Lectures on the modal μ-calculus

Yde Venema[*]

Abstract

These notes give an introduction to the theory of the modal μ-calculus and other modal fixpoint logics.

[*]Institute for Logic, Language and Computation, University of Amsterdam, Plantage Muidergracht 24, NL–1018 TV Amsterdam. E-mail: `yde@science.uva.nl`.

Preface

This text is written for students and researchers in logic and (theoretical) computer science who are interested in (modal) logic and its connections with automata theory. The current version assumes some familiarity with the basic definitions of modal logic, as can be found in any recent text book. It is very much work in progress: Some parts of the text are still missing, the notes are still very rudimentary, and there are not nearly as many exercises as are needed in a text of this kind. The final version of these notes will have full proofs of all results mentioned here. It will also contain material on modal fixpoint logics other than the modal μ-calculus, such as LTL and CTL, on complete derivation systems for various fixpoint logics, and on coalgebras and the automata operating on them. Also, in the final version I intend to pay more attention to computational aspects of fixpoint logics such as model checking, to the complexity of various problems related to logic and automata theory, and to the model theory of modal fixpoint logics.

In their present incarnation, these notes serve as material for a course, *Modal Fixpoint Logics*, to be given at Renmin University in Beijing (China) in June 2008. Earlier version accompanied a course, *The modal μ-calculus*, given at the ESSLLI summer school in Malaga, Spain, 2006, and a number of editions of the course, *Advanced Modal Logic*, that I taught at the University of Amsterdam. I am very grateful for the comments that I received on earlier versions of these notes, both from students and from colleagues.

Comments on the present version will also be appreciated very much!

Yde Venema

Amsterdam, June 12, 2008

Contents

Introduction

The study of the modal μ-calculus can be motivated from various (not necessarily disjoint!) directions.

Process Theory In this area of theoretical computer science, one studies formalisms for describing and reasoning about labelled transition systems — these being mathematical structures that model processes. Here the modal μ-calculus strikes a very good balance between computational efficiency and expressiveness. On the one hand, the presence of fixpoint operators make it possible to express most, if not all, of the properties that are of interest in the study of (ongoing) behavior. But on the other hand, the formalism is still simple enough to allow an (almost) polynomial model checking complexity and an exponential time satisfiability problem.

Modal Logic From the perspective of modal logic, the modal μ-calculus is a well-behaved extension of the basic formalism, with a great number of attractive logical properties. For instance, it is the bisimulation invariant fragment of second order logic, it enjoys uniform interpolation, and the set of its validities admits a transparent, finitary axiomatization, and has the finite model property. In short, the modal μ-calculus shares (or naturally generalizes) all the nice properties of ordinary modal logic.

Mathematics and Theoretical Computer Science More generally, the modal μ-calculus has a very interesting theory, with lots of connections with neighboring areas in mathematics and theoretical computer science. We mention automata theory (more specifically, the theory of finite automata operating on infinite objects), game theory, universal algebra and lattice theory, and the theory of universal coalgebra.

Open Problems Finally, there are still a number of interesting *open problems* concerning the modal μ-calculus. For instance, it is unknown whether the characterization of the modal μ-calculus as the bisimulation invariant fragment of monadic second order logic still holds if we restrict attention to finite structures, and in fact there are many open problems related to the expressiveness of the formalism. Also, the exact complexity of the model checking problem is not known. And to mention a third example: the completeness theory of modal fixpoint logics is still a largely undeveloped field.

Summarizing, the modal μ-calculus is a formalism with important applications in the field of process theory, with interesting metalogical properties, various nontrivial links with other areas in mathematics and theoretical computer science, and a number of intriguing open problems. Reason enough to study it in more detail.

Part I
Modal Logic

1 Basics

As mentioned in the preface, we assume familiarity with the basic definitions concerning the syntax and semantics of modal logic. The purpose of this first chapter is to briefly recall notation and terminology, and to provide an introduction to the coalgebraic perspective on modal logic (this perspective will not play a role until Chapter 8).

Convention 1.1 Throughout this text we let P be a set of *proposition letters*, whose elements are usually denoted as $p, q, r, x, y, z, \ldots$, and let D be a set of (atomic) *actions*, whose elements are usually denoted as $d, e, c, \ldots$ In practice we will often suppress explicit reference to P and D.

1.1 Basics

Structures

► `Introduce LTSs as process graphs`

Definition 1.2 A (P, D)-*(labelled) transition system* or (P, D)-*Kripke model* is a triple $\mathbb{S} = \langle S, V, R\rangle$ such that S is a set of objects called *states* or *points*, $V : \mathsf{P} \to \wp(S)$ is a *valuation*, and $R = \{R_d \subseteq S \times S \mid d \in \mathsf{D}\}$ is a family of binary *accessibility relations*. The pair (P, D) is called the *type* of the transition system.

Elements of the set $R_d[s] := \{t \in S \mid (s, t) \in R_d\}$ are called *d-successors* of s. A transition system is called *image-finite* or *finitely branching* if $R_d[s]$ is finite, for every $d \in \mathsf{D}$ and $s \in S$.

A *pointed* transition system or Kripke model is a pair $(\mathbb{S}, s)$ consisting of a transition system $\mathbb{S}$ and a designated state s in $\mathbb{S}$. $\triangleleft$

Remark 1.3 It will be convenient to have an alternative, *coalgebraic* presentation of transition systems. Intuitively, it should be clear that instead of having a valuation $V : \mathsf{P} \to \wp(S)$, telling us at which states each proposition letter is true, we could just as well have a map $\sigma_V : S \to \wp(\mathsf{P})$ informing us which proposition letters are true at each state. Also, a binary relation R on a set S can be represented as a map $R[\cdot] : S \to \wp(S)$ mapping a state s to the collection $R[s]$ of its successors. In this line, a family $R = \{R_d \subseteq S \times S \mid d \in \mathsf{D}\}$ accessibility relations can be seen as a map $\sigma_R : S \to \wp(S)^{\mathsf{D}}$, where $\wp(S)^{\mathsf{D}}$ denotes the set of maps from D to $\wp(S)$.

Combining these two maps into one single function, we see that a transition system $\mathbb{S} = \langle S, V, R\rangle$ of type (P, D) can be seen as a pair $\langle S, \sigma\rangle$, where $\sigma : S \to \wp(\mathsf{P}) \times \wp(S)^{\mathsf{D}}$ is the map given by $\sigma(s) := (\sigma_V(s), \sigma_R(s))$. $\triangleleft$

For future reference we define the notion of a *Kripke functor*.

Definition 1.4 Fix a set P of proposition letters and a set D of atomic actions. Given a set S, let $\mathsf{K}_{\mathsf{D},\mathsf{P}}S$ denote the set

$$\mathsf{K}_{\mathsf{D},\mathsf{P}}S := \wp(\mathsf{P}) \times \wp(S)^{\mathsf{D}}.$$

This operation will be called the *Kripke functor* associated with D and P.

A typical element of $\mathsf{K}_{\mathsf{D},\mathsf{P}}S$ will be denoted as (π, X), with $\pi \subseteq \mathsf{P}$ and $X = \{X_d \mid d \in \mathsf{D}\}$ with $X_d \subseteq S$ for each $d \in \mathsf{D}$.

When we take this perspective we will sometimes refer to Kripke models as $\mathsf{K}_{\mathsf{D},\mathsf{P}}S$-*coalgebras* or *Kripke coalgebras.* ◁

Given this definition we may summarize Remark 1.3 by saying that any transition system can be presented as a pair $\mathbb{S} = \langle S, \sigma : S \to \mathsf{K}S\rangle$ where K is the Kripke functor associated with $\mathbb{S}$. In practice, we will often usually write K rather than $\mathsf{K}_{\mathsf{D},\mathsf{P}}$.

Syntax

Working with fixpoint operators, we may benefit from a set-up in which the use of the negation symbol may only be applied to atomic formulas. The price that one has to pay for this is an enlarged arsenal of primitive symbols. In the context of modal logic we then arrive at the following definition.

Definition 1.5 The set $\mathrm{PML}_{\mathsf{D}}(\mathsf{P})$ of *Polymodal Logic* in D and P is defined as follows:

$$\varphi ::= p \mid \neg p \mid \bot \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond_d \varphi \mid \Box_d \varphi$$

where $p \in \mathsf{P}$, and $d \in \mathsf{D}$. Elements of $\mathrm{PML}_{\mathsf{D}}(\mathsf{P})$ are called *(poly-)modal formulas*, or briefly, *formulas.* Formulas of the form p or $\neg p$ are called *literals.*

In case the set D is a singleton, we speak of the language $\mathrm{BML}(\mathsf{P})$ of *basic modal logic* or *monomodal* logic; in this case we will denote the modal operators by $\Diamond$ and $\Box$, respectively. ◁

Often the sets P and D are implicitly understood, and suppressed in the notation. Generally it will suffice to treat examples, proofs, etc., from basic modal logic.

Remark 1.6 The *negation* $\sim\!\varphi$ of a formula φ can inductively be defined as follows:

$$\begin{array}{lcllcl}
\sim\!\bot & := & \top & \sim\!\top & := & \bot \\
\sim\! p & := & \neg p & \sim\!\neg p & := & p \\
\sim\!(\varphi \vee \psi) & := & \sim\!\varphi \wedge \sim\!\psi & \sim\!(\varphi \wedge \psi) & := & \sim\!\varphi \vee \sim\!\psi \\
\sim\!\Box_d \varphi & := & \Diamond_d \sim\!\varphi & \sim\!\Diamond_d \varphi & := & \Box_d \sim\!\varphi
\end{array}$$

On the basis of this, we can also define the other standard abbreviated connectives, such as $\to$ and $\leftrightarrow$. ◁

We assume that the reader is familiar with standard syntactic notions such as those of a *subformula* or the *construction tree* of a formula, and with standard syntactic operations such as *substitution.* Concerning the latter, we let $\varphi[\psi/p]$ denote the formula that we obtain by substituting all occurrences of p in φ by ψ.

Semantics

The *relational* semantics of modal logic is well known. The basic idea is that the modal operators $\Diamond_d$ and $\Box_d$ are both interpreted using the *accessibility* relation R_d.

The notion of truth (or satisfaction) is defined as follows.

Definition 1.7 Let $\mathbb{S} = \langle S, \sigma \rangle$ be a transition system of type (P, D). Then the *satisfaction relation* $\Vdash$ between states of $\mathbb{S}$ and formulas of PML is defined by the following formula induction.

$$\begin{array}{lll}
\mathbb{S}, s \Vdash p & \text{if} & s \in V(p), \\
\mathbb{S}, s \Vdash \neg p & \text{if} & s \notin V(p), \\
\mathbb{S}, s \Vdash \bot & & \text{never}, \\
\mathbb{S}, s \Vdash \top & & \text{always}, \\
\mathbb{S}, s \Vdash \varphi \lor \psi & \text{if} & \mathbb{S}, s \Vdash \varphi \text{ or } \mathbb{S}, s \Vdash \psi, \\
\mathbb{S}, s \Vdash \varphi \land \psi & \text{if} & \mathbb{S}, s \Vdash \varphi \text{ and } \mathbb{S}, s \Vdash \psi, \\
\mathbb{S}, s \Vdash \Diamond_d \varphi & \text{if} & \mathbb{S}, t \Vdash \varphi \text{ for some } t \in R_d[s], \\
\mathbb{S}, s \Vdash \Box_d \varphi & \text{if} & \mathbb{S}, t \Vdash \varphi \text{ for all } t \in R_d[s].
\end{array}$$

We say that φ *is true* or *holds* at s if $\mathbb{S}, s \Vdash \varphi$, and we let the set

$$[\![\varphi]\!]^{\mathbb{S}} := \{s \in S \mid \mathbb{S}, s \Vdash \varphi\}.$$

denote the *meaning* or *extension* of φ in $\mathbb{S}$. $\lhd$

Alternatively (but equivalently), one may define the semantics of modal formulas directly in terms of this meaning function $[\![\varphi]\!]^{\mathbb{S}}$. This approach has some advantages in the context of fixpoint operators, since it brings out the role of the powerset algebra $\wp(S)$ more clearly.

Remark 1.8 Fix an LTS $\mathbb{S}$, then define $[\![\varphi]\!]^{\mathbb{S}}$ by induction on the complexity of φ, where the operations $\langle R_d \rangle$ and $[R_d]$ are defined in Appendix A:

$$\begin{array}{lcl}
[\![p]\!]^{\mathbb{S}} & = & V(p) \\
[\![\neg p]\!]^{\mathbb{S}} & = & S \setminus V(p) \\
[\![\bot]\!]^{\mathbb{S}} & = & \varnothing \\
[\![\top]\!]^{\mathbb{S}} & = & S \\
[\![\varphi \lor \psi]\!]^{\mathbb{S}} & = & [\![\varphi]\!]^{\mathbb{S}} \cup [\![\psi]\!]^{\mathbb{S}} \\
[\![\varphi \land \psi]\!]^{\mathbb{S}} & = & [\![\varphi]\!]^{\mathbb{S}} \cap [\![\psi]\!]^{\mathbb{S}} \\
[\![\Diamond_d \varphi]\!]^{\mathbb{S}} & = & \langle R_d \rangle [\![\varphi]\!]^{\mathbb{S}} \\
[\![\Box_d \varphi]\!]^{\mathbb{S}} & = & [R_d] [\![\varphi]\!]^{\mathbb{S}}
\end{array}$$

The satisfaction relation $\Vdash$ may be recovered from this by putting $\mathbb{S}, s \Vdash \varphi$ iff $s \in [\![\varphi]\!]^{\mathbb{S}}$.
$\lhd$

Definition 1.9 Let s and s' be two states in the transition systems $\mathbb{S}$ and $\mathbb{S}'$ of type (P, D), respectively. Then we say that s and s' are *modally equivalent*, notation: $\mathbb{S}, s \leftrightsquigarrow_{(\mathsf{P},\mathsf{D})} \mathbb{S}', s'$, if s and s' satisfy the same modal formulas, that is, $\mathbb{S}, s \Vdash \varphi$ iff $\mathbb{S}', s' \Vdash \varphi$, for all modal formulas $\varphi \in \mathrm{PML}_{\mathsf{D}}(\mathsf{P})$. $\lhd$

Flows, trees and streams

In these notes we will devote a lot of attention to *deterministic* transition systems,

Definition 1.10 A transition system $\mathbb{S} = \langle S, V, R \rangle$ is called *deterministic* if each $R_d[s]$ is a singleton. $\lhd$

Note that our definition of determinism does not allow $R_d = \varnothing$ for any point s. We first consider the monomodal case.

Definition 1.11 Let P be a set of proposition letters. A deterministic monomodal Kripke model for this language is called a *flow model for* P, or a $\wp(\mathsf{P})$*-flow.* In case such a structure is of the form $\langle \omega, V, \mathit{Succ} \rangle$, where Succ is the standard successor relation on the set ω of natural numbers, we call the structure a *stream model for* P, or a $\wp(\mathsf{P})$*-stream.* $\lhd$

In case the set D of actions is finite, we may just as well identify it with the set $k = \{0, \ldots, k-1\}$, where k is the size of D. We usually restrict to the binary case, that is, $k = 2$. Our main interest will be in Kripke models that are based on the *binary tree*, i.e., a tree in which every node has exactly two successors, a left and a right one.

Definition 1.12 With $2 = \{0, 1\}$, we let 2^* denote the set of finite strings of 0s and 1s. We let ϵ denote the empty string, while the left- and right successor of a node s are denoted by $s0$ and $s1$, respectively. Written as a relation, we put

$$\mathit{Succ}_i = \{(s, si) \mid s \in 2^*\}.$$

A *binary tree over* P, or a *binary* $\wp(\mathsf{P})$*-tree* is a Kripke model of the form $\langle 2^*, V, \mathit{Succ}_0, \mathit{Succ}_1 \rangle$.

Generalizing the tree models, deterministic Kripke model of type $(\mathsf{P}, 2)$ will often be referred to as *biflow models for* P, or a $\wp(\mathsf{P})$*-biflows.* $\lhd$

Remark 1.13 In the general case, the *k-ary tree* is the structure $(k^*, \mathit{Succ}_0, \ldots, \mathit{Succ}_{k-1})$, where k^* is the set of finite sequences of natural numbers smaller than k, and Succ_i is the *i-th successor relation* given by

$$\mathit{Succ}_i = \{(s, si) \mid s \in k^*\}.$$

A k-flow model is a Kripke model $\mathbb{S} = \langle S, V, R \rangle$ with k many deterministic accessibility relations, and a *k-ary tree model* is a k-flow model which is based on the k-ary tree. $\lhd$

In deterministic transition systems, the distinction between boxes and diamonds evaporates. It is then convenient to use a single symbol $\bigcirc_i$ to denote either the box or the diamond.

Definition 1.14 The set $\mathrm{MFL}_k(\mathsf{P})$ of formulas of *k-ary Modal Flow Logic* in P is given as follows:

$$\varphi ::= p \mid \neg p \mid \bot \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc_i \varphi$$

where $p \in \mathsf{P}$, and $i < k$. In case $k = 1$ we will also speak of *modal stream logic*, notation: $\mathrm{MSL}(\mathsf{P})$. $\lhd$

1.2 Game semantics

We will now describe the semantics defined above in game-theoretic terms. That is, we will define the *evaluation game* $\mathcal{E}(\xi, \mathbb{S})$ associated with a (fixed) formula ξ and a (fixed) LTS $\mathbb{S}$. This game is an example of a *board game*. In a nutshell, board games are games in which the players move a token along the edge relation of some graph, so that a match of the game corresponds to a (finite or infinite) path through the graph. Furthermore, the winning conditions of a match are determined by the nature of this path. We will meet many examples of board games in these notes, and in Chapter 7 we will study them in more detail.

The evaluation game $\mathcal{E}(\xi, \mathbb{S})$ is played by two *players*: Éloise ($\exists$ or 0) and Abélard ($\forall$ or 1). Given a player σ, we always denote the *opponent* of σ by $\bar{\sigma}$. As mentioned, a *match* of the game consists of the two players moving a *token* from one position to another. *Positions* are of the form (φ, s), with φ a *subformula* of ξ, and s a state of $\mathbb{S}$.

It is useful to assign *goals* to both players: in an arbitrary position (φ, s), think of $\exists$ trying to show that φ is *true* at s in $\mathbb{S}$, and of $\forall$ of trying to convince her that φ is *false* at s.

Depending on the type of the position (more precisely, on the formula part of the position), one of the two players may move the token to a next position. For instance, in a position of the form $(\Diamond_d \varphi, s)$, it is $\exists$'s turn to move, and she must choose an arbitrary d-successor t of s, thus making (φ, t) the next position. Intuitively, the idea is that in order to show that $\Diamond\varphi$ is true at s, $\exists$ has to come up with a successor of s where φ holds. Formally, we say that the set of *(admissible) next positions* that $\exists$ may choose from is given as the set $\{(\varphi, t) \mid t \in R_d[s]\}$.

In the case there is no successor of s to choose, she immediately *loses* the game. This is a convenient way to formulate the rules for winning and losing this game: if a position (φ, s) has no admissible next positions, the player whose turn it is to play at (φ, s) immediately loses the game.

This convention gives us a nice handle on positions of the form (p, s) where p is a proposition letter: we always assign such a position an *empty* set of admissible moves, but we make $\exists$ responsible for (p, s) in case p is *false* at s, and $\forall$ in case p is *true* at s.

Position	Player	Admissible moves
$(\varphi_1 \vee \varphi_2, s)$	$\exists$	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\varphi_1 \wedge \varphi_2, s)$	$\forall$	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\Diamond_d \varphi, s)$	$\exists$	$\{(\varphi, t) \mid t \in R_d[s]\}$
$(\Box_d \varphi, s)$	$\forall$	$\{(\varphi, t) \mid t \in R_d[s]\}$
$(\bot, s)$	$\exists$	$\varnothing$
$(\top, s)$	$\forall$	$\varnothing$
$(p, s), s \in V(p)$	$\forall$	$\varnothing$
$(p, s), s \notin V(p)$	$\exists$	$\varnothing$
$(\neg p, s), s \notin V(p)$	$\forall$	$\varnothing$
$(\neg p, s), s \in V(p)$	$\exists$	$\varnothing$

Table 1: Evaluation game for modal logic

In this way, $\exists$ immediately wins if p is true at s, and $\forall$ if it is otherwise. The rules for the negative literals ($\neg p$) and the constants, $\bot$ and $\top$, follow a similar pattern.

The full set of rules of the game is given in Table 1. Observe that all matches of this game are finite, since at each move of the game the active formula is reduced in size. (From the general perspective of board games, this means that we need not worry about winning conditions for matches of infinite length.) We may now summarize the game as follows.

Definition 1.15 Given a modal formula ξ and a transition system $\mathbb{S}$, the *evaluation game* $\mathcal{E}(\xi, \mathbb{S})$ is defined as the board game given by Table 1. The instantiation of this game with starting point (ξ, s) is denoted as $\mathcal{E}(\xi, \mathbb{S})@(\xi, s)$. $\lhd$

An *instance* of an evaluation game is a pair consisting of an evaluation game and a *starting position* of the game. Such an instance will also be called an *initialized game*, or sometimes, if no confusion is likely, simply a game.

A *strategy* for a player σ in an (initialized) game is a method that σ uses to select his moves during the play. Such a strategy is *winning for* σ if every match of the game (starting at the given position) is won by σ, provided σ plays according to this strategy. A position (φ, s) is *winning* for σ if σ has a winning strategy for the game initialized in that position. (This is independent of whether it is σ's turn to move at the position.) The set of winning positions in $\mathcal{E}(\xi, \mathbb{S})$ for σ is denoted as $\mathrm{Win}_\sigma(\mathcal{E}(\xi, \mathbb{S}))$.

The main result concerning these games is that they provide an alternative, but equivalent, semantics for modal logic.

Theorem 1.16 *Let ξ be a modal formula, and let $\mathbb{S}$ be an LTS. Then for any state s in $\mathbb{S}$ it holds that*

$$(\xi, s) \in \mathrm{Win}_\exists(\mathcal{E}(\xi, \mathbb{S})) \iff \mathbb{S}, s \Vdash \xi.$$

The proof of this Theorem is left to the reader.

1.3 Bisimulations and bisimilarity

One of the most fundamental notions in the model theory of modal logic is that of a bisimulation between two transition systems.

▶ `discuss bisimilarity as a notion of behavioral equivalence`

Definition 1.17 Let $\mathbb{S}$ and $\mathbb{S}'$ be two transition systems of the same type (P, D). Then a non-empty relation $Z \subseteq S \times S'$ is a *bisimulation* if the following hold, for every $(s, s') \in Z$.

(prop) $s \in V(p)$ iff $s' \in V'(p)$, for all $p \in \mathsf{P}$;

(forth) for all actions d, and for all $t \in R_d[s]$ there is a $t' \in R'_d[s']$ with $(t, t') \in Z$;

(back) for all actions d, and for all $t' \in R'_d[s']$ there is a $t \in R_d[s]$ with $(t, t') \in Z$.

Two states s and s' are called *bisimilar*, notation: $\mathbb{S}, s \leftrightarroweq \mathbb{S}', s'$ if there is some bisimulation Z with $(s, s') \in Z$.

Relations satisfying the back and forth clauses, but the (prop) clause only for a subset $\mathsf{Q} \subseteq \mathsf{P}$ are called Q-*bisimulations*, and the corresponding notion of bisimilarity is denoted by $\leftrightarroweq_{\mathsf{Q}}$. $\triangleleft$

Bisimilarity and modal equivalence

In order to understand the importance of this notion for modal logic, the starting point should be the observation that the truth of modal formulas is *invariant* under bisimilarity. Recall that $\leftrightsquigarrow$ denotes the relation of modal equivalence.

Theorem 1.18 (Bisimulation Invariance) *Let $\mathbb{S}$ and $\mathbb{S}'$ be two transition systems of the same type. Then*

$$\mathbb{S}, s \leftrightarroweq \mathbb{S}', s' \;\Rightarrow\; \mathbb{S}, s \leftrightsquigarrow \mathbb{S}', s'$$

for every pair of states s in $\mathbb{S}$ and s' in $\mathbb{S}'$.

Proof. By a straightforward induction on the complexity of modal formulas one proves that bisimilar states satisfy the same formulas. QED

But there is much more to say about the relation between modal logic and bisimilarity than Theorem 1.18. In particular, for some classes of models, one may prove a converse statement, which amounts to saying that the notions of bisimilarity and modal equivalence coincide. Such classes are said to have the *Hennessy-Milner* property. As an example we mention the class of finitely branching transition systems.

Theorem 1.19 (Hennessy-Milner Property) *Let $\mathbb{S}$ and $\mathbb{S}'$ be two finitely branching transition systems of the same type. Then*

$$\mathbb{S}, s \leftrightarrows \mathbb{S}', s' \iff \mathbb{S}, s \leftrightsquigarrow \mathbb{S}', s'$$

for every pair of states s in $\mathbb{S}$ and s' in $\mathbb{S}'$.

Proof. The direction from left to right follows from Theorem 1.18. In order to prove the opposite direction, one may show that the relation $\leftrightsquigarrow$ of modal equivalence itself is a bisimulation. Details are left to the reader. QED

This theorem can be read as indication of the expressiveness of modal logic: any difference in behaviour between two states in finitely branching transition systems can in fact be witnessed by a concrete modal formula. As another witness to this expressivity, in section 1.5 we will see that modal logic is sufficiently rich to express all bisimulation-invariant first-order properties. Obviously, this result also adds considerable strength to the link between modal logic and bisimilarity.

As a corollary of the bisimulation invariance theorem, modal logic has the *tree model property*, that is, every satisfiable modal formula is satisfiable on a structure that has the shape of a tree. For the definition of a path through a relational structure, and that of a tree, we refer to the appendix.

Definition 1.20 A transition system $\mathbb{S}$ of type (P, D) is called *tree-like* if the structure $\langle S, \bigcup_{d \in \mathsf{D}} R_d \rangle$ is a tree. $\triangleleft$

The key step in the proof of the tree model property of modal logic is the observation that every transition system can be 'unravelled' into a bisimilar tree-like model. The basic idea of such an unravelling is the new states encode (part of) the *history* of the old states. Technically, the new states are the *paths* through the old system.

Definition 1.21 Let $\mathbb{S} = \langle S, V, R \rangle$ be a transition system of type (P, D). A *path* through $\mathbb{S}$ is a nonempty sequence of the form $(s_0, d_1, s_1, \ldots, d_n, s_n)$ such that $R_{d_i} s_{i-1} s_i$ for all $i \leq n$. The set of paths through $\mathbb{S}$ is denoted as $\mathit{Paths}(\mathbb{S})$; we use the notation $\mathit{Paths}_s(\mathbb{S})$ for the set of paths starting at s.

The *unravelling* of $\mathbb{S}$ *around* a state s is the transition system $\vec{\mathbb{S}}_s$ which is coalgebraically defined as the structure $\langle \mathit{Paths}_s(\mathbb{S}), \vec{\sigma} \rangle$, where the coalgebra map $\vec{\sigma} = (\sigma_V, (\sigma_d \mid d \in \mathsf{D}))$ is defined by putting

$$\begin{aligned} \vec{\sigma}_V(s_0, d_1, s_1, \ldots, d_n, s_n) &:= \sigma_V(s_n), \\ \vec{\sigma}_d(s_0, d_1, s_1, \ldots, d_n, s_n) &:= \{(s_0, d_1, s_1, \ldots, d_n, s_n, d, t) \in \mathit{Paths}_s(\mathbb{S}) \mid R_d s_n t\}. \end{aligned}$$

Finally, the unravelling of a pointed transition system $(\mathbb{S}, s)$ is the pointed structure $(\vec{\mathbb{S}}_s, (s))$, where (s) denotes the empty path starting and finishing at s. $\triangleleft$

Clearly, unravellings are tree-like structures, and any pointed transition system is bisimilar to its unravelling. But then the following theorem is immediate by Theorem 1.18.

Theorem 1.22 (Tree Model Property) *Let φ be a satisfiable modal formula. Then φ is satisfiable at the root of a tree-like model.*

Bisimilarity game

We may also give a game-theoretic characterization of the notion of bisimilarity. We first give an informal description. A match of the *bisimilarity game* between two Kripke models $\mathbb{S}$ and $\mathbb{S}'$ is played by two players, $\exists$ and $\forall$. As in the evaluation game, these players move a token around from one *position* of the game to the next one. In the game there are two kinds of positions: pairs of the form $(s, s') \in S \times S'$ are called *basic positions* and belong to $\exists$. The other positions are of the form $Z \subseteq S \times S'$ and belong to $\forall$.

The idea of the game is that at a position (s, s'), $\exists$ claims that s and s' are bisimilar, and to substantiate this claim she proposes a *local bisimulation* Z (see below) for s and s'. $\forall$ then challenges her by picking a pair $(t, t') \in Z$ as the next basic position.

Definition 1.23 Let $\mathbb{S}$ and $\mathbb{S}'$ be two transition systems of the same type (P, D). Then a relation $Z \subseteq S \times S'$ is a *local bisimulation* for two points $s \in S$ and $s' \in S'$, if it satisfies the properties (prop), (back) and (forth) of Definition 1.17 for this specific s and s'. $\lhd$

Note that if s and s' disagree about the truth of some proposition letter, then there is *no* local bisimulation for s and s'. Also observe that a bisimulation is a relation which is a local bisimulation for each of its members.

Implicitly, $\exists$'s claim at a position $Z \subseteq S \times S'$ is that *all* pairs in Z are bisimilar, so $\forall$ can pick an arbitrary pair $(t, t') \in Z$ and challenge $\exists$ to show that these t and t' are bisimilar.

If a player gets stuck in a match of this game, then the opponent wins the match. For instance, if s and s' disagree about some proposition letter, then the corresponding position is an immediate loss for $\exists$. Or, if neither s nor s' has successors, and agree on the truth of all proposition letters, then $\exists$ could choose the *empty* relation as a local bisimulation, so that $\forall$ would lose the match at his next move.

A new option arises if neither player gets stuck: this game may also have matches that last *forever*. Nevertheless, we can still declare a winner for such matches, and the agreement is that $\exists$ is the winner of any infinite match. Formally, we put the following.

Definition 1.24 The *bisimilarity game* $\mathcal{B}(\mathbb{S}, \mathbb{S}')$ between two Kripke models $\mathbb{S}$ and $\mathbb{S}'$ is the board game given by Table 2, with the winning condition that finite matches are lost by the player who got stuck, while all infinite matches are won by $\exists$. $\lhd$

Position	Player	Admissible moves
$(s, s') \in S \times S'$	$\exists$	$\{Z \in \wp(S \times S') \mid Z$ is a local bisimulation for s and $s'\}$
$Z \in \wp(S \times S')$	$\forall$	$Z = \{(t, t') \mid (t, t') \in Z\}$

Table 2: Bisimilarity game for Kripke models

The following theorem states that the collection of basic winning positions for $\exists$ forms the *largest bisimulation* between $\mathbb{S}$ and $\mathbb{S}'$.

Theorem 1.25 *Let* $(\mathbb{S}, s)$ *and* $(\mathbb{S}', s')$ *be two pointed Kripke models. Then* $\mathbb{S}, s \leftrightarrows \mathbb{S}, s'$ *iff* $(s, s') \in \mathrm{Win}_\exists(\mathcal{B}(\mathbb{S}, \mathbb{S}'))$.

Proof. For the direction from left to right: suppose that Z is a bisimulation between $\mathbb{S}$ and $\mathbb{S}'$ linking s and s'. Suppose that $\exists$, starting from position (s, s'), always chooses the relation Z itself as the local bisimulation. A straightforward verification, by induction on the length of the match, shows that this strategy always provides her with a legitimate move, and that it keeps her alive forever. This proves that it is a winning strategy.

