0}$. This tree will simplify our induction argument in the completeness proof.

Since the unwinding of $\mathcal{G}_{\varphi_0}$ is accepted by $\mathcal{TA}_{\varphi_0}$, we know that on any infinite path of it there is a *confirming vertex*, i.e. a vertex which lights green i.o. and disappears only finitely many times. The crucial observation is that for any state s of $\mathcal{G}_{\varphi_0}$ and any path $\mathcal{P}$ passing i.o. through s, a confirming vertex of $\mathcal{P}$ belongs to s and that we can linearly order such confirming vertices. Intuitively, if we have this ordering $Ord(s) = (v_1, \ldots, v_i)$ we know that on any infinite path passing i.o. through the state s, some v_j is a confirming vertex and that none of $v_1, \ldots, v_{j-1}$ will ever disappear on this path.

More formally let us start by fixing some arbitrary linear ordering on the set of all vertices occurring in $\mathcal{G}_{\varphi_0}$, then $Ord(s)$ can be described as follows:

Definition 4.2.1 For any node s of $\mathcal{G}_{\varphi_0}$ we define an ordering of some of the vertices from s. Let $Ord(s)$ be the list $(v_1, \ldots, v_k)$ such that for each $i = 1, \ldots, k$, v_i is defined by the following rule:

Consider a graph obtained from $\mathcal{G}_{\varphi_0}$ by deleting all nodes where one of the vertices $v_1, \ldots, v_{i-1}$ lights green. Let $\mathcal{C}$ be a nontrivial strongly connected component, of this graph containing node s. Then v_i is the smallest vertex, in our fixed linear ordering on all vertices, such that v_i does not disappear in any node of $\mathcal{C}$ and lights green in some node of $\mathcal{C}$. ■

Observe that we can always find a vertex v_i required in the above definition because otherwise if we took a cycle through all the nodes of the strongly connected component and unwind it to an infinite path, it would not be accepted by the automaton $\mathcal{A}_{\varphi_0}$ which contradicts our initial assumptions.

The ordering $Ord(s)$ allows us to distinguish some special nodes of $\mathcal{G}_{\varphi_0}$, namely such that a vertex from $Ord(s)$ lights green in s. Observe that such a vertex must be the last vertex in $Ord(s)$.

Definition 4.2.2 A node s of $\mathcal{G}_{\varphi_0}$ is a *loop node* if $Ord(s) = (v_1, \ldots, v_i)$ and v_i lights green in s. The vertex v_i will be called the *green vertex* of s

This notions allow us, among others, to define a special unwinding of $\mathcal{G}_{\varphi_0}$.

Definition 4.2.3 We will unwind the graph $\mathcal{G}_{\varphi_0}$ to the finite "tree with loops" $\mathcal{T}_{\varphi_0} =< T, S >$, labeled with nodes of $\mathcal{G}_{\varphi_0}$, such that $\mathcal{G}_{\varphi_0}$ and $\mathcal{T}_{\varphi_0}$ unwind to the same infinite tree. We proceed as follows:

- The root of $\mathcal{T}_{\varphi_0}$ will be labeled by the start node of $\mathcal{G}_{\varphi_0}$.
- Suppose n is already constructed node labeled by $S(n)$. If there is an edge from $S(n)$ to a state s in $\mathcal{G}_{\varphi_0}$ then we add a son of n labeled by s to $\mathcal{T}_{\varphi_0}$. The only exception to this rule is when s is a loop node, there is already an ancestor m of n labeled with s, and the only vertex from $Ord(s)$ which lights green on the path from m to n is the green vertex of s. In this case we just add a back edge from n to m without creating any son for s. ■

Fact 4.2.4 The tree $\mathcal{T}_{\varphi_0}$ constructed above is finite.

Proof
Suppose that there is an infinite path in $\mathcal{T}_{\varphi_0}$ without taking a back edge. Let us take any state s of $\mathcal{G}_{\varphi_0}$ which occurs infinitely often on this path. Consider

$Ord(s) = (v_1, \ldots, v_k)$. Between any two occurrences of s some vertex from $v_1, \ldots, v_k$ must light green. Let us take the smallest i s.t. v_i lights green i.o. on the path.

Let s' be the state which occurs i.o. on the path and where v_i lights green. This means that there is a cycle from s to s' and back to s on which none of $v_1, \ldots, v_{i-1}$ lights green. It follows from the definition of $Ord(s)$ that none of $v_1, \ldots, v_i$ disappears on such a cycle. Hence $Ord(s') = (v_1, \ldots, v_i)$. So s' is a loop node and v_i is its green vertex. In this case we can find two nodes m, n of $\mathcal{T}_{\varphi_0}$ labeled with s', such that v_i does not disappear and $v_1, \ldots, v_{i-1}$ do not light green on the path from m to n. From the definition of $\mathcal{T}_{\varphi_0}$ it follows that the path should end before the node n. Contradiction. □

The next concept is very important although quite straightforward. The idea is to distinguish, for a node of $\mathcal{G}_{\varphi_0}$, these loop nodes to which we can eventually arrive by taking back edges.

Definition 4.2.5 For any node m of $\mathcal{T}_{\varphi_0}$ we define the set of *active nodes* of m, $AN(m)$ as the smallest set such that:

- For any, maybe not proper, ancestor n of m such that there is a back edge from some, maybe not proper, descendant of m to n, let $n \in AN(m)$
- if $n \in AN(m)$ then $AN(n) \subseteq AN(m)$

Let $AV(m)$, the set of *active vertices* of m, be the set of the green vertices of all the nodes from $AN(m)$. Observe that only loop nodes can belong to $AN(m)$. ■

Final lemma of this section presents the properties of active vertices and loop nodes we will need in the definition of the coding and in the final induction argument.

Lemma 4.2.6 For any node m of $\mathcal{T}_{\varphi_0}$ and $v \in AV(m)$, v is a vertex of $S(m)$. If there is an edge from m to m' in $\mathcal{T}_{\varphi_0}$ then: if there is no back edge to n then $AN(m') \subseteq AN(m)$; otherwise $AN(m') \subseteq AN(m) \cup \{m'\}$.

Proof

To prove the first point of the lemma we will show that for any node $n \in AN(m)$, $Ord(S(n))$ is a prefix of $Ord(S(m))$ hence a green node of $S(n)$

occurs in $Ord(S(m))$. Directly from the definition it follows that all vertices from $Ord(S(m))$ occur in $S(m)$.

We do the proof by induction on the height of the vertex m. We only present induction step because it is more general. Observe that all vertices in $AN(m)$ are ancestors of m hence by yet another induction on the distance from m we can prove that for any $n \in AN(m)$, $Ord(S(n))$ is a prefix of $Ord(S(m))$. We have two cases:

— There is a back edge from some, maybe not proper, descendant of m, say m', to n. From the definition of $\mathcal{T}_{\varphi_0}$, we know that n is a loop node and on the path from n to m', going through m, none of the vertices from $Ord(S(n))$ disappears and the last one lights green. Directly from the definition of the ordering follows that $Ord(S(n))$ is a prefix of $Ord(S(m))$.

— There is a node $n' \in AN(m)$ such that $n \in AN(n')$. Then because n' is an ancestor of m we have that $Ord(n)$ is a prefix of $Ord(n')$ from the outermost induction assumption. On the other hand because n' is closer to m than n then $Ord(n')$ is a prefix of $Ord(m)$ from the nested induction assumption. From this two facts we have that $Ord(n)$ is a prefix of $Ord(m)$.

The second part of the lemma follows directly from the definition of $AN(m)$. □

4.3 Coding states into formulas

The next step is to develop some means to code the states occurring in $\mathcal{G}_{\varphi_0}$ as a formulas of the μ-calculus. This is the object of this section together with presenting the main properties of the defined coding.

The coding $F_f(n)$ of a node n, will depend on a function f designating the part of the structure which needs to be coded. The coding will allow us to reduce the task of proving $F_f(n) \vdash \Delta$ to constructing proofs for $F_f(m_1) \vdash \Delta, \ldots, F_f(m_i) \vdash \Delta$, where $m_1, \ldots, m_i$ are all sons of n in $\mathcal{G}_{\varphi_0}$. Of course the coding must be such that proving sequents constructed for the sons is simpler then the original one. As we will see in the next section rule (ind) allows to "remember" some regenerations in sequents. This information takes the form of a fresh variable Z introduced in the rule. The coding must be such that the important part of this information, i.e. this designated by function f, is not forgotten.