For the converse direction, it suffices to show that the relation $\mathrm{Win}_\exists(\mathcal{B}(\mathbb{S}, \mathbb{S}'))$ itself is in fact a bisimulation. We leave the details for the reader. QED

Bisimulations via relation lifting

Together, the back- and forth clause of the definition of a bisimulation express that the pair of respective successor sets of two bisimilar states must belong to the so-called *Egli-Milner lifting* $\overline{\wp}Z$ of the bisimulation Z. In fact, the notion of a bisimilation can be completely defined in terms of *relation lifting*.

Definition 1.26 Given a relation $Z \subseteq A \times A'$, define the relation $\overline{\wp}Z \subseteq \wp A \times \wp A'$ as follows:

$$\begin{array}{rcll} \overline{\wp}Z & := & \{(X, X') \mid & \text{for all } x \in X \text{ there is an } x' \in X' \text{ with } (x, x') \in Z \\ & & \& & \text{for all } x' \in X' \text{ there is an } x \in X \text{ with } (x, x') \in Z\}. \end{array}$$

Similarly, define, for a Kripke functor $\mathsf{K} = \mathsf{K}_{\mathsf{D},\mathsf{P}}$, the relation $\overline{\mathsf{K}}Z \subseteq \mathsf{K}A \times \mathsf{K}A'$ as follows:

$$\overline{\mathsf{K}}Z := \{((\pi, X), (\pi', X')) \mid \pi = \pi' \text{ and } (X_d, X'_d) \in \overline{\wp}Z \text{ for each } d \in \mathsf{D}\}.$$

The relations $\overline{\wp}Z$ and $\overline{\mathsf{K}}Z$ are called the *lifting* of Z with respect to $\wp$ and K, respectively. We say that $Z \subseteq A \times A'$ is *full on* $B \in \wp A$ and $B' \in \wp A'$, notation: $Z \in B \bowtie B'$, if $(B, B') \in \overline{\wp}Z$. $\lhd$

It is completely straightforward to check that a nonempty relation Z linking two transition systems $\mathbb{S}$ and $\mathbb{S}'$ is a local bisimulation for two states s and s' iff $(\sigma(s), \sigma'(s')) \in \overline{\mathsf{K}}Z$. In particular, $\exists$'s move in the bisimilarity game at a position (s, s') consists of choosing a binary relation Z such that $(\sigma(s), \sigma'(s')) \in \overline{\mathsf{K}}Z$. The following characterization of bisimulations is also an immediate consequence.

Proposition 1.27 *Let $\mathbb{S}$ and $\mathbb{S}'$ be two Kripke coalgebras for some Kripke functor K, and let $Z \subseteq S \times S'$ be some relation. Then*

$$Z \text{ is a bisimulation } \text{ iff } (\sigma(s), \sigma'(s')) \in \overline{\mathsf{K}}Z \text{ for all } (s, s') \in Z. \tag{1}$$

1.4 Finite models and computational aspects

▶ `complexity of model checking`

▶ `filtration & polysize model property`

▶ `complexity of satisfiability`

▶ `complexity of global consequence`

1.5 Modal logic and first-order logic

▶ `modal logic is the bisimulation invariant fragment of first-order logic`

1.6 The cover modality

As we will see now, there is an interesting alternative for the standard formulation of basic modal logic in terms of boxes and diamonds. This alternative set-up is based on a connective which turns *sets* of formulas into formulas.

Definition 1.28 Let Φ be a finite set of formulas. Then $\nabla\Phi$ is a formula, which holds at a state s in a Kripke model if *every* formula in Φ holds at *some* successor of s, while at the same time, *every* successor of s makes *some* formula in Φ true. The operator ∇ is called the *cover modality.* $\lhd$

Observe that this definition involves the $\forall\exists\&\forall\exists$ pattern that we know from the notion of *relation lifting* $\overline{\wp}$ defined in the previous section. In other words, the semantics of the cover modality can be expressed in terms of relation lifting. For that purpose, observe that we may think of the forcing or satisfaction relation $\Vdash$ simply as a binary relation between states and formulas.

Proposition 1.29 *Let s be some state in a Kripke model $\mathbb{S}$, and let Φ be a set of formulas. Then*

$$\mathbb{S}, s \Vdash \nabla\Phi \text{ iff } (\sigma_R(s), \Phi) \in \overline{\wp}(\Vdash).$$

Proof. Immediate by unravelling the definitions. QED

It is not so hard to see that the cover modality can be defined in the standard modal language:

$$\nabla\Phi \equiv \Box \bigvee \Phi \wedge \bigwedge \Diamond\Phi, \tag{2}$$

where $\Diamond\Phi$ denotes the set $\{\Diamond\varphi \mid \varphi \in \Phi\}$.

Things start to get interesting once we realize that *both* the ordinary diamond $\Diamond$ *and* the ordinary box $\Box$ can be expressed in terms of the cover modality (and the disjunction):

$$\begin{array}{lcl} \Diamond\varphi & \equiv & \nabla\{\varphi, \top\}, \\ \Box\varphi & \equiv & \nabla\varnothing \vee \nabla\{\varphi\}. \end{array} \tag{3}$$

Here, as always, we use the convention that $\bigvee \varnothing = \bot$ and $\bigwedge \varnothing = \top$.

Making the above observations more precise, we arrive at the following definition and proposition.

Definition 1.30 Formulas of the language BML_∇ are given by the following recursive definition:

$$\varphi \ ::= \ p \mid \neg p \mid \bot \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \nabla\Phi$$

where Φ denotes a finite set of formulas. $\lhd$

Proposition 1.31 *The languages* BML *and* BML_∇ *are equally expressive.*

Proof. Immediate by (2) and (3). QED

The *real* importance of the cover modality is that it allows us to almost completely eliminate the Boolean *conjunction.* This remarkable fact is based on the following distributive law. Recall from Definition 1.26 that we write $Z \in A \bowtie A'$ if a relation $Z \subseteq A \times A'$ is *full on* A and A', that is, if $(A, A') \in \overline{\wp}Z$.

Proposition 1.32 *For all pairs* Φ, Φ' *of sets of formulas, the following two formulas are equivalent:*

$$\nabla\Phi \wedge \nabla\Phi' \ \equiv \bigvee_{Z \in \Phi \bowtie \Phi'} \nabla\{\varphi \wedge \varphi' \mid (\varphi, \varphi') \in Z\}. \tag{4}$$

Proof. For the direction from left to right, suppose that $\mathbb{S}, s \Vdash \nabla\Phi \wedge \nabla\Phi'$. Let $Z \subseteq \Phi \times \Phi'$ consist of those pairs (φ, φ') such that the conjunction $\varphi \wedge \varphi'$ is true at some successor t of s. It is then straightforward to verify that Z is full on Φ and Φ', and that $\mathbb{S}, s \Vdash \nabla\{\varphi \wedge \varphi' \mid (\varphi, \varphi') \in Z\}$.

The converse direction follows fairly directly from the definitions. QED

1.7 Coalgebraic modal logic

Using the cover modality introduced in the previous section, we can show that we can restrict the use of conjunction in modal logic to that of the *special conjunction* connective $\bullet$. First however, we take care of the proposition letters.

Definition 1.33 Fix a finite set P of proposition letters. Given a subset $\pi \subseteq \mathsf{P}$, we let $\odot\pi$ denote the formula with semantics given by

$$\mathbb{S}, s \Vdash \odot\pi \text{ iff } \sigma_V(s) = \pi$$

for any $\mathsf{K}_{\mathsf{D},\mathsf{P}}$-coalgebra $\mathbb{S} = \langle S, \sigma \rangle$. $\lhd$

In words, the formula $\odot\pi$ holds at a state s iff π consists precisely of those proposition letters in P that are true at s, or equivalently,

$$\odot\pi := \bigwedge_{p \in \pi} p \wedge \bigwedge_{p \notin \pi} \neg p.$$

It is not difficult to see that every propositional formula with proposition letters from P can be expressed as disjunctions of formulas of the form $\odot\pi$. In particular, it is straightforward to verify that

$$q \;\equiv\; \bigvee_{q \in \pi} \odot\pi$$

for every $q \in \mathsf{P}$.

We are now ready for the introduction of the coalgebraic modal connective $\bullet$.

Definition 1.34 Fix finite sets P of proposition letters and D of atomic actions, respectively. Given a subset $\pi \subseteq \mathsf{P}$, and a D-indexed family $\Phi = \{\Phi_d \mid d \in \mathsf{D}\}$ of formulas, then $\pi \bullet \Phi$ is a formula, of which the semantics is defined by the following equivalence:

$$\pi \bullet \Phi \;\equiv\; \odot\pi \wedge \bigwedge_{d \in \mathsf{D}} \nabla_d \Phi_d.$$

Here ∇_d is the cover modality associated with the accessibility relation R_d of d.

The set $\mathrm{CML}_{\mathsf{D}}(\mathsf{P})$ of *coalgebraic modal formulas* is given as follows:

$$\varphi \;::=\; \bot \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \pi \bullet \Phi. \qquad \lhd$$

In words, $\pi \bullet \Phi$ is the conjunction of (i) a complete description of the *local* situation in terms of the proposition letters being true or false, and (ii) for each action d, a description of the d-successor set of the current state, using the cover modality for R_d.

▶ `Explain why` $\bullet$ `and the language` CML `are called ‘coalgebraic’.`

Proposition 1.35 *Fix sets* P *of proposition letters and* D *of atomic actions, respectively. Then we have*

$$\mathbb{S}, s \Vdash \pi \bullet \Phi \text{ iff } (\sigma(s), (\pi, \Phi)) \in \overline{\mathsf{K}}(\Vdash) \tag{5}$$

for any $\pi \in \wp(\mathsf{P})$ *any* D*-indexed family* $\Phi = \{\Phi_d \mid d \in \mathsf{D}\}$*, and for any pointed transition system* $(\mathbb{S}, s)$ *of type* (P, D)*, the following equivalence holds.*

Proof. Simply spell out the definitions. QED

In fact, we could have taken (5) below as the *definition* of the semantics of the bullet modality.

The following result is not very hard to prove.

Theorem 1.36 *For any* P *and* D*, the languages* $\mathrm{PML}_{\mathsf{D}}(\mathsf{P})$ *and* $\mathrm{CML}_{\mathsf{D}}(\mathsf{P})$ *are expressively equivalent.*

Proof. There is a straightforward translation from CML-formulas to ordinary modal formulas, so we focus on the other direction.

It is not hard to verify that every polymodal formula can be rewritten to an equivalent formula using the connectives $\top, \bot, \wedge, \vee, \odot$ and ∇_d. But then the proof of the theorem is straightforward by the observation that both $\odot\pi$ and $\nabla_d\Phi$ can be rewritten to formulas using the bullet connective, using the equivalences below:

$$\begin{array}{rcl} \top & \equiv & \nabla_d\varnothing \vee \nabla_d\{\top\} \\ \top & \equiv & \bigvee_{\pi \subseteq \mathsf{P}} \odot\pi. \end{array}$$

For instance, this allows us to write

$$\begin{array}{rcl} \odot\pi & \equiv & \odot\pi \wedge \bigwedge_{d \in \mathsf{D}} \top \\ & \equiv & \odot\pi \wedge \bigwedge_{d} (\nabla_d\varnothing \vee \nabla_d\{\top\}) \\ & \equiv & \bigvee_{\Phi:\mathsf{D} \to \{\varnothing, \{\top\}\}} \pi \bullet \{\Phi(d) \mid d \in \mathsf{D}\}. \end{array}$$

QED

It may come as a surprise to the reader that the bullet operator is in fact the *only* form of conjunction that we need! More precisely, Theorem 1.38 below states that every formula of CML can be rewritten into an equivalent version that does not use the ordinary Boolean conjunction, but only the special 'bullet conjunction'.

Definition 1.37 Formulas of the language $\mathrm{CML}^{-}_{\mathsf{D}}(\mathsf{P})$ are given by the following recursive definition:

$$\varphi \ ::= \ \top \mid \bot \mid \varphi \vee \varphi \mid \pi \bullet \Phi$$

where π denotes a subset of P, and Φ a D-indexed set of $\mathrm{CML}^{-}_{\mathsf{D}}(\mathsf{P})$-formulas. $\lhd$

Theorem 1.38 *For any* P *and* D*, the languages* $\mathrm{PML}_{\mathsf{D}}(\mathsf{P})$ *and* $\mathrm{CML}^{-}_{\mathsf{D}}(\mathsf{P})$ *are expressively equivalent.*

Proof. Obviously it suffices to prove that every CML-formula φ has an equivalent formula $\overline{\varphi}$ that does not use the conjunction symbol. We will prove this result by induction on the length of a formula, confining ourselves to the case of basic modal logic (with one action).

In the base step of this induction there is nothing to prove. In the inductive step, the clauses for the disjunction and the cover modality speak for themselves:

$$\begin{array}{rcl} \overline{\varphi \vee \psi} & := & \overline{\varphi} \vee \overline{\psi}, \\ \overline{\pi \bullet \Phi} & := & \pi \bullet \{\overline{\varphi} \mid \varphi \in \Phi\}. \end{array}$$

This leaves the case of a conjunction $\varphi \wedge \varphi'$, where we make a further case distinction. If either of the formulas is of the form $\top$ or $\bot$ it is obvious how to proceed: $\overline{\bot \wedge \varphi} := \bot$, $\overline{\top \wedge \varphi} := \overline{\varphi}$, etc. Also, in case either of the two conjuncts is a disjunction, say $\varphi = \varphi_0 \vee \varphi_1$, using induction loading we may correctly define $\overline{\varphi \wedge \varphi'} := \overline{\varphi_0 \wedge \varphi'} \vee \overline{\varphi_0 \wedge \varphi'}$.

The heart of the proof lies in the one remaining inductive case, namely, where $\varphi = \pi \bullet \Phi$ and $\varphi' = \pi' \bullet \Phi'$. Here we put

$$\overline{\varphi \wedge \varphi'} \ := \ \begin{cases} \bot & \text{if } \pi \neq \pi', \\ \bigvee_{Z \in \Phi \bowtie \Phi'} (\pi \bullet \{\overline{\varphi \wedge \varphi'} \mid (\varphi, \varphi') \in Z\}) & \text{if } \pi \neq \pi'. \end{cases}$$

It then follows immediately from the inductive assumptions that $\overline{\varphi \wedge \varphi'}$ is a $\mathrm{BML}^{-}_{\nabla}$-formula, and from Proposition 1.32 that $\overline{\varphi \wedge \varphi'}$ is equivalent to $\varphi \wedge \varphi'$. QED

Notes

Modal logic has a long history in philosophy and mathematics, for an overview we refer to Blackburn, de Rijke and Venema [3] The use of modal formalisms as specification languages in process theory goes back at least to the 1970s, with Pratt [26] and Pnueli [25] being two influential early papers.

The notion of bisimulation, which plays an important role in modal logic and process theory alike, was first introduced in a modal logic context by van Benthem [2], who proved that modal logic is the bisimulation invariant fragment of first-order logic. The notion was later, but independently, introduced in a process theory setting by Park [24]. At the time of writing we do not know who first took a game-theoretical perspective on the semantics of modal logic. The cover modality ∇ was introduced independently by Moss [17] and Janin & Walukiewicz [10].

Readers who want to study modal logic in more detail are referred to Blackburn, de Rijke and Venema [3] or Chagrov & Zakharyaschev [5].

Exercises

Exercise 1.1 Prove Theorem 1.16.

Exercise 1.2 Consider the following version $\mathcal{B}_\omega(\mathbb{S}, \mathbb{S}')$ of the bisimilarity game between two transition systems $\mathbb{S}$ and $\mathbb{S}'$. Positions of this game are of the form either (s, s', α) or (Z, α), with $s \in S$, $s' \in S'$, $Z \subseteq S \times S'$ and α either a natural number or ω. The admissible moves for $\exists$ and $\forall$ are displayed in the following table:

Position	Player	Admissible moves
(s, s', α)	$\exists$	$\{(Z, \alpha) \mid Z$ is a local bisimulation for s and s' $\}$
(Z, α)	$\forall$	$\{(s, s', \beta) \mid (s, s') \in Z$ and $\beta < \alpha\}$

Note that all matches of this game have finite length.

We write $\mathbb{S}, s \leftrightarrows_\alpha \mathbb{S}', s'$ to denote that $\exists$ has a winning strategy in the game $\mathcal{B}_\omega(\mathbb{S}, \mathbb{S}')$ starting at position (s, s', α).

(a) Give concrete examples such that $\mathbb{S}, s \leftrightarrows_n \mathbb{S}', s'$ for all $n < \omega$, but not $\mathbb{S}, s \leftrightarrows_\omega \mathbb{S}', s'$.
(Hint: think of two modally equivalent but not bisimilar states.)

(b) Let k be a natural number. Prove that, for all $\mathbb{S}, s$ and $\mathbb{S}', s'$:

$$\mathbb{S}, s \leftrightarrows_k \mathbb{S}', s' \;\Rightarrow\; \mathbb{S}, s \leftrightsquigarrow_k \mathbb{S}', s'.$$

Here $\leftrightsquigarrow_k$ denotes the modal equivalence relation with respect to formulas of modal depth at most k.

(c) Let $\mathbb{S}$ and $\mathbb{S}'$ be finitely branching transition systems. Prove *directly* (i.e., without using part (b)) that (i) $\Rightarrow$ (ii), for all $s \in S$ and $s' \in S'$:

(i) $\mathbb{S}, s \leftrightarrows_k \mathbb{S}', s'$ for all $k < \omega$,

(ii) $\mathbb{S}, s \leftrightarrows \mathbb{S}', s'$.

Exercise 1.3 Let Φ and Θ be sets of formulas. Prove that

$$\nabla(\Phi \cup \{\bigvee \Theta\}) \;\equiv\; \bigvee \Big\{ \nabla(\Phi \cup \Theta') \;\mid\; \varnothing \neq \Theta' \subseteq \Theta \Big\}$$

Part II
Modal Fixpoint Logics

2 The modal μ-calculus

This chapter is a first introduction to the modal μ-calculus. We define the language, discuss some syntactic issues, and then proceed to its game-theoretic semantics. As a first result, we prove that the modal μ-calculus is bisimulation invariant, and has a strong, 'bounded' version of the tree model property. We start with an example.

Example 2.1 Consider the formula $\langle d^* \rangle p$ from propositional dynamic logic. By definition, this formula holds at those points in an LTS $\mathbb{S}$ from which there is a finite R_d-path, of unspecified length, leading to a state where p is true.

We leave it for the reader to prove that

$$\mathbb{S} \Vdash \langle d^* \rangle p \leftrightarrow (p \vee \langle d \rangle \langle d^* \rangle p)$$

for any transition system $\mathbb{S}$ (here we write $\langle d \rangle$ rather than $\Diamond_d$). Informally, one might say that $\langle d^* \rangle p$ is a *fixed point* or solution of the 'equation'

$$x \leftrightarrow p \vee \langle d \rangle x. \tag{6}$$

One may show, however, that $\langle d^* \rangle p$ is not the only fixpoint of (6). If we let ∞_d denote a formula that is true at those states of a transition system from which an infinite d-path emanates, then the formula $\langle d^* \rangle p \vee \infty_d$ is another fixed point of (6).

In fact, one may prove that the two mentioned fixpoints are the smallest and largest possible solutions of (6), respectively. $\lhd$

As we will see in this chapter, the modal μ-calculus allows one to explicitly refer to such smallest and largest solutions. For instance, as we will see further on, the smallest and largest solution of the 'equation' (6) will be written as $\mu x.p \vee \langle d \rangle x$ and $\nu x.p \vee \langle d \rangle x$, respectively.

To arrive at the semantics of modal fixpoint formulas one can take two roads. In Chapter 3 we will introduce the algebraic semantics of $\mu x.\varphi$ and $\nu x.\varphi$ in an LTS $\mathbb{S}$, in terms of the *least* and *greatest fixpoint*, respectively, of some algebraically defined meaning function. For this purpose, we will consider the formula φ as an *operation* on the power set of (the state space of) $\mathbb{S}$, and we have to prove that this operation indeed has a least and a greatest fixpoint. As we will see, this formal definition of the semantics of the modal μ-calculus may be mathematically transparent, but it is of little help when it comes to unravelling and understanding the actual meaning of individual formulas. In practice, it is much easier to work with the *evaluation games* that we will introduce in this chapter.

This framework builds on the game-theoretical semantics for ordinary modal logic as described in subsection 1.2, extending it with features for the fixpoint operators and for the bound variables of fixpoint formulas (such as x in the formula $\mu x.p \vee \Diamond x$). The key difference lies in the fact that when a match of an evaluation game reaches a position of the form (x, s), with x a *bound* variable, then an equation such as (6) is used to *unfold* the variable x into its associated formula (in the example, the formula $p \vee \Diamond x$).

As a consequence, the flavour of these games is remarkably different from the evaluation games we met before. Recall that in evaluation matches for *basic* modal formulas, the formula is broken down step by step. From this it follows that the length of such a match is *bounded* by the length of the formula. Evaluation matches for fixpoint formulas, on the other hand, can last infinitely long, if some fixpoint variables are unfolded infinitely often. Hence, the game-theoretic semantics for fixpoint logics takes us to the area of *infinite games*.

2.1 Syntax

As announced already in the previous chapter, in the case of fixpoint formulas we will usually work with formulas in *positive normal form* in which the only admissible occurrences of the negation symbol is in front of atomic formulas.

Definition 2.2 Given sets P and D of proposition letters and atomic actions, respectively, define the collection $\mu\mathrm{PML}(\mathsf{D}, \mathsf{P})$ of *(poly-)modal fixpoint formulas* as follows:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Diamond_d \varphi \mid \Box_d \varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

where $p, x \in \mathsf{P}$, $d \in \mathsf{D}$. There is a restriction on the formation of the formulas $\mu x.\varphi$ and $\nu x.\varphi$, namely, that all occurrences of x in φ are *positive*. That is, no occurrence of x in φ may be in the scope of the negation operator $\neg$.

As before, we will usually write $\mu\mathrm{PML}$ rather than $\mu\mathrm{PML}(\mathsf{D}, \mathsf{P})$ in order not to clutter up notation. In case the set D of atomic actions is a singleton, we will simply speak of the *modal μ-calculus*, notation: $\mu\mathrm{ML}(\mathsf{P})$, or $\mu\mathrm{ML}$ if P is understood.

The syntactic combinations μx and νx are called the *least* and *greatest fixpoint operators*, respectively. We use the symbol η to denote either μ or ν. A fixpoint formula of the form $\mu x.\varphi$ is called a *μ-formula*, while *ν-formulas* are the ones of the form $\nu x.\varphi$. $\lhd$

Definition 2.3 The concepts of *subformula* and *proper subformula* are defined as usual. We write $\varphi \trianglelefteq \psi$ if φ is a subformula of ψ. The set of subformulas of ψ is denoted as $\mathit{Sfor}(\psi)$. $\lhd$

Syntactically, the fixpoint operators are very similar to the quantifiers of first-order logic in the way they *bind* variables.

Definition 2.4 Fix a formula φ. The sets $FV(\varphi)$ and $BV(\varphi)$ of *free* and *bound variables* of φ are defined by the following induction on φ:

$$\begin{array}{lcl@{\qquad\qquad}lcl}
FV(\bot) & := & \varnothing & BV(\bot) & := & \varnothing \\
FV(\top) & := & \varnothing & BV(\top) & := & \varnothing \\
FV(p) & := & \{p\} & BV(p) & := & \varnothing \\
FV(\neg p) & := & \{p\} & BV(\neg p) & := & \varnothing \\
FV(\varphi \vee \psi) & := & FV(\varphi) \cup FV(\psi) & BV(\varphi \vee \psi) & := & BV(\varphi) \cup BV(\psi) \\
FV(\varphi \wedge \psi) & := & FV(\varphi) \cup FV(\psi) & BV(\varphi \wedge \psi) & := & BV(\varphi) \cup BV(\psi) \\
FV(\Diamond_d \varphi) & := & FV(\varphi) & BV(\Diamond_d \varphi) & := & BV(\varphi) \\
FV(\Box_d \varphi) & := & FV(\varphi) & BV(\Box_d \varphi) & := & BV(\varphi) \\
FV(\eta x.\varphi) & := & FV(\varphi) \setminus \{x\} & BV(\eta x.\varphi) & := & BV(\varphi) \cup \{x\}
\end{array}$$

$\lhd$

Formulas like $x \vee \mu x.((p \vee x) \wedge \Box \nu x.\Diamond x)$ may be well formed, but in practice they are very hard to read and work with. In the sequel we will almost exclusively work with formulas in which every bound variable uniquely determines a fixpoint operator binding it, and in which there is no overlap between free and bound variables.

Definition 2.5 A formula $\varphi \in \mu$PML is *clean* if no two distinct (occurrences of) fixed point operators in φ bind the same variable, and no variable has both free and bound occurrences in φ. If x is a bound variable of the clean formula φ, we let $\varphi_x = \eta_x x.\delta_x$ denote the unique subformula of φ where x is bound by the fixpoint operator η_x. $\lhd$

An important role in the theory of the modal μ-calculus is played by a certain order on its bound variables. The idea behind this 'dependency order' is that if $x \leq y$, the meaning of φ_x is (in principle) dependent on the meaning of y, because y may occur freely in φ_x.

Definition 2.6 Given a clean formula φ, we define a *dependency order* on the set $BV(\varphi)$, saying that *y ranks higher* than x, notation: $x \leq_\varphi y$ iff $\varphi_x \trianglelefteq \varphi_y$. $\lhd$

We finish our sequence of syntactic definitions with the notion of guardedness, which will become important later on.

Definition 2.7 A variable x is *guarded* in a μPML-formula φ if every occurrence of x in φ is in the scope of a modal operator. A formula $\xi \in \mu$PML is *guarded* if for every subformula of ξ of the form $\eta x.\delta$, x is guarded in δ. $\lhd$

2.2 Game semantics

For a definition of the evaluation game of the modal μ-calculus, fix a *clean* formula ξ and an LTS $\mathbb{S}$. Basically, the game $\mathcal{E}(\xi, \mathbb{S})$ for ξ a fixpoint formula is defined in the same way as for plain modal logic formulas.

Definition 2.8 Given a clean modal μ-calculus formula ξ and a transition system $\mathbb{S}$, we define the *evaluation game* $\mathcal{E}(\xi, \mathbb{S})$ as a board game with players $\exists$ and $\forall$ moving a token around positions of the form $(\varphi, s) \in Sfor(\xi) \times S$. The rules, determining the admissible moves from a given position, together with the player who is supposed to make this move, are given in Table 3. $\lhd$

One might expect that the main difference with the evaluation game for basic modal formulas would involve the new formula constructors of the μ-calculus: the fixpoint operators. Perhaps surprisingly, the fixpoint operators are dealt with in the most straightforward way possible: the successor of a position of the form $(\eta x.\delta, s)$ is simply obtained as the pair (δ, s). Since this next position is thus uniquely determined, the position $(\eta x.\delta, s)$ will not be assigned to either of the players.

The crucial difference lies in the treatment of the *bound variables* of a fixpoint formula ξ. Previously, and still in the case of free variables, positions of the form (p, φ) would be *final positions* of the game, immediately determining the winner of the match. However, at a position (x, s) with x *bound*, the fixpoint variable x gets *unfolded*; this means that the new position is given as (δ_x, s), where $\eta_x x.\delta_x$ is the unique subformula of ξ where x is bound. Note that for this to be well defined, we need ξ to be clean. The disjointness of $FV(\xi)$ and $BV(\xi)$ ensures that it is always clear whether a variable is to be unfolded or not, and the fact that bound formulas are bound by unique occurrences of fixpoint operators guarantees that δ_x is uniquely determined. Finally, since in this case the next position is also completely determined by the current one, positions of the form (x, s) with x *bound* are assigned to neither of the players.

Example 2.9 Let $\mathbb{S} = \langle S, R, V \rangle$ be the Kripke model based on the set $S = \{0, 1, 2\}$, with $R = \{(0, 1), (1, 1), (1, 2), (2, 2)\}$, and V given by $V(p) = \{2\}$. Now let ξ be the formula $\eta x.p \vee \Box x$, and consider the game $\mathcal{E}(\xi, \mathbb{S})$ initialized at $(\xi, 0)$.

The second position of any match of this game will be $(p \vee \Box x, 0)$ belonging to $\exists$. Assuming that she wants to win, she chooses the disjunct $\Box x$ since otherwise p being false at 0 would mean an immediate loss for her. Now the position $(\Box x, 0)$ belongs to $\forall$ and he will make the only move allowed to him, choosing $(x, 1)$ as the next position. Here an automatic move is made, *unfolding* the variable x, and thus changing the position to $(p \vee \Box x, 1)$. And as before, $\exists$ will choose the right disjunct: $(\Box x, 1)$.

At $(\Box x, 1)$, $\forall$ does have a choice. Choosing $(x, 2)$, however, would mean that $\exists$ wins the match since p being true at 2 enables her to finally choose the first disjunct of the formula $p \vee \Box x$. So $\forall$ chooses $(x, 1)$, a position already visited by the match before.

Position	Player	Admissible moves
$(\varphi_1 \vee \varphi_2, s)$	$\exists$	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\varphi_1 \wedge \varphi_2, s)$	$\forall$	$\{(\varphi_1, s), (\varphi_2, s)\}$
$(\Diamond_d \varphi, s)$	$\exists$	$\{(\varphi, t) \mid t \in \sigma_d(s)\}$
$(\Box_d \varphi, s)$	$\forall$	$\{(\varphi, t) \mid t \in \sigma_d(s)\}$
$(\bot, s)$	$\exists$	$\varnothing$
$(\top, s)$	$\forall$	$\varnothing$
(p, s), with $p \in FV(\xi)$ and $s \in V(p)$	$\forall$	$\varnothing$
(p, s), with $p \in FV(\xi)$ and $s \notin V(p)$	$\exists$	$\varnothing$
$(\neg p, s)$, with $p \in FV(\xi)$ and $s \in V(p)$	$\exists$	$\varnothing$
$(\neg p, s)$, with $p \in FV(\xi)$ and $s \notin V(p)$	$\forall$	$\varnothing$
$(\eta_x x.\delta_x, s)$	$-$	$\{(\delta_x, s)\}$
(x, s), with $x \in BV(\xi)$	$-$	$\{(\delta_x, s)\}$

Table 3: Evaluation game for modal fixpoint logic

This means that these strategies force the match to be *infinite*, with the variable x unfolding infinitely often at positions of the form $(x, 1)$, and the match taking the following form:

$$(\xi, 0)(p \vee \Box x, 0)(\Box x, 0)(x, 1)(p \vee \Box x, 1)(\Box x, 1)(x, 1)(p \vee \Box x, 1) \ldots$$

So who is declared to be the winner of this match? This is where the difference between the two fixpoint operators shows up. In case $\eta = \mu$, the above infinite match is *lost* by $\exists$ since the fixpoint variable that is unfolded infinitely often is a μ-variable, and μ-variables are to be unfolded only finitely often. In case $\eta = \nu$, the variable unfolded infinitely often is a ν-variable, and this is unproblematic: $\exists$ wins the match. $\lhd$

The above example shows the principle of unfolding at work. Its effect is that matches may now be of infinite length since formulas are no longer deconstructed at every move of the game. Nevertheless, as we will see, it will still be very useful to declare a *winner* of such an infinite game. Here we arrive at one of the key ideas underlying the semantics of fixpoint formulas, which in a slogan can be formulated as follows:

$$\boxed{\nu \text{ means unfolding, } \mu \text{ means finite unfolding.}}$$

Giving a more detailed implementation of this slogan, in case of a unique variable that is unfolded infinitely often during a match π, we will declare $\exists$ to be the winner of π if this variable is a ν-variable, and $\forall$ in case we are dealing with a μ-variable. But what happens in case that *various* variables are unfolded infinitely often? As we shall see, in these cases there is always a *unique* such variable that ranks higher than any other such variable.

Definition 2.10 Let ξ be a clean μPML-formula, and $\mathbb{S}$ a labelled transition system. A *match* of the game $\mathcal{E}(\xi, \mathbb{S})$ is a (finite or infinite) sequence of positions

$$(s, \xi) = (s_0, \varphi_0)(s_1, \varphi_1)(s_2, \varphi_2)\dots$$

which are in accordance with the rules of Table 3. A *full match* is either an infinite match, or a finite match in which the player responsible for the last position got stuck. In practice we will always refer to full matches simply as *matches.* A match that is not full is called *partial.*

Given an infinite match π, we let $\mathit{Unf}^{\infty}(\pi) \subseteq BV(\xi)$ denote the set of variables that are unfolded infinitely often during π. $\lhd$

Proposition 2.11 *Let ξ be a clean μPML-formula, and $\mathbb{S}$ a labelled transition system. Then for any infinite match π of the game $\mathcal{E}(\xi, \mathbb{S})$, the set $\mathit{Unf}^{\infty}(\pi)$ has a highest ranking member, in terms of the dependency order of Definition 2.6.*

Proof. Since ξ consists of finitely many symbols, $\mathit{Unf}^{\infty}(\pi)$ is not empty. We claim that it is in fact *directed* (with respect to the ranking order). That is, for any x and y in $\mathit{Unf}^{\infty}(\pi)$ there is a variable $z \in \mathit{Unf}^{\infty}(\pi)$ such that $x \leq_{\xi} z$ and $y \leq_{\xi} z$.

For suppose otherwise. Then in particular, $\varphi_x = \eta_x x.\delta_x$ and $\varphi_y = \eta_y y.\delta_y$ are not subformulas of one another. However, π goes through both φ_x and φ_y infinitely often. Now the only way it can move from φ_x to φ_y is by unfolding some variable z such that both φ_x and φ_y are subformulas of φ_z, that is, $x \leq_{\xi} z$ and $y \leq_{\xi} z$. Since this happens infinitely often, one of these variables z must belong to $\mathit{Unf}^{\infty}(\pi)$, as required.

But if $\mathit{Unf}^{\infty}(\pi)$ is directed, being finite it must have a maximum. That is, there is indeed a highest variable in $BV(\xi)$ that gets unfolded infinitely often during π. QED

Given this result, there is now a natural formulation of the winning conditions for infinite matches of evaluation games.

Definition 2.12 Let ξ be a clean μPML-formula. The winning conditions of the game $\mathcal{E}(\xi, \mathbb{S})$ are given in Table 4. $\lhd$

	$\exists$ wins π	$\forall$ wins π
π is finite	$\forall$ got stuck	$\exists$ got stuck
π is infinite	$\max(\mathit{Unf}^{\infty}(\pi))$ is a ν-variable	$\max(\mathit{Unf}^{\infty}(\pi))$ is a μ-variable

Table 4: Winning conditions of $\mathcal{E}(\xi, \mathbb{S})$

We can now formulate the game-theoretic semantics of the modal μ-calculus as follows.

Definition 2.13 Let ξ be a clean formula of the modal μ-calculus, and let $\mathbb{S}$ be a transition system of the appropriate type. Then we say that ξ is (game-theoretically) *satisfied* at s, notation: $\mathbb{S}, s \Vdash_g \xi$ if $(s, \xi) \in \mathrm{Win}_\exists(\mathcal{E}(\xi, \mathbb{S}))$. $\lhd$

▶ `define satisfaction relation of arbitrary formulas via clean alphabetical variants`

2.3 Examples

Example 2.14 As a first example, consider the formulas $\eta x.p \vee x$, and fix a Kripke model $\mathbb{S}$. Observe that any match of the evaluation game $\mathcal{E}(\eta x.p \vee x, \mathbb{S})$ starts with the two positions $(\eta x.p \vee x, s)(p \vee x, s)$, after which $\exists$ can make a choice. We claim that

$$\mathbb{S}, s \Vdash_g \mu x.p \vee x \text{ iff } s \in V(p).$$

For the direction from right to left, assume that $s \in V(p)$. Now, if $\exists$ chooses the disjunct p at the position $(s, p \vee x)$, she wins the match because $\forall$ will get stuck at (s, p). Hence $s \in \mathrm{Win}_\exists(\mathcal{E}(\eta x.p \vee x, \mathbb{S}))$.

On the other hand, if $s \notin V(p)$, then $\exists$ will lose if she chooses disjunct p at position $(s, p \vee x)$. So she must choose the disjunct x which then unfolds to $p \vee x$ so that $\exists$ is back at the position $(s, p \vee x)$. Thus if $\exists$ does not want to get stuck her only way to survive is to keep playing the position (s, x), thus causing the match to be infinite. But such a match is won by $\forall$ since the only variable that gets unfolded infinitely often is a μ-variable. Hence in this case we see that $s \notin \mathrm{Win}_\exists(\mathcal{E}(\eta x.p \vee x, \mathbb{S}))$.

If on the other hand we take $\eta = \nu$, then $\exists$ can win any match:

$$\mathbb{S}, s \Vdash_g \nu x.p \vee x.$$

It is easy to see that the strategy of always choosing the disjunct x at a position of the form $(s, p \vee x)$ is winning. For, it forces all games to be infinite, and since the only fixpoint variable that gets ever unfolded here is a ν-variable, all infinite matches are won by $\exists$.

Concluding, we see that $\mu x.p \vee x$ is equivalent to the formula p, and $\nu x.p \vee x$, to the formula $\top$. $\lhd$

Example 2.15 Now we turn to the formulas $\mu x.\Diamond x$ and $\nu x.\Diamond x$. First consider how a match for any of these formulas proceeds. The first two positions of such a match will be of the form $(\eta x.\Diamond x, s)(\Diamond x, s)$, at which point it is $\exists$'s turn to make a move. Now she either is stuck (in case the state s has no successor) or else the next two positions are $(x, t)(\Diamond x, t)$ for some successor t of s, chosen by $\exists$. Continuing this analysis, we see that there are two possibilities for a match of the game $\mathcal{E}(\eta x.\Diamond x, \mathbb{S})$:

1. the match is an infinite sequence of positions

$$(\eta x.\Diamond x, s_0)(\Diamond x, s_0)(x, s_1)(\Diamond x, s_1)(x, s_2)\ldots$$

   corresponding to an infinite path $s_0 R s_1 R s_2 R \ldots$ through $\mathbb{S}$.

2. the match is a finite sequence of positions

$$(\eta x.\Diamond x, s_0)(\Diamond x, s_0)(x, s_1)(\Diamond x, s_1)\ldots(\Diamond x, s_k)$$

   corresponding to a finite path $s_0 R s_1 R \ldots s_k$ through $\mathbb{S}$, where s_k has no successors.

Note too that in either case it is only $\exists$ who has turns, and that her strategy corresponds to choosing a *path* through $\mathbb{S}$. From this it is easy to derive that
• $\mu x.\Diamond x$ is equivalent to the formula $\bot$,
• $\mathbb{S}, s \Vdash_g \nu x.\Diamond x$ iff there is an infinite path starting at s. $\lhd$

▶ `Until operator`

The examples that we have considered so far involved only a single fixpoint operator. We now look at an example containing both a least and a greatest fixpoint operator.

Example 2.16 Let ξ be the following formula:

$$\xi = \nu x.\mu y. \underbrace{(p \wedge \Diamond x)}_{\alpha_p} \vee \underbrace{(\neg p \wedge \Diamond y)}_{\alpha_{\neg p}}$$

Then we claim that for any LTS $\mathbb{S}$, and any state s in $\mathbb{S}$:

$$\mathbb{S}, s \Vdash_g \xi \text{ iff there is some path from } s \text{ on which } p \text{ is true infinitely often.} \qquad (7)$$

To see this, first suppose that there is a path $\pi = s_0 s_1 s_2 \ldots$ as described in the right hand side of (7) and suppose that $\exists$ plays according to the following strategy:

(a) at a position $(\alpha_p \vee \alpha_{\neg p}, t)$, choose (α_p, t) if $\mathbb{S}, t \Vdash_g p$ and choose $(\alpha_{\neg p}, t)$ otherwise;

(b) at a position $(\Diamond\varphi, t)$, distinguish cases:
- if the match so far has followed the path, with $t = s_k$, choose (φ, s_{k+1});
- otherwise, choose an arbitrary successor (if possible).

We claim that this is a winning strategy for $\exists$ in the evaluation game initialized at (ξ, s). Indeed, since $\exists$ always chooses the propositionally safe disjunct of $\alpha_p \vee \alpha_{\neg p}$, she forces $\forall$, when faced with a position of the form $(\alpha_{\pm p}, t) = (\pm p \wedge \Diamond z, t)$ to always choose the diamond conjunct $\Diamond z$, or lose immediately. In this way she guarantees to always get to positions of the form $(s_i, \Diamond z)$, and thus she can force the match to last infinitely long, following the infinite path π. But why does she actually *win* this match? The point is that, whenever she chooses α_p, three positions later, x will be unfolded, and likewise with $\alpha_{\neg p}$ and y. Thus, p being true infinitely often on π means that the ν-variable x gets unfolded infinitely often. And so, even though the μ-variable y might get unfolded infinitely often as well, she wins the match since x ranks higher than y anyway.

For the other direction, assume that $\mathbb{S}, s \Vdash_g \xi$ so that $\exists$ has a winning strategy in the game $\mathcal{E}(\xi, \mathbb{S})$ initialized at (ξ, s). It should be clear that any winning strategy must follow (a) above. So whenever $\forall$ faces a position $(p \wedge \Diamond z, t)$, p will be true, and likewise with positions $(\neg p \wedge \Diamond z, t)$. Now consider a match in which $\forall$ plays propositionally sound, that is, always chooses the diamond conjunct of these positions. This match must be infinite since both players will stay alive forever: $\forall$ because he can always choose a diamond conjunct, and $\exists$ because we assumed her strategy to be winning. But a second consequence of $\exists$ playing a winning strategy, is that it cannot happen that y is unfolded infinitely often, while x is not. So x is unfolded infinitely often, and as before, x only gets unfolded right after the match passed a world where p is true. Thus the path chosen by $\exists$ must contain infinitely many states where p holds. $\lhd$

2.4 Memory-free determinacy

From a theoretical perspective, the importance of the game-theoretical semantics of fixpoint logics lies in the fact that the evaluation games are so-called *parity games* (see Chapter 7 for more details). Parity games have a number of very useful properties. In particular, it can be shown that winning strategies for either player can always be assumed to be positional, that is, do not depend on moves made earlier in the match, but only on the current position. As we will see further on, this property is crucial in establishing various results about the modal μ-calculus.

- ▶ Every evaluation game $\mathcal{E}(\xi, \mathbb{S})$ is *determined* in the sense that every position (φ, s) is winning for exactly one of the two players.
- ▶ In addition, evaluation games enjoy *history-free* or *memory-free determinacy*. This means that each player $\sigma \in \{\exists, \forall\}$ has a *positional* strategy f_σ which is winning for the game $\mathcal{E}(\xi, \mathbb{S})@(\varphi, s)$ for every position (φ, s) that is winning for σ.
- ▶ A strategy is *positional* if it only depends on the current position (that is, the final position of the partial play).

2.5 Bounded tree model property

Given the game-theoretic characterization of the semantics, it is rather straightforward to prove that formulas of the modal μ-calculus are bisimulation invariant. From this it is immediate that the modal mu-calculus has the tree model property. But in fact, we can use the game semantics to do better than this, proving that every satisfiable modal fixpoint formula is satisfied in a tree of which the branching degree is *bounded* by the size of the formula.

Theorem 2.17 (Bisimulation Invariance) *Let ξ be a modal fixpoint formula with $FV(\xi) \subseteq \mathsf{P}$, and let $\mathbb{S}$ and $\mathbb{S}'$ be two labelled transition systems with points s and s', respectively. If $\mathbb{S}, s \leftrightarroweq_{\mathsf{P}} \mathbb{S}', s'$, then*

$$\mathbb{S}, s \Vdash_g \xi \textit{ iff } \mathbb{S}', s' \Vdash_g \xi.$$

Proof. Assume that $s \leftrightarroweq_{\mathsf{P}} s'$ and that $\mathbb{S}, s \Vdash_g \xi$, with $FV(\xi) \subseteq \mathsf{P}$. We will show that $\mathbb{S}', s' \Vdash_g \xi$. By Memory-Free Determinacy we may assume that $\exists$ has a positional winning strategy f in the evaluation game $\mathcal{E} := \mathcal{E}(\xi, \mathbb{S})$ initialized at (ξ, s). We need to provide her with a winning strategy in the game $\mathcal{E}' := \mathcal{E}(\xi, \mathbb{S}')@(\xi, s')$. She obtains her strategy f' in $\mathcal{E}'$ from playing a *shadow match* of $\mathcal{E}$, using the bisimilarity relation to guide her choices.

To see how this works, let's simply start with comparing the initial position (ξ, s') of $\mathcal{E}'$ with its counterpart (ξ, s) of $\mathcal{E}$. (From now on we will write $s \leftrightarroweq s'$ instead of $s \leftrightarroweq_{\mathsf{P}} s'$).

In case ξ is an atomic formula, then it is easy to see that both (ξ, s) and (ξ, s') are final positions. Also, since f is assumed to be winning, ξ must be true at s, and so it must hold at s' as well. Hence, $\exists$ wins the match.

If ξ is not atomic, we distinguish cases. First suppose that $\xi = \xi_1 \vee \xi_2$. If f tells $\exists$ to choose disjunct ξ_i at (ξ, s), then she chooses the same disjunct ξ_i at position (ξ, s'). If $\xi = \xi_1 \wedge \xi_2$, it is $\forall$ who moves. Suppose in $\mathcal{E}'$ he chooses ξ_i, making (ξ_i, s') the next position. We now consider in $\mathcal{E}$ the same move of $\forall$, so that the next position in the shadow match is (ξ_i, s).

A third possibility is that $\xi = \Diamond\psi$. In order to make her move at (ξ, s'), $\exists$ first looks at (ξ, s). Since f is a winning strategy, it indeed picks a successor t of s. Then because $s \leftrightarroweq s'$, there is a successor t' of s' such that $t \leftrightarroweq t'$. This t' is $\exists$'s move in $\mathcal{E}$, so that (ψ, t) and (ψ, t') are the next positions in $\mathcal{E}$ and $\mathcal{E}'$, respectively.

Finally, if $\xi = \Box\psi$, we are dealing again with positions for $\forall$. Suppose in $\mathcal{E}'$ he chooses the successor t' of s', so that the next position is (ψ, t'). (In case s' has no successors, $\forall$ immediately loses, so that there is nothing left to prove.) Now again we turn to the shadow match; by bisimilarity of s and s' there is a successor t of s such that $t \leftrightarroweq t'$. So we may assume that $\forall$ moves the game token of $\mathcal{E}$ to position (ψ, t).

The crucial observation is that if $\exists$ does not win immediately, then at least she can guarantee that the next positions in $\mathcal{E}$ and $\mathcal{E}'$ are of the form (φ, u) and (φ, u') respectively, with $u \leftrightarroweq u'$, and such that the move in $\mathcal{E}$ is consistent with f.

Continuing in this fashion, $\exists$ is able to maintain the condition (*) that for any match

$$\beta' = (\varphi_0, s'_0)(\varphi_1, s'_1) \dots (\varphi_n, s'_n)$$

played thus far, there is a shadow match

$$\beta = (\varphi_0, s_0)(\varphi_1, s_1)\dots(\varphi_n, s_n)$$

in $\mathcal{E}$ which is consistent with f, and such that $Z : s_i \leftrightarrows s'_i$ for all $i \leq n$.

It is not hard to see why this suffices to prove the theorem; for infinite matches, the key observation is that the two sequences of formulas, in the $\mathcal{E}'$-match and its $\mathcal{E}$-shadow, respectively, are exactly the same. QED

As an immediate corollary, we obtain the tree model property for the modal μ-calculus.

Theorem 2.18 (Tree Model Property) *Let ξ be a modal fixpoint formula. If ξ is satisfiable, then it is satisfiable at the root of a tree model.*

Proof. For simplicity, we confine ourselves to the basic modal language. Suppose that ξ is satisfiable at state s of the Kripke model $\mathbb{S}$. Then by bisimulation invariance, ξ is satisfiable at the root of the *unravelling* $\vec{\mathbb{S}}_s$ of $\mathbb{S}$ around s. This unravelling clearly is a tree model. QED

For the next theorem, recall that the size of a formula is simply defined as its length, that is, the number of symbols occurring in it.

Theorem 2.19 (Bounded Tree Model Property) *Let ξ be a modal fixpoint formula. If ξ is satisfiable, then it is satisfiable at the root of a tree, of which the branching degree is bounded by the size $|\xi|$ of the formula.*

Proof. Suppose that ξ is satisfiable. By the Bisimulation Invariance Theorem it follows that ξ is satisfiable at the root r of some tree model $\mathbb{T} = \langle T, R, V\rangle$. So $\exists$ has a winning strategy f in the game $\mathcal{E} := \mathcal{E}(\xi, \mathbb{T})$ starting at position (ξ, r). By the Memory-Free Determinacy of evaluation game, we may assume that this strategy is positional — this will simplify our argument a bit. We may thus represent this strategy as a map f that, among other things, maps positions of the form $(s, \Diamond\varphi)$ to positions of the form (t, φ) with Rst.

We will prune the tree $\mathbb{T}$, keeping only the nodes that $\exists$ needs in order to win the match. Formally, define subsets $(T_n)_{n\in\omega}$ as follows:

$$\begin{array}{lcl}
T_0 & := & \{r\}, \\
T_{n+1} & := & T_n \cup \{s \mid (\varphi, s) = f(\Diamond\varphi, t) \text{ for some } t \in T_n \text{ and } \Diamond\varphi \trianglelefteq \xi\}, \\
T_\omega & := & \bigcup_{n\in\omega} T_n.
\end{array}$$

Let $\mathbb{T}_\omega$ be the subtree of $\mathbb{T}$ based on T_ω ($\mathbb{T}_\omega$ is in general not a generated submodel of $\mathbb{T}$). From the construction it is obvious that the branching degree of $\mathbb{T}_\omega$ is bounded by the length of ξ, because ξ has at most $|\xi|$ diamond subformulas.

We claim that $\mathbb{T}_\omega, r \Vdash_g \xi$. To see why this is so, let $\mathcal{E}' := \mathcal{E}(\xi, \mathbb{T}_\omega)$ be the evaluation game played on the pruned tree. It suffices to show that the strategy f', defined as the restriction of f to positions of the game $\mathcal{E}'$, is winning for $\exists$ in the game starting at (ξ, r). Consider an arbitrary $\mathcal{E}'$-match $\pi = (\xi, r)(\varphi_1, t_1) \dots$ which is consistent with f'. The key observation of the proof is that π is also a match of $\mathcal{E}@(\xi, r)$, that is consistent with f. To see this, simply observe that all moves of $\forall$ in π could have been made in the game on $\mathbb{T}$ as well, whereas by construction, all f' moves of $\exists$ in $\mathcal{E}'$ are f moves in $\mathcal{E}$.

Now by assumption, f is a winning strategy for $\exists$ in $\mathcal{E}$, so she wins π in $\mathcal{E}$. But then π is winning as such, i.e., no matter whether we see it as a match in $\mathcal{E}$ or in $\mathcal{E}'$. In other words, π is also winning as an $\mathcal{E}'$-match. And since π was an arbitrary $\mathcal{E}'$ match starting at (ξ, r), this shows that f' is a winning strategy, as required. QED

Notes

The modal μ-calculus was introduced by D. Kozen [13]. Its game-theoretical semantics goes back to at least Emerson & Jutla [9] (who use alternating automata as an intermediate step). As far as we are aware, the bisimulation invariance theorem, with the associated tree model property, is a folklore result. The bounded tree model property is due to Kozen & Parikh [15].

Exercises

3 Fixpoints

The game-theoretic semantics of the modal μ-calculus introduced in the previous chapter has some attractive characteristics. It is intuitive, relatively easy to understand, and, as we shall see further on, it can be used to prove some strong properties of the formalism. However, there are drawbacks as well. For instance, the evaluation games of Definition 2.8 can only be played with *clean* formulas. The semantics of arbitrary formulas is defined in a slightly artificial way. Perhaps more importantly, the game-theoretical semantics is not *compositional*; that is, the meaning of a formula is not defined in terms of the meanings of its subformulas. These shortcomings vanish in the *algebraic semantics* that we are about to introduce. In order to define this term, we first consider an example.

Example 3.1 Recall that in Example 2.1, we informally introduced the formula $\mu x.p \vee \Diamond_d x$ as the smallest fixpoint or solution of the 'equation' $x \leftrightarrow p \vee \Diamond_d x$.

To make this intuition more precise, we have to look at the formula $\delta = p \vee \Diamond_d x$ as an operation. The idea is that the value (that is, the extension) of this formula is a function of the value of x, provided that we keep the value of p constant. Varying the value of x boils down to considering 'x-variants' of the valuation V of $\mathbb{S} = \langle S, R, V \rangle$. Let, for $X \subseteq S$, $V[x \mapsto X]$ denote the valuation that is exactly like V apart from mapping x to X, and let $\mathbb{S}[x \mapsto X]$ denote the x-variant $\langle S, R, V[x \mapsto X] \rangle$ of $\mathbb{S}$. Then $[\![\delta]\!]^{\mathbb{S}[x\mapsto X]}$ denotes the extension of δ in this x-variant. It follows from this that the formula δ *induces* the following function $\delta^{\mathbb{S}}_x$ on the power set of S:

$$\delta^{\mathbb{S}}_x(X) := [\![\delta]\!]^{\mathbb{S}[x\mapsto X]}.$$

In our example we have

$$\delta^{\mathbb{S}}_x(X) = V(p) \cup \langle R \rangle(X).$$

Now we can make precise why $\mu x.p \vee \Diamond_d x$ is a fixpoint formula: its extension, the set $[\![\mu x.p \vee \Diamond_d x]\!]$, is a fixpoint of the map $\delta^{\mathbb{S}}_x$:

$$[\![\mu x.p \vee \Diamond_d x]\!] = V(p) \cup \langle R \rangle([\![\mu x.p \vee \Diamond_d x]\!]).$$

In fact, as we shall see in this chapter, the formulas $\mu x.p \vee \Diamond_d x$ and $\nu x.p \vee \Diamond_d x$ are such that their extensions are the *least* and *greatest* fixpoints of the map $\delta^{\mathbb{S}}_x$, respectively. $\lhd$

It is worthwhile to discuss the theory of fixpoint operators at a more general level than that of modal logic. Before we turn to the definition of the algebraic semantics of the modal μ-calculus, we first discuss the general fixpoint theory of monotone operations on complete lattices.

3.1 General fixpoint theory

Basics

In this chapter we assume some familiarity[1] with partial orders and lattices (see Appendix A).

Definition 3.2 Let $\mathbb{P}$ and $\mathbb{P}'$ be two partial orders and let $f : P \to P'$ be some map. Then f is called *monotone* if $f(x) \leq' f(y)$ whenever $x \leq y$, and *antitone* if $f(x) \geq' f(y)$ whenever $x \leq y$. $\lhd$

Definition 3.3 Let $\mathbb{P} = \langle P, \leq \rangle$ be some partial order, and let $f : P \to P$ be some map. Then an element $p \in P$ is called a *prefixpoint* of f if $f(p) \leq p$, a *postfixpoint* of f if $f(p) \geq p$, and a *fixpoint* if $f(p) = p$. The sets of prefixpoints, postfixpoints, and fixpoints of f are denoted respectively as $\mathrm{PRE}(f)$, $\mathrm{POS}(f)$ and $\mathrm{FIX}(f)$.

In case the set of fixpoints of f has a least (respectively greatest) member, this element is denoted as $\mathrm{LFP}.f$ ($\mathrm{GFP}.f$, respectively). These least and greatest fixpoints may also be called *extremal fixpoints*. $\lhd$

The following theorem is a celebrated result in fixpoint theory.

Theorem 3.4 (Knaster-Tarski) *Let $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$ be a complete lattice, and let $f : C \to C$ be monotone. Then f has both a least and a greatest fixpoint, and these are given as*

$$\mathrm{LFP}.f = \bigwedge \mathrm{PRE}(f), \tag{8}$$

$$\mathrm{GFP}.f = \bigvee \mathrm{POS}(f). \tag{9}$$

Proof. We will only prove the result for the least fixpoint, the proof for the greatest fixpoint is completely analogous.

Define $q := \bigwedge \mathrm{PRE}(f)$, then we have that $q \leq x$ for all prefixpoints x of f. From this it follows by monotonicity that $f(q) \leq f(x)$ for all $x \in \mathrm{PRE}(f)$, and hence by definition of prefixpoints, $f(q) \leq x$ for all $x \in \mathrm{PRE}(f)$. In other words, $f(q)$ is a lower bound of the set $\mathrm{PRE}(f)$. Hence, by definition of q as the *greatest* such lower bound, we find $f(q) \leq q$, that is, q itself is a prefixpoint of f.

It now suffices to prove that $q \leq f(q)$, and for this we may show that $f(q)$ is a prefixpoint of f as well, since q is by definition a lower bound of the set of prefixpoints. But in fact, we may show that $f(y)$ is a prefixpoint of f for *every* prefixpoint y of f — by monotonicity of f it immediately follows from $f(y) \leq y$ that $f(f(y)) \leq f(y)$. QED

Another way to obtain least and greatest fixpoint is to *approximate* them from below and above, respectively.

[1]Readers lacking this may take abstract complete lattices to be concrete power set algebras.

Definition 3.5 Let $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$ be a complete lattice, and let $f : C \to C$ be some map. Then by ordinal induction we define the following maps on C:

$$\begin{array}{rcl} f_\mu^0(c) & := & c, \\ f_\mu^{\alpha+1}(c) & := & f(f_\mu^\alpha(c)) \\ f_\mu^\lambda(c) & := & \bigvee_{\alpha<\lambda} f_\mu^\alpha(c), \end{array}$$

where λ denotes an arbitrary limit ordinal. Dually, we put

$$\begin{array}{rcl} f_\nu^0(c) & := & c, \\ f_\nu^{\alpha+1}(c) & := & f(f_\nu^\alpha(c)), \\ f_\nu^\lambda(c) & := & \bigwedge_{\alpha<\lambda} f_\nu^\alpha(c), \end{array}$$

$\lhd$

Proposition 3.6 *Let $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$ be a complete lattice, and let $f : C \to C$ be monotone. Then f is inductive, that is, $f_\mu^\alpha(\bot) \leq f_\mu^\beta(\bot)$ for all ordinals α and β such that $\alpha < \beta$.*

Proof. We leave this proof as an exercise to the reader. QED

Given a set C, we let $|C|$ denote its cardinality or size.

Corollary 3.7 *Let $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$ be a complete lattice, and let $f : C \to C$ be monotone. Then there is some α of size at most $|C|$ such that* LFP.$f = f_\mu^\alpha(\bot)$.

Proof. By Proposition 3.6, f is inductive, that is, $f_\mu^\alpha(\bot) \leq f_\mu^\beta(\bot)$ for all ordinals α and β such that $\alpha < \beta$. It follows from elementary set theory that there cannot be an injection from the set of ordinals of cardinality at most $|C|^+$ into C. From these two observations it is immediate that there must be two ordinals $\alpha, \beta < |C|^+$ such that $f_\mu^\alpha(\bot) = f_\mu^\beta(\bot)$. From the definition of the approximations it then follows that there must be an ordinal α such that $f_\mu^\alpha(\bot) = f_\mu^{\alpha+1}(\bot)$, or, equivalently, $f_\mu^\alpha(\bot)$ is a fixpoint of f. To show that it is the *smallest* fixpoint, one may prove that $f_\mu^\beta(\bot) \leq$ LFP.f for every ordinal β. This follows from a straightforward ordinal induction. QED

Definition 3.8 Let $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$ be a complete lattice, and let $f : C \to C$ be monotone. The least ordinal α such that $f_\mu^\alpha(\bot) = f_\mu^{\alpha+1}(\bot)$ is called the *closure ordinal* of f. $\lhd$

Multidimensional fixpoints

Suppose that we are given a finite family $\{\mathbb{C}_1, \ldots, \mathbb{C}_n\}$ of complete lattices, and put $\mathbb{C} = \prod_{1 \leq i \leq n} \mathbb{C}_i$. Given a finite family of monotone maps $f_1, \ldots, f_n$ with $f_i : C \to C_i$, we may define the map $f : C \to C$ given by $f(c) = (f_1(c), \ldots, f_n(c))$. Monotonicity of f is an easy consequence of the monotonicity of the individual f_i, and so by completeness of $\mathbb{C}$, f has a least and a greatest fixpoint. An obvious question is whether one may express these multi-dimensional fixpoints in terms of one-dimensional fixpoints of maps that one may associate with $f_1, \ldots, f_n$.

It will be convenient to introduce some notation. Given a monotone map $g : C \to C_i$ and an $n-1$-tuple $\bar{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, we let $g_{\bar{x}} : C_i \to C_i$ denote the map given by

$$g_{\bar{x}}(x_i) := g(x_1, x_2, \ldots, x_n).$$

The least and greatest fixpoints of this operation will be denoted as $\mu x_i.g(x_1, x_2, \ldots, x_n)$ and $\nu x_i.g(x_1, x_2, \ldots, x_n)$, respectively. Furthermore, in this context we will also use vector notation, for instance writing

$$\mu \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} . \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots \\ f_2(x_1, \ldots, x_n) \end{pmatrix}$$

for LFP.f.

The basic observation facilitating the computation of multidimensional fixpoints is the following so-called *Bekič principle.*

Proposition 3.9 *Let $\mathbb{D}_1$ and $\mathbb{D}_2$ be two complete lattices, and let $f_i : D_1 \times D_2 \to D_i$ for $i = 1, 2$ be monotone maps. Then*

$$\eta \begin{pmatrix} x \\ y \end{pmatrix} . \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \end{pmatrix} = \begin{pmatrix} \eta x.f_1(x, \eta y.f_2(x, y)) \\ \eta y.f_2(\eta x.f_1(x, y), y) \end{pmatrix}$$

where η uniformly denotes either μ or ν.

Proof. Define $\mathbb{D} := \mathbb{D}_1 \times \mathbb{D}_2$, and let $f : D \to D$ be given by putting $f(d) := (f_1(d), f_2(d))$. Then f is clearly monotone, and so it has both a least and a greatest fixpoint.

By the order duality principle it suffices to consider the case of least fixed points only. Suppose that (a_1, a_2) is the least fixpoint of f, and let b_1 and b_2 be given by

$$\begin{cases} b_1 & := & \eta x.f_1(x, \eta y.f_2(x, y), \\ b_2 & := & \eta y.f_2(\eta x.f_1(x, y), y). \end{cases}$$

Then we need to show that $a_1 = b_1$ and $a_2 = b_2$.

By definition of (a_1, a_2) we have

$$\begin{cases} a_1 &= f_1(a_1, a_2), \\ a_2 &= f_2(a_1, a_2), \end{cases}$$

whence we obtain

$$\begin{cases} \mu x.f_1(x, a_2) &\leq a_1 \quad \text{and} \\ \mu y.f_2(a_1, y) &\leq a_2, \end{cases}$$

From this we obtain by monotonicity that

$$f_1(\mu x.f_1(x, a_2)) \leq f_1(a_1, a_2) = a_1,$$

so that we find $b_1 \leq a_1$. Likewise we may show that $b_2 \leq a_2$.