We will use here a convention that U_i will stand for the i-th μ-constant in $\mathcal{D}_{\varphi_0}$ and $\mu X.\alpha_i(X)$ for the formula it defines. Additionally let d^μ denote the number of μ–constants in $\mathcal{D}_{\varphi_0}$.

Definition 4.3.1 Let s be any state of $\mathcal{G}_{\varphi_0}$. A function from vertices occurring in the states of $\mathcal{G}_{\varphi_0}$ to $\mathcal{N} \cup \{\infty\}$ will be called *admissible* for s iff for any vertex v of s, such that $f(v) = \infty$, the vertex v has no $el(v)$-sons in s.

For any vertex v of s we also define the set of close ancestors of v

$$\{v\}\!\uparrow_s = \{u : u \text{ ancestor (maybe not proper) of } v \text{ in } s,\ el(u) = el(v)\}$$

Intuitively, an admissible function f designates the part of a state which has to be coded. This can be seen in the following notion of the signature of a vertex. This syntactic signature has some degree of resemblance to the semantic signature from the Definition 2.3.1. Instead of ordinals we use here fresh variables to distinguish approximations of fixpoints.

Definition 4.3.2 For any state s of $\mathcal{G}_{\varphi_0}$, f admissible function for s and any vertex v of s, we define a *signature* of v, denoted $\parallel v \parallel_s^f$, as a sequence of formulas $(\delta_1, \ldots, \delta_{d^\mu})$. Formula δ_i is defined as follows. Let u be the lowest (maybe not proper) ancestor of v, such that $el(u) = U_i$, or the root if there is no such ancestor. Then:

- if $f(u) = \infty$ then $\delta_i = \mathit{ff}$,
- if $f(u) = 0$ then $\delta_i = \mu X.\mathcal{V} \wedge \alpha_i(X)$,
- if $\infty > f(u) > 0$ then $\delta_i = \mathcal{V}' \wedge \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))$,

where:

$$\mathcal{V} = \bigwedge\{Z_w^j : w \in \{u\}\!\uparrow_s, 1 \le j \le f(w)\}$$
$$\mathcal{V}' = \bigwedge\left(\{Z_w^j : w \in \{u\}\!\uparrow_s, 1 \le j \le f(w)\} \setminus \{Z_u^{f(u)}\}\right)$$

The variables Z_w^j, we have used, are assumed to be variables not occurring free in φ_0.

If s is a loop node then we also define the signature $\overline{\parallel v \parallel}_s^f$. The signature $\overline{\parallel v \parallel}_s^f$ differs from $\parallel v \parallel_s^f$ only if v is the green vertex of s and $\infty > f(v) > 0$. In this case signatures differ only at position i, such that $U_i = el(v)$. Namely, the i-th coordintate of $\overline{\parallel v \parallel}_s^f$ is $\mu X.\mathcal{V} \wedge \alpha_i(X)$ instead of $\mathcal{V}' \wedge \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))$ in the signature $\parallel v \parallel_s^f$. Here

$$\mathcal{V}' = \bigwedge\left(\{Z_w^j : w \in \{v\}\!\uparrow_s, 1 \le j \le f(w)\} \setminus \{Z_v^{f(v)}\}\right) \quad \mathcal{V} = \mathcal{V}' \wedge Z_v^{f(v)}$$

Using our "syntactical" signatures the coding of states is defined below.

Definition 4.3.3 Let ψ be a formula occurring in a state s of $\mathcal{G}_{\varphi_0}$. Let v be a ψ-vertex of s, i.e. the lowest vertex of s containing ψ, and let $\| v \|_s^f = (\delta_1, \dots, \delta_{d^\mu})$. We define

$$\| \psi \|_s^f = \psi[\gamma'_d/W_d] \dots [\gamma'_1/W_1]$$

where $\mathcal{D}_{\varphi_0} = (W_1 = \gamma_1) \dots (W_d = \gamma_d)$ and $\gamma'_i = \gamma_i$ if W_i is a ν-constant or $\gamma'_i = \delta_j$ if $W_i = U_j$ is the j-th μ-constant of $\mathcal{D}_{\varphi_0}$.

If Γ is a set of formulas occurring in vertices of s then let

$$\| \Gamma \|_s^f = \{\| \psi \|_s^f : \psi \in \Gamma\}$$

Finally $F_f(s)$ will stand for $\bigwedge \| \Sigma \|_s^f$, i.e. the conjunction of all formulas from $\| \Sigma \|_s^f$, where Σ is the set of all formulas occurring in s. By definition of the states, Σ is also the label of the root of s.

If s is a loop node then we define $\overline{\| \psi \|}_s^f$ and $\overline{F}_f(s)$ in a similar way using $\overline{\| v \|}_s^f$ instead of $\| v \|_s^f$ in the above clauses. ∎

The following lemma expresses the fact that lower vertices contain more information in its signature then their ancestors. If we look closer, this additional information is a "regeneration history" of a formula.

Lemma 4.3.4 Let s be a state of $\mathcal{G}_{\varphi_0}$, let f be admissible function for s and let v, w be two vertices of s, such that v is an ancestor of w. For any $i = 1, \dots, d^\mu$, if δ_i is the i-th element of $\| v \|_s^f$ and δ'_i is the i-th element of $\| w \|_s^f$ then the sequent $\delta'_i \vdash \delta_i$ is provable.

Proof

Let us choose any $i \in \{1, \dots, d^\mu\}$. If w', the lowest ancestor of w s.t. $el(w') = U_i$, is also an ancestor of v then $\delta_i = \delta'_i$ and we are done. Otherwise let v' be either the lowest ancestor of v s.t. $el(v') = U_i$ or the root if there is no such ancestor.

- if $f(w') = \infty$ then $\delta'_i = ff$ and of course $\delta'_i \vdash \delta_i$ is provable.
- if $f(w') = 0$ then

$$\delta'_i = \mu X.\mathcal{V} \wedge \alpha(X) \quad \mathcal{V} = \bigwedge \left(\{Z_u^j : u \in \{w'\}\!\uparrow_s, 1 \leq j \leq f(u)\}\right)$$

and there are two possibilities for δ_i

$$\delta_i = \mathcal{V}_1 \wedge \alpha(\mu X.\mathcal{V}_2 \wedge \alpha(X)) \quad \text{or} \quad \delta_i = \mu X.\mathcal{V}_2 \wedge \alpha(X)$$
$$\mathcal{V}_1 = \bigwedge\left(\{Z_u^j : u \in \{v'\}\!\uparrow_s, 1 \le j \le f(u)\} \setminus \{Z_{v'}^{f(v')}\}\right) \quad \mathcal{V}_2 = \mathcal{V}_1 \wedge Z_{v'}^{f(v')}$$

But clearly $\mathcal{V} \vdash \mathcal{V}_2$ is provable because w' is a proper descendant of v'. Hence, because all occurrences of $\mathcal{V}, \mathcal{V}_1, \mathcal{V}_2$ are positive we have the proof of $\delta_i' \vdash \delta_i$ by Lemma 3.1.4.

- If $f(w') > 0$ then the reasoning is very similar.

□

As was already mentioned the amount of information about s which is coded in $F_f(s)$ depends on the function f. There is a natural ordering of functions which reflects this dependence.

Definition 4.3.5 Let $\sqsubseteq$ be a binary relation on functions from vertices occurring in the nodes of $\mathcal{G}_{\varphi_0}$ to $\mathcal{N} \cup \{\infty\}$ such that $f \sqsubseteq g$ iff $f(v) \le g(v)$ for every vertex v . Let $\perp$ denote the function constantly equal 0.

Lemma 4.3.6 For any state s of $\mathcal{G}_{\varphi_0}$ and functions f, g admissible for s such that $f \sqsubseteq g$, the sequents $F_g(s) \vdash F_f(s)$ and $\overline{F}_g(s) \vdash \overline{F}_f(s)$ are provable.

Proof
The lemma follows directly from the observation that if $f \sqsubseteq g$ then $\{Z_u^j : u \in \{v\}\!\uparrow_s, 1 \le j \le f(u)\} \subseteq \{Z_u^j : u \in \{v\}\!\uparrow_s, 1 \le j \le g(u)\}$, for any vertex v of s. Hence the sequent $\parallel \psi \parallel_s^g \vdash \parallel \psi \parallel_s^f$ is provable for any ψ occurring in s. □

Finally we show the most important fact about our labeling. It will allow us to transfer the labeling from one node of the graph to the other.

Lemma 4.3.7 For any state s of $\mathcal{G}_{\varphi_0}$ and function f, admissible for s and equal 0 for all vertices not occurring in s, we have:

1. if there is an edge from s to t and a vertex w of t s.t. $f(w) = \infty$, then $F_f(s) \vdash ff$ is provable if either w lights green in t or w has an $el(w)$-son in t.

2. $F_f(s) \vdash F_f(t)$ is provable if the tableau rule other then $(\langle\rangle)$ or (or) was used in s and t is the resulting state.

3. $F_f(s) \vdash \langle a \rangle F_f(t)$ is provable if the reduction of the action a was used in s and t is the resulting state.

4. $F_f(s) \vdash F_f(t_1), F_f(t_2)$ is provable if the tableau (or) rule was used and t_1, t_2 are the resulting states.

5. if t is a loop node then we can replace $F_f(t)$ by $\overline{F}_f(t)$ in the above clauses (and similarly $F_f(t_1), F_f(t_2)$ by $\overline{F}(t_1)$ and $\overline{F}(t_2)$ respectively in the last clause).

Proof

Proof will proceed by cases depending on the rule which was used in a node s but first we will show the lemma which will be used in each of the cases.

Lemma 4.3.8 Suppose r is obtained from s by applying all but the last fifth step of the transition function and t is the state resulting after the application of this last step, i.e., some of the vertices of r might light green in t and have all its sons deleted. For any f as in Lemma 4.3.7: if t is a loop node of $\mathcal{G}_{\varphi_0}$ then the sequent $F_f(r) \vdash \overline{F}_f(t)$ is provable and if t is not a loop node then $F_f(r) \vdash F_f(t)$ is provable

Proof
Let Γ be the label of the root of r which is the same as the label of the root of t. By the construction of the states, Γ is also the set of all formulas occurring in r as well as in t. For any formula $\psi \in \Gamma$, the ψ-vertex in t, denoted v_t is an ancestor of the ψ-vertex in r, denoted v_r or vertices v_t and v_r are the same. If they are the same then $\parallel \psi \parallel_r^f = \parallel \psi \parallel_t^f$. Otherwise let $(\delta_1, \ldots, \delta_{d^\mu}) = \parallel v_t \parallel_r^f = \parallel v_t \parallel_t^f$ and $(\delta'_1, \ldots, \delta'_{d^\mu}) = \parallel v_r \parallel_r^f$. Because v_r is a descendant of v_t in r then from Lemma 4.3.4 it follows that $\delta'_i \vdash \delta_i$ is provable for $i = 1, \ldots, d^\mu$. Hence the sequent $\parallel \psi \parallel_r^f \vdash \parallel \psi \parallel_t^f$ is provable by Lemma 3.1.4 because all occurrences of definition constants are positive.

This shows that $F_f(r) \vdash F_f(t)$ is provable. To see why $F_f(r) \vdash \overline{F}_f(t)$ is provable when t is a loop node, first observe that $\parallel \psi \parallel_t^f = \overline{\parallel \psi \parallel}_t^f$ if a ψ-vertex of t is not the green vertex of t.

Let u be the green vertex of t, $\psi \in nl(u)$ and let $el(u) = U_i$ be the i-th μ-defintion constant in $\mathcal{D}_{\varphi_0}$. Because u lights green in t there must be a son u' of u, in r s.t. $\psi \in nl(u')$ and $el(u') = el(u)$. The only difference between $\overline{\parallel u \parallel}_t^f = (\delta_1, \ldots, \delta_{d^\mu})$ and $\parallel u' \parallel_r^f = (\delta'_1, \ldots, \delta'_{d^\mu})$ is at position i. But from the definition of signatures it follows that $\delta_i' \vdash \delta_i$ is provable. Then using once again Lemma 3.1.4 we have that $\| \psi \|_r^f \vdash \overline{\| \psi \|_t^f}$ is provable. This completes the proof. □

Going back to the proof of Lemma 4.3.7 we consider the rules of the tableau system $\mathcal{S}_{ref}$ one by one.

— Suppose the rule applied in s is:

$$\frac{\alpha \wedge \beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{\alpha, \beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

Let r be a state obtained from s by applying all but the last fifth step of the transition function on the input $\alpha, \beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}$. This means that the color of all vertices in s is set to white, the formula $\alpha \wedge \beta$ is replaced by two formulas, α and β. Next, if a formula α or β occurs in a vertex v and occurs in a vertex to the left of it then the formula is deleted from the label of v. Finally the vertices with empty labels are removed. Observe that in r there may still occur vertices which would light green and have all its sons removed if the last step of transition function were applied. We will show that

$$\| \alpha \wedge \beta, \Gamma \|_s^f \vdash \bigwedge \| \alpha, \beta, \Gamma \|_r^f$$

is provable.

For any formula $\varphi \in \Gamma, \varphi \neq \alpha, \varphi \neq \beta$ we have $\| \varphi \|_s^f = \| \varphi \|_r^f$ because φ-vertices in s and r are the same.

If also $\alpha \wedge \beta$-vertex in s is the same as α-vertex in r then clearly $\| \alpha \wedge \beta \|_s^f \vdash \| \alpha \|_r^f$. If it is not the case then it must be because $\alpha \in \Gamma$ and α occurs to the left of $\alpha \wedge \beta$ in s. But then α-vertices in s and r are the same and $\| \alpha \|_s^f = \| \alpha \|_r^f$. We can use similar argument for formula β.

This shows that the sequent $\| \alpha \wedge \beta, \Gamma \|_s^f \vdash \bigwedge \| \alpha, \beta, \Gamma \|_r^f$ is provable. From Lemma 4.3.8 we obtain the hypothesis.

— If the rule applied in s is:

$$\frac{\mu X.\alpha_i(X), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{U_i, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

then, as before, let r be a state obtained form s by applying all but the last fifth step of the transition function on the input $U_i, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}$. We will show that the sequent

$$\| \mu X.\alpha_i(X), \Gamma \|_s^f \vdash \bigwedge \| U_i, \Gamma \|_r^f$$

is provable.

For any $\varphi \in \Gamma$, $\varphi \neq U_i$, φ-vertices in s and r are the same hence $\| \varphi \|_s^f = \| \varphi \|_r^f$. For the formula U_i we have two possibilities. The first possibility is that U_i-vertices in s and r are the same. In this case also $\| U_i \|_s^f = \| U_i \|_r^f$.

Otherwise the U_i-vertex in r is a $\mu X.\alpha(X)$-vertex in s, call it v. By definition 4.3.3, $\| \mu X.\alpha_i(X) \|_s^f = \mu X.\alpha_i(X)[\gamma_d/W_d] \ldots [\gamma_1/W_1]$ where each γ_i is determined using the signature $\| v \|_s^f$. Now $\| U_i \|_r^f = U_i[\gamma_d/W_d] \ldots [\gamma_1/W_1]$ for the same formulas γ_j, $j = 1, \ldots, d$, because v is the U_i-vertex in r.

It is enough to show that $\| \mu X.\alpha_i(X) \|_s^f = \| U_i \|_r^f$. By construction of the definition list, only constants older then U_i can appear in $\mu X.\alpha_i(X)$. From Lemma 4.1.4 it follows that there is no ancestor of v, with edge label U_i. This means that in signature $\| v \|_s^f = (\delta_1, \ldots, \delta_{d^\mu})$, its i-th coordinate δ_i is $\mu X.\alpha_i(X)$. Hence

$$\begin{aligned} \| \mu X.\alpha_i(X) \|_s^f &= \mu X.\alpha_i(X)[\gamma_j/W_j] \ldots [\gamma_1/W_1] \\ \| U_i \|_r^f &= U_i[\mu X.\alpha_i(X)/U_i][\gamma_j/W_j] \ldots [\gamma_1/W_1] \end{aligned}$$

where W_j is the youngest definition constant older than U_i.