Conversely, by definition of b_1 and b_2 we have

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} f_1(b_1, \mu y.f_2(b_1, y)) \\ f_2(\mu x.f_1(x, b_2), b_2) \end{pmatrix}.$$

Then with $c_2 := \mu y.f_2(b_1, y)$, we have $b_1 = f_1(b_1, c_1)$. Also, by definition of c_2 as a fixpoint, $c_2 = f_2(b_1, c_2)$. Putting these two identities together, we find that

$$\begin{pmatrix} b_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} f_1(b_1, c_2) \\ f_2(b_1, c_2) \end{pmatrix} = f \begin{pmatrix} b_1 \\ c_2 \end{pmatrix}.$$

Hence by definition of (a_1, a_2), we find that $a_1 \leq b_1$ (and that $a_2 \leq c_2$), but that is of less interest now). Analogously, we may show that $a_2 \leq b_2$. QED

Using induction on the dimension, Proposition 3.9 allows us to compute the least and greatest fixpoints of any monotone map f on a finite product of complete lattices in terms of the least and greatest fixpoints of operations on the factors of the product. The correctness of this *elimination method*, which is reminiscent of Gauss elimination in linear algebra, is a direct consequence of Proposition 3.9.

To see how it works, suppose that we are dealing with lattices $\mathbb{C}_1, \ldots, \mathbb{C}_{n+1}, \mathbb{C}$ and maps $f_1, \ldots, f_{n+1}, f$, just as described above, and that we want to compute $\eta\vec{x}.f$, that is, find the elements $a_1, \ldots, a_{n+1}$ such that

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n+1} \end{pmatrix} = \eta \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \end{pmatrix} . \begin{pmatrix} f_1(x_1, \ldots, x_n, x_{n+1}) \\ f_2(x_1, \ldots, x_n, x_{n+1}) \\ \vdots \\ f_{n+1}(x_1, \ldots, x_n, x_{n+1}) \end{pmatrix}$$

We may define

$$g_{n+1}(x_1, \ldots, x_n) := \eta x_{n+1}.f_{n+1}(x_1, \ldots, x_n),$$

and then use Proposition 3.9, with $\mathbb{D}_1 = \mathbb{C}_1 \times \cdots \times \mathbb{C}_n$, and $\mathbb{D}_2 = \mathbb{C}_{n+1}$, to obtain

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \eta \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} . \begin{pmatrix} f_1(x_1, \ldots, x_n, g_{n+1}(x_1, \ldots, x_n)) \\ f_2(x_1, \ldots, x_n, g_{n+1}(x_1, \ldots, x_n)) \\ \vdots \\ f_n(x_1, \ldots, x_n, g_{n+1}(x_1, \ldots, x_n)) \end{pmatrix}$$

We may then inductively assume to have obtained the tuple $(a_1, \ldots, a_n)$. Finally, we may compute $a_{n+1} := g_{n+1}(a_1, \ldots, a_n)$.

Observe that in case $\mathbb{C}_i = \mathbb{C}_j$ for all i, j and the operations f_i are all term definable in some formal algebraic fixpoint language, then each the components a_i of the extremal fixpoints of f can also be expressed in this language.

3.2 Boolean algebras

In the special case that the complete lattice is in fact a (complete) *Boolean algebra*, there is more to be said.

Dual maps

In the case of monotone maps on complete Boolean algebras, the least and greatest fixed points become interdefinable, using the notion of (Boolean) *duals* of maps.

Definition 3.10 A *complete* Boolean algebra is a structure $\mathbb{B} = \langle B, \bigvee, \bigwedge, - \rangle$ such that $\langle B, \bigvee, \bigwedge \rangle$ is a complete lattice, and $- : B \to B$ is an antitone map such that $x \wedge -x = \bot$ and $x \vee -x = \top$ for all $x \in B$. $\lhd$

In a complete Boolean algebra $\mathbb{B} = \langle B, \bigvee, \bigwedge, - \rangle$, it holds that $-\bigvee X = \bigwedge\{-x \mid x \in X\}$ and $-\bigwedge X = \bigvee\{-x \mid x \in X\}$.

Definition 3.11 Let $\mathbb{B} = \langle B, \bigvee, \bigwedge, - \rangle$ be a complete Boolean algebra, and consider $f : B \to B$ be an arbitrary map. Then the *(Boolean) dual map* of f is defined as the map $\tilde{f} : B \to B$ given by

$$\tilde{f}(b) := -f(-b).$$

$\lhd$

Proposition 3.12 *Let* $\mathbb{B} = \langle B, \bigvee, \bigwedge, - \rangle$ *be a complete Boolean algebra, and let* $f : B \to B$ *be monotone. Then* $\tilde{f}$ *is monotone as well,* $\tilde{\tilde{f}} = f$*, and*

$$\begin{aligned} \mathrm{LFP}.\tilde{f} &= -\mathrm{GFP}.f, \\ \mathrm{GFP}.\tilde{f} &= -\mathrm{LFP}.f. \end{aligned}$$

Proof. We only prove that LFP.$\tilde{f} = -$GFP.f, leaving the other parts of the proof as exercises to the reader.

First, note that by monotonicity of $\tilde{f}$, the Knaster-Tarski theorem gives that

$$\mathrm{LFP}.\tilde{f} = \bigwedge \mathrm{PRE}(\tilde{f}).$$

But as a consequence of the definitions, we have that

$$b \in \mathrm{PRE}(\tilde{f}) \iff -b \in \mathrm{POS}(f).$$

From this it follows that

$$\begin{aligned}\mathrm{LFP}.\tilde{f} &= \bigwedge\{-b \mid b \in \mathrm{POS}(f)\} \\ &= -\bigvee \mathrm{POS}(f) \\ &= -\mathrm{GFP}.f\end{aligned}$$

which finishes the proof of the Theorem. QED

Further on we will see that Proposition 3.12 allows us to define negation as an abbreviated operator in the modal μ-calculus.

Games

In case the Boolean algebra in question is in fact a *power set algebra*, a nice game-theoretic characterization of least and greatest fixpoint operators is possible.

Definition 3.13 Let S be some set and let $F : \wp(S) \to \wp(S)$ be a monotone operation. Consider the *unfolding games* $\mathcal{U}^\mu(F)$ and $\mathcal{U}^\nu(F)$. The positions and admissible moves of these two graph games are the same, see Table 5.

Position	Player	Admissible moves
$s \in S$	$\exists$	$\{A \in \wp(S) \mid s \in F(A)\}$
$A \in \wp(S)$	$\forall$	$A (= \{s \in S \mid s \in A\})$

Table 5: Unfolding games for $F : \wp(S) \to \wp(S)$

The *winning conditions* of finite matches are standard (the player that got stuck loses the match). The difference between $\mathcal{U}^\mu(F)$ and $\mathcal{U}^\nu(F)$ shows up in the winning conditions of infinite matches: $\exists$ wins the infinite matches of $\mathcal{U}^\nu(F)$, but $\forall$ those of $\mathcal{U}^\mu(F)$. $\lhd$

Then the following proposition substantiates the slogan that 'ν means unfolding, μ means finite unfolding'.

Theorem 3.14 *Let S be some set and let $F : \wp(S) \to \wp(S)$ be a monotone operation. Then*

1. $\mathrm{GFP}.F = \{s \in S \mid \mathrm{Win}_\exists(\mathcal{U}^\nu(F))\}$,

2. $\mathrm{LFP}.F = \{s \in S \mid \mathrm{Win}_\exists(\mathcal{U}^\mu(F))\}$,

Proof. For the inclusion $\supseteq$ of part 1, it suffices to prove that $W := S \cap \mathrm{Win}_\exists(\mathcal{U}^\nu(F))$ is a postfixpoint of F:

$$W \subseteq F(W). \tag{10}$$

Let s be an arbitrary point in W, and suppose that $\exists$'s winning strategy tells her to choose $A \subseteq S$ at position s. Then no matter what element $s_1 \in A$ is picked by $\forall$, $\exists$ can continue the match and win. Hence, all elements of A are winning positions for $\exists$. But from $A \subseteq W$ it follows that $F(A) \subseteq F(W)$, and by the legitimacy of $\exists$'s move A at s it follows that $s \in F(A)$. We conclude that $s \in F(W)$, which proves (10).

For the converse inclusion $\subseteq$ of part 1 of the proposition, take an arbitrary point $s \in \mathrm{GFP}.F$. We need to provide $\exists$ with a winning strategy in the unfolding game $\mathcal{U}^\nu(F)$ starting at s. This strategy is actually as simple as can be: $\exists$ should always play $\mathrm{GFP}.F$. Since $\mathrm{GFP}.F = F(\mathrm{GFP}.F)$, this strategy prescribes legitimate moves for $\exists$ at every point in $\mathrm{GFP}.F$. And, if she sticks to this strategy, $\exists$ will stay alive forever and thus win the match, no matter what $\forall$'s responses are.

For the second part of the theorem, let W denote the set $\mathrm{Win}_\exists(\mathcal{U}^\mu(F))$ of $\exists$'s winning positions in $\mathcal{U}^\mu(F)$. We first prove the inclusion $W \subseteq \mathrm{LFP}.F$. Clearly it suffices to show that all points outside the set $\mathrm{LFP}.F$ are winning positions for $\forall$.

Consider a point $s \notin \mathrm{LFP}.F$. If $s \notin F(A)$ for any $A \subseteq S$ then $\exists$ loses immediately, and we are done. Otherwise, suppose that $\exists$ starts a match of $\mathcal{U}^\mu(F)$ by playing some set $B \subseteq S$ with $s \in F(B)$. We claim that B is not a subset of $\mathrm{LFP}.F$, since otherwise we would have $F(B) \subseteq F(\mathrm{LFP}.F) \subseteq \mathrm{LFP}.F$; which would contradict the fact that $s \notin \mathrm{LFP}.F$. But if $B \not\subseteq \mathrm{LFP}.F$ then $\forall$ may continue the match by choosing a point $s_1 \in B \setminus \mathrm{LFP}.F$. Now $\forall$ can use the same strategy from s_1 as he used from s, and so on. This strategy guarantees that either $\exists$ gets stuck after finitely many rounds (in case $\forall$ manages to pick an s_n for which there is no A such that $s_n \in F(A_n)$), or else the match will last forever. In both cases $\forall$ wins the match.

The other inclusion $\subseteq$ of part 2 is easily proved using the ordinal approximation of least fixpoints. Using the fact that $\mathrm{LFP}.F = \bigcup\{F^\alpha_\mu(\varnothing) \mid \alpha \text{ an ordinal }\}$, it suffices to prove that

$$F^\alpha_\mu(\varnothing) \subseteq \mathrm{Win}_\exists(\mathcal{U}^\mu(F))$$

for all α. This proof proceeds by a transfinite induction, of which we only provide the case for successor ordinals. Let $\alpha = \beta + 1$ be some successor ordinal and inductively assume that $\exists$ has a winning strategy f_t for every point $t \in F^\beta_\mu(\varnothing)$. We need to provide her with a strategy which is winning from an arbitrary position $s \in F^\alpha_\mu(\varnothing)$. By definition $F^\alpha_\mu(\varnothing) = F(F^\beta_\mu(\varnothing))$, so $\exists$ may legitimately choose the set $F^\beta_\mu(\varnothing)$ as her first move at position s, and then, confronted with $\forall$ choosing a point, say, t, from $F^\beta_\mu(\varnothing)$, continue with the strategy f_t. It is almost immediate that this is a winning strategy for $\exists$. QED

Remark 3.15 Note that the proof of Theorem 3.14 witnesses a fundamental *asymmetry* in the treatment of least and greatest fixpoints in the unfolding game. In order to show that a state s belongs to one of the extremal fixpoints of a monotone map F, in both cases the approach is 'from below', i.e., in the game $\exists$ tries to provide positive evidence that s belongs to the given kind of fixpoint. However, in the case of the *least* fixpoint, this evidence from below consists of the ordinal approximations of LFP.F, whereas in the case of the *greatest* fixpoint, in the end what she tries to show is that the point in question belongs to *some* postfixpoint. Phrased differently, the game characterization of the greatest fixpoint of F uses the Knaster-Tarski characterization (8), whereas the characterization of the least fixpoint uses the ordinal approximation of Corollary 3.7. $\lhd$

3.3 Algebraic semantics for the modal μ-calculus

Basic definitions

In order to define the algebraic semantics of the modal μ-calculus, we need to consider formulas as *operations* on the power set of the (state space of a) transitions system, and we have to prove that such operations indeed have least and greatest fixpoints. In order to make this precise, we need some preliminary definitions.

Definition 3.16 Given an LTS $\mathbb{S} = \langle S, V, R\rangle$ and subset $X \subseteq S$, define the valuation $V[x \mapsto X]$ by putting

$$V[x \mapsto X](y) \;:=\; \begin{cases} V(y) & \text{if } y \neq x, \\ X & \text{if } y = x. \end{cases}$$

Then, the LTS $\mathbb{S}[x \mapsto X]$ is given as the structure $\langle S, V[x \mapsto X], R\rangle$. $\lhd$

Now inductively assume that $[\![\varphi]\!]^{\mathbb{S}}$ has been defined for all LTSs. Given a labelled transition system $\mathbb{S}$ and a propositional variable $x \in \mathsf{P}$, each formula φ induces a map $\varphi^{\mathbb{S}}_x : \wp(S) \to \wp(S)$ defined by

$$\varphi^{\mathbb{S}}_x(X) := [\![\varphi]\!]^{\mathbb{S}[x \mapsto X]}$$

Example 3.17 a) Where $\varphi_a = p \vee x$ we have $(\varphi_a)^{\mathbb{S}}_x(X) = [\![p \vee x]\!]^{\mathbb{S}[x\mapsto X]} = V(p) \cup X$.
b) Where $\varphi_b = \neg x$ we have $(\varphi_b)^{\mathbb{S}}_x(X) = [\![\neg x]\!]^{\mathbb{S}[x\mapsto X]} = S \setminus X$.
c) Where $\varphi_c = p \vee \Diamond_d x$ we find $(\varphi_c)^{\mathbb{S}}_x(X) = [\![p \vee \Diamond_d x]\!]^{\mathbb{S}[x\mapsto X]} = V(p) \cup \langle R_d\rangle X$.
d) Where $\varphi_d = \Diamond_d \neg x$ we find $(\varphi_d)^{\mathbb{S}}_x(X) = [\![\Diamond_d \neg x]\!]^{\mathbb{S}[x\mapsto X]} = \langle R_d\rangle(S \setminus X)$. $\lhd$

Alternatively but equivalently, X is a fixpoint of $\varphi_x^{\mathbb{S}}$ iff $\mathbb{S}[x \mapsto X] \Vdash x \leftrightarrow \varphi$. Likewise, X is a *prefixpoint* of $\varphi_x^{\mathbb{S}}$ iff $\mathbb{S}[x \mapsto X] \Vdash \varphi \to x$, and a *postfixpoint* of $\varphi_x^{\mathbb{S}}$ iff $\mathbb{S}[x \mapsto X] \Vdash x \to \varphi$. Here we write $\mathbb{S}, s \Vdash \varphi$ for $s \in [\![\varphi]\!]^{\mathbb{S}}$.

Example 3.18 Consider the formulas of Example 3.17.

a) The sets $V(p)$ and S are fixpoints of φ_a, as is in fact any X with $V(p) \subseteq X \subseteq S$.

b) Since we do not consider structures with empty domain, the formula $\neg x$ has no fixpoints at all. (Otherwise we would find $X = {\sim_S} X$ for some $S \neq \varnothing$, a contradiction.)

c) Two fixpoints of φ_c were already given in Example 2.1.

d) Consider any model $\mathbb{Z} = \langle Z, S, V \rangle$ based on the set Z of integers, where $S = \{(z, z+1) \mid z \in Z\}$ is the successor relation. Then the only two fixpoints of φ_d are the sets of even and odd numbers, respectively. $\lhd$

In particular, it is not the case that every formula has a least fixpoint. If we can guarantee that the induced function $\varphi_x^{\mathbb{S}}$ of φ is monotone, however, then the Knaster-Tarski theorem (Theorem 3.4) provides both least and greatest fixpoints of $\varphi_x^{\mathbb{S}}$. Precisely for this reason, in the definition of fixpoint formulas, we imposed the condition in the clauses for $\eta x.\varphi$, that x may only occur positively in φ. As we will see, this condition on x guarantees monotonicity of the function $\varphi_x^{\mathbb{S}}$.

Definition 3.19 Given a μPML-formula φ and a labelled transition system $\mathbb{S} = \langle S, V, R \rangle$, we define the *meaning* $[\![\varphi]\!]^{\mathbb{S}}$ of φ in $\mathbb{S}$, together with the map $\varphi_x^{\mathbb{S}} : \wp(S) \to \wp(S)$ by the following simultaneous formula induction:

$$\begin{array}{lcl}
[\![\bot]\!]^{\mathbb{S}} & = & \varnothing \\
[\![\top]\!]^{\mathbb{S}} & = & S \\
[\![p]\!]^{\mathbb{S}} & = & V(p) \\
[\![\neg p]\!]^{\mathbb{S}} & = & S \setminus V(p) \\
[\![\varphi \lor \psi]\!]^{\mathbb{S}} & = & [\![\varphi]\!]^{\mathbb{S}} \cup [\![\psi]\!]^{\mathbb{S}} \\
[\![\varphi \land \psi]\!]^{\mathbb{S}} & = & [\![\varphi]\!]^{\mathbb{S}} \cap [\![\psi]\!]^{\mathbb{S}} \\
[\![\Diamond_d \varphi]\!]^{\mathbb{S}} & = & \langle R_a \rangle [\![\varphi]\!]^{\mathbb{S}} \\
[\![\Box_d \varphi]\!]^{\mathbb{S}} & = & [R_a] [\![\varphi]\!]^{\mathbb{S}} \\
[\![\mu x.\varphi]\!]^{\mathbb{S}} & = & \bigcap \mathrm{PRE}(\varphi_x^{\mathbb{S}}) \\
[\![\nu x.\varphi]\!]^{\mathbb{S}} & = & \bigcup \mathrm{POS}(\varphi_x^{\mathbb{S}})
\end{array}$$

The map $\varphi_x^{\mathbb{S}}$, for $x \in \mathsf{P}$, is given by $\varphi_x^{\mathbb{S}}(X) = [\![\varphi]\!]^{\mathbb{S}[x \mapsto X]}$. $\lhd$

Theorem 3.20 *Let φ be an μPML-formula, in which x occurs only positively, and let $\mathbb{S}$ be a labelled transition system. Then $[\![\mu x.\varphi]\!]^{\mathbb{S}} = \mathrm{LFP}.\varphi_x^{\mathbb{S}}$, and $[\![\nu x.\varphi]\!]^{\mathbb{S}} = \mathrm{GFP}.\varphi_x^{\mathbb{S}}$.*

Proof. This is an immediate consequence of the Knaster-Tarski theorem, provided we can prove that $\varphi_x^{\mathbb{S}}$ is monotone in x if all occurrences of x in φ are positive. We leave the details of this proof to the reader (see Exercise 3.2). QED

Immediate consequences

It follows from the definitions that the set μPML is closed under taking *negations*. Informally, let $\sim\varphi$ be the result of simultaneously replacing all occurrences of $\top$ with $\bot$, of p with $\neg p$ (for *free* variables p), of $\wedge$ with $\vee$, of $\Box_d$ with $\Diamond_d$, of μx with νx, and vice versa, while leaving occurrences of bound variables unchanged. As an example, $\sim(\mu x.p \vee \Diamond x) = \nu x.\neg p \wedge \Box x$. Formally, we define $\sim$ as follows.

Definition 3.21 Given a modal fixpoint formula φ, define $\sim\varphi$ inductively as follows:

$$\begin{array}{lcllcl}
\sim\bot & := & \top & \sim\top & := & \bot \\
\sim\neg p & := & p & \sim p & := & \neg p \\
\sim\varphi \vee \psi & := & \sim\varphi \wedge \sim\psi & \sim\varphi \wedge \psi & := & \sim\varphi \vee \sim\psi \\
\sim\Box_d\varphi & := & \Diamond_d\sim\varphi & \sim\Diamond_d\varphi & := & \Box_d\sim\varphi \\
\sim\mu x.\varphi & := & \nu x.\sim\varphi[x/\neg x] & \sim\nu x.\varphi & := & \mu x.\sim\varphi[x/\neg x]
\end{array}$$

Here $\varphi[x/\neg x]$ is the formula φ with (note!) all occurrences of $\neg x$ replaced with x. $\lhd$

Perhaps the clause for the fixpoint operators requires some explanation. Consider for instance the case of $\sim\mu x.\varphi$. First observe that since in φ no x occurs in a subformula $\neg x$, in $\sim\varphi$ *all* occurrences of x are negated. Hence, if we replace every occurrence of $\neg x$ with x, we again obtain a formula in which no occurrence of x is below a negation sign, and hence, we may legitimately put a fixpoint operator in front of $\sim\varphi[\neg x/x]$. Note that the net effect of these syntactic transformations is that the *bound* variables of a fixpoint formula remain unchanged. As an example, the reader is invited to check that, indeed, $\sim(\mu x.p \vee \Diamond x) = \nu x.\neg p \wedge \Box x$.

The following proposition states that $\sim$ functions as a standard Boolean negation. We let $\sim_S X = S \setminus X$ denote the complement of X in S.

Proposition 3.22 *Let φ be a modal fixpoint formula. Then $\sim\varphi$ corresponds to the negation of φ, that is,*

$$[\![\sim\varphi]\!]^{\mathbb{S}} = \sim_S[\![\varphi]\!]^{\mathbb{S}} \tag{11}$$

for every labelled transition system $\mathbb{S}$.

Proof. We prove this proposition by induction on the complexity of φ. Leaving all other cases as exercises for the reader, we concentrate on the inductive case where φ is of the form $\mu x.\psi$.

In this case, the right hand side of (11), denotes the complement of the least fixed point of the operation $\psi^{\mathbb{S}}_x$. By Proposition 3.12, this is the same as the *greatest* fixpoint of the *dual* map $\widetilde{\psi^{\mathbb{S}}_x}$ given by $\widetilde{\psi^{\mathbb{S}}_x}(A) := \sim_S\psi^{\mathbb{S}}_x(\sim_S A)$. The left hand side of (11) denotes the greatest fixpoint of the map $(\sim\psi[x/\neg x])^{\mathbb{S}}_x$. Thus we are done if we can show that

$$(\sim\psi[x/\neg x])^{\mathbb{S}}_x \;=\; \widetilde{\psi^{\mathbb{S}}_x}. \tag{12}$$

For this purpose, take an arbitrary set $A \subseteq S$. Since all occurrences of x in $\sim\psi$ are inside a subformula $\neg x$, it follows that $(\sim\psi[x/\neg x])^{\mathbb{S}}_{x}(A) = (\sim\psi)^{\mathbb{S}}_{x}(\sim_S A) = [\![\sim\psi]\!]^{\mathbb{S}[x\mapsto\sim_S A]}$. Inductively, this is identical to the set $\sim_S[\![\psi]\!]^{\mathbb{S}[x\mapsto\sim_S A]} = \sim_S\psi^{\mathbb{S}}_{x}(\sim_S A)$.

This proves (12), and hence, the proposition. QED

Remark 3.23 It follows from the Proposition above that we could indeed have based the language of the modal μ-calculus on a far smaller alphabet of primitive symbols. Given sets P and D of proposition letters and atomic actions, respectively, we could have defined the set of modal fixpoint formulas using the following induction:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond_d\varphi \mid \mu x.\varphi$$

where $p, x \in \mathsf{P}$, $a \in \mathsf{D}$, and in $\mu x.\varphi$, all free occurrences of x must be positive (that is, under an even number of negation symbols). Here we define $FV(\neg\varphi) = FV(\varphi)$ and $BV(\neg\varphi) = BV(\varphi)$.

In this set-up, the connectives $\wedge$ and $\Box_d$ are defined using the standard abbreviations, while for the greatest fixpoint operator we may put

$$\nu x.\varphi := \neg\mu x.\neg\varphi(\neg x).$$

Note the *triple* use of the negation symbol that is required to maintain the positivity of x — explained by the earlier remarks. $\lhd$

Earlier on we defined the notions of *clean* and *guarded* formulas.

Proposition 3.24 *Every fixpoint formula is equivalent to a clean one.*

Proof. We leave this proof as an exercise for the reader. QED

Proposition 3.25 *Every fixpoint formula is equivalent to a guarded one.*

Proof.(Sketch) We prove this proposition by formula induction. Clearly the only non-trivial case to consider concerns the fixpoint operators. Consider a formula of the form $\eta x.\delta(x)$, where $\delta(x)$ is guarded and clean, and suppose that x has an unguarded occurrence in δ.

First consider an unguarded occurrence of x in $\delta(x)$ inside a fixpoint subformula, say, of the form $\theta y.\gamma(x, y)$. By induction hypothesis, all occurrences of y in $\gamma(x, y)$ are guarded. Obtain the formula $\overline{\delta}$ from δ by replacing the subformula $\theta y.\gamma(x, y)$ with $\gamma(x, \theta y.\gamma(x, y))$. Then clearly $\overline{\delta}$ is equivalent to δ, and all of the unguarded occurrences of x in $\overline{\delta}$ are outside of the scope of the fixpoint operator θ.

Continuing like this we obtain a formula $\eta x.\overline{\delta}(x)$ which is equivalent to $\eta x.\delta(x)$, and in which none of the unguarded occurrences of x lies inside the scope of a fixpoint operator. That leaves $\wedge$ and $\vee$ as the only operation symbols in the scope of which we may find unguarded occurrences of x.

From now on we only consider the case that $\eta = \mu$, the case where $\eta = \nu$ is very similar. Clearly, using the laws of classical propositional logic, we may bring the formula $\overline{\delta}$ into conjunctive normal form

$$(x \vee \alpha_1(x)) \wedge \cdots \wedge (x \vee \alpha_n(x)) \wedge \beta(x), \tag{13}$$

where all occurrences of x in $\alpha_1, \ldots, \alpha_n$ and β are guarded. (Note that we may have $\beta = \top$, or $\alpha_i = \bot$ for some i.)

Clearly (13) is equivalent to the formula

$$\delta'(x) := (x \vee \alpha(x)) \wedge \beta(x),$$

where $\alpha = \alpha_1 \wedge \cdots \wedge \alpha_n$. Thus we are done if we can show that

$$\mu x.\delta'(x) \;\equiv\; \mu x.\alpha(x) \wedge \beta(x). \tag{14}$$

Since $\alpha \wedge \beta$ implies δ', it is easy to see (and left for the reader to prove) that $\mu x.\alpha \wedge \beta$ implies $\mu x.\delta'$. For the converse, it suffices to show that $\varphi := \mu x.\alpha(x) \wedge \beta(x)$ is a prefixpoint of $\delta'(x)$. But it is not hard to derive from $\varphi \equiv \alpha(\varphi) \wedge \beta(\varphi)$ that

$$\delta'(\varphi) = (\varphi \vee \alpha(\varphi)) \wedge \beta(\varphi) \equiv ((\alpha(\varphi) \wedge \beta(\varphi)) \vee \alpha(\varphi)) \wedge \beta(\varphi) \equiv \alpha(\varphi) \wedge \beta(\varphi) \equiv \varphi,$$

which shows that in fact, φ is a fixpoint, and hence certainly a prefixpoint, of $\delta'(x)$. QED

Combining the proofs of the previous two propositions one easily shows the following.

Proposition 3.26 *Every fixpoint formula is equivalent to a clean, guarded one.*

3.4 Adequacy

In this section we prove the prove the *equivalence* of the two semantic approaches towards the modal μ-calculus. Since the algebraic semantics is usually taken to be the more fundamental notion, we refer to this result as the *Adequacy Theorem*: informally, it states that games are an adequate way of working with the algebraic semantics.

Theorem 3.27 (Adequacy) *Let ξ be a clean μPML-formula. Then for all labelled transition systems $\mathbb{S}$ and all states s in $\mathbb{S}$:*

$$s \in [\![\xi]\!]^{\mathbb{S}} \iff (\xi, s) \in \mathrm{Win}_{\exists}(\mathcal{E}(\xi, \mathbb{S})). \tag{15}$$

Proof. The theorem is proved by induction on the complexity of ξ. We only discuss the inductive step where ξ is of the form $\eta x.\delta$, leaving the other cases as exercises to the reader.

To start with, by definition of the map $\delta^{\mathbb{S}}_x$ and the inductive hypothesis we have, for all $A \subseteq S$,

$$s \in \delta^{\mathbb{S}}_x(A) \iff (\delta, s) \in \mathrm{Win}_\exists(\mathcal{E}(\delta, \mathbb{S}[x \mapsto A]). \tag{16}$$

Comparing (15) and (16), it will be important to observe that $\mathcal{G} := \mathcal{E}(\xi, \mathbb{S})$ and $\mathcal{G}_A := \mathcal{E}(\delta, \mathbb{S}[x \mapsto A])$ are *very* similar games. For a start, the positions of the two games are effectively the same. Positions of the form (ξ, t), which exist in the first game but not in the second, are the only exception — but in $\mathcal{G}$, any position (ξ, t) is immediately and automatically succeeded by the position (δ, t) which does exist in the second game. The only real difference between the games shows up in the rule concerning positions of the form (x, u). In $\mathcal{G}_A$, x is a *free* variable ($x \in FV(\delta)$), so in a position (u, x) the game is over, the winner being determined by u being a member of A or not. In $\mathcal{G}$ however, x is *bound*, so in position (x, u), the variable x will get unfolded.

In order to prove (15) for $\xi = \eta x.\delta$, by definition of $[\![\eta x.\delta]\!]^{\mathbb{S}}$ and Theorem 3.14, it suffices to show that

$$s \in \mathrm{Win}_\exists(\mathcal{U}^\eta(\delta^{\mathbb{S}}_x)) \iff (\xi, s) \in \mathrm{Win}(\mathcal{E}(\xi, \mathbb{S})). \tag{17}$$

In other words, the key insight in proof of the inductive step concerns the transformation winning strategies for $\exists$ from one game to another. The fundamental link between the two kinds of games in (17) is to think of $\mathcal{E}(\xi, \mathbb{S})$ as the unfolding game $\mathcal{U} := \mathcal{U}^\eta(\delta^{\mathbb{S}}_x)$ where each round of $\mathcal{U}$ corresponds to a version of the game $\mathcal{G}_A$ for some (dynamically varying) set A. We are now ready for the details of the proof.

For the direction from left to right of (17), suppose that $\exists$ has a winning strategy in the game $\mathcal{U}$ starting at some position s_0. Without loss of generality (see Exercise 3.6) we may assume that this strategy is *positional*, so we may represent it as a map $T : S \to \wp(S)$. By the legitimacy of this strategy, for every $s \in \mathrm{Win}_\exists(\mathcal{U})$ it holds that $s \in \delta^{\mathbb{S}}_x(T_s)$. So by the inductive hypothesis (16), for each such s we may assume the existence of a winning strategy f_s for $\exists$ in the game $\mathcal{G}_{T_s}@(\delta, s)$. Given the similarities between the games $\mathcal{G}@(x, s)$ and $\mathcal{G}_{T_s}@(\delta, s)$ (see the discussion above), this strategy is also valid in the game $\mathcal{G}@(x, s)$, at least, until a new position of the form (x, t) is reached.

This suggests the following strategy g for $\exists$ in $\mathcal{G}@(\xi, s_0)$:

- after the initial automatic move, the position of the match is (δ, s_0); $\exists$ first plays her strategy f_{s_0};
- each time a position (x, s) is reached, distinguish cases:

  (a) if $s \in \mathrm{Win}_\exists(\mathcal{U})$ then $\exists$ continues with f_s;

(b) if $s \notin \text{Win}_{\exists}(\mathcal{U})$ then $\exists$ continues with a random strategy.

First we show that this strategy guarantees that whenever a position of the form (x, s) is visited, s belongs to $\text{Win}_{\exists}(\mathcal{U})$, so that case (b) mentioned above never occurs. The proof is by induction on the number of positions (x, s) that have been visited already. For the inductive step, if s is a winning position for $\exists$ in $\mathcal{U}$, then, as we saw, f_s is a winning strategy for $\exists$ in the game $\mathcal{G}_{T_s}@(\delta, s)$. This means that if a position of the form (x, t) is reached, the variable x must be *true* at t in the model $\mathbb{S}[x \mapsto T_s]$, and so t must belong to the set T_s. But by assumption of the map $T : S \to \wp(S)$ being a winning strategy in $\mathcal{U}$, any element of T_s is again a member of $\text{Win}_{\exists}(\mathcal{U})$.

In fact we have shown that every unfolding of the variable x in $\mathcal{G}$ marks a new round in the unfolding game $\mathcal{U}$. To see why the strategy g guarantees a win for $\exists$ in $\mathcal{G}@(\xi, s_0)$, consider an arbitrary $\mathcal{G}@(\xi, s_0)$-match π in which $\exists$ plays g. Distinguish cases.

First suppose that x is unfolded only finitely often. Let (x, s) be the last basic position in π where this happens. Given the similarities between the games $\mathcal{G}$ and $\mathcal{G}_{T_s}$, the match from this moment on can be seen as both a g-conform $\mathcal{G}$-match and an f_s-conform $\mathcal{G}_{T_s}$-match. As we saw, f_s is a winning strategy for $\exists$ in the game $\mathcal{G}_{T_s}@(\delta, s)$. But since no further position of the form (x, t) is reached, and $\mathcal{G}$ and $\mathcal{G}_{T_s}$ only differ when it comes to x, this means that π is also a win for $\exists$ in $\mathcal{G}$.

If x is unfolded infinitely often, then because it is the *highest* variable of ξ, $\exists$ can only be the winner of the match π if x is a *greatest fixpoint variable*. In other words, we have to verify that $\eta = \nu$. Suppose that $s_1, s_2, \ldots$ are the positions where x is unfolded. Then it easy to verify that the sequence $s_0 T_{s_0} s_1 T_{s_1} \ldots$ constitutes a $\mathcal{U}$-match in which $\exists$ plays her strategy T. Since this strategy was supposed to be winning, we must be dealing with an unfolding game $\mathcal{U}^{\eta}$ for a *greatest* fixpoint, that is, we are indeed dealing with the case $\eta = \nu$.

For the converse implication of (17), assume that $\exists$ has a winning strategy, say f, in the game $\mathcal{G}@(\xi, s_0)$. We need to supply her with a winning strategy in the unfolding game $\mathcal{U}@s_0$. The basic idea is that while playing $\mathcal{U}$, $\exists$ builds an f-conform shadow match of $\mathcal{G}@(\xi, s_0)$ such that the positions in the $\mathcal{U}$-match of the form s_i exactly correspond to the basic positions of the form (δ, s_i) in the $\mathcal{G}$-match. (After the initial position (s_0, δ), these are exactly the ones where the variable x has just been unfolded.)

Clearly this holds for the initial position, where the partial $\mathcal{U}$-match consists of the trivial sequence s_0. As the associated partial $\mathcal{G}$-match we may take the sequence $(\eta x.\delta, s_0)(\delta, s_0)$. Inductively suppose that $\exists$ has kept the above condition for a partial match π of $\mathcal{U}@s_0$ with $\mathit{last}(\pi) = s_k$. Let $s_0, \ldots, s_k$ (in that order) be the state positions in π. By the inductive assumption, there is a f-conform partial match $\bar{\pi}$ ending at position (δ, s_k) (and in which (δ, s_0), ..., (δ, s_k) are the successive positions of the form (δ, s)). Let T_π be the set of states $t \in S$ for which there is some partial g-conform match ρ, *extending* $\bar{\pi}$, with $\mathit{last}(\rho)$ of the form (x, t), and which has no proper initial segment with these properties. That is, in the continuation (x, t) is the *first* position where x is unfolded. This set T_π will be $\exists' s$ move in $\mathcal{U}$ at position s_k.

We first show that T_π is actually a legitimate move. We only consider the case where $k = 0$. (The general case, which is conceptually the same but technically slightly more involved, is left as an exercise to the reader.)

The main observation in this proof is that $\exists$'s strategy f, by assumption a winning strategy for her in the game $\mathcal{G}@(\delta, s_0)$, also is winning for her in the game $\mathcal{G}_{T_\pi}@(\delta, s_0)$. To see why this is so, conclude from the similarities between $\mathcal{G}$ and $\mathcal{G}_{T_\pi}$ that f is well-defined and legitimate as a strategy in the latter game. Now consider an f-conform (full) match ρ of $\mathcal{G}_{T_\pi}@(\delta, s_0)$. In case ρ contains a position of the form (x, t), this must be the *final* position since x cannot be unfolded in $\mathcal{G}_{T_\pi}$. Then by definition of T_π we obtain $t \in T_\pi$. Thus seen as a $\mathcal{G}_{T_\pi}$-match, ρ is won by $\exists$. If on the other hand ρ does not contain *any* position of the form (x, t), given that $\mathcal{G}$ and $\mathcal{G}_{T_\pi}$ completely coincide as long as x does not come into the picture, it follows that ρ can also be seen as a full $\mathcal{G}$-match. And since ρ is conform the strategy f, its winner in $\mathcal{G}$ is $\exists$. But then, once more by the fact that ρ contains no x-positions, its winner in $\mathcal{G}_{T_\pi}$ must be $\exists$ too. In other words, we have prove that (δ, s_0) is a winning position for for $\exists$ in $\mathcal{G}_{T_\pi}$. Then by the inductive hypothesis, s_0 belongs to the set $\delta_x^{\mathbb{S}}(T_\pi)$, which amounts to saying that indeed T_π is a legitimate move for $\exists$ at s_0 in $\mathcal{U}$.

Second, we prove that $\exists$ can keep the mentioned condition concerning the shadow match for one more round of $\mathcal{U}$. This proof is in fact easy. Suppose that in $\mathcal{U}$, following $\exists$'s move T_π, $\forall$ picks an element $s_{k+1} \in T_\pi$, thus constituting a partial $\mathcal{U}$-match $\rho = \pi T_\pi s_{k+1}$. By definition of T_π, there is a partial g-conform match $\bar{\rho}$, extending $\bar{\pi}$, with $last(\bar{\rho})$ of the form (x, t), and which is minimal (in length) with respect to these two properties. It is then straightforward to verify that $\bar{\rho}$ has all the required properties. In particular, the fact that $\bar{\rho}$ is a *minimal* extension of $\bar{\pi}$ ending in a position (x, t) with $t \in T_\pi$, ensures that the successive positions of the form (δ, s) in ρ are exactly $(\delta, s_0), \ldots, (\delta, s_{k+1})$, as required.

Finally, in order to see why this strategy is winning for $\exists$, observe that it follows from the set-up that she never gets stuck, so we only have to worry about infinite matches. Let $s_0 s_1 s_2 \ldots$ be the sequence of state positions in such an infinite match of $\mathcal{U}$. In the associated $\mathcal{G}$-match, each state s_i corresponds to a position of the form (δ, s_i). This can only be the case if the variable x is unfolded infinitely often, and since the $\mathcal{G}$-match is conform $\exists$'s winning strategy g, this cannot happen if x is a μ-variable. But then x is a ν-variable, and so $\eta = \nu$ and hence the $\mathcal{U}$-match is won by $\exists$. QED

Convention 3.28 In the sequel we will use the Adequacy Theorem without further notice. Also, we will write $\mathbb{S}, s \Vdash \varphi$ in case $s \in [\![\varphi]\!]^{\mathbb{S}}$.

Notes

What we now call the Knaster-Tarski Theorem (Theorem 3.4) was first proved by Knaster [12] in the context of power set algebras, and subsequently generalized by Tarski [29] to the setting of complete lattices. The Bekič principle (Proposition 3.9) stems from an unpublished technical report.

As far as we know, the results in section 3.2 on the duality between the least and the greatest fixpoint of a monotone map on a complete Boolean algebra, are folklore. The characterization of least and greatest fixpoints in game-theoretic terms is fairly standard in the theory of (co-)inductive definitions, see for instance Aczel [1]. The equivalence of the algebraic and the game-theoretic semantics of the modal μ-calculus (here formulated as the Adequacy Theorem 3.27) was first established by Emerson & Jutla [9].

Exercises

Exercise 3.1 Prove Proposition 3.6: show that monotone maps on complete lattices are inductive.

Exercise 3.2 Prove Theorem 3.20.
(Hint: given complete lattices $\mathbb{C}$ and $\mathbb{D}$, and a monotone map $f : C \times D \to C$, show that the map $g : D \to C$ given by

$$g(d) := \mu x.f(x, d)$$

is monotone. Here $f(x, d)$ is the least fixpoint of the map $f_d : C \to C$ given by $f_d(c) = f(c, d)$.)

Exercise 3.3 Let $F : \wp(S) \to \wp(S)$ be some monotone map. A collection $\mathcal{D} \in \wp\wp(S)$ of subsets of S is *directed* if for every two sets $D_0, D_1 \in \mathcal{D}$, there is a set $D \in \mathcal{D}$ with $D_i \subseteq D$ for $i = 0, 1$. Call F *(Scott) continuous* if it preserves directed unions, that is, if $F(\bigcup \mathcal{D}) = \bigcup_{D \in \mathcal{D}} F(D)$ for every directed $\mathcal{D}$.
Prove the following:

(a) F is Scott continuous iff for all $X \subseteq S$: $F(X) = \bigcup\{F(Y) \mid Y \subseteq_\omega X\}$.
(Here $Y \subseteq_\omega X$ means that Y is a finite subset of X.)

(b) If F is Scott continuous then the closure ordinal of F is at most ω.

Exercise 3.4 Let $F : \wp(S) \to \wp(S)$ be a monotone operation, and let γ_F be its closure ordinal. Sharpen Corollary 3.7 by proving that the cardinality of γ_F is bounded by $|S|$ (rather than by $|\wp(S)|$).

Exercise 3.5 Describe the bisimilarity game as an unfolding game.

Exercise 3.6 Prove that the unfolding game of Definition 3.13 satisfies *memoryless determinacy*. That is, let $\mathcal{U}^{\mu}(F)$ be the least fixpoint unfolding game for some monotone map $F : \wp(S) \to \wp(S)$. Prove the existence of two *positional* strategies $f_{\exists} : S \to \wp(S)$ and $f_{\forall} : \wp(S) \to S$ such that for every position p of the game, either $f_{\exists}$ is a winning strategy for $\exists$ in $\mathcal{U}^{\mu}(F)@p$, or else $f_{\forall}$ is a winning strategy for $\forall$ in $\mathcal{U}^{\mu}(F)@p$.

Part III
Streams and Trees

4 Stream automata

As we already mentioned in the introduction in the theory of the modal μ-calculus and other fixpoint logics a fundamental role is played by automata. As we will see further on, these devices provide a very natural generalization to the notion of a formula. This chapter gives an introduction to the theory of automata operating on (potentially infinite) objects. Whereas in the next chapters we will meet various kinds of automata for classifying trees and general transition systems, here we confine our attention to the devices that operate on *streams* or infinite words, these being the simplest nontrivial examples of infinite behavior.

Convention 4.1 Throughout this chapter (and the next), we will be dealing with some finite *alphabets* C. Elements of C will be sometimes denoted as $c, d, c_0, c_1, \ldots$, but often it will be convenient to think of C as a set of *colors*. In this case we will denote the elements of C with lower case roman letters that are mnemonic of the most familiar corresponding color ('b' for *blue*, 'g' for *green*, etcetera).

Definition 4.2 Given an alphabet C, a *C-stream* is just an infinite C-sequence, that is, a map $\gamma : \omega \to C$ from the natural numbers to C (see Appendix A). C-streams will also be called *infinite words* or *ω-words* over C. Sets of C-streams are called *stream languages* or *ω-languages* over C. $\lhd$

Remark 4.3 This definition is consistent with the terminology we introduced in Chapter 1. There we defined a $\wp(\mathsf{P})$*-stream* or *stream model for* P to be a Kripke model of the form $\mathbb{S} = \langle \omega, V, \mathit{Succ} \rangle$, where *Succ* is the standard successor relation on the set ω of natural numbers, and $V : \mathsf{P} \to \wp(\omega)$ is a valuation. If we represent V coalgebraically as a map $\sigma_V : \omega \to \wp(\mathsf{P})$ (cf. Remark 1.3), then in the terminology of Definition 4.2, $\mathbb{S}$ is indeed a $\wp(\mathsf{P})$*-stream*. $\lhd$

4.1 Deterministic stream automata

We start with the standard definition.

Definition 4.4 Given an *alphabet* C, a *deterministic C-automaton* is a quadruple $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$, where A is a finite set, $a_I \in A$ is the *initial state* of $\mathbb{A}$, $\delta : A \times C \to A$ its *transition function*, and $Acc \subseteq A^\omega$ its *acceptance condition*. The pair $\langle A, \delta \rangle$ is called the *transition diagram* of $\mathbb{A}$. $\lhd$

Example 4.5 The transition diagram and initial state of a deterministic automaton can nicely be represented graphically, as in the picture below, where $C = \{b, r, g\}$:

$$\begin{tikzcd}
\Rightarrow a_0 \arrow[loop above, "{r,g}"] \arrow[rr, "b"] & & a_1 \arrow[loop above, "b"] \arrow[rr, bend left, "{r,g}"] & & a_2 \arrow[loop above, "{r,g}"] \arrow[ll, bend left, "b"]
\end{tikzcd}$$

◁

An automaton comes to life if we supply it with input, in the form of a stream over its alphabet: It will *process* this stream, as follows. Starting from the initial state a_I, the automaton will step by step pass through the stream, jumping from one state to another as prescribed by the transition function.

Example 4.6 Let $\mathbb{A}_0$ be any automaton with transition diagram and initial state as given above, and suppose that we give this device as input the stream $\alpha = brgbrgbrgbrgbrgb\cdots$. Then we find that $\mathbb{A}_0$ will make an infinite series of transitions, determined by α:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{b} a_1 \cdots$$

Thus the machine passes through an infinite sequence of states:

$$\rho = a_0a_1a_2a_2a_1a_2a_2a_1a_2a_2\ldots$$

This sequence is called the *run* of the automaton on the word α — a run of $\mathbb{A}$ is thus an A-stream.

For a second example, on the word $\alpha' = brbgbrgrgrgrgrgr\cdots$ the run of the automaton $\mathbb{A}_0$ looks as follows:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{b} a_1 \xrightarrow{g} a_2 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{r} a_2 \xrightarrow{g} \cdots$$

we see that from the sixth step onwards, the machine device remains circling in its state a_2: $\cdots\ a_2 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{r} \cdots$. ◁

Definition 4.7 Given a finite automaton $\mathbb{A} = \langle A, \delta, Acc, a_I\rangle$, we will write $a \xrightarrow{c} a'$ if $a' = \delta(a, c)$. We extend this inductively to the relation $\twoheadrightarrow \subseteq A \times C^* \times A$:

- if $w = \epsilon$ then $a \overset{\epsilon}{\twoheadrightarrow} a'$ iff $a = a'$,
- if $w = vc$ then $a \overset{wc}{\twoheadrightarrow} a'$ iff there is a a'' such that $a \overset{w}{\twoheadrightarrow} a''$ and $a'' \xrightarrow{c} a'$.

In words, $a \overset{w}{\twoheadrightarrow} a'$ if there is a w-labelled path from a to a'.

The *run* of $\mathbb{A}$ on a C-stream $\gamma = c_0c_1c_2\ldots$ is the infinite A-sequence

$$\rho = a_0a_1a_2\ldots$$

such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for every $i \in \omega$. ◁

Generally, whether or not an automaton *accepts* an infinite word, depends on the existence of a successful run — note that in the present deterministic setting, this run is unique. In order to determine which runs are successful, we need the acceptance condition.

Definition 4.8 A run $\rho \in A^\omega$ of an automaton $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$ is *successful* with respect to an acceptance condition Acc if $\rho \in Acc$.

A finite C-automaton $\mathbb{A} = \langle A, \delta, Acc, a_I \rangle$ *accepts* a C-stream γ if the run of $\mathbb{A}$ on γ is successful. The *ω-language* $L_\omega(\mathbb{A})$ associated with $\mathbb{A}$ is defined as the set of streams that are accepted by $\mathbb{A}$. Two automata are called *equivalent* if they accept the same streams. ◁

A natural requirement on the acceptance condition is that it only depends on a bounded amount of information about the run.

Remark 4.9 In the case of automata running on *finite words*, there is a very simple and natural acceptance criterion. The point is that runs on finite words are themselves finite too. For instance, suppose that in Example 4.6 we consider the run on the finite word *brgb*:

$$a_0 \xrightarrow{b} a_1 \xrightarrow{r} a_2 \xrightarrow{g} a_2 \xrightarrow{b} a_1.$$

Then this runs *ends* in the state a_1. In this context, a natural criterion for the acceptance of the word *abca* by the automaton is to make it dependent on the membership of this final state a_1 in a designated set $F \subseteq A$ of *accepting* states.

A structure of the form $\mathbb{A} = \langle A, \delta, F, a_I \rangle$ with $F \subseteq A$ may be called a *finite word automaton*, and we say that such a structure *accepts* a finite word w if the unique state a such that $a_I \overset{w}{\twoheadrightarrow} a$ belongs to F. ◁

4.2 Acceptance conditions

For runs on infinite words, a natural acceptance criterion would involve the collection of states that occur infinitely often in the run.

Definition 4.10 Given an infinite sequence α over some finite set A, let $Occ(\alpha)$ and $Inf(\alpha)$ denote the set of elements of A that occur in α at least once and infinitely often, respectively. ◁

Definition 4.11 Given a transition diagram $\langle A, \delta \rangle$, we define the following types of acceptance conditions:

- A *Muller* condition is given as a collection $\mathcal{M} \subseteq \wp(A)$ of subsets of A. The corresponding acceptance condition is defined as

$$Acc_{\mathcal{M}} := \{\alpha \in A^\omega \mid Inf(\alpha) \in \mathcal{M}\}.$$

- A *Büchi* condition is given as a subset $F \subseteq A$. The corresponding acceptance condition is defined as

$$Acc_F := \{\alpha \in A^\omega \mid \mathit{Inf}(\alpha) \cap F \neq \varnothing\}.$$

- A *parity condition* is given as a map $\Omega : A \to \omega$. The corresponding acceptance condition is defined as

$$Acc_\Omega := \{\alpha \in A^\omega \mid \max\{\Omega(a) \mid a \in \mathit{Inf}(\alpha)\} \text{ is even }\}.$$

Automata with these acceptance conditions are called *Muller*, *Büchi* and *parity automata*, respectively. $\lhd$

Of these three types of acceptance conditions, the Muller condition perhaps is the most natural. It exactly and directly specifies the subsets of A that are admissible as the set $\mathit{Inf}(\rho)$ of a successful run. The Büchi condition is also fairly intuitive: an automaton with Büchi condition F accepts a stream α if the run on α passes through some state in F infinitely often. This makes Büchi automata the natural analog of the automata that operate on *finite* words, see Remark 4.9.

The parity condition may be slightly more difficult to understand. The idea is to give each state a of $\mathbb{A}$ a weight $\Omega(a) \in \omega$. Then any infinite A-sequence $\alpha = a_0a_1a_2\dots$ induces an infinite sequence $\Omega(a_0)\Omega(a_1)\dots$ of natural numbers. Since the range of Ω is finite this means that there is a *largest* natural number N_α occurring infinitely often in this sequence, $N_\alpha = \max\{\Omega(a) \mid a \in \mathit{Inf}(\alpha)\}$. Now, a parity automaton accepts an infinite word iff the number N_ρ of the associated run ρ is *even*.

At first sight, this condition will seem rather contrived and artificial. Nevertheless, for a number of reasons the parity automaton is destined to play the leading role in these notes. Most importantly, the distinction between even and odd parities directly corresponds to that between least and greatest fixpoint operators, so that parity automata are the more direct automata-theoretic counterparts of fixpoint formulas. An additional theoretic motivation to use parity automata is that their associated acceptance games have some very nice game-theoretical properties, as we will see further on.

Let us now first discuss some examples of automata with these three acceptance conditions.

Example 4.12 Suppose that we supply the device of Example 4.5 with the Büchi acceptance condition $F_0 = \{a_1\}$. That is, the resulting automaton $\mathbb{A}_0$ accepts a stream α iff the run of $\mathbb{A}_0$ passes through the state a_1 infinitely often. For instance, $\mathbb{A}_0$ will accept the word $\alpha = brgbrgbrgbrgbrgbrgb\cdots$, because the run of $\mathbb{A}_0$ is the stream $a_0a_1a_2a_2a_1a_2a_2a_1a_2a_2\dots$ which indeed contains a_1 infinitely many times. On the other hand, as we saw already, the run of $\mathbb{A}_0$ on the stream $\alpha' = brbgbrgrgrgrgrgr\cdots$ loops in state a_2, and so α' will not be accepted.

In general, it is not hard to prove that $\mathbb{A}_0$ accepts a C-stream γ iff γ contains infinitely many b's. $\lhd$

Example 4.13 Consider the automaton $\mathbb{A}_1$ given by the following diagram and initial state:

$$\begin{tikzcd}
\Rightarrow a_0 \arrow[loop above, "{r,g}"] \arrow[r, "b"] & a_b \arrow[r, bend left, "g"] \arrow[d, "b"'] \arrow[dr, "r"] & a_g \arrow[loop above, "{r,g}"] \arrow[l, bend left, "b"] \\
 & a_f \arrow[loop below, "{b,r,g}"] & a_r \arrow[l, "b"'] \arrow[u, "g"'] \arrow[loop below, "r"]
\end{tikzcd}$$

As an example of a Muller acceptance condition, consider the set

$$\{\ \{a_0\}\ ,\ \{a_g\}\ ,\ \{a_b, a_g\}\ ,\ \{a_b, a_r, a_g\}\ \}$$

The resulting automaton accepts those infinite streams in which every b is followed by a finite number of r's, followed by a g. We leave the details of this proof as an exercise to the reader, confining ourselves to a brief description of the intuitive meaning of the states.

a_0 represents the situation where the automaton has not encountered any b's;

a_f is the 'faulty' state;

a_b is the state where the automaton has just processed a b; it now has to pass through a finite sequence of r's, eventually followed by a g;

a_r represents the situation where the automaton, after seeing a b, has processed a finite, non-empty, sequence of r's;

a_g is the state where the automaton, after passing the last b, has fulfilled its obligation to process a g.

$\lhd$

Example 4.14 For an example of a parity automaton, consider the transition diagram of Example 4.5, and suppose that we endow the set $\{a_0, a_1, a_2\}$ with the priority map Ω given by $\Omega(a_i) = i$. Given the shape of the transition diagram, it then follows more or less directly from the definitions that the resulting automaton accepts an infinite word over $C = \{b, r, g\}$ iff it either stays in a_0, or visits a_2 infinitely often. From this one may derive that $L_\omega(\mathbb{A})$ consists of those C-streams containing infinitely many r's or infinitely many g's (or both). $\triangleleft$

It is important to understand the relative strength of Muller, Büchi and parity automata when it comes to recognizing ω-languages. The Muller acceptance condition is the more fundamental one in the sense that the other two are easily represented by it.

Proposition 4.15 *There is an effective procedure transforming a deterministic Büchi stream automaton into an equivalent deterministic Muller stream automaton.*

Proof. Given a Büchi condition F on a set A, define the corresponding Muller condition $\mathcal{M}_F \subseteq \wp(A)$ as follows:

$$\mathcal{M}_F := \{B \subseteq A \mid B \cap F \neq \varnothing\}.$$

Clearly then, $Acc_{\mathcal{M}_F} = Acc_F$. It is now immediate that any Büchi automaton $\mathbb{A} = \langle A, \delta, F, a_I\rangle$ is equivalent to the Muller automaton $\langle A, \delta, \mathcal{M}_F, a_I\rangle$. QED

Proposition 4.16 *There is an effective procedure transforming a deterministic parity stream automaton into an equivalent deterministic Muller stream automaton.*

Proof. Analogous to the proof of the previous proposition, we put

$$\mathcal{M}_\Omega := \{B \subseteq A \mid \max(\Omega[B]) \text{ is even }\},$$

and leave it for the reader to verify that this is the key observation in turning a parity acceptance condition into a Muller one. QED

Interestingly enough, Muller automata can be simulated by devices with a parity condition.

Proposition 4.17 *There is an effective procedure transforming a deterministic Muller stream automaton into an equivalent deterministic parity stream automaton.*

Proof. Given a Muller automaton $\mathbb{A} = \langle A, \delta, \mathcal{M}, a_I\rangle$, define the corresponding parity automaton $\mathbb{A}' = \langle A', \delta', \Omega, a'_I\rangle$ as follows. The crucial concept used in this construction is that of *latest appearance records*. The following notation will be convenient: given a finite sequence in A^*, say, $\alpha = a_1 \dots a_n$, we let $\widetilde{\alpha}$ denote the set $\{a_1, \dots, a_n\}$, and $\alpha[\nabla/a]$ the sequence α with every occurrence of a being replaced with the symbol ∇.

To start with, the set A' of states is defined as the collection of those finite sequences over the set $A \cup \{\nabla\}$ in which every symbol occurs exactly once:

$$A' = \{a_1 \dots a_k \nabla a_{k+1} \dots a_m \mid A = \{a_1, \dots, a_m\}\}.$$

The intuition behind this definition is that a state in $\mathbb{A}'$ encodes information about the states of $\mathbb{A}$ that have been visited during the initial part of its run on some word. More specifically, the state $a_1 \dots a_k \nabla a_{k+1} \dots a_m$ encodes that the states visited by $\mathbb{A}$ are $a_{n+1}, \dots, a_m$ (for some $n \leq m$, not necessarily $n = k$), and that of these, a_m is the state visited most recently, a_{m-1} the one before that, etc. The symbol ∇ marks the *previous* position of a_m in the list.

For a proper understanding of $\mathbb{A}'$ we need to go into more detail. First, for the initial position of $\mathbb{A}'$, fix some enumeration $d_1, \dots, d_m$ of A with $a_I = d_m$, and define

$$a'_I \;:=\; d_1 \dots d_m \nabla.$$

For the transition function, consider a state $\alpha = a_1 \dots a_k \nabla a_{k+1} \dots a_m$ in A', and a color $c \in C$. To obtain the state $\delta'(\alpha, c)$, replace the occurrence of $\delta(a_m, c)$ in $a_1 \dots a_m$ with ∇, and make the state $\delta(a_m, c)$ itself the rightmost element of the resulting sequence. Thus the ∇ in the new sequence marks the latest appearance of the state $\delta(a_m, c)$. Formally, we put

$$\delta'(a_1 \dots a_k \nabla a_{k+1} \dots a_m, c) \;:=\; (a_1 \dots a_m)[\nabla/\delta(a_m, c)]\delta(a_m, c).$$

For an example, see 4.18 below.

Now consider the runs ρ and ρ' of $\mathbb{A}$ and $\mathbb{A}'$, respectively, on some C-stream γ. Let $P = \mathit{Inf}(\rho)$ denote the set of states of $\mathbb{A}$ that are visited infinitely often during ρ. From a certain moment on, ρ will *only* pass through states in P; let $\mathbb{A}$ continue its run until it has passed through each state in P at least one more time. It is not too hard to see that from that same moment on, ρ' will only pass through states of the form $a_1 \dots a_k \nabla a_{k+1} \dots a_m$ such that the states in P form a final segment $a_{l+1} \dots a_m$ of the sequence $a_1 \dots a_m$. Also, since ∇ marks the previous position of a_m, it must occur before one of the a_i with $l + 1 \leq i < m$. In other words, we have

$$\mathit{Inf}(\rho') \subseteq \{\alpha\nabla\beta \in A' \mid \widetilde{\beta} \subseteq P\}. \tag{18}$$

Furthermore, it is crucial to note that among the states $\alpha\nabla\beta = a_1 \dots a_k \nabla a_{k+1} \dots a_m$ in $\mathit{Inf}(\rho')$, the ones with the *longest tail* $\beta = a_{k+1} \dots a_m$ (i.e., with maximal $|\beta|$), are exactly the ones where $\mathit{Inf}(\rho)$ is *identical* to the set $\{a_{k+1}, \dots, a_m\} = \widetilde{\beta}$. This shows how we can encode the success of runs of $\mathbb{A}$ in a parity condition for $\mathbb{A}'$. Putting

$$\Omega(\alpha\nabla\beta) \;:=\; \begin{cases} 2 \cdot |\beta| + 1 & \text{if } \widetilde{\beta} \notin \mathcal{M}, \\ 2 \cdot |\beta| + 2 & \text{if } \widetilde{\beta} \in \mathcal{M}, \end{cases}$$

we ensure that for any word γ, we have the following equivalences:

$$\begin{aligned}
\mathbb{A} \text{ accepts } \gamma &\iff \mathit{Inf}(\rho) \in \mathcal{M} \\
&\iff \{\widetilde{\beta} \mid \alpha\nabla\beta \in \mathit{Inf}(\rho') \text{ with } \beta \text{ of maximal length } \} \in \mathcal{M} \\
&\iff \max\{\Omega(\alpha\nabla\beta) \mid \alpha\nabla\beta \in \mathit{Inf}(\rho')\} \text{ is even} \\
&\iff \mathbb{A}' \text{ accepts } \gamma.
\end{aligned}$$

This suffices to prove the equivalence of $\mathbb{A}$ and $\mathbb{A}'$. QED

Example 4.18 With $\mathbb{A}_1$ the Muller automaton of Example 4.13, here are some examples of the transition function δ' of its parity equivalent $\mathbb{A}'$:

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\delta'(a_b a_r a_g a_f a_0 \nabla, b) & := & \nabla a_r a_g a_f a_0 a_b & \delta'(\nabla a_r a_g a_f a_0 a_b, b) & := & a_r a_g \nabla a_0 a_b a_f \\
\delta'(a_b a_r a_g a_f a_0 \nabla, r) & := & a_b a_r a_g a_f \nabla a_0 & \delta'(\nabla a_r a_g a_f a_0 a_b, r) & := & \nabla a_g a_f a_0 a_b a_r \\
\delta'(a_b a_r a_g a_f a_0 \nabla, g) & := & a_b a_r a_g a_f \nabla a_0 & \delta'(\nabla a_r a_g a_f a_0 a_b, g) & := & a_r \nabla a_f a_0 a_b a_g
\end{array}$$

Likewise, a few examples of the priority map:

$$\begin{array}{rcl}
\Omega(a_b a_r a_g a_f \nabla a_0) & := & 4 \\
\Omega(a_g a_f a_0 a_b \nabla a_r) & := & 3 \\
\Omega(a_f a_r a_0 \nabla a_b a_g) & := & 6 \\
\Omega(a_f a_0 \nabla a_b a_r a_g) & := & 8
\end{array}$$

As the initial state of $\mathbb{A}'$, one could for instance take the sequence $a_r a_r a_g a_f a_0 \nabla$. $\lhd$

The following example shows that, in the case of deterministic stream automata, the recognizing power of Muller and parity automata is *strictly* stronger than that of Büchi automata.

Example 4.19 Consider the following language over the alphabet $C = \{b, r\}$:

$$L = \{\alpha \in C^\omega \mid r \notin \mathit{Inf}(\alpha)\}.$$

That is, L consists of those C-streams that contain at most finitely many red items (that is, the symbol r occurs at most finitely often). We will give both a Muller and a parity automaton to recognize this language, and then show that there is no Büchi automaton for L.

It is not difficult to see that there is a deterministic Muller automaton recognizing this language. Consider the automaton $\mathbb{A}_2$ given by the following diagram,

$$\begin{tikzcd}
{} \arrow[r, Rightarrow] & a_b \arrow[loop above, "b"] \arrow[r, bend left, "r"] & a_r \arrow[loop above, "r"] \arrow[l, bend left, "b"]
\end{tikzcd}$$

and Muller acceptance condition $\mathcal{M}_2 := \{\{a_b\}\}$. It is straightforward to verify that the run of $\mathbb{A}_2$ on an $\{b, r\}$-stream α keeps circling in a_b iff from a certain moment on, α only produces b's.