Having shown that $\| \mu X.\alpha_i(X), \Gamma \|_s^f \vdash \bigwedge \| U_i, \Gamma \|_r^f$ is provable it is enough to use Lemma 4.3.8 to complete this case.

— If the rule applied in s is:

$$\frac{\nu X.\beta_i(X), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{V_i, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

then the reasoning is entirely the same as in the preceding case

— Suppose the rule applied in s is:

$$\frac{U_i, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{\alpha_i(U_i), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

where U_i is the i-th μ-constant in the definition list $\mathcal{D}_{\varphi_0}$. Let, as before, r be a state obtained form s by applying all but the last fifth step of the transition function on the input $\alpha_i(U_i), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}$ and t be a state obtained by applying this last step to r.

Till now we have not considered clause 1 of the lemma's hypothesis because in order to satisfy the premise of the clause some new vertex must be created or light green. To light green, a vertex must first have some sons. If a vertex w has no $el(w)$-sons in s and lights green in t then it must have some $el(w)$-son in an intermediate state r. The rule we consider now is the only rule when this can happen.

Let v be the lowest vertex in s such that $U_i \in nl(v)$ and $el(v)$ is not younger than U_i, or let v be the root of s if there is no such vertex. By the definition of transition function only one new vertex is added and it becomes a son of v. Hence for the conditions of clause 1 to be satisfied, v must have a U_i son in r, $f(v) = \infty$ and $el(v) = U_i$.

If $f(v) = \infty$ then from the definition of a signature the i-th element of $\| v \|_s^f$ is *ff* and, because $U_i \in nl(v)$, we have that $ff \in F_f(s)$ and $F_f(s) \vdash ff$ is provable. This completes the proof for the case when the assumptions of clause 1 are satisfied.

We focus now on the proof of clause 2. We will show that:

$$\| U_i, \Gamma \|_s^f \vdash \bigwedge \| \alpha_i(U_i), \Gamma \|_r^f$$

is provable.

For any $\varphi \in \Gamma$, such that $\varphi \neq \alpha_i(U_i)$, the φ-vertices in s and r are the same hence $\| \varphi \|_s^f = \| \varphi \|_r^f$. If also $\alpha_i(U_i)$-vertices of r and s are the same then we are done.

If this is not the case then let w be the U_i-vertex in s and let v be the lowest vertex in s such that $U_i \in nl(v)$ and $el(v)$ not younger U_i or let v be the root of s if there is no such vertex. By the definition of transition function the $\alpha(U_i)$-vertex u in r, is a son of the vertex v.

Observe that only definition constants not younger than U_i can appear in $\alpha_i(U_i)$. Hence by definition of the signature:

$$\begin{aligned} \| U_i \|_s^f &= U_i[\gamma_k/U_i] \ldots [\gamma_1/W_1] \\ \| \alpha_i(U_i) \|_r^f &= \alpha_i(U_i)[\gamma'_k/U_i] \ldots [\gamma'_1/W_1] \end{aligned}$$

where, according to Definition 4.3.3, $\gamma_1, \ldots, \gamma_k, \gamma'_1, \ldots, \gamma'_k$ are defined using the first i elements of signatures $\parallel w \parallel^f_r$ and $\parallel u \parallel^f_r$ respectively. Because v is the lowest ancestor of w s.t. $el(v)$ is not younger U_i, the signatures $\parallel v \parallel^f_s$ and $\parallel w \parallel^f_s$ are the same up to position i. Hence we can compare $\parallel u \parallel^f_r$ with $\parallel v \parallel^f_s$ not with $\parallel w \parallel^f_s$.

Because u is a son of v and $el(u) = U_i$, the signatures of $\parallel v \parallel^f_s$ and $\parallel u \parallel^f_r$ may differ at most on the position i. That is, $\gamma_l = \gamma'_l$ for $l \in \{1, \ldots, k-1\}$. Let us denote by δ_i and δ'_i the formulas at the i-th position in the signatures $\parallel v \parallel^f_t$ and $\parallel u \parallel^f_t$ respectively.

We have already considered the case when $f(v) = \infty$ and v has an $el(v)$-son in r when we took clause 1 into account. Hence we can assume that either $f(v) < \infty$ or $el(v) \neq U_i$. Because u is a new vertex by assumption of the lemma we know that $f(u) = 0$ and the i-th element of $\parallel u \parallel^f_r$ is

$$\delta'_i = \mu X.\mathcal{V}' \wedge \alpha_i(X) \quad \text{where} \quad \mathcal{V}' = \bigwedge\{Z^j_w : w \in \{u\}\!\uparrow_s, 1 \leq j \leq f(w)\}$$

If $el(v) \neq U_i$ then $\mathcal{V}' = \mathit{true}$ and $\delta_i = \mu X.\alpha_i(X) = \delta'_i$ in this case. Otherwise we have two cases depending on whether $f(v) = 0$ or $\infty > f(v) > 0$:

- If $f(v) = 0$ then

  $$\delta_i = \mu X.\mathcal{V} \wedge \alpha_i(X) \qquad \text{where } \mathcal{V} = \bigwedge\{Z^j_w : w \in \{v\}\!\uparrow_s, 1 \leq j \leq f(w)\}$$

  But of course $\mathcal{V}' = \mathcal{V}$ and

  $$\begin{aligned} \parallel U_i \parallel^f_s &= \mu X.\mathcal{V} \wedge \alpha_i(X)[\gamma_{k-1}/W_{k-1}]..[\gamma_1/W_1] \\ \parallel \alpha_i(U_i) \parallel^f_r &= \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))[\gamma_{k-1}/W_{k-1}]..[\gamma_1/W_1] \end{aligned}$$

  Hence the sequent $\parallel U_i \parallel^f_s \vdash \parallel \alpha_i(U_i) \parallel^f_r$ is provable in this case.

- If $f(v) > 0$ then

  $$\delta_i = \mathcal{V}_1 \wedge \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))$$

  where $\mathcal{V}$ is as above and $\mathcal{V}_1 = \mathcal{V} \wedge Z^{f(v)}_v$. Furthermore we have:

  $$\begin{aligned} \parallel U_i \parallel^f_s &= \mathcal{V}_1 \wedge \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))[\gamma_{k-1}/W_{k-1}]..[\gamma_1/W_1] \\ \parallel \alpha_i(U_i) \parallel^f_r &= \alpha_i(\mu X.\mathcal{V} \wedge \alpha_i(X))[\gamma_{k-1}/W_{k-1}]..[\gamma_1/W_1] \end{aligned}$$

  Hence the sequent $\parallel U_i \parallel^f_s \vdash \parallel \alpha_i(U_i) \parallel^f_r$ is provable also in this case.

Having shown that the sequent $\| U_i, \Gamma \|_s^f \vdash \bigwedge \| \alpha_i(U_i), \Gamma \|_r^f$ is provable we apply Lemma 4.3.8 to show the desired conclusion.

— If the rule applied is:

$$\frac{V, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{\beta(V), \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

where V is a ν-constant then the reasoning is the same as in the preceding case.

— If the rule applied in s is

$$\frac{\langle a\rangle\alpha, \Gamma \vdash_{\mathcal{D}}}{\alpha, \{\beta : [a]\beta \in \Gamma\} \vdash_{\mathcal{D}_{\varphi_0}}}$$

then let, as before, r be a state obtained form s by applying all but the last fifth step of the transition function on the input $\alpha, \{\beta : [a]\beta \in \Gamma\} \vdash_{\mathcal{D}_{\varphi_0}}$. To show that

$$\| \langle a\rangle\alpha, \Gamma \|_s^f \vdash \langle a\rangle \bigwedge \| \alpha, \{\beta : [a]\beta \in \Gamma\} \|_r^f$$

is provable it is enough to show that we can prove:

$$\| \langle a\rangle\alpha, \Gamma \|_s^f \vdash \langle a\rangle \| \alpha \|_r^f \wedge \bigwedge\{[a]\| \beta \|_r^f : [a]\beta \in \Gamma\}$$

For each formula $\varphi \in \{\beta : [a]\beta \in \Gamma\}, \varphi \neq \alpha$ we have that the $[a]\varphi$-vertex in s is the same as the φ-vertex in r. Hence $\| [a]\varphi \|_s^f = [a]\| \varphi \|_r^f$.