For a parity automaton recognizing L, endow the diagram above with the priority map Ω_2 given by $\Omega_2(a_b) = 0$, $\Omega_2(a_r) = 1$. With this definition, there can only be one set of states of which the maximum priority is even, namely, the singleton $\{a_b\}$. Hence, this parity acceptance condition is the same as the Muller condition $\{\{a_b\}\}$.

However, there is *no* deterministic *Büchi* automaton recognizing L. Suppose for contradiction that $L = L_\omega(\mathbb{A})$, where $\mathbb{A} = \langle A, \delta, F, a_I\rangle$ is some Büchi automaton. Since the stream $\alpha_0 = bbb\ldots$ belongs to L, it is accepted by $\mathbb{A}$. Hence in particular, the run ρ_0 of $\mathbb{A}$ on α_0 will pass some state $f_0 \in F$ after a finite number, say n_0, of steps.

Now consider the stream $\alpha_1 = b^{n_0}rbbb\ldots$. Since runs are uniquely determined, the initial n_0 steps of the run ρ_1 of $\mathbb{A}$ on α_1 are identical to the first n_0 steps of $\mathbb{A}$ on α_0. But since α_1 belongs to L too, it too is accepted by $\mathbb{A}$. Thus on input α_1, $\mathbb{A}$ will visit a state in F infinitely often. That is, we may certainly choose an $n_1 \geq 1$ such that ρ_1 passes some state $f_1 \in F$ after $n_0 + n_1$ steps. Now consider the stream $\alpha_2 = b^{n_0}rb^{n_1}rbbb\ldots$, and analyze the run ρ_2 of $\mathbb{A}$ on α_2. Continuing like this, we can find positive numbers $n_0, n_1, \ldots$ such that for every $k \in \omega$, the stream

$$\alpha_k = b^{n_0}rb^{n_1}\ldots rb^{n_k}rbbb\ldots \in L, \text{ for all } k. \tag{19}$$

Consider the stream

$$\alpha = (b^{n_0}r)(b^{n_1}r)\ldots(b^{n_k}r)\ldots$$

Containing infinitely many r's, α does not belong to L. Nevertheless, it follows from (19) that the run ρ of $\mathbb{A}$ on α passes through the states $f_0, f_1, \ldots$ as described above. Since F is finite, there is then at least one $f \in F$ appearing infinitely often in this sequence. Thus we have found an $f \in F$ that is passed infinitely often by ρ, showing that $\mathbb{A}$ accepts α. This gives the desired contradiction. $\lhd$

4.3 Nondeterministic automata

Nondeterministic automata generalize deterministic ones in that, given a state and a color, the next state is not *uniquely* determined, and in fact need not exist at all.

Definition 4.20 Given an *alphabet* C, a *nondeterministic C-automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$, where A is a finite set, $a_I \in A$ is the *initial state* of $\mathbb{A}$, $\Delta : A \times C \to \wp(A)$ its *transition function* of $\mathbb{A}$, and $Acc \subseteq A$ its *acceptance condition.* $\lhd$

As a consequence, the run of an nondeterministic automaton on a stream is no longer uniquely determined either.

Definition 4.21 Given a nondeterministic automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I \rangle$, we define the relations $\rightarrow \subseteq A \times C \times A$ and $\twoheadrightarrow \subseteq A \times C^* \times A$ in the obvious way: $a \xrightarrow{c} a'$ if $a' \in \Delta(a, c)$, $a \overset{\epsilon}{\twoheadrightarrow} a'$ if $a = a'$, and $a \overset{wc}{\twoheadrightarrow} a'$ if there is a a'' such that $a \overset{w}{\twoheadrightarrow} a'' \xrightarrow{c} a'$. A *run* of a nondeterministic automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I \rangle$ on an C-stream $\gamma = c_0c_1c_2 \dots$ is an infinite A-sequence

$$\rho = a_0a_1a_2 \dots$$

such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for every $i \in \omega$. $\lhd$

Now that runs are no longer unique, an automaton may have both successful and unsuccessful runs on a given stream. Consequently, there is a choice to make concerning the definition of the notion of acceptance.

Definition 4.22 A nondeterministic C-automaton $\mathbb{A} = \langle A, \Delta, Acc, a_I \rangle$ *accepts* a C-stream γ if there is a successful run of $\mathbb{A}$ on γ. $\lhd$

Further concepts, such as the language recognized by an automaton, the notion of equivalence of two automata, and the Büchi, Muller and parity acceptance conditions, are defined as for deterministic automata. Also, the transformations given in the Propositions 4.15, 4.16 and 4.17 are equivalence-preserving for nondeterministic automata just as for deterministic one. *Different* from the deterministic case, however, is that *nondeterministic* Büchi automata have the *same* accepting power as their Muller and parity variants.

Proposition 4.23 *There is an effective procedure transforming a nondeterministic Muller stream automaton into an equivalent nondeterministic Büchi stream automaton.*

Proof. Let $\mathbb{A} = \langle A, \Delta, \mathcal{M}, a_I \rangle$ be a nondeterministic Muller automaton. The idea underlying the definition of the Büchi equivalent $\mathbb{A}'$ is that $\mathbb{A}'$, while copying the behavior of $\mathbb{A}$, *guesses* the set $M = \mathit{Inf}(\rho)$ of a successful run of $\mathbb{A}$, and at a certain (nondeterministically chosen) moment confirms this choice by moving to a position of the form $(a, M, \varnothing)$. In order to make sure that not too many streams are accepted, the device has to keep track which of the states in M have been visited by $\mathbb{A}$, resetting this counter to the empty set every time when *all* M-states have been passed.

$$\begin{array}{rcl}
A' & := & A \cup \bigcup_{M \in \mathcal{M}} \{(a, M, P) \mid a \in M, P \subseteq M\}, \\
a'_I & := & a_I \\
\Delta'(a, c) & := & \Delta(a, c) \cup \bigcup_{M \in \mathcal{M}} \{(b, M, \varnothing) \mid b \in \Delta(a, c) \cap M\} \\
\Delta'((a, M, P), c) & := & \left\{ \begin{array}{ll} \{(b, M, P \cup \{a\}) \mid b \in \Delta(a, c) \cap M\} & \text{if } P \neq M, \\ \{(b, M, \{a\}) \mid b \in \Delta(a, c) \cap M\} & \text{if } P = M, \end{array} \right. \\
F & := & \{(a, M, P) \in A' \mid P = M\}.
\end{array}$$

We leave it as an exercise for the reader to verify that the resulting automaton is indeed equivalent to $\mathbb{A}$. QED

We now turn to the *determinization* problem for stream automata. In the case of automata operating on finite words, it is not difficult to prove that nondeterminism does not really add recognizing power: any nondeterministic finite automaton $\mathbb{A}$ may be 'determinized', that is, transformed into an equivalent deterministic automaton $\mathbb{A}^d$.

Remark 4.24 Finite word automata (see Example 4.9) can be determinized by a fairly simple *subset construction*.

Let $\mathbb{A} = \langle A, \Delta, F, a_I \rangle$ be a nondeterministic finite word automaton. A run of $\mathbb{A}$ on a finite word $w = c_1 \cdots c_n$ is defined as a finite sequence $a_0 a_1 \cdots a_n$ such that $a_0 = a_I$ and $a_i \xrightarrow{c_i} a_{i+1}$ for all $i < n$. $\mathbb{A}$ *accepts* a finite word w if there is a successful run, that is, a run $a_0 a_1 \cdots a_n$ ending in an accepting state a_n.

Given such a nondeterministic automaton, define a deterministic automaton $\mathbb{A}^+$ as follows. For the states of $\mathbb{A}^+$ we take the *macro-states* of $\mathbb{A}$, that is, the nonempty subsets of A. The deterministic transition function δ is given by

$$\delta(P, c) := \bigcup_{a \in P} \Delta(a, c).$$

In words, $\delta(P, c)$ consists of those states that can be reached from some state in P by making one a-step in $\mathbb{A}$. The accepting states of $\mathbb{A}^+$ are those macro-states that contain an accepting state from $\mathbb{A}$: $F^+ := \{P \in A^+ \mid P \cap F \neq \varnothing\}$, and its initial state is the singleton $\{a_I\}$.

In order to establish the equivalence of $\mathbb{A}$ and $\mathbb{A}^+$, we need to prove that for every word w, $\mathbb{A}$ has an accepting run on w iff the unique run of $\mathbb{A}^+$ on w is successful. The key claim in this proof is the following statement:

$$\{a_I\} \overset{w}{\twoheadrightarrow}_{\mathbb{A}^+} P \iff a_I \overset{w}{\twoheadrightarrow}_{\mathbb{A}} a \text{ for all } a \in P. \tag{20}$$

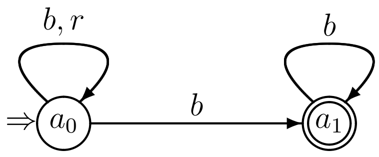
Intuitively, (20) states that $\{a_I\} \overset{w}{\twoheadrightarrow}_{\mathbb{A}^+} P$ iff P contains all the states that $\mathbb{A}$ can reach on input w. We leave the straightforward inductive proof of (20) as an exercise for the reader. $\lhd$

Unfortunately, the class of Büchi automata does not admit such a determinization procedure. As a consequence of Proposition 4.23 below, and witnessed by the Examples 4.19 and 4.25, the recognizing power of nondeterministic Büchi automata is strictly greater than that of their deterministic variants.

Example 4.25 For a nondeterministic Büchi automaton recognizing the language

$$L = \{\alpha \in C^\omega \mid r \notin \mathit{Inf}(\alpha)\}$$

of Example 4.19, consider the automaton given by the following picture:



In general, the Büchi acceptance condition $F \subseteq A$ of an automaton $\mathbb{A}$ is depicted by the set of states with *double circles.* So in this case, $F = \{a_1\}$. $\lhd$

There is positive news as well. A key result in automata theory states that when we turn to Muller and parity automata, nondeterminism does *not* increase recognizing power. This result follows from Proposition 4.23 and Theorem 4.26 below.

Theorem 4.26 *There is an effective procedure transforming a nondeterministic Büchi stream automaton into an equivalent deterministic Muller stream automaton.*

The *proof* of Theorem 4.26 will be given in the next section. As an important application we mention the following *Complementation Lemma.*

Proposition 4.27 *Let $\mathbb{A}$ be a nondeterministic Muller or parity automaton. Then there is an automaton $\overline{\mathbb{A}}$ of the same kind, such that $L_\omega(\overline{\mathbb{A}})$ is the complement of the language $L_\omega\mathbb{A}$.*

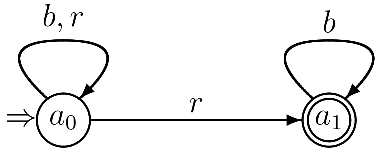
Leaving the proof of this proposition as an exercise for the reader, we finish this section with a summary of the relative power of the automata concept in the diagram below. Arrows indicate the reducibility of one concept to another, 'D' and 'ND' are short for 'deterministic' and 'nondeterministic', respectively.

$$\boxed{\begin{tikzcd}
\text{D Büchi} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{D Muller} \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \text{D parity} \arrow[d, Leftrightarrow] \\
\text{ND Büchi} \arrow[r, Leftrightarrow] & \text{ND Muller} \arrow[r, Leftrightarrow] & \text{ND parity}
\end{tikzcd}}$$

4.4 The Safra construction

This section is devoted to the proof of Theorem 4.26, which is based on a modification of the subset construction of Remark 4.24.

Remark 4.28 This modification has to be fairly substantial: Theorem 4.26 cannot be proved by a straightforward adaptation of the subset construction discussed in Remark 4.24. Consider the Büchi automaton $\mathbb{A}$ given by the following picture:



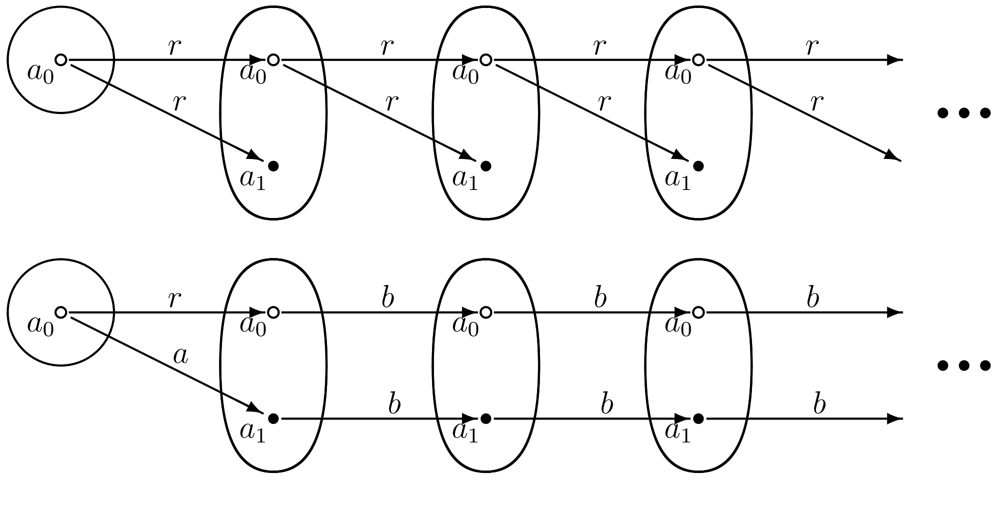
We leave it for the reader to verify that $L_\omega(\mathbb{A})$ consists of those streams of bs and rs that contain at least one and at most finitely many red items. In particular, the stream $r^\omega = rrrrr\ldots$ is rejected, while the stream $rb^\omega = rbbbb\ldots$ is accepted.

Now consider a deterministic automaton $\mathbb{A}^+$ of which the transition diagram is given by the subset construction. Then the run of the automaton $\mathbb{A}^+$ on r^ω is *identical* to its run on rb^ω:

$$a_0\{a_0, a_1\}\{a_0, a_1\}\{a_0, a_1\}\ldots$$

In other words, no matter which acceptance condition we give to $\mathbb{A}^+$, the automaton will accept either both r^ω and rb^ω, or neither. In either case $L_\omega(\mathbb{A}^+)$ will be different from $L_\omega(\mathbb{A})$.

As a matter of fact, it will be instructive to see in a bit more detail how the runs of $\mathbb{A}$ on r^ω and rb^ω, respectively, appear as 'traces' in the run of $L_\omega(\mathbb{A}^+)$ on these two streams:



Clearly, where the second run contains one single trace that corresponds to a successful run of the automaton $\mathbb{A}$, in the first run, all traces that reach a successful state are aborted immediately. These two pictures make clear that there are some some subtle but crucial distinctions that get lost if we do a straightforward subset construction. ◁

In Safra's modification of the subset construction, the states of the deterministic automaton can be seen as *finite sets of macro-states* that are ordered by the inclusion relation to form a certain kind of tree. The key idea underlying this modification is that at each step of the run, the accepting elements of each macro-state are given some special treatment. In the end this enables one to single out the runs with a sequence of macro-states containing a good trace (that is, an infinite sequence of states constituting an accepting run of the nondeterministic automaton). Formally, we define these 'tree-ordered finite sets of macro-states' as Safra trees.

Definition 4.29 An ordered tree is a structure $\langle S, r, \lhd, <_H \rangle$ such that $\langle S, \lhd \rangle$ is a tree with root r, and $<_H$ is a *sibling ordering relation*, that is, a partial order on S which totally orders the children of every node. Given two nodes s and t, we say that s is *to the left of* t if s and t have ancestors s' and t', respectively, such that $s' <_H t'$.

A *Safra tree* over a set B is a pair (S, L) where S is a finite ordered tree, and $L : S \to \wp^+(B)$ is a labelling such that $L(t)$ is a *proper* subset of $L(s)$ if t is a child of s, and $L(s) \cap L(t) = \varnothing$ if s and t are siblings (that is, have the same parent). $\lhd$

▶ `number of Safra trees`

It is not hard to see that for any Safra tree (S, L) and for every state $b \in B$, b belongs to some label set of the tree iff it belongs to the label of the root. And, if b belongs to the label of the root, then there is a *unique* node $s \in S$ such that $b \in L(s)$ but s has no child t with $b \in L(t)$. This node s is called *the lowest node of* b.

We now turn to the definition of the Safra construction.

Definition 4.30 Let $\mathbb{B}$ be a nondeterministic Büchi automaton $\mathbb{B} = \langle B, b_I, \Delta, F \rangle$. We will define a deterministic Muller automaton $\mathbb{B}^S = \langle B^S, a_I, \delta, \mathcal{M} \rangle$.

Assume that B has n states. The carrier B^S will consist of the collection of all *Safra trees* (S, L) over B with $S \subseteq \{0, 1, \dots, 2n-1\}$, that in addition have a map γ coloring nodes of the tree either white or green. The initial state of $\mathbb{B}^S$ will be the Safra tree consisting of a single white node 0 labelled with the singleton $\{b_I\}$.

For the transition function on B^S, take an arbitrary colored Safra tree (S, L, γ). On input $c \in C$, the deterministic transition function δ on B^S transforms (S, L, γ) into a new colored, labelled Safra tree, by performing the following sequence of actions:

1. Separate accepting states For any node s such that $L(s)$ contains accepting states, add a (canonically chosen) new node $s' \notin S$ to S as the youngest child of s, and label s' with the set $L(s) \cap F$. (Note that such an s' can always be found).

2. Make macro-move Apply the power set construction to the individual nodes: for each node s, replace its label $A \subseteq B$ with the set $\bigcup_{a \in A} \Delta(a, c)$.

3. Merge traces For each node s, remove those members from its label that already belong to the label of an older sibling of s (3a). After that, remove all nodes with empty labels (3b).

4. Mark successful nodes For every node s of which the label is *identical* to the the union of the labels of its children, remove all the descendants of s, and *mark* s by coloring it green. All other nodes are colored white.

For the Muller acceptance condition $\mathcal{M}$ of $\mathbb{B}^S$, put $M \in \mathcal{M}$ if there is some $s \in \{0, \dots, 2n-1\}$ such that s is a node of every tree in M, and s is colored green in some tree in M. $\lhd$

Example 4.31 ▶ `Example to be supplied`

$\lhd$

It is obvious from the construction that $\mathbb{B}^S$ is a deterministic automaton, so all that is left for the proof of Theorem 4.26 is to establish the equivalence of $\mathbb{B}$ and $\mathbb{B}^S$.

Proposition 4.32 *Let $\mathbb{B}$ be a nondeterministic Büchi automaton. Then*

$$L_\omega(\mathbb{B}) = L_\omega(\mathbb{B}^S).$$

Proof. For the inclusion $\subseteq$, suppose that there is a successful run $\rho = b_0 b_1 \dots$ of $\mathbb{B}$ on some C-stream $\gamma = c_0 c_1 \dots$. Consider the (unique) run $\sigma = (S_0, L_0, \theta_0)(S_1, L_1, \theta_1) \dots$ of $\mathbb{B}^S$ on γ. Here each (S_i, L_i, γ_i) is a Safra tree with labeling L and coloring θ_i. We claim that there is a object s which after some initial phase belongs to each Safra tree of σ, and which is marked green infinitely often.

To see why this must be the case, first note that at every stage i, the state b_i of ρ belongs to the label $L_i(r_i)$ of the root r_i of the Safra tree S_i. It follows that the root is always nonempty, and hence never removed; with $r := r_0$ we have $r_i = r$ for all $i > 0$. Now if r is colored green infinitely often, we are done.

So suppose that this is not the case. In other words, after a certain moment i, r will no longer be marked; consider the first time $j > i$ for which b_j is an accepting state (such a j must exist since ρ is by assumption an accepting run). According to the definition of δ, being an accepting state, in the next stage $j+1$, first b_j is put in the label set of one of the children of r, and so after step 2 of stage $j+1$, the next state b_{j+1} of ρ belongs to one of the children of the root. In subsequent steps of this stage, and in subsequent stages of the run σ, the contemporary state of ρ can be moved to an older sibling (step 3a). Such a shift merge into an older sibling can only happen finitely often, so there is some object s such that after some stage, s remains in every Safra tree of σ as a child of r, and its label contains the contemporary state of ρ.

We can now repeat the argument with s taking the role of r: either s itself is marked infinitely often, or else the state of ρ is eventually placed at the next level. Since the depth of the Safra trees involved is bounded, there must be some node s which after some initial phase belongs to each Safra tree in σ, and which is marked infinitely often.

For the opposite inclusion $\supseteq$, suppose that the (unique) run $\sigma = (S_0, L_0, \theta_0)(S_1, L_1, \theta_1)\ldots$ of $\mathbb{B}^S$ on γ is successful. Then by definition there is some node $s \in \{0, \ldots, 2n-1\}$ which after some initial phase will belong to each Safra tree in σ and which will subsequently be marked infinitely often, say at stages $k_1 < k_2 < \cdots$. For each $i > 0$, let A_i denote the macro-state of s at stage k_i, that is: $A_i := L_{k_i}(s)$.

Recall that γ is the infinite input stream $c_0c_1c_1\cdots$. Let $\gamma[p, q)$ denote the finite word $c_p \cdots c_{q-1}$. Since our construction is a refinement of the standard subset construction of Remark 4.24, it easily follows from the definitions of δ, that for every state $a \in A_1$ there is a $\gamma[0, k_1)$-labeled path from b_I to a, or briefly:

$$\text{for all } a \in A_1 \text{ we have } b_I \overset{\gamma[0,k_0)}{\twoheadrightarrow} a. \tag{21}$$

With a little more effort, crucially involving the conditions for marking nodes, and the rules governing the creation and maintenance of nodes, one may prove that for

$$\text{for all } i > 0 \text{ and for all } a \in A_{i+1} \text{ there is a } a' \in A_i \text{ such that } a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a. \tag{22}$$

Here $a' \overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F} a$ means that there is a $\gamma[k_i, k_{i+1})$-labelled path from a' to a which passes through some state in F. Details of this proof are left as an exercise to the reader.

The remainder of the proof consists of showing how to find a successful run of $\mathbb{B}$ on γ as the concatenation of a run segment given by (21) and infinitely many run segments given by (22). For this we use König's Lemma.

Defining $A_0 := \{b_I\}$, construct a tree whose nodes are all pairs of the form (a, i) with $a \in A_i$. As the parent of a node $(a, i+1)$ we pick one of the pairs (a', i) given by (21) and (22), respectively. Obviously this is a well-formed, infinite, finitely branching tree. So by König's Lemma, there is an infinite branch $(a_0, 0)(a_1, 1)\ldots$. By construction, we have $a_0 = b_I$, while for each $i \geq 0$ there is an $\overset{\gamma[k_i,k_{i+1})}{\twoheadrightarrow_F}$-labelled path in $\mathbb{B}$ from a_i to a_{i+1} which passes through some accepting state of $\mathbb{B}$. The infinite concatenation of these paths gives a run of $\mathbb{B}$ which visits infinitely often an accepting state of $\mathbb{B}$, and hence by finiteness of B, it visits some state of $\mathbb{B}$ infinitely often. Clearly then this run is accepting. QED

4.5 A coalgebraic perspective

In this section we introduce a coalgebraic perspective on stream automata. We have two reasons for doing so. First, we hope that this coalgebraic presentation will facilitate the introduction, further on, of automata operating on different kinds of structures. And second, we also believe that the coalgebraic perspective, in which the similarities between automata and the objects they classify comes out more clearly, makes it easier to understand some of the fundamental concepts and results in the area.

In this context, it makes sense to consider a slightly wider class than streams only.

Definition 4.33 A *C-flow* is a pair $\mathbb{S} = \langle S, \sigma\rangle$ with $\sigma : S \to C \times S$. Often we will write $\sigma(s) = (\sigma_C(s), \sigma_0(s))$. If we single out an (initial) state $s_0 \in S$ in such a structure, we obtain a *pointed C-flow* $(\mathbb{S}, s_0)$. $\lhd$

Example 4.34 Streams over an alphabet C can be seen as pointed C-flows: simply identify the word $\gamma = c_0c_1c_2\ldots$ with the pair $(\langle\omega, \lambda n.(c_n, n+1)\rangle, 0)$. Conversely, with any pointed flow $\langle\mathbb{S}, s\rangle$ we may associate a unique stream $\gamma_{\mathbb{S},s}$ by inductively defining $s_0 := s$, $s_{i+1} := \sigma_0(s_i)$, and putting $\gamma_{\mathbb{S}}(n) := \sigma_C(s_n)$. $\lhd$

It will be instructive to define the following notion of equivalence between flows. As its name already indicates, we are dealing with the analog of the notion of a bisimulation between two Kripke models. Since flows, having a deterministic transition structure, are less complex objects than Kripke models, the notion of bisimulation is also, and correspondingly, simpler.

Definition 4.35 Let $\mathbb{S}$ and $\mathbb{S}'$ be two C-flows. Then a nonempty relation $Z \subseteq S \times S'$ is a *bisimulation* if the following holds, for every $(s, s') \in Z$:

(color) $\sigma_C(s) = \sigma'_C(s')$;

(successor) $(\sigma_0(s), \sigma'_0(s')) \in Z$.

Two pointed flows $(\mathbb{S}, s)$ and $(\mathbb{S}', s')$ are called *bisimilar*, notation: $\mathbb{S}, s \leftrightarrows \mathbb{S}', s'$ if there is some bisimulation Z linking s to s'. In case the flows $\mathbb{S}$ and $\mathbb{S}'$ are implicitly understood, we may drop reference to them and simply call s and s' bisimilar. $\lhd$

As an example, it is not hard to see that any pointed flow $(\mathbb{S}, s)$ is bisimilar to the stream $\gamma_{\mathbb{S},s}$ that we may associate with it (see Example 4.34). Restricted to the class of streams, bisimilarity means *identity*.

Definition 4.36 A stream is called *regular* if it is bisimilar to a finite pointed flow. $\lhd$

Associated is a new perspective on nondeterministic stream automata which makes them very much *resemble* these flows. Roughly speaking the idea is this. Think of establishing a bisimulation between two pointed flows in terms of one structure $\langle A, a_I, \alpha\rangle$ *classifying* the other, $\langle S, s_C, \sigma\rangle$.

Now on the one hand make a restriction in the sense that the classifying flow must be finite, but on the other hand, instead of demanding its transition function to be of the form $\alpha : A \to C \times A$, allow objects $\alpha(a)$ to be *sets* of pairs in $C \times A$, rather than single pairs. That is, introduce *non-determinism* by letting the transition map Δ of $\mathbb{A}$ be of the form

$$\Delta : A \to \wp(C \times A). \tag{23}$$

Remark 4.37 This presentation (23) of nondeterminism is completely *equivalent* to the one given earlier. The point is that there is a natural bijection between maps of the above kind, and the ones given in Definition 4.20 as the transition structure of nondeterministic automata:

$$A \to \wp(C \times A) \;\cong\; (A \times C) \to \wp(A). \tag{24}$$

To see why this is so, an easy proof suffices. Using the principle of currying we can show that

$$A \to ((C \times A) \to 2) \cong (A \times C \times A) \to 2 \cong (A \times C) \to (A \to 2),$$

where the first and last set can be identified with respectively the left and right hand side of (24) using the bijection between subsets and their characteristic functions.

Concretely, we may identify a map $\Delta : (A \times C) \to \wp(A)$ with the map $\Delta' : A \to \wp(C \times A)$ given by

$$\Delta'(a) := \{(c, a') \mid a' \in \Delta(a, c)\}. \tag{25}$$

$\lhd$

Thus we arrive at the following reformulation of the definition of nondeterministic automata. Note that with this definition, a stream automaton can be seen as a kind of 'multi-stream' in the sense that every state harbours a *set* of potential 'local realizations' as a flow. Apart from this, an obvious difference with flows is that stream automata also have an acceptance condition.

Definition 4.38 A *nondeterministic C-stream automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, a_I \rangle$ such that $\Delta : A \to \wp(C \times A)$ is the *transition function*, $Acc \subseteq A^\omega$ is the *acceptance condition*, and $a_I \in A$ is the *initial state* of the automaton. $\lhd$

Finally, it makes sense to formulate the notion of an automaton *accepting* a flow in terms that are related to that of establishing the existence of a bisimulation. The nondeterminism can nicely be captured in game-theoretic terms — note however, that here we are dealing with a single player only.

In fact, bisimilarity between two pointed flows can itself be captured game-theoretically, using a trivialized version of the bisimilarity game for Kripke models of Definition 1.24. Consider two flows $\mathbb{A}$ and $\mathbb{S}$. Then the *bisimulation game* $\mathcal{B}(\mathbb{A}, \mathbb{S})$ between $\mathbb{A}$ and $\mathbb{S}$ is defined as a board game with positions of the form $(a, s) \in A \times S$, all belonging to $\exists$. At position (a, s), if a and s have a different color, $\exists$ loses immediately; if on the other hand $\alpha_C(a) = \sigma_C(s)$, then as the next position of the match she 'chooses' the pair consisting of the successors of a and s, respectively. These rules can concisely be formulated as in the following Table:

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\alpha_0(a), \sigma_0(s)) \mid \alpha_C(a) = \sigma_C(s)\}$

Finally, the winning conditions of the game specify that $\exists$ wins all infinite games. We leave it for the reader to verify that a pair $(a, s) \in A \times S$ is a winning position for $\exists$ iff a and s are bisimilar.

In order to proceed, however, we need to make a slight modification. We add positions of the form $(\alpha, s) \in (C \times A) \times S$, and insert an 'automatic' move immediately after a basic position, resulting in the following Table.

Position	Player	Admissible moves
$(a, s) \in A \times S$	-	$\{(\alpha(a), s)\}$
$(\alpha, s) \in (C \times A) \times S$	$\exists$	$\{(\alpha_0, \sigma_0(s)) \mid \alpha_C = \sigma_C(s)\}$

The acceptance game of a nondeterministic automaton $\mathbb{A}$ and a flow $\mathbb{S}$ can now be formulated as a natural generalization of this game.

Definition 4.39 Given a nondeterministic C-stream automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ and a pointed flow $\mathbb{S} = \langle S, s_0, \sigma\rangle$, we now define the *acceptance game* $\mathcal{A}(\mathbb{A}, \mathbb{S})$ as the following board game.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\alpha, s) \in (C \times A) \times S \mid \alpha \in \Delta(a)\}$
$(\alpha, s) \in (C \times A) \times S$	$\exists$	$\{(\alpha_0, \sigma_0(s)) \mid \alpha_C = \sigma_C(s)\}$

Table 6: Acceptance game for nondeterministic stream automata

Its positions and rules are given in Table 6, whereas the winning conditions of infinite matches are specified as follows. Given an infinite match of this game, first select the sequence

$$(a_0, s_0)(a_1, s_1)(a_2, s_2)\dots$$

of *basic positions*, that is, the positions reached during play that are of the form $(a, s) \in A \times S$. Then the match is winning for $\exists$ if the 'A-projection' $a_0 a_1 a_2 \dots$ of this sequence belongs to Acc. $\lhd$

Definition 4.40 A nondeterministic C-stream automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ *accepts* a pointed flow $\mathbb{S} = \langle S, s_0, \sigma\rangle$ if the pair (a_I, s_0) is a winning position for $\exists$ in the game $\mathcal{A}(\mathbb{A}, \mathbb{S})$. $\lhd$

The following proposition states that the two ways of looking at nondeterministic automata are equivalent.

Proposition 4.41 *Let $\mathbb{A} = \langle A, a_I, \Delta, Acc \rangle$, with $\Delta : (A \times C) \to \wp(A)$ be a nondeterministic C-automaton, and let $\mathbb{A}'$ be the nondeterministic C-stream automaton $\langle A, a_I, \Delta', Acc \rangle$, where $\Delta' : A \to \wp(C \times A)$ is given by (25). Then $\mathbb{A}$ and $\mathbb{A}'$ are equivalent.*

In the sequel we will *identify* the two kinds of nondeterministic automata, speaking of the *coalgebraic presentation* $\langle A, a_I, \Delta' : A \to \wp(C \times A), Acc \rangle$ of an automaton $\langle A, a_I, \Delta : (A \times C) \to \wp(A), Acc \rangle$.