Looking at the formula α, the α-vertex in r can be either the same as the $[a]\alpha$-vertex in s (if $[a]\alpha \in \Gamma$ and occurred to the left of $\langle a\rangle\alpha$ in s) or the same as the $\langle a\rangle\alpha$-vertex in s. In the first case $\| [a]\alpha \|_s^f = [a]\| \alpha \|_r^f$. In the second $\| \langle a\rangle\alpha \|_s^f = \langle a\rangle\| \alpha \|_r^f$.

As before we use Lemma 4.3.8 to finish the proof of the hypothesis.

— If the rule applied in s is:

$$\frac{\alpha \vee \beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}{\alpha, \Gamma \vdash_{\mathcal{D}_{\varphi_0}} \quad \beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}}$$

then let r_1 and r_2 be the states obtained from s by applying all but the last fifth step of the transition function on the inputs $\alpha, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}$ and $\beta, \Gamma \vdash_{\mathcal{D}_{\varphi_0}}$ respectively. We will show how to prove:

$$\| \alpha \vee \beta, \Gamma \|_s^f \vdash \bigwedge \| \alpha, \Gamma \|_{r_1}^f, \bigwedge \| \beta, \Gamma \|_{r_2}^f$$

For every formula $\varphi \in \Gamma$, such that $\varphi \neq \alpha, \varphi \neq \beta$ we know that φ-vertices r_1 and r_2, in s are the same, hence $\| \varphi \|_s^f = \| \varphi \|_{r_1}^f = \| \varphi \|_{r_2}^f$. If the α-vertex in r_1 and the β-vertex in r_2 are the same as the $\alpha \vee \beta$-vertex in s then of course $\| \alpha \vee \beta \|_s^f = \| \alpha \|_{r_1}^f \vee \| \beta \|_{r_2}^f$ and we are done.

If the α-vertex in r_1 is different from the $\alpha \vee \beta$-vertex in s then it must be the case that $\alpha \in \Gamma$ and the α-vertices in s and r_1 are the same. Then of course $\| \alpha \|_s^f = \| \alpha \|_{r_1}^f$. The same reasoning shows that $\| \alpha \vee \beta, \Gamma \|_s^f \vdash \| \beta, \Gamma \|_{r_2}^f$ is provable. An application of Lemma 4.3.8 gives us the desired conclusion.

□

4.4 Completeness proof

In this final section we describe the construction of a proof of the sequent $\varphi_0 \vdash$ in our system. As we will see this proof can be "read" from the tree $\mathcal{T}_{\varphi_0}$, i.e., a finite tree with back edges labeled with states of $\mathcal{A}_{\varphi_0}$ as described in definition 4.2.3. This means that using our coding of states the construction of the proof of $\varphi_0 \vdash$ will be directed by the structure of $\mathcal{T}_{\varphi_0}$.

$\mathcal{T}_{\varphi_0} = \langle T, S \rangle$, as we have already mentioned during its construction, is not a real tree but rather a finite tree with "loops", i.e., back edges from leaves to some ancestors. Sometimes we prefer to treat $\mathcal{T}_{\varphi_0}$ as a finite tree and when we say, for example, that a node is an ancestor of some other node we mean that it is an ancestor in the tree without back edges. We write $F_f(n)$ instead of $F_f(S(n))$ and similarly $\overline{F}_f(n)$ for $\overline{F}_f(S(n))$. We will also use one more abbreviation, let $a_1, \ldots, a_k$ be all actions occurring in φ_0, then P will stand for the PDL program $a_1 \cup \ldots \cup a_k$.

Let us assume that $\varphi_0 \vdash$ is unprovable. We will show that from this assumption it follows that we can find an infinite path in $\mathcal{T}_{\varphi_0}$ without taking back edges, i.e. that $\mathcal{T}_{\varphi_0}$ is infinite, which contradicts Fact 4.2.4. This infinite path will start in n_0, the root of $\mathcal{T}_{\varphi_0}$. The following lemma will allow us to extend the path

Lemma 4.4.1 Suppose that we have constructed a path up to a node m of $\mathcal{T}_{\varphi_0}$ and to each node m' on the path we have assigned a function $f_{m'}$, admissible for $S(m')$. Assume also that the sequent:

$$F_{f_m}(m) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \tag{4.1}$$

is not provable, and moreover the following conditions hold:

I1 $\{v : f_m(v) > 0\} = AV(m)$

I2 $\{n_1, .., n_j\} = \{n : n \in AN(m), \infty > f_m(v) > 0,\ v$ green vertex of $n\}$

I3 for each $i = 1, \ldots, j$, f_i is the function assigned to n_i, $f_i \sqsubseteq f_m$.

Then we can find a son n of m, in the tree $\mathcal{T}_{\varphi_0}$ without back edges, and an admissible function f_n such that for node n sequent similar to (4.1) satisfying conditions I1, I2 and I3 can be constructed. This sequent is uniquely determined by n and f_n.

If n_0 is the root of $\mathcal{T}_{\varphi_0}$ then $S(n_0)$, the start state of $\mathcal{A}_{\varphi_0}$, consists only of the root labeled $\{\varphi_0\}$. Of course n_0 cannot be a loop node because it has no vertices to light green. From the definition of the coding follows that $F_\perp(n_0) = \varphi_0$. Therefore, if $\varphi_0 \vdash$ is unprovable, we can start our path from the node n_0 with the function $\perp$ and the sequent $F_\perp(n_0) \vdash$. It remains to prove Lemma 4.4.1 to obtain a contradiction. We will call conditions I1, I2 and I3 invariants.

Proof of Lemma 4.4.1

Let us suppose that we have extended our path up to a node m and have an admissible function f and an unprovable sequent:

$$F_f(m) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \tag{4.2}$$

which satisfy all the conditions of Lemma 4.4.1. We will show that there must be a son n of m, in $\mathcal{T}_{\varphi_0}$ without back edges, for which an unprovable sequent satisfying all the invariants can be constructed. We reason by cases depending on the status of node m.

1 If m is a leaf of $\mathcal{T}_{\varphi_0}$, i.e. there are no edges of any kind going from m, then by definition of the refutation the label of the leaf is a tableau axiom, i.e., a tableau sequent where some constant p and its negation occur. In this case $p, \neg p \in F_f(m)$. We obtain a contradiction with the unprovability of (4.2).

2 If there are edges going from m then by Lemma 4.2.6 and invariant I1, f is zero for all vertices not occurring in $S(m)$. This means that we can use Lemma 4.3.7 to find an edge leading from m to some n such that if no back edge leads to n, the sequent:

$$F_f(n) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \tag{4.3}$$

is unprovable and if there are back edges leading to n, the sequent

$$\overline{F}_f(n) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \tag{4.4}$$

is unprovable. Observe that in the last case $S(n)$ is a loop node.

To finish the proof it is enough to construct an unprovable sequent for n satisfying all the invariants. It is also important to make sure that n is not an ancestor of m in the tree $\mathcal{T}_{\varphi_0}$ without back edges.

2.a If no back edges lead to n then it is easy to force the sequent (4.3) to satisfy all the invariants.

Because $AV(n) \subseteq AV(m)$, by Lemma 4.2.6, invariant I1 will be satisfied if we just make f equal 0 for all vertices not occurring in $AV(n)$. Let us call the resulting function g. Clearly g is admissible for $S(n)$, because otherwise, by first clause of Lemma 4.3.7 the sequent $F_f(m) \vdash$ would be provable. We have also that $g \sqsubseteq f$, hence the sequent $F_f(n) \vdash F_g(n)$ is provable by Lemma 4.3.6.

Because $AN(n) \subseteq AN(m)$, we must throw away some formulas from the right side of the sequent (4.3) to satisfy invariant I2. Let $\{n_{l_1}, \ldots, n_{l_k}\} = \{n' : n' \in AN(n), \infty > g(v) > 0,\ v \text{ is the green vertex of } n'\}$.

To see why invariant I3 still holds one must observe that directly from the definition of $AV(n)$ it follows that if $n_i \in AN(n)$ then $AV(n_i) \subseteq AV(n)$. Hence $g(v) = f(v)$ for every $v \in AV(n_i)$ and because $f_i \sqsubseteq f$ then $f_i \sqsubseteq g$. Therefore the sequent

$$F_g(n) \vdash \langle P^* \rangle \overline{F}_{f_{l_1}}(n_{l_1}), \ldots, \langle P^* \rangle \overline{F}_{f_{l_k}}(n_{l_k})$$

satisfies all the invariants and is unprovable if only sequent (4.3) is.

2.b If there are back arcs leading to n then the operation is a little bit more difficult. Let v_n be the green vertex of $S(n)$. The first observation is that $f(v_n) \neq \infty$. This follows directly from Lemma 4.3.7 because otherwise $F_f(m) \vdash \mathit{ff}$ would be provable.

2.b.i We must show that n is really a descendant of m. Lemma 4.3.7 guarantees only that there is an edge from m to n in $\mathcal{T}_{\varphi_0}$. But this edge can be a back edge, i.e. n can be an ancestor of m in the tree $\mathcal{T}_{\varphi_0}$ without back edges.

Suppose that we have taken a back edge from m to n. According to the definition of $AN(m)$, it holds that $n \in AN(m)$ and from invariant I1 it follows that $f(v_n) > 0$. Because $f(v_n) \neq \infty$ then also $f(v_n) < \infty$.

Now from invariant I2 we know that $n \in \{n_1, \ldots, n_j\}$, suppose $n = n_k$ for some k. From invariant I3 we have $f_k \sqsubseteq f$, hence by Lemma 4.3.6 the sequent

$$\overline{F}_f(n) \vdash \overline{F}_{f_k}(n_k)$$

is provable which implies that sequent (4.4) is provable, a contradiction.

2.b.ii So we know that n is not an ancestor of m. We must construct an unprovable sequent for n satisfying all the invariants. The difficulty we face is that in this case, by Lemma 4.2.6, $AN(n) \subseteq AN(m) \cup \{n\}$, i.e. a new node to take care of is added. One more difficulty is that the green vertex v_n of n, may be already in use, i.e., $v_n \subseteq AV(m)$. This last observation is the reason why the range of admissible functions is $\mathcal{N} \cup \infty$ rather then just $\{0, 1, \infty\}$. Intuitively $f(v) = i > 0$ means that v is the green vertex of i loop nodes among those we consider.

For simplicity we can assume that $f(v) = 0$ for all $v \notin AV(n)$. If it is not the case then we can always take a function h equal to f on $AV(n)$ and equal 0 for other vertices. Then because $h \sqsubseteq f$ we would have that $\overline{F}_f(n) \vdash \overline{F}_h(n)$ Hence from the assumption that the sequent (4.4) is unprovable, the sequent:

$$\overline{F}_h(n) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), .., \langle P^* \rangle \overline{F}_{f_j}(n_j)$$

would be unprovable.

We know that $f(v_n) < \infty$ and we can define two functions g, g':

- $g(v_n) = f(v_n) + 1$, $g'(v_n) = \infty$.
- $g(u) = g'(u) = f(u)$ for $u \neq v_n$,

Clearly functions g and g' are admissible for $S(n)$ because v_n is the green vertex of $S(n)$ which means that it lights green in $S(n)$ and has no sons in this state.

We would like to show that from the assumption that the sequent (4.4) is unprovable we can deduce that one of the sequents

$$F_{g'}(n) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \tag{4.5}$$

$$F_g(n) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j), \langle P^* \rangle \overline{F}_g(n) \tag{4.6}$$

must be unprovable. The reason is that, as we will show, sequents (4.5) and (4.6) are assumptions and (4.4) is the conclusion in an instance of the (*ind*) rule.

Let $\Delta = nl(v_n)$ and let Γ be the set of all formulas occurring in $S(n)$ which are not in $nl(v_n)$. We have

$$F_h(n) = \bigwedge \| \Gamma \cup \Delta \|^h_{S(n)} = \bigwedge \| \Gamma \|^h_{S(n)} \wedge \bigwedge \| \Delta \|^h_{S(n)} \quad \text{for} \quad h = g, g'$$
$$\overline{F}_h(n) = \bigwedge \overline{\| \Gamma \cup \Delta \|}^h_{S(n)} = \bigwedge \overline{\| \Gamma \|}^h_{S(n)} \wedge \bigwedge \overline{\| \Delta \|}^h_{S(n)} \quad \text{for} \quad h = f, g$$

But

$$\overline{\| \Gamma \|}^f_{S(n)} = \| \Gamma \|^g_{S(n)} = \overline{\| \Gamma \|}^g_{S(n)} = \| \Gamma \|^{g'}_{S(n)}$$

because Γ is the set of formulas not occurring in $nl(v_n)$ and the signatures $\overline{\| u \|}^f_{S(n)}$, $\| u \|^g_{S(n)}$, $\overline{\| u \|}^g_{S(n)}$, $\| u \|^{g'}_{S(n)}$ differ only when $u = v_n$.

To compare

$$\| \Delta \|^h_{S(n)} = \{\| \varphi \|^h_{S(n)} : \varphi \in \Delta\}, \quad h = g, g'$$
$$\overline{\| \Delta \|}^h_{S(n)} = \{\overline{\| \varphi \|}^h_{S(n)} : \varphi \in \Delta\}, \quad h = f, g$$

we must look at

$$\| v_n \|^h_{S(n)} = (\delta^h_1, \ldots, \delta^h_{d^\mu}) \quad h = g, g'$$
$$\overline{\| v_n \|}^h_{S(n)} = (\gamma^h_1, \ldots, \gamma^h_{d^\mu}) \quad h = f, g$$

because v_n is the lowest vertex in $S(n)$ containing formulas from Δ. From the definition of signatures it follows that the only difference between these signatures is at position i s.t. $U_i = el(v_n)$ is the i-th definition constant in $\mathcal{D}_{\varphi_0}$, namely

$$\begin{array}{ll} \delta^{g'}_i = \mathit{ff} & \delta^g_i = \mathcal{V} \wedge \alpha_i(\mu X.Z^{g(v_n)}_{v_n} \wedge \mathcal{V} \wedge \alpha_i(X)) \\ \gamma^f_i = \mu X.\mathcal{V} \wedge \alpha_i(X) & \gamma^g_i = \mu X.Z^{g(v_n)}_{v_n} \wedge \mathcal{V} \wedge \alpha_i(X) \end{array}$$

where

$$\begin{aligned} \mathcal{V} &= \bigwedge \{Z^j_w : w \in \{v_n\}\!\uparrow_{S(n)}, 1 \leq j \leq f(w)\} \\ &= \bigwedge \Big(\{Z^j_w : w \in \{v_n\}\!\uparrow_{S(n)}, 1 \leq j \leq g(w)\} \setminus \{Z^{g(v_n)}_{v_n}\}\Big) \end{aligned}$$

Using our notation that $\mathcal{D}_{\varphi_0} = (W_1 = \gamma_1) \ldots (W_d = \gamma_d)$ we can see that

$$\begin{aligned} \overline{F}_f(n) &= \| \Gamma \|^f_{S(n)} \wedge \bigwedge \{\delta[\gamma_d/W_d]..[\mu X.\alpha(X)/U_i]..[\gamma_1/W_1] : \delta \in \Delta\} \\ F_{g'}(n) &= \| \Gamma \|^f_{S(n)} \wedge \bigwedge \{\delta[\gamma_d/W_d]..[\mathit{ff}/U_i]..[\gamma_1/W_1] : \delta \in \Delta\} \\ F_g(n) &= \| \Gamma \|^f_{S(n)} \wedge \bigwedge \{\delta[\gamma_d/W_d]..[\alpha(\mu X.Z \wedge \alpha(X))/U_i]..[\gamma_1/W_1] : \delta \in \Delta\} \\ \overline{F}_g(n) &= \| \Gamma \|^f_{S(n)} \wedge \bigwedge \{\delta[\gamma_d/W_d]..[\mu X.Z \wedge \alpha(X)/U_i]..[\gamma_1/W_1] : \delta \in \Delta\} \end{aligned}$$

where $\alpha(X) = \mathcal{V} \wedge \alpha_i(X)$, $Z = Z_{v_n}^{g(v_n)}$ and $\gamma_d, \ldots, \gamma_1$ are formulas constructed from $\| v_n \|_{S(n)}^{f}$ according to Definition 4.3.3. Hence sequents 4.4, 4.5 and 4.6 can be respectively presented as

$$\begin{array}{r}
\psi(\mu X.\beta(X)) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \\
\psi(\mathit{ff}) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j) \\
\psi(\beta(\mu X.Z \wedge \beta(x))) \vdash \langle P^* \rangle \overline{F}_{f_1}(n_1), \ldots, \langle P^* \rangle \overline{F}_{f_j}(n_j), \langle P^* \rangle \psi(\mu X.Z \wedge \beta(x))
\end{array}$$

for suitable ψ and β. Of course $Z = Z_i^{g(v_n)}$ is a new variable because $g(v_n) = f(v_n) + 1$ and from invariant I3 it follows that $f_k(v_n) \leq f(v_n)$ for $k = 1, .., j$.