Notes

The idea to use finite automata for the classification of infinite words originates with Büchi. In [4] he used stream automata with (what we now call) a Büchi acceptance condition to prove the decidability of the second-order theory of the natural numbers (with the successor relation). In the subsequent development of the theory of stream automata, other acceptance conditions were introduced. The Muller condition is named after the author of [19]. The invention of the parity condition, which can be seen as a refinement of the Rabin condition, is usually attributed to Emerson & Jutla [9], Mostowski [18], and/or Wagner.

The first construction of a deterministic equivalent to a nondeterministic Muller automaton was given by McNaughton [16]. The construction we presented in section 4.4 is due to Safra [28]. Finally, the coalgebraic perspective on stream automata presented in the final section of this chapter is the author's.

Exercises

Exercise 4.1 Provide Büchi automata recognizing exactly the following stream languages:

(a) $L_a = \{\alpha \in \{a, b, c\}^\omega \mid a \text{ and } b \text{ occur infinitely often in } \alpha\}$

(b) $L_b = \{\alpha \in \{a, b, c\}^\omega \mid \text{any } a \text{ in } \alpha \text{ is eventually followed by a } b\}$

(c) $L_c = \{\alpha \in \{a, b\}^\omega \mid \text{between any two } a\text{'s is an even number of } b\text{'s}\}$

(d) $L_d = \{\alpha \in \{a, b, c\}^\omega \mid ab \text{ and } cc \text{ occur infinitely often in } \alpha\}$

Exercise 4.2 Let C be a finite set. A C-stream language $L \subseteq C^\omega$ is called *ω-regular* if there exists a parity C-stream automaton $\mathbb{A} = (A, \Delta, \Omega, a_I)$ such that $L = L_\omega(\mathbb{A})$. Show that the class of ω-regular languages is closed under the Boolean operations, i.e., show that

(a) If $L \subseteq C^\omega$ is ω-regular then its complement $\overline{L} := \{\gamma \in C^\omega \mid \gamma \notin L\}$ is ω-regular.

(b) If L_1 and L_2 are ω-regular C-stream languages, then $L_1 \cup L_2$ is ω-regular.

(c) If L_1 and L_2 are ω-regular C-stream languages, then $L_1 \cap L_2$ is ω-regular.

Exercise 4.3 Show the following, for any deterministic Büchi automaton $\mathbb{A}$:

$$L_\omega(\mathbb{A}) = \{\alpha \in \Sigma^\omega \mid \text{infinitely many prefixes of } \alpha \text{ belong to } L(\mathbb{A})\}.$$

Exercise 4.4 Let C and D be finite sets and let $f : C \to D$ be a function. The function f can be extended to a function $\overline{f} : C^\omega \to D^\omega$ in the obvious way by putting $\overline{f}(\gamma) := f(c_0)f(c_1)f(c_2)\ldots \in D^\omega$ for any C-stream $\gamma \in C^\omega$. For a given C-stream language $L \subseteq C^\omega$ we define

$$\overline{f}(L) := \{f(\gamma) \mid \gamma \in L\} \subseteq D^\omega.$$

(a) Show that $L \subseteq C^\omega$ is ω-regular implies $f(L) \subseteq D^\omega$ is ω-regular.

(b) Show that there is a C-stream language $L \subseteq C^\omega$ such that $L = L_\omega(\mathbb{A})$ for some *deterministic* Büchi automaton $\mathbb{A}$ and such that $f(L) \subseteq D^\omega$ is not recognizable by any deterministic Büchi automaton.

Exercise 4.5 Show that nondeterministic Büchi automata have the same recognizing power as their Muller variants by showing that the automata $\mathbb{A}'$ and $\mathbb{A}$ in the proof of Proposition 4.23 are indeed equivalent.

Exercise 4.6 Consider the language L_d of exercise 4.1.

(a) Give a clear description of the complement $\overline{L_d}$ of L_d.

(b) Give a nondeterministic Büchi automaton recognizing exactly the language $\overline{L_d}$.

(c) Prove that there is no deterministic Büchi automaton recognizing the language $\overline{L_d}$. (Hint: use the theorem from Exercise 4.3.)

Exercise 4.7 Provide deterministic Muller automata recognizing the following languages:

(a) L_d of exercise 4.1.

(b) $L_a = \{\alpha \in \{a, b, c\}^\omega \mid \text{ between every pair of } a\text{'s is an occurrence of } bb \text{ or } cc\ \}$.

Exercise 4.8 Describe the languages that are recognized by the following Muller automata (presented in tabular form, with $\Rightarrow$ indicating the initial state):

(a)

$\mathbb{A}$	a	b
$\Rightarrow$ q_0	q_1	q_2
q_1	q_0	q_2
q_2	q_1	q_0

with $\mathcal{F} := \{\{q_0, q_1\}, \{q_0, q_2\}\}$.

(b) The same automaton as in (a) but with $\mathcal{F} := \{\{q_1, q_2\}, \{q_0, q_1, q_2\}\}$.

(c)

$\mathbb{A}$	a	b	c
$\Rightarrow$ q_0	q_1	q_0	q_0
q_1	q_0	q_2	q_0
q_2	q_0	q_0	q_3
q_3	q_0	q_0	q_0

with $\mathcal{F} := \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_1, q_2\}\}$.

5 Tree automata

In this chapter we consider a second classic type of automata, namely, those operating on infinite trees, of which the nodes are labeled by some finite alphabet C (to be fixed for the remainder of this section).

For simplicity we will restrict to the binary case, i.e., trees where every node has exactly two successors. Recall from Definition 1.12 that 2^* denotes the set of finite strings of 0s and 1s, that ϵ denotes the empty string, and that the left- and right successor of a node s are denoted by $s0$ and $s1$, respectively.

Definition 5.1 Given an alphabet C, a *binary C-labelled tree* or *binary C-tree* is a map $\tau : 2^* \to C$. $\triangleleft$

However, just as in the case of streams, we will take a coalgebraic approach, considering a wider class of structures than trees only.

Definition 5.2 Given an alphabet C, for any set S we denote the set $C \times S \times S$ as $\mathsf{B}_C S$. A *binary C-flow* or *C-biflow* is a structure $\mathbb{S} = \langle S, \sigma : S \to \mathsf{B}_C S \rangle$. We often write $\sigma(s) = (\sigma_C(s), \sigma_0(s), \sigma_1(s))$, where $\sigma_C(s)$, $\sigma_0(s)$, and $\sigma_1(s)$, denote the *color*, the *left* successor, and the *right* successor of s, respectively. A *pointed* biflow is a pair $(\mathbb{S}, s)$ with $\mathbb{S}$ a C-biflow, and s some designated point in $\mathbb{S}$. $\triangleleft$

It should be clear that, indeed, binary C-trees are examples of such pointed biflows — we standardly take the root as the designated point. Observe too that Definition 5.2 is consistent with the terminology we introduced in Chapter 1.

Definition 5.3 A *(C)-biflow language* is a class of pointed C-biflows, and a *$(C$-$)$tree language* is a set(!) of binary (C-)trees. $\triangleleft$

5.1 Nondeterministic tree automata

In this section we introduce nondeterministic tree automata, and we discuss, in game-theoretic terms, the associated notion of acceptance. Right from the start we take a coalgebraic perspective on these automata, and so we immediately define how these automata operate on arbitrary binary C-flows, rather than on binary C-trees only.

Automata and their acceptance games

Definition 5.4 A *nondeterministic tree automaton* is a structure $\mathbb{A} = \langle A, a_I, \Delta, Acc \rangle$, where A is a finite set, $a_I \in A$ is the initial state of $\mathbb{A}$, $\Delta : A \to \wp(\mathsf{B}_C A)$ its transition function, and $Acc \subseteq A^\omega$ its acceptance condition. Such an automaton is called a *Büchi*, *Muller*, or *parity* automaton, respectively, if the acceptance condition is expressed in the corresponding format. $\triangleleft$

Remark 5.5 The presentation in Definition 5.4 is slightly nonstandard, in the sense that the transition map of a nondeterministic tree automaton is usually given as a map $\Delta : A \times C \to \wp(A \times A)$. However, similar to the case of stream automata (see Remark 4.37), one may use the natural bijection

$$A \to \wp(C \times A \times A) \cong A \times C \to \wp(A \times A) \tag{26}$$

to show that our presentation is equivalent to the standard one. This bijection associates, with a map $\Delta : A \to \wp(C \times A \times A)$ the function $\Delta' : A \times C \to \wp(A \times A)$ given by $\Delta'(a, c) := \{(a_0, a_1) \in A \times A \mid (c, a_0, a_1) \in \Delta(a)\}$.

Our motivation from this minor deviation for the standard format is that in our approach, automata closely resemble the structures that they are supposed to classify: pointed biflows. Whereas the transition structure of such a biflow is a map of type $S \to \mathsf{B}_C(S)$, that of an automaton is a nondeterministic version of this, namely a map $A \to \wp(\mathsf{B}_C A)$. $\lhd$

Nondeterministic tree automata operate on pointed C-biflows. The acceptance criterion is formulated in terms of a so-called *acceptance game* associated with an automaton $\mathbb{A}$ and a structure $\mathbb{S}$. Matches of this game proceed in rounds, which start and finish with a *basic* position, that is, a position of the form $(a, s) \in A \times S$:

- At a basic position (a, s), $\exists$ chooses an element $(c, a_1, a_2) \in \Delta(a)$; the new position is $((c, a_1, a_2), s)$.
- At position $((c, a_1, a_2), s)$, $\exists$ immediately loses if $c \neq \sigma_C(s)$; otherwise, the next position is the set $\{(a_0, \sigma_0(s)), (a_1, \sigma_1(s))\}$;
- At position $\{(a_0, s_0), (a_1, s_1)\}$, $\forall$ chooses an element (a_i, s_i), which is the new basic position.

Observe that, in each round of the game, it is $\forall$ who chooses the *direction* to take in the flow: left or right. He can thus effectively determine the *path* through the flow that is taken during the match. This explains the name 'Pathfinder', that one may often find for $\forall$ in the literature.

Concerning the winning conditions of this game, the clause concerning finite matches is the same as always: any player that gets stuck loses the match immediately. In order to determine the winner of an infinite match, we look at the associated sequence of *basic positions*, i.e., positions of the form $(a, s) \in A \times S$. Given such a sequence

$$\rho = (a_0, s_0)(a_1, s_1)(a_2, s_2)\ldots,$$

we consider the 'projection' of the sequence on A:

$$\pi_A(\rho) := a_0 a_1 a_2 \ldots$$

We declare $\exists$ as the winner of the game if this sequence meets the acceptance condition, that is, if $\pi_A(\rho) \in Acc$; but if $\pi_A(\rho) \notin Acc$, the infinite match is won by $\forall$.

Definition 5.6 Let $\mathbb{A} = \langle A, a_I, \Delta, Acc \rangle$ be a nondeterministic tree automaton. For any binary C-flow $\mathbb{S}$, we define the *acceptance game* $\mathcal{A}(\mathbb{A}, \mathbb{S})$ as the two-player board game given by the rules of Table 7, together with the following winning condition: To determine the winner of an infinite match, consider the sequence $(a_0, s_0)(a_1, s_1)\ldots$ of basic positions in the match; $\exists$ is the winnner of the match if the sequence $a_0 a_1 \ldots$ belongs to the set Acc, and $\forall$ if it does not.

Given a point $s \in S$, we say that $\mathbb{A}$ *accepts* the pointed structure $(\mathbb{S}, s)$ if $\exists$ has a winning strategy in the game $\mathcal{A}(\mathbb{A}, \mathbb{S})@(a_I, s)$, that is, the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ *initialized* at position (a_I, s).

Given an automaton $\mathbb{A}$, we let $L(\mathbb{A})$ denote the biflow language *recognized* by $\mathbb{A}$, that is, the class of C-biflows that are accepted by $\mathbb{A}$. Likewise, $L_t(\mathbb{A})$ denotes the tree language recognized by $\mathbb{A}$, i.e., $L_t(\mathbb{A})$ is the set of binary C-trees in $L(\mathbb{A})$. $\lhd$

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\alpha, s) \mid \alpha \in \Delta(a)\}$
$((c, a_0, a_1), s) \in \mathsf{B}_C A \times S$	$\exists$	$\{\{(a_0, \sigma_0(s)), (a_1, \sigma_1(s))\} \mid c = \sigma_C(s)\}$
$\{(a_0, s_0), (a_1, s_1)\} \subseteq A \times S$	$\forall$	$\{(a_0, s_0), (a_1, s_1)\}$

Table 7: Acceptance game for nondeterministic tree automaton

Examples

Example 5.7 Let K_1 be the class of pointed biflows over the language $C = \{g, y\}$ ('green' and 'yellow') in which there is a (completely) green path starting from the designated point. We claim that K is recognized by the following Muller automaton $\mathbb{A}_1$. $\mathbb{A}_1$ has two states, a_g (the initial state) and $a_\top$. The transition structure is given by the map Δ:

$$\begin{array}{lcl} \Delta(a_g) & := & \{(g, a_g, a_\top), (g, a_\top, a_g)\}, \\ \Delta(a_\top) & := & \{(g, a_\top, a_\top), (y, a_\top, a_\top)\}, \end{array}$$

and the Muller acceptance is simply the set $\mathcal{M} := \wp(A_0)$. That is, all that $\exists$ has to do in order to win is to stay alive. (Alternatively, given this particular Δ, we could also define $\mathcal{M} := \big\{\{a_g\}, \{a_\top\}\big\}$.) Roughly, when operating on a pointed biflow $(\mathbb{S}, s)$, $\mathbb{A}_1$ 'guesses' a path starting at s, and checks whether this path contains green nodes only. For some more detailed intuitions, think of $a_\top$ as the 'surely successful state': Once $\exists$ has managed to reach $a_\top$ (i.e., a position of the form $(a_\top, s)$), she can no longer loose the match. Hence, $\forall$ will avoid $a_\top$ as much as he can: When faced with a choice between two positions (a_g, s_0) and $(a_\top, s_1)$, he will always choose the first.

Think of a_g as the state that marks the path through the flow. To see this, consider a position (a_g, s) in the acceptance game. If s is a yellow state, then $\exists$ looses immediately, so suppose otherwise. If she picks the element $(g, a_g, a_\top)$ from $\Delta(a_g)$, then $\forall$ has to choose between the positions $(a_g, \sigma_0(s))$ ('go left') and $(a_\top, \sigma_1(s))$ ('go right'), but as we saw, he will avoid $a_\top$ and always go left. Likewise, if $\exists$ picks the other element, $(g, a_\top, a_g)$ from $\Delta(a_g)$, she forces $\forall$ to go right in his next move. Thus effectively, by the way we have defined Δ, $\exists$ chooses the successor state in the biflow. As a consequence, she can indeed 'guess a path' through the structure.

Details of the proof that $L(\mathbb{A}_1)$ is indeed the class K_1 are left to the reader. $\lhd$

Example 5.8 For a second example, let K_2 be the class of pointed $\{g, y\}$-biflows containing a path (starting at the designated point of the biflow) on which every green node is eventually followed by a yellow one. Consider the Muller automaton $\mathbb{A}_2 = \langle A_2, a_p, \Delta_2, \mathcal{M}_2 \rangle$, where $A_2 = \{a_p, a_y, a_\top\}$, $\mathcal{M}_2 = \big\{\{a_\top\}, \{a_p\}, \{a_p, a_y\}\big\}$, and Δ_2 is given by

$$\begin{array}{lcl} \Delta_2(a_p) & := & \{(g, a_y, a_\top), (g, a_\top, a_y), (y, a_p, a_\top), (y, a_\top, a_p)\}, \\ \Delta_2(a_y) & := & \{(g, a_y, a_\top), (g, a_\top, a_y), (y, a_p, a_\top), (y, a_\top, a_p)\}, \\ \Delta_2(a_\top) & := & \{(g, a_\top, a_\top), (y, a_\top, a_\top)\}, \end{array}$$

As in the previous example, the presence of the 'surely successful state' $a_\top$, and the shape of Δ_2 makes that $\mathbb{A}_2$ guesses a path and checks whether it is of the right shape. The state a_p simply encodes that the automaton is on the path; the a_y in addition remembers that the path has passed a green state, but is still waiting to encounter a yellow state. As before, proof details are left to the reader. $\lhd$

Example 5.9 For a slightly different example, consider the Büchi tree automaton $\mathbb{A}_3 = \langle A_3, a_s, \Delta_3, F_3 \rangle$ given by $A_3 = \{a_s, a_g, a_y\}$, $F_3 = \{a_g\}$, and

$$\Delta_3(a) := \{(g, a_g, a_g), (y, a_y, a_y)\}$$

for each $a \in A_3$.

The basic intuition underlying the definition of $\mathbb{A}_3$ is that the state a_g (a_y, respectively), encode that the previous node in the tree was colored green (yellow, respectively). From this it is not too hard to derive that this automaton accepts exactly those trees in which every path contains infinitely many green points. $\lhd$

Example 5.10 Let K_4 denote the class of green/yellow trees in which immediately after each green node, a path starts which contains infinitely many yellow nodes. We leave it for the reader to verify that K_4 is the language recognized by the automaton $\mathbb{A}_4 = \langle A_4, a_s, \Delta_4, \mathcal{M}_4 \rangle$, where $A_4 = \{a_s, a_y\}$, $\mathcal{M}_4 = \big\{\{a_s\}, \{a_y\}, \{a_s, a_y\}\big\}$, and Δ_4 is given by

$$\begin{array}{lcl} \Delta_4(a_s) & := & \{(g, a_y, a_s), (g, a_s, a_y), (y, a_s, a_s)\}, \\ \Delta_4(a_y) & := & \{(y, a_y, a_s), (y, a_s, a_y)\}, \end{array}$$

Here the state a_s is a bit like the surely successful state $a_\top$ of earlier examples, the difference being that every time a green node is met, the automaton starts the search for a completely yellow path by activating the state a_y. $\lhd$

Acceptance and bisimilarity

As in the case of stream automata, our approach towards tree automata is strongly inspired by the links between the notions of bisimilarity between biflows and the acceptance of a biflow by an automaton. For C-biflows, the natural notion of a bisimulation is the following.

Definition 5.11 Let $\mathbb{S}$ and $\mathbb{S}'$ be two C-biflows. Then a nonempty relation $Z \subseteq S \times S'$ is a *bisimulation* if the following holds, for every $(s, s') \in Z$:

(color) $\sigma_C(s) = \sigma'_C(s')$;

(successor) both $(\sigma_0(s), \sigma'_0(s'))$ and $(\sigma_1(s), \sigma'_1(s'))$ belong to Z.

Two pointed biflows $(\mathbb{S}, s)$ and $(\mathbb{S}', s')$ are called *bisimilar*, notation: $\mathbb{S}, s \leftrightarroweq \mathbb{S}', s'$ if there is some bisimulation Z linking s to s'. In case the biflows $\mathbb{S}$ and $\mathbb{S}'$ are implicitly understood, we may drop reference to them and simply call s and s' bisimilar. $\lhd$

On the class of binary *trees*, the notion of bisimilarity reduces to the identity relation, and so it is not of use as a tool for comparing trees. This is not to say that it is not of interest in the study of trees. For instance, some concepts pertaining to trees can nicely be defined in terms of bisimilarity.

Definition 5.12 A binary tree over an alphabet C is called *regular* if it is bisimilar to a finite biflow over C. $\lhd$

To see how the acceptance game for tree automata relates to the notion of bisimilarity, we first characterize the notion of bisimilarity by game-theoretic means.

Definition 5.13 Let $\mathbb{A} = \langle A, \alpha, a\rangle$ and $\mathbb{S} = \langle S, \sigma, s\rangle$ be two pointed biflows. The *bisimilarity game* $\mathcal{B}(\mathbb{A}, \mathbb{S})$ is the two player board game, of which the set of positions and admissible moves are specified by Table 8. The winning condition of the game is simple: $\exists$ wins all infinite games. $\lhd$

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{\{(\alpha_0(a), \sigma_0(s)), (\alpha_1(a), \sigma_1(s))\} \mid \alpha_C(a) = \sigma_C(s)\}$
$\{(a_0, s_0), (a_1, s_1)\} \subseteq A \times S$	$\forall$	$\{(a_0, s_0), (a_1, s_1)\}$

Table 8: Bisimilarity game for biflows

We leave it for the reader to verify that a pair $(a, s) \in A \times S$ is a winning position for $\exists$ iff a and s are bisimilar.

As in the case of stream automata, one can see the acceptance game as a more sophisticated version of the bisimilarity game. Again, the starting point is to think of establishing a bisimulation between two pointed biflows in terms of one structure $\langle A, \alpha, a_I \rangle$ *classifying* the other, $\langle S, \sigma, s_0 \rangle$. Then to obtain the acceptance game for tree automata, three modifications are to be made. On the one hand, make a restriction in the sense that the classifying structure (the automaton) must be finite. On the other hand, instead of demanding that the transition function be of the form $\alpha : A \to \mathsf{B}_C A$, allow the one-step unfolding of an automaton state a to be dynamically determined, with $\exists$ choosing from a *set* $\Delta(a) \subseteq \mathsf{B}_C A$ of options. And finally, add an acceptance condition allowing $\forall$ to win infinite matches as well.

It is straightforward to turn a finite biflow into an automaton which accepts the biflow modulo bisimilarity.

Proposition 5.14 *Let $\mathbb{S} = \langle S, \sigma \rangle$ be a finite C-biflow. Define the map $\hat{\sigma} : S \to \wp(S)$ by putting $\hat{\sigma}(s) := \{\sigma(s)\}$, and let $\mathcal{M}_S$ be the set $\wp(S)$. Then for any point $s \in S$, and any pointed C-biflow $(\mathbb{S}', s')$,*

$$\mathbb{S}, s \leftrightarrows \mathbb{S}', s' \text{ iff the automaton } \mathbb{A}_{\mathbb{S},s} := \langle S, s, \hat{\sigma}, \mathcal{M}_S \rangle \text{ accepts } (\mathbb{S}', s').$$

Proof. A straightforward inspection reveals that the acceptance game $\mathcal{A}(\mathbb{A}_{\mathbb{S},s}@(s, s'))$ is essentially *identical* to the bisimilarity game $\mathcal{B}(\mathbb{S}, \mathbb{S}')@(s, s')$. The point is that at a position (t, t'), in the acceptance game, $\exists$ has no choice but to pick the element $\sigma(t)$ as the one-step realization of t, and after the resulting move $(\sigma(s), s')$ the match proceeds as in the bisimilarity game. The winning condition of the acceptance game specifies that $\exists$ should win all infinite matches, exactly as in the bisimilarity game. From these observations the proof is immediate. QED

The following proposition, the *proof* of which is left as an exercise for the reader, states that for any kind of tree automata, the associated recognizable languages are closed under bisimilarity.

Proposition 5.15 *Let $\mathbb{A}$ be a nondeterministic tree automaton, and let $(\mathbb{S}, s) \leftrightarrows (\mathbb{S}', s')$ be two bisimilar pointed biflows. Then $\mathbb{A}$ accepts $\mathbb{S}$ iff it accepts $\mathbb{S}'$.*

As a corollary, it follows that every biflow language is completely determined by its tree members.

Recognizability

Just as for stream automata, an important topic in the theory of tree automata is to compare the recognizing power associated with various acceptance conditions. As always, one can use the method of latest appearance records to show the equivalence of Muller and parity automata. It is *not* the case, however, that every language that is recognized by a Muller or parity automaton is also recognized by a Büchi automaton. Let us see an example.

Example 5.16 Let K_5 be the class of pointed biflows $(\mathbb{S}, s)$ over the alphabet $C = \{g, y\}$ such that every path starting from s contains at most finitely many green points. We leave it to the reader to find a Muller or parity automaton for this language. It is more interesting to see that there is *no* Büchi automaton $\mathbb{B}$ recognizing K.

▶ `Further details to be supplied`

◁

When it comes to recognizability, another obvious question is how the power of *deterministic* automata relates to that of the nondeterministic ones. We call an automaton $\langle A, a_I, \Delta : A \to \wp(\mathsf{B}_C A), Acc\rangle$ deterministic if for all $a \in A$ and all $c \in C$ there is a *unique* pair (a_0, a_1) such that $(c, a_0, a_1) \in \Delta(a)$. (Modulo the equivalence (26), this is the standard definition of determinism.)

Here we arrive at a significant difference with stream automata. Deterministic tree automata, even when equipped with a parity or Muller acceptance condition, do *not* have the same recognizing power as the nondeterministic ones, as the following example shows.

Example 5.17 Let $\mathbb{T}_0$ be the binary $\{g, y\}$-tree in which the left successor $0 = \sigma_0(\epsilon)$ of the root ϵ is the unique green element, and let $\mathbb{T}_1$ be defined analogously with respect to the right successor $1 = \sigma_1(\epsilon)$. It is not very hard to prove that any *deterministic* tree automaton $\mathbb{A}$ that accepts both $\mathbb{T}_0$ and $\mathbb{T}_1$, will also accept the tree in which 0 and 1 *together* form the complete set of green elements. But then such an automaton $\mathbb{A}$ cannot recognize the tree language L in which the root has exactly one green successor (i.e., *either* 0 *or* 1 is green).

On the other hand, it is easily seen that $\mathsf{L} = L(\mathbb{A}_6)$, where $A_6 = \{a_I, a_g, a_y, a_\top\}$, a_I is the initial state, Δ_6 is given by

$$\begin{array}{lcl}
\Delta_6(a_I) & := & \{(g, a_g, a_y), (g, a_y, a_g), (y, a_g, a_y), (y, a_y, a_g)\}, \\
\Delta_6(a_g) & := & \{(g, a_\top, a_\top)\}, \\
\Delta_6(a_y) & := & \{(y, a_\top, a_\top)\}, \\
\Delta_6(a_\top) & := & \{(g, a_\top, a_\top), (y, a_\top, a_\top)\},
\end{array}$$

and $\mathcal{M}_6 = \big\{\{a_\top\}\big\}$. ◁

This weakness of deterministic automata causes some nontrivial complications when it comes to applications in logic. To see why this is so, recall that in the previous Chapter we proved a Complementation Lemma (Proposition 4.27) as an direct corollary of the determinization of stream automata. We can prove a complementation lemma for tree automata as well, but instead of proving closure under complementation for a 'simpler', that is, deterministic, automata type, here we have to move to a *more complex* kind of automaton. This new type of automaton, involving the notion of *alternation*, will be introduced in section 5.3.

5.2 Emptiness

This section concerns the problem, whether a given tree automaton recognizes any pointed flow at all. For nondeterministic parity automata we shall prove two important results. First, we show that if a nondeterministic parity automaton accepts a pointed binary flow at all, it accepts a *finite* one. As an immediate corollary of this, we show that every nonempty recognizable tree language contains a regular tree. As the second result, we give an algorithm which *decides*, given a nondeterministic tree automaton $\mathbb{A}$, whether $L(\mathbb{A}) = \varnothing$ or not. As we shall see later on, these two properties are crucial in proving, respectively, the finite model property and decidability of the modal μ-calculus for binary flows. In both cases, the key tool in our proof will be the following *emptiness game* that we may associate with an automaton.

Definition 5.18 Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be a nondeterministic parity tree automaton. Then the *emptiness game* $\mathcal{G}_\varnothing(\mathbb{A})$ is given by Table 9. The winning condition for infinite matches is defined using the priority map for game positions (see the table) as a parity condition. $\lhd$

Position	Player	Admissible moves	Priority
$a \in A$	$\exists$	$\Delta(a)$	$\Omega(a)$
$((c, a_0, a_1), s) \in \mathsf{B}_C A$	$\forall$	$\{a_0, a_1\}$	0

Table 9: Emptiness game for nondeterministic parity tree automaton

Intuitively the reader may think of this game as the simultaneous projection on $\mathbb{A}$ of all acceptance games of $\mathbb{A}$, as should become clear from the proof of the theorem below.

This result, Theorem 5.19, establishes a kind of strong finite model property for tree automata: if a tree automaton accepts some biflow, it accepts a finite one. The 'strength' of the property lies in the fact, that the size of the finite biflow accepted by the automaton is bounded by the size of the automaton. In fact, the biflow $\langle S, \sigma \rangle$ 'lives inside' the automaton $\langle A, a_I, \Delta, \Omega \rangle$, in the following sense. The set S of states is a *subset* of the carrier A of the automaton, and for each $s \in S$, the coalgebra unfolding $\sigma(s)$ is one of the realizations enabled by $\mathbb{A}$: $\sigma(s) \in \Delta(s)$.

One final remark: the proof of Theorem 5.19 involves a crucial application of the Memory-Free Determinacy of parity games, a fundamental result that we will prove and discuss in Chapter 7 on Board Games.

Theorem 5.19 *Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be a nondeterministic parity tree automaton. Then the following are equivalent:*

1. *$L(\mathbb{A}) \neq \varnothing$;*
2. *$a_I \in \mathrm{Win}_\exists(\mathcal{G}_\varnothing(\mathbb{A}))$;*
3. *$\mathbb{A}$ accepts a pointed biflow (S, σ, a_I) with $S \subseteq A$ and $\sigma(a) \in \Delta(a)$ for all $a \in A$.*

Proof. $\boxed{1 \Rightarrow 2}$ Suppose that $\mathbb{A}$ accepts some pointed flow $(\mathbb{S}, s_0)$. Then by definition, $\exists$ has a winning strategy in the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})@(a_I, s_0)$. This strategy will be the basis of her winning strategy in the emptiness game of $\mathbb{A}$.

Concretely, in $\mathcal{G}_\varnothing(\mathbb{A})@a_I$, $\exists$ will maintain the following condition. Let

$$a_I\gamma_1 a_1 \gamma_2 \dots a_k,$$

be an initial segment of an $\mathcal{G}_\varnothing(\mathbb{A})$-match (with $\gamma_{i+1} \in \Delta(a_i)$ being the move of $\exists$ at position a_i). Then $\exists$ sees this match as the projection of a parallel match of $\mathcal{A}(\mathbb{A}, \mathbb{S})@(a_I, s_0)$ where $\exists$ plays her winning strategy Φ:

$$\begin{array}{cccccccc}
(a_I, s_0) & (\gamma_1, b_0^1, b_1^1) & \{(b_0^1, \sigma_0(s_0)), (b_1^1, \sigma_1(s_0))\} & (a_1 = b_i^1, s_1 = \sigma_i(s_0)) & \dots & (a_k, s_k) & \dots \\
\Downarrow & \Downarrow & \Downarrow & \Downarrow & & \Downarrow & \\
a_I & \gamma_1 & - & a_1 = b_i^1 & \dots & a_n & \dots
\end{array}$$

The existence of such a parallel match is easily proved by an inductive argument, of which the base case is immediate by the shape (a_I versus (a_I, s_0)) of the initial game positions. Inductively assume that at stage k, the matches of $\mathcal{G}_\varnothing(\mathbb{A})$ and $\mathcal{A}(\mathbb{A}, \mathbb{S})$ have arrived at the positions a_k and (a_k, s_k) respectively. Suppose that $\exists$'s winning strategy in $\mathcal{A}(\mathbb{A}, \mathbb{S})@(a_I, s_0)$ tells her to choose $\gamma = (c, b_0, b_1) \in \Delta(a_k)$ at this position. Then define $\gamma_{k+1} := \gamma$, and let $\exists$ choose this γ at position a_k of $\mathcal{G}_\varnothing(\mathbb{A})$. Note that since this strategy of $\exists$ is supposed to be winning, it must be the case that $c = \sigma_C(s_k)$, so that $\exists$ does not get stuck. Now suppose that in the match of $\mathcal{G}_\varnothing(\mathbb{A})$, $\forall$ chooses the direction i, picking $a_{k+1} := b_i$ as the next position. Then, returning to the match of the acceptance game, we may let $\forall$ choose the same direction i there, making $(b_i, \sigma_i(s))$ the next position. In other words, we have proved that $\exists$ can maintain the parallel match for one more round.