In conclusion, sequents (4.5) and (4.6) are assumptions and (4.4) is the conclusion in an instance of (ind) rule. This means that either (4.5) or (4.6) must be unprovable.

If sequent 4.6 is unprovable then we transform it to the sequent satisfying all the invariants in a similar way as we did in the case 2a.

Because we have assumed that $f(v) = 0$ for all $v \notin AV(n)$ invariant I1 is already satisfied.

Because $AN(n) \subseteq AN(m) \cup \{n\}$ we must throw away some formulas from the right side of the sequent 4.6 to satisfy invariant I2. Let $\{n_{l_1}, \ldots, n_{l_k}\} = \{n' : n' \in AN(n) \setminus \{n\}, \infty > f(v) > 0,\ v \text{ is the green vertex of } n'\}$.

To see why invariant I3 still holds one must observe that directly from the definition of $AV(n)$ it follows that if $n_i \in AN(n)$ then $AV(n_i) \subseteq AV(n)$. Hence for any vertex v if $f_i(v) > 0$ then $g(v) > 0$. Thus because $f_i \sqsubseteq f$ holds then also $f_i \sqsubseteq g$ holds. Therefore the sequent

$$F_g(n) \vdash \langle P^* \rangle \overline{F}_{f_{l_1}}(n_{l_1}), .., \langle P^* \rangle \overline{F}_{f_{l_k}}(n_{l_k}), \langle P^* \rangle \overline{F}_g(n)$$

satisfies all the invariants and is unprovable. The function g is the function assigned to n.

If sequent (4.5) is unprovable then the transformations are exactly the same as above. The function assigned to n in this case will be the function g'.

We can summarize this section with the thorem:

Theorem 4.4.2 (Completeness) *For ever unsatisfiable formula φ_0 of the μ-calculus, the sequent $\varphi_0 \vdash$ is provable.*

Proof
By Proposition 3.1.6 we can restrict ourselfs to possitive guarded formulas.

Suppose conversely that a possitive guarded formula φ_0 is unsatisfiable but the sequent $\varphi_0 \vdash$ is unporvable.

To the root n_0 of $\mathcal{T}_{\varphi_0}$, as defined in Definition 4.2.3, we can assign an addmissable function $\perp$. It is easy to see that $\varphi_0 = F_\perp(n_0)$, where n_0 is the root of $\mathcal{T}_{\varphi_0}$. It is also strightforward to check that the seuqent $F_\perp(n_0)\vdash$ satisfies all the conditions of Lemma 4.4.1.

This means that we can find a son n_1 of n_0 and an addmissable function f_{n_1} such that a sequent, as (4.1) but for node n_1, will be unprovable and invariants I1, I2 and I3 will be satisfied. Then we can find a son n_2 of n_1 with the same property and so on. This way we can construct an infinite path in the tree $\mathcal{T}_{\varphi_0}$ without back edges. This is a contradiction with Fact 4.2.4. $\square$

Chapter 5

Conclusions

Propositional μ-calculus is a very interesting logic from the viewpoint of program verification and specification. It is as expressive as very strong monadic second order logic $S2S$ and yet checking the satisfiability of the μ-calculus formulas is as easy as for much weaker logic PDL. This can be done in EXPTIME which is surprisingly close to the lower bound on checking the satisfiability of classical propositional logic. In contrast checking satisfiability of $S2S$ formulas is not elementary.

The other property of the μ-calculus which facilitates description of program properties is that the recursion operator is a basic operator in this logic. This means that translations of programs into formulas can be very natural.

These features of the μ-calculus make it a logic widely used for program verification. The problem of efficient model checking in the μ-calculus was and still is an important issue from the practical as well as theoretical point of view.

If program verification using model-checking in the μ-calculus received a great deal of attention it was not so with "proof-theoretic" approach to this task. In this approach one codes a program as a formula of a logic which reduces the problem of verifying that the program has a property α to proving that α follows from the formula describing the program.

One of the obvious reasons for this was the lack of a complete axiom system for the μ-calculus. This obstacle was solved in this work and the presented system is, in our opinion, quite "user friendly". With such a system it may be worth revising the idea of Manna and Pnueli [28] which may result in some new methods for program verification. One immediate gain is that while model checkers are completely automatic, provers can be made to interact with the user who can help in what is essentially an exponential task.

We hope that the presented axiomatization gives or will give some insight into the μ-calculus. Because of close relationship between μ-calculus, automata on infinite objects and $S2S$ it may also help in better understanding of these theories. It must be mentioned that we do not know whether Kozen's axiomatization of the μ-calculus [24] is complete or not. The question of a complete axiom system for $S2S$ also remains open.

The presented method of completeness proof is also quite different from that used for other propositional logics of programs. It must be mentioned that there is a degree of resemblance between this proof and the proof of Kozen [24] for the fragment of the μ-calculus.

Our approach has this advantage over completeness proofs by Henkin method that it is direct. That is for a valid formula it gives a method for constructing a proof of it. Of course a proof constructed by this method can be sometimes quite large.

The size of the proof is mainly due to restrictions on a proof shape. If effectiveness is important, refutations can serve as a proof "substitute" with their good exponential bound on their size. As completeness proof shows, refutations have all the information needed to construct a proof. We think that this information is also quite apparent, i.e. refutations can be as human readable as proofs.

Further research topics are numerous. One could try to obtain complete axiomatizations of CTL* or ECTL* possibly by modification of the presented method. Axiomatizations of this logics are still interesting because no simple and direct codings from CTL* or ECTL* to the μ-calculus are known.

There are various decidable extensions of $S2S$ usually obtained by adding some new predicates to the logic. Similar predicates can be added to the μ-calculus. Some of this extensions can be interesting both from the theoretical and practical point of view. For most of them it is not even clear whether there is a finite model property, not to mention complexity of decision procedures, axiomatization and other more specific issues.

The other extension, suggested by the induction rule (ind), is to introduce a new sort of variables $\{\tau, \varsigma, \ldots\}$ which we call *index variables.* Then we add to the μ-calculus a new construct $\tau X.\alpha(X)$, let us call it *approximation formula.* Index variables range in the model over natural numbers (or over all ordinals if one prefers to) and the intended meaning of $\tau X.\alpha(X)$ is the n-fold iteration of $\alpha(X)$ on ff, where n is the value of τ.

In this extended calculus we can formulate an induction rule similar to (ind) but without need to use additional fresh variable. The obtained system can be shown complete for the formulas of the μ-calculus but it is not clear whether it is complete for the whole extension. On the other hand if we restrict valuation of index variables to natural numbers then the small model property for such a logic is a consequence of the small model property of the μ-calculus. It remains to be seen whether extensions of this sort possess some interesting properties.

The ability to perform "surgery" on models, i.e. cutting or expanding models in such a way that the truth of some formulas remain intact, can be very important especially in such topics as model checking. Traditionally one used sets of formulas to distinguish important properties of states of a model. This approach has some drawbacks when used to such an expressive logic as the μ-calculus. Using trees of formulas, representing states of an automaton, as opposed to sets of formulas can possibly solve some of the problems.

Acknowledgments

Unfortunately it is not possible to mention all the people who helped me in work on this thesis, but I would like to mention few of them. I should start with Leszek Holenderski, who helped me from the very beginning of my studies, being my first guide to the computer science world. His influence is hard to overestimate. Professor Grażyna Mirkowska, my first supervisor, introduced me to the modal logics and served with her guidance and advice during my years of study. I would like also to thank professor Dexter Kozen for his introduction to the μ-calculus. Damian Niwiński was the one from whom I learned automata theory. His openness and ability to share the knowledge had great influence on the ideas of my thesis. Professor Jerzy Tiuryn guided development of the thesis and was always ready with help and advise. I am greatly obliged for his time and effort he devoted to my work. Finally I would like to thank all the people from the professor Tiuryn's group at the Institute of Informatics, Warsaw University, for creating wonderful environment to learn and work in.

Bibliography

[1] J.Richard Büchi. On the decision method in restricted second-order arithmetic. In *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford Univ. Press, 1960.

[2] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[3] Mads Dam. CTL* and ECTL* as a fragments of the modal μ-calculus. In *CAAP'92*, volume 581 of *LNCS*, pages 145–165, 1992.

[4] E. Allen Emereson. Temporal and modal logic. In J.van Leeuven, editor, *Handbook of Theoretical Computer Science Vol.B*, pages 995–1072. Elsvier, 1990.

[5] E. Allen Emerson. Automata, tableaux and temporal logic. In *Colledge Conference on Logic of Programs*, volume 193 of *LNCS*. Springer-Verlag, 1985.

[6] E. Allen Emerson and C.L.Lei. Efficient model checking in fragments of propositional mu-calculus. In *First IEEE Symp. on Logic in Computer Science*, pages 267–278, 1986.

[7] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. In *29th IEEE Symp. on Foundations of Computer Science*, 1988.

[8] E. Allen Emerson and Charanjit S. Jutla. On simultaneously determinizing and complementing ω-automata. In *LICS'89*, 1989.

[9] E.Allen Emerson and J. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33:175–211, 1986.

[10] E.Allen Emerson and J.Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[11] E.Allen Emerson and C.S. Jutla. Tree automata, mu calculus and determinacy. In *Proc. FOCS 91*, 1991.

[12] M.J. Fisher and R.E. Ladner. Propositional modal logic of programs. In *9th ACM Ann. Aymp. on Theory of Computing*, pages 286–294, 1977.

[13] M.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[14] D. Gabbay. Axiomatizations of logics of programs. Unpublished manuscript, Bar-Ilan Univ., 1977.

[15] D. Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1981.

[16] Yuri Gurevich and Leo Harrington. Trees, automata and games. *Journal of the ACM*, 1982.

[17] D. Harel, D Kozen, and R. Parikh. Process logic: Expressiveness, decidability and completeness. *Journal of Computer and System Sciences*, 25:144–201, 1982.

[18] David Harel. Dynamic logic. In *Handbook of Philosophical Logic Vol II*, pages 497–604. D.Reidel Publishing Company, 1984.

[19] M. Henessy and R. Milner. Algebraic laws for nondeterminism and concurency. *Journal of the ACM*, 32:137–161, 1985.

[20] P. Hitchcock and D. Park. Induction rules and termination proofs. volume Proc. 1st Internat. Coll. on Autoamta, Languages and Programming, pages 225–251, 1973.

[21] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.

[22] Nils Klarund. Progress measures, immediate determinacy and a subset construction for tree automata. In *IEEE LICS*, pages 382–393, 1992.

[23] P.M.W. Knijnenburg and J. van Leeuwen. On models for propositional dynamic logic. *Theoretical Computer Science*, 91:181–203, 1991.

[24] Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[25] Dexter Kozen. A completeness theorem for Kleene algebras and the algebras of regular events. In *IEEE LICS*, pages 214–225, 1992.

[26] Dexter Kozen and R. Parikh. An elementary proof of the completeness of the PDL. *Theoretical Computer Science*, 14:113–118, 1981.

[27] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In J.van Leeuven, editor, *Handbook of Theoretical Computer Science Vol.B*, pages 789–840. Elsvier, 1990.

[28] Z. Manna and A. Pnueli. Verification of the concurrent programs: the temporal framework. In R.Boyer and J.Moore, editors, *The Correctness Problem in Computer Scince*, pages 215–273. Academic Press, 1981.

[29] D.A. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.

[30] R. McNaughton. Testing and generating infinte sequences by a finite automaton. *Information & Computation*, 9:521–530, 1966.

[31] A.R. Meyer. Weak monadic second order theory of one successor is not elementary. In *Lecture Notes in Mathematics*, volume 453, pages 132–154. Springer-Verlag, 1975.

[32] Grażyna Mirkowska. PAL — propositional algorithmic logic. In *LNCS 125*, pages 23–101. Springer-Verlag, 1981.

[33] Y.N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Logic*. North-Holand, 1980.

[34] A.A. Muchnik. Games on infinite trees and automata with dead ends. *Semiotics and Information*, 24:17–44, 1984. in Russian.

[35] D. Niwiński and I. Walukiewicz. Games for μ-calculus. *To appear.*

[36] Damian Niwiński. The propositional μ-calculus is more expressive than the PDL with looping. Unpublished manuscript, 1984.

[37] Damian Niwiński. On fixed-point clones. In *Proc. 13th ICALP*, volume 226 of *LNCS*, pages 464–473, 1986.

[38] Damian Niwiński. Fixed points vs. infinite generation. In *Proc. 3rd. IEEE LICS*, pages 402–409, 1988.

[39] R. Parikh. The completeness of propositional dynamic logic. In *Proc 7th Symp on Mathematical Foundations of Computer Science*, volume 64 of *LNCS*, pages 403–415, 1978.

[40] D. Park. Finiteness is μ-ineffable. *Theoretical Computer Science*, 3:173–181, 1976.

[41] V.R. Pratt. Semantical considerations on Floyd-Hoare logic. In *17th Ann. IEEE Symp. on Foundations of Computer Science*, pages 109–121, 1976.

[42] V.R. Pratt. Models of program logics. In *20th IEEE Symp. Found. Comput. Sci.*, pages 115–122, 1979.

[43] V.R. Pratt. A decidable μ-calculus: preliminary report. In *Proc. 22nd Ann IEEE Symp. on Foundations of Computer Science*, pages 421–427, 1981.

[44] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[45] Shmuel Safra. On the complexity of ω-automata. In *29th IEEE Symp. on Foundations of Computer Science*, 1988.

[46] Andrzej Salwicki. Formalized algorithmic languages. *Bull. Acad. Polon. Sci., Ser. Sci. Math. Astron. Phys.*, 18:227–232, 1970.

[47] D.S. Scott and J.W. de Bakker. A theory of programs. Unpublished notes, IBM, Vienna, 1969.

[48] Dirk Siefkes. *Büchi's Monadic Second Order Successor Arithmetic*, volume 120 of *Lecture Notes in Mathematics*. Springer-Verlag, 1970.

[49] A.P Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with application to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[50] Colin P. Stirling and David J. Walker. Local model checking in the modal mu-calculus. In *International Joint Conference in Theory and Practice of Software Development*, volume 351 of *LNCS*, pages 369–382. Springer-Verlag, 1989.

[51] C.S. Stirling. Modal and temporal logics. In S.Abramsky, D.Gabbay, and T.Maibaum, editors, *Handbook of Logic in Comuter Science*, pages 477–563. Oxford University Press, 1991.

[52] Robert S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information & Computation*, 54:121–141, 1982.

[53] Robert S. Streett and E. Allan Emerson. An automata theoretic procedure for the propositional mu-calculus. *Information & Computation*, 81:249–264, 1989.

[54] Wolfgang Thomas. Computation tree logic and regular ω-languages. In *LNCS 354*, pages 690–713. Springer-Verlag, 1988.

[55] M.Y. Vardi and L.Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *17th ASM STOC*, pages 240–251, 1985.

[56] M.Y Vardi and P.Wolper. Reasoning about infite computation paths. to appear.

[57] M.Y. Vardi and P.Wolper. Automata theoretic techniques for modal logics of programs. In *Sixteenth ACM Symposium on the Theoretical Computer Science*, 1984.

[58] M.Y. Vardi and P.Wolper. Yet another process logic. In *Proc. Workshop on Logics of Programs*, volume LNCS 164, pages 501–512, 1984.

[59] G. Winskel. Model checking in the modal nu-calculus. volume 372 of *LNCS*. Springer-Verlag, 1989.

[60] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.