Using this strategy in the emptiness game will then guarantee her to win the match, since the associated sequence of $\mathbb{A}$-states is the same for both matches, and the $\mathcal{A}(\mathbb{A}, \mathbb{S})$-match $\exists$ plays according to her supposedly winning strategy.

$\boxed{2 \Rightarrow 3}$ Assume that $\exists$ has a winning strategy in the emptiness game starting from the initial state a_I of $\mathbb{A}$. The key point of the emptiness game for parity automata is that $\mathcal{G}_\varnothing(\mathbb{A})@a_I$ is a parity game, and so we may without loss of generality assume that this strategy is *positional*, see Theorem 7.21. In other words, we may represent it as a map $\sigma : A \to \mathsf{B}_C A$. Let $W := \mathrm{Win}_\exists(\mathcal{G}_\varnothing(\mathbb{A}))$ be the set of positions in A that are winning for $\exists$. We invite the reader to check that $\sigma(a) \in \mathsf{B}_C W$ for all $a \in W$. Now define $\mathbb{S}$ be the binary flow $\langle W, \sigma \rangle$. We claim that $\mathbb{A}$ accepts $(\mathbb{S}, a_I)$

To see why this is the case, we will prove that (a_I, a_I) is a winning position in the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ initialized at (a_I, s). The winning strategy that we may equip $\exists$ with in this game is in fact very simple:

- at position (a, s), pick $(\sigma(a), s)$ as the next position if $a = s \in \mathrm{Win}_{\exists}(\mathcal{G}_{\varnothing}(\mathbb{A}))$, and choose a random element otherwise.

It can be proved that any match of the acceptance game in which $\exists$ uses this strategy, can be 'projected' onto a match of the emptiness game in which she plays her winning strategy:

$$\begin{array}{ccccccccc}
(a_I, a_I) & (\sigma(a_I), a_I) & \{(\sigma_i(a_I), \sigma_i(a_I)) \mid i \in 2\} & (a_1, a_1) & (\sigma(a_1), a_1) & \dots & (a_n, a_n) & \dots \\
\Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow & & \Downarrow & \\
a_I & \sigma(a_I) & - & a_1 & \sigma(a_1) & \dots & a_n & \dots
\end{array}$$

Given the winning conditions of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}_{\varnothing}(\mathbb{A})$ it is then immediate that the given strategy indeed guarantees that $\exists$ wins any match starting at position (a_I, a_I).

$\boxed{3 \Rightarrow 1}$ This implication is a direct consequence of the definitions. QED

From the computational version of Memory-Free Determinacy we can derive the following decidability result.

Theorem 5.20 *There is an algorithm deciding, for a given nondeterministic parity tree automaton $\mathbb{A}$, whether $L(\mathbb{A})$ is empty or not.*

Proof. Consider the associated emptiness game of the input automaton $\mathbb{A}$. Given the fact that $\mathcal{G}_{\varnothing}(\mathbb{A})$ is a parity game, it follows from **??** that it is computable whether a_I is a winning position in this game or not. By the previous proposition, this provides an algorithm which determines whether $L(\mathbb{A})$ is empty or not. QED

5.3 Alternation

Earlier on we saw that we can model the acceptance procedure of a nondeterministic tree automaton as a two-player game. However, the interaction between the two players is fairly limited; in particular, the only role of $\forall$ in the game is to select the direction of the path in the flow. Allowing for some more interaction, we arrive at a fundamental concept from theoretical computer science, viz., that of machine models based on *alternation*. Roughly, the idea underlying the alternating machine model is that, apart from *existential* choices made by the player that is working towards *some* successful run of the machine, there are also *universal* choices yielding parallel runs *all* of which have to be successful. For a more precise formulation of the concept, a game-theoretic framework is best. For instance, game theory allows us to naturally generalize the notion of a *run* of a machine on an input object, to that of a *match* being played in order to determine the behavior of a machine on a given input object.

There are many ways to cast this idea of interaction between players into a formal model (see Remark 5.23 for some more discussion). In this Chapter, we will represent the transition function of an alternating tree automaton as a map

$$\Delta : A \to \wp\wp(\mathsf{B}_C A).$$

Here the first power set symbol represents a choice for $\exists$, and the second one, a choice for $\forall$.

Convention 5.21 In the sequel, when giving the transition function of an automaton in set-theoretical format, we will frequently make an explicit reference to the player whose choice is represented.

For instance, suppose that we are in a context where positions of the form $(X, s) \in \wp(Q) \times S$ belong to player σ, and that the move that σ makes is determined by picking an element $q \in Q$. Then we may write, whenever it may be helpful, $\wp_\sigma(Q)$ rather than $\wp(Q)$. In particular, we will often write

$$\Delta : A \to \wp_\exists\wp_\forall(\mathsf{B}_C A).$$

for the transition map of an alternating tree automaton.

This brings us to the following definition.

Definition 5.22 An *alternating C-tree automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, a_I\rangle$ such that $\Delta : A \to \wp_\exists\wp_\forall(\mathsf{B}_C A)$ is the *transition function*, $Acc \subseteq A^\omega$ is the *acceptance condition*, and $a_I \in A$ is the *initial state* of the automaton.

The admissible moves of the *acceptance game* associated with these automata are given in Table 10, and its winning conditions are standardly derived from the acceptance condition Acc. An alternating C-tree automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ *accepts* a C-biflow $\mathbb{S} = \langle S, s_0, \sigma\rangle$, if the pair (a_I, s_0) is a winning position for $\exists$ in the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$. $\lhd$

Any round of this acceptance game consists of four moves, which can be naturally grouped together as follows:

static part: $(a, s) \xrightarrow{\exists} (\Gamma, s) \xrightarrow{\forall} (\gamma, s)$;

dynamic part $((c, a_0, a_1), s) \xrightarrow{\exists} \{(a_0, s_0), (a_1, s_1)\} \xrightarrow{\forall} (a_i, \sigma_i(s))$.

The name ‘static’ refers to the fact that the match stays in the same point s of the biflow, while the players interactively determine the ‘successor’ $\gamma \in \mathsf{B}_C A$ of a. Then, the ‘dynamic’ stage of the round is as in the bisimilarity game for biflows, see Definition 5.13.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\Gamma, s) \in \wp_\forall(\mathsf{B}_C A) \times S \mid \Gamma \in \Delta(a)\}$
$(\Gamma, s) \in \wp_\forall(\mathsf{B}_C A) \times S$	$\forall$	$\{(\gamma, s) \in \mathsf{B}_C A \times S \mid \gamma \in \Gamma\}$
$((c, a_0, a_1), s) \in \mathsf{B}_C A \times S$	$\exists$	$\{\{(a_0, \sigma_0(s)), (a_1, \sigma_1(s))\} \mid c = \sigma_C(s)\}$
$\{(a_0, s_0), (a_1, s_1)\} \subseteq A \times S$	$\forall$	$\{(a_0, s_0), (a_1, s_1)\}$

Table 10: Acceptance game for alternating tree automata

▶ `Example to be supplied`

Remark 5.23 The choice to represent alternation as a map

$$\Delta : A \to \wp_\exists \wp_\forall(\mathsf{B}_C A).$$

may look rather arbitrary. Why not take a map from A to $\wp_\forall \wp_\exists(\mathsf{B}_C A)$, or to $\wp_\exists \wp_\forall \wp_\exists(\mathsf{B}_C A)$? As we will see in Chapter 6 when we discuss a logical presentation of alternation, all these formalizations are in fact equivalent. The one we chose in Definition 5.22 provides a natural choice in this spectrum, because it corresponds to some *disjunctive normal form.* $\lhd$

5.4 From alternation to nondeterminism

In this section we will see that every alternating tree automaton can be effectively transformed into an equivalent nondeterministic one. As we will see later on, this result is of fundamental importance in the theory of tree automata and its application in logic.

Theorem 5.24 *There is an effective procedure transforming an alternating parity tree automaton into an equivalent nondeterministic parity tree automaton.*

Before going into the technical details, let us first give some intuitions behind the proof of Theorem 5.24. Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be a fixed alternating parity tree automaton. Our construction of its nondeterministic equivalent $\widehat{\mathbb{A}}$ can be split into two steps.

In the first (and most important) step of the construction, we define an nondeterministic automaton $\mathbb{A}^\sharp$ via a variation of the power set construction. Roughly, the idea is that a match of $\mathcal{A}(\mathbb{A}^\sharp, \mathbb{S})$ corresponds to $\exists$ playing *various matches* of $\mathcal{A}(\mathbb{A}, \mathbb{S})$, all on the same path through $\mathbb{S}$. The automaton $\mathbb{A}^\sharp$ will allow $\exists$ to take care of all $\forall$'s $\mathbb{A}$-moves in parallel. (This only applies to his moves in the static part of the game. We need not worry about $\forall$'s moves in the dynamic part since these concern a choice of direction only, and this choice will still be assigned to $\forall$ in the nondeterministic case.)

As a consequence, each basic position (R, s) of $\mathcal{A}(\mathbb{A}^\sharp, \mathbb{S})$ should somehow encode a collection of basic positions in $\mathcal{A}(\mathbb{A}, \mathbb{S})$. However, if we would simply take the states of $\mathbb{A}^\sharp$ to be *macro-states* of $\mathbb{A}$, i.e., subsets of $\mathbb{A}$, we would get into trouble when defining the acceptance condition of $\mathbb{A}$, similar to the problems one encounters when determinizing stream automata, see 4.4. An elegant way out is provided by defining the carrier set $A^\sharp$ of $\mathbb{A}^\sharp$ to be the set of *binary relations* over A, and to link $A^\sharp$-sequences and A-sequences via the notion of a *trace*.

Definition 5.25 Given an infinite word $\rho = R_1R_2R_3\dots$ over the set $A^\sharp$ of binary relations over a set A, a *trace* through ρ is a finite A-word $\alpha = a_0a_1a_2\dots a_k$, or an A-stream $\alpha = a_0a_1a_2\dots$, such that $a_iR_{i+1}a_{i+1}$ for all $i < k$ (respectively, for all $i < \omega$). $\lhd$

The key idea behind the definition of $\mathbb{A}^\sharp$ and the proof of its equivalence to $\mathbb{A}$, is that with each $\mathcal{A}(\mathbb{A}^\sharp, \mathbb{S})$-match

$$(R_1, s_1)(R_2, s_2)(R_3, s_3)\dots$$

and each trace $a_0a_1a_2$ through $R_1R_2R_3\dots$ we may associate an $\mathcal{A}(\mathbb{A}, \mathbb{S})$-match

$$(a_1, s_1)(a_2, s_2)(a_3, s_3)\dots$$

and conversely. This explains the winning condition of the automaton $\mathbb{A}^\sharp$: a $A^\sharp$-stream should be winning for $\exists$ if *all* traces through it are winning according to the acceptance condition of $\mathbb{A}$. From now on we focus on automata with parity conditions.

Definition 5.26 Relative to a parity condition Ω on A, call a trace $\alpha \in A^\omega$ *bad* if it is infinite and the maximum priority occurring infinitely often on α, is an odd number. Let NBT_Ω denote the set of infinite $A^\sharp$-words that contain no bad traces relative to Ω. $\lhd$

While we can establish that $\mathbb{A}^\sharp$, equipped with the acceptance condition NBT_Ω, is equivalent to $\mathbb{A}$, its own acceptance condition clearly is not a parity condition. The second part of the construction then consists of showing that $\mathbb{A}^\sharp$ can be replaced with a nondeterministic automaton of which the acceptance condition is of the required format.

The automaton $\mathbb{A}^\sharp$

Before giving the formal details, let us first provide some further intuitions behind the definition of $\mathbb{A}^\sharp$. Our starting point is that a state R of $\mathbb{A}^\sharp$ encodes the macro-state $\mathsf{Ran}(R) := \{b \in A \mid (a, b) \in R \text{ for some } a \in A\}$, that is, the range of R. This already suffices to motivate the definition of the initial state of $\mathbb{A}^\sharp$:

$$R_I := \{(a_I, a_I)\}$$

In order to gather some intuitions concerning the definition of $\Delta^\sharp$, consider a biflow $\mathbb{S}$ and a position of the form (R, s) in the acceptance game $\mathcal{G}^\sharp = \mathcal{A}(\mathbb{A}^\sharp, \mathbb{S})$. Let c denote the color of s. Take a state $a \in \mathsf{Ran}(R)$, and consider the set $\Delta(a)$ of $\exists$'s choices at position (a, s) in the game $\mathcal{G} = \mathcal{A}(\mathbb{A}, \mathbb{S})$. The first observation is that if $\exists$ picks a set $\Gamma \in \Delta(a)$, she better makes sure that *all* elements of Γ are of the form (c', b_0, b_1) with $c' = c$, or else she would give $\forall$ an almost immediate win.

So, assume that $\exists$ has chosen an element $\Gamma \in \Delta^c(a)$, with

$$\Delta^c(a) := \{\Gamma \in \Delta(a) \mid c' = c \text{ for all } (c', b_0, b_1) \in \Gamma\}.$$

(If no such Γ exists, then it is to be expected that $\Delta^\sharp(R) = \varnothing$, with the effect that $\exists$ loses any match at any position (R, s).)

The crucial question is now, for each $i = 0, 1$: For which states b in A can $\exists$ expect the pair $(b, \sigma_i(s))$ as the next basic position of the $\mathcal{G}$-match (i.e., after (a, s))? Given the key idea behind the definition of the automaton $\mathbb{A}^\sharp$ (see the discussion following Definition 5.25), the point is that, for each such b, the pair (a, b) would form a possible continuation of a trace through any $A^\sharp$-sequence ending at relation R. For an answer to the crucial question, observe that $b_i \in A$ is such a state iff $\forall$ can choose a triple $(c, b_0, b_1) \in \Gamma$, and then decide to go in the direction i. Thus the set of these positions can be denoted as $\pi_i(\Gamma)$, where $\pi_i : C \times A \times A \to A$ denotes the appropriate projection function.

Finally, for the definition of $\Delta^\sharp$, the idea is that for each $c \in C$, the elements of $\Delta^\sharp(R)$ of the form (c, Q_0, Q_1) are in direct correspondence with the set

$$\{\Gamma : \mathsf{Ran}(R) \to \wp(\mathsf{B}_C A) \mid \Gamma(a) \in \Delta^c(a) \text{ for all } a \in \mathsf{Ran}(R)\},$$

which represents the vector of reasonable choices available to $\exists$, on the assumption that the current state of the automaton is an element of $\mathsf{Ran}(R)$. More precisely, $\Delta^\sharp(R)$ will be given by putting $(c, Q_0, Q_1) \in \Delta^\sharp(R)$ iff, for some Γ in the above set, we have $Q_0[a] = \pi_0[\Gamma(a)]$ and $Q_1[a] = \pi_1[\Gamma(a)]$, for all $a \in \mathsf{Ran}(R)$.

Definition 5.27 Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$ be an alternating parity tree automaton. $\mathbb{A}^\sharp$ is given as the nondeterministic tree automaton $\mathbb{A}^\sharp := \langle A^\sharp, R_I, \Delta^\sharp, \mathrm{NBT}_\Omega\rangle$. Here $A^\sharp = \wp(A \times A)$ is the set of binary relations on A, the initial state R_I is the relation $R_I := \{(a_I, a_I)\}$, and the transition function $\Delta^\sharp$ is given by

$$\Delta^\sharp(R) := \big\{(c, Q_0, Q_1) \in \mathsf{B}_C A^\sharp \mid \forall a \in \mathsf{Ran}(R) \exists \Gamma_a \in \Delta^c(a)\, (Q_0[a], Q_1[a]) = (\pi_0[\Gamma_a], \pi_1[\Gamma_a])\big\}.$$

Finally, the acceptance condition NBT_Ω is as in Definition 5.26. $\lhd$

In the sequel, we will standardly abbreviate $\mathcal{G} := \mathcal{A}(\mathbb{A}, \mathbb{S})$ and $\mathcal{G}^\sharp := \mathcal{A}(\mathbb{A}^\sharp, \mathbb{S})$.

Proposition 5.28 *Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$ be an alternating parity tree automaton. Then $\mathbb{A}$ is equivalent to $\mathbb{A}^\sharp$.*

Proof. Let $\mathbb{S} = \langle S, \sigma \rangle$ be an arbitrary binary C-biflow, and let r be an arbitrary point in $\mathbb{S}$. We will prove that

$$\mathbb{A} \text{ accepts } (\mathbb{S}, r) \text{ iff } \mathbb{A}^\sharp \text{ accepts } (\mathbb{S}, r), \tag{27}$$

which clearly suffices to prove the proposition.

$\boxed{\Rightarrow}$ For the direction from left to right of (27), assume that $\mathbb{A}$ accepts the binary C-biflow $(\mathbb{S}, r)$. Then by definition, (a_I, r) is a winning position for $\exists$ in the acceptance game $\mathcal{G} := \mathcal{A}(\mathbb{A}, \mathbb{S})$. The argument can be simplified a bit because $\mathcal{G}$ is a board game with a *parity* acceptance condition: Without loss of generality assume the existence of a winning strategy that is *positional*, that is, we may represent it as a map $\Phi : A \times S \to \wp(\mathsf{B}_C A)$.

Define the following (positional) strategy for $\exists$ in $\mathcal{G}^\sharp$. At position (R, s), she picks the pair $\Psi(R, s) := (\sigma_C(s), Q_0, Q_1) \in C \times A^\sharp \times A^\sharp$, where Q_0 and Q_1 are given, by putting for $a \in A$:

$$Q_i[a] := \begin{cases} \pi_i[\Phi(a, s)] & \text{if } a \in \mathsf{Ran}(R), \\ \varnothing & \text{otherwise.} \end{cases}$$

Note that $\Psi(R, s)$ is not guaranteed to belong to $\Delta^\sharp(R)$, so it need not be a legitimate move for $\exists$ at every position (R, s) of $\mathcal{G}^\sharp$. Nevertheless, we claim that Ψ is a winning strategy for $\exists$ in the game $\mathcal{G}^\sharp$ starting at (R_I, r).

Our main technical claim in our proof is the following. Call a position (R, s) of $\mathcal{G}^\sharp$ *safe* if $(a, s) \in \mathrm{Win}_\exists(\mathcal{G})$, for each $a \in \mathsf{Ran}(R)$.

Claim 1 Let (R, s) be a safe position of $\mathcal{G}^\sharp$, and let $\Psi(R, s) = (c, Q_0, Q_1)$. Then for each $a \in \mathsf{Ran}(R)$, each $i \in \{0, 1\}$, and each $b \in Q_i[a]$ there is a scenario for a round of the game $\mathcal{G}$, starting at position (a, s) and ending at $(b, \sigma_i(s))$, in which $\exists$ plays her strategy Φ.

Proof of Claim Assume the conditions in the claim, and let a, b and i be as stated. Consider the following round of the game $\mathcal{G}$, starting at position (a, s).

- At position (a, s), $\exists$ plays her winning strategy, picking $\Phi(a, s) \in \Delta(a)$, thus making $(\Phi(a, s), s)$ the next position.

Recall that $c = \sigma_C(s)$ by definition of Ψ. We may assume that $\Phi(a, s) \in \Delta^c(a)$, for otherwise, $\forall$ could pick an element $(c', b_0, b_1) \in \Phi(a, s)$ with $c' \neq c$. Then $\exists$ would get stuck at her next move, contradicting the assumptions that $(a, s) \in \mathrm{Win}_\exists(\mathcal{G})$ and Φ is a winning strategy. But if $\Phi(a, s) \in \Delta^c(a)$, it follows from the definition of Ψ that $Q_i[a] = \pi_i[\Phi(a, s)]$. So from $b \in Q_i[a]$ we may infer the existence of a triple $(c', b_0, b_1) \in \Phi(a, s)$ with $b = b_i$. Now continue the round of $\mathcal{G}$ as follows.

- At position $(\Phi(a, s), s)$, $\forall$ picks $((c', b_0, b_1), s)$ as the next position.

It follows from $\Phi(a, s) \in \Delta^c(a)$ that $c' = c$, and so $\exists$ will not get stuck here.

- At position $((c', b_0, b_1), s)$, $\exists$ moves to position $\{(b_0, s_0), (b_1, s_1)\}$.
- Subsequently, $\forall$ chooses (b_i, s) as the next position, finishing the round.

The claim then follows from the fact that $b = b_i$. ◀

The following claim justifies the terminology for safe positions.

Claim 2 Let (R, s) be a safe position of $\mathcal{G}^\sharp$. Then $\Psi(R, s) = (c, Q_0, Q_1)$ is a legitimate move for $\exists$, she will not get stuck after this move, and the next basic position of $\mathcal{G}^\sharp$ will also be safe.

Proof of Claim Write $\Psi(R, s) = (c, Q_0, Q_1)$. In order to prove that $\Psi(R, s)$ is a legitimate move at position (R, s), we need to show that $\Psi(R, s) \in \Delta^\sharp(R)$, and for this purpose it suffices to find, for each $a \in \mathsf{Ran}(R)$, a $\Gamma \in \Delta^c(a)$ with $Q_i[a] = \pi_i[\Gamma]$. But in the proof of Claim 1 we showed that the set $\Gamma = \Phi(s, a)$ meets all these criteria. Note too that since $\Gamma \in \Delta^c(a)$, and $c = \sigma_C(s)$ by definition of Ψ, $\exists$ will not get stuck after this move.

For the final statement of the claim, let $(Q_i, \sigma_i(s))$ be the next basic position of the game $\mathcal{G}^\sharp$ after (R, s). Take an arbitrary element $b \in \mathsf{Ran}(Q_i)$. It follows from the definition of $\Psi(R, s)$ that $b \in Q_i[a]$ for some $a \in \mathsf{Ran}(R)$, and so $(b, \sigma_i(s)) \in \mathrm{Win}_\exists(\mathcal{G})$ by Claim 1. Since b was arbitrary, this shows that $(Q_i, \sigma_i(s))$ is a safe position. ◀

Turning our attention to $\mathcal{G}^\sharp$-matches starting at position (R_I, r), we first observe that by definition of R_I and the assumption that $(a_I, r) \in \mathrm{Win}_\exists(\mathcal{G})$, (R_I, r) is a safe position. But then it follows by iterative applications of Claim 2, that any Ψ-conform match of $\mathcal{G}^\sharp@(R_I, r)$ is infinite, and that all basic positions of such a match are safe. Let

$$(R_1, s_1)(R_2, s_2)(R_3, s_3)\ldots$$

be the sequence of basic positions of such a match. In order to prove that Ψ is a winning strategy, it suffices to prove that there are no bad traces through $R_1R_2\ldots$.

But, again using Claim 1, we may inductively show that with each trace $\rho = a_0a_1a_2\ldots$ through the $A^\sharp$-stream $R_1R_2R_3\ldots$, we may associate a $\mathcal{G}$-match with basic positions

$$(a_1, s_1)(a_2, s_2)\ldots$$

starting at position $(a_1, s_1) = (a_I, r)$, in which $\exists$ plays her strategy Φ. Since the strategy Φ was assumed to be winning for the game initialized at (a_I, r), this means that the trace ρ cannot be bad.

From this it follows that the $A^\sharp$-stream $R_1R_2R_3\ldots$ does not contain a bad trace, and so $\exists$ wins any match of $\mathcal{G}^\sharp$ starting at position (R_I, r), as long as she sticks to her strategy Ψ. In other words: $(R_I, r) \in \mathrm{Win}_\exists(\mathcal{G}^\sharp)$, and so by definition, $\mathbb{A}^\sharp$ accepts $(\mathbb{S}, r)$, as required.

$\boxed{\Leftarrow}$ For the opposite direction of (27), assume that $\mathbb{A}^\sharp$ accepts $(\mathbb{S}, r)$. In other words, we may assume that there is a strategy f which is winning for $\exists$ in the game $\mathcal{G}^\sharp$ starting at (R_I, r). In order to show that $\mathbb{A}$ accepts $(\mathbb{S}, r)$, we need to prove that (a_I, r) is a winning position for $\exists$ in $\mathcal{G}$.

We will equip $\exists$ with a strategy f', in the game $\mathcal{G}$ initialized at (a_I, r), which has the following property. For any (possibly finite) f'-conform match $(a_1, s_1)(a_2, s_2)\dots$ of $\mathcal{G}$, with $a_1 = a_I$ and $s_1 = r$, there is an f-conform match $(R_1, s_1)(R_2, s_2)\dots$ of $\mathcal{G}^\sharp$, with $R_1 = R_I$, satisfying the condition that

$$a_{i+1} \in R_{i+1}[a_i] \text{ for every stage } i. \tag{28}$$

Hence, the sequence of $\mathbb{A}$-states $a_0a_1a_2\dots$ (with $a_0 = a_1$) induced by such a match is a *trace* of the $A^\sharp$-sequence $R_0R_1R_2\dots$ which we may associate with the f-conform match. Since f is by assumption winning for $\exists$, by definition of the winning condition NBT_Ω of $\mathbb{A}^\sharp$, the (maximum parity occurring infinitely often on) the trace must be *even.* This will guarantee that $\exists$ wins all *infinite* matches of the game. Hence, it suffices to prove that at any finite stage of an f'-conform match, she either wins immediately, or else she can keep the above condition for one more round.

Suppose then that $\exists$ has been able to keep this condition for k steps. That is, with the partial $\mathcal{G}$-match $(s_0, a_0)\dots(s_k, a_k)$ (where $a_I = a_0$) we may associate a partial, f-conform $\mathcal{G}^\sharp$-match $(s_0, R_0)\dots(s_k, R_k)$ such that $R_0 = R_I$ and

$$a_{i+1} \in R_{i+1}[a_i] \text{ for all } 0 < i < k. \tag{29}$$

For notational convenience, write $a = a_k$, $R = R_k$ and $s = s_k$, so we have $a \in \mathsf{Ran}(R)$. Let $(c, Q_0, Q_1) \in \mathsf{B}_C A^\sharp$ be the move dictated by $\exists$'s strategy f in $\mathcal{G}^\sharp$. Since f is a winning strategy for $\exists$ this move must be legitimate, that is, $(c, Q_0, Q_1) \in \Delta^\sharp(R)$. We may also infer that $c = \sigma_C(s)$, for otherwise, $\exists$ would get stuck immediately after this move, and lose.

Let us now define $\exists$'s strategy f' in round $k+1$ of $\mathcal{G}$. Since $(c, Q_0, Q_1) \in \mathsf{B}_C A^\sharp$, by definition of $\Delta^\sharp$, and the fact that $a \in \mathsf{Ran}(R)$, there is some $\Gamma \in \Delta^c(a)$ such that $Q_i[a] = \pi_i[\Gamma]$, for $i \in \{0, 1\}$. This Γ is the next move of $\exists$ in the game $\mathcal{G}$.

If $\Gamma = \varnothing$, then $\exists$ wins right away, in which case we are done. So assume that $\Gamma \neq \varnothing$, and suppose that $\forall$ responds to $\exists$'s move with a triple $(c', b_0, b_1) \in \Gamma$. Observe first that $c' = c = \sigma_C(s)$, since $\Gamma \in \Delta^c(a)$ and $c = \sigma_C(s)$. From this it follows that $\exists$ does not get stuck at position $((c', b_0, b_1), s)$, and so she may safely move to the next position, $\{(b_0, \sigma_0(s)), (b_1, \sigma_1(s))\}$.

Now suppose that at this stage of the match, $\forall$ chooses $(a_{k+1}, s_{k+1}) \in \{(b_0, \sigma_0(s)), (b_1, \sigma_1(s))\}$ as the next basic position. Let i be such that $b_i = a_{k+1}$ and $\sigma_i(s) = s_{k+1}$, and let $\forall$, in

$\mathcal{G}^\sharp$, at position $((c, Q_0, Q_1), s_{k+1})$ choose direction i, making (Q_i, s_{k+1}) the next (basic) position of the match. That is, we define $R_{k+1} := Q_i$.

In order to prove (29) for $k + 1$, it suffices to show that $(a_k, a_{k+1}) \in R_{k+1}$, that is, $(a, b_i) \in Q_i$. But this follows from the observation that $b_i = \pi_i(c, b_0, b_1) \in \pi_i[\Gamma] = Q_i[a]$. QED

Regular automata

In the previous subsection we defined a nondeterministic tree automaton $\mathbb{A}^\sharp$ and proved it to be equivalent to the given automaton $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$. The problem with the automaton $\mathbb{A}^\sharp$ is that its acceptance condition $\mathrm{NBT}_\Omega \subseteq (A^\sharp)^\omega$ is not given by a parity function. We will now see that this problem can easily be overcome since NBT_Ω has the form of an ω-regular language over the language $A^\sharp$, that is, it is recognized by some stream automaton.

Definition 5.29 An automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ is called *ω-regular* if $Acc \subseteq A^\omega$ is an ω-regular language. $\lhd$

In this section we shall prove that, given an regular tree automaton $\mathbb{A}$ of which the acceptance condition is given by some deterministic stream automaton $\mathbb{Z}$, we can effectively construct a parity automaton $\mathbb{A} \odot \mathbb{Z}$ that is equivalent to $\mathbb{A}$. First however, we show that, indeed, $\mathbb{A}^\sharp$ is a regular automaton, by constructing a stream automaton recognizing the ω-language NBT_Ω.

Proposition 5.30 *Let A be some finite set, and let $\Omega : A \to \omega$ be a parity function on A. Then the set NBT_Ω is an ω-regular language over the alphabet $A^\sharp$.*

Proof. First we define a nondeterministic $A^\sharp$-stream parity automaton $\mathbb{B}$ which accepts exactly those infinite $A^\sharp$-streams that *do* contain a bad trace. Given the properties of parity stream automata it is fairly straightforward to continue from here. First, take a deterministic equivalent $\mathbb{B}'$ of $\mathbb{B}$; such an automaton exists by Theorem 4.26. And second, since $\mathbb{B}'$ is deterministic, it is easy to perform complementation on it, that is, define an automaton $\mathbb{C}$ that accepts exactly those $A^\sharp$-streams that are rejected by $\mathbb{B}'$. In short: $L_\omega(\mathbb{C}) = (A^\sharp)^\omega \setminus L_\omega(\mathbb{B}') = (A^\sharp)^\omega \setminus L_\omega(\mathbb{B})$. Clearly then $L_\omega(\mathbb{C}) = \mathrm{NBT}_\Omega$.

For the definition of $\mathbb{B}$, take an object $b_I \notin A$, and define $B := A \cup \{b_I\}$. Let $\Delta : B \times A^\sharp \to \wp(B)$ be given by putting

$$\Delta(b, R) := \begin{cases} \mathsf{Ran}(R) & \text{if } b = b_I, \\ R[b] & \text{if } b \in A, \end{cases}$$

and define Ω^{+1} by putting $\Omega^{+1}(a) := \Omega(a) + 1$ for $a \in A$, and $\Omega^{+}(b_I) := 0$. Then $\mathbb{B}$ is the automaton $\langle B, b_I, \Delta, \Omega^{+1}\rangle$.

It is immediate from the definitions that $b_I \xrightarrow{R} a$ iff $a \in \mathsf{Ran}(R)$, that is, if there is some $a' \in A$ such that $a'Ra$. From this and the definition of Δ it follows that

$$b_I \xrightarrow{R_1} a_1 \xrightarrow{R_2} a_2 \xrightarrow{R_3} \dots$$

is a run of $\mathbb{B}$ iff there is some $a_0 \in A$ such that $a_0 a_1 a_2 \dots$ is a trace through $R_1 R_2 \dots$ Then the definition of Ω^{+1} ensures that $\mathbb{B}$ indeed accepts those $A^\sharp$-streams that contain a bad trace. QED

It follows from Proposition 5.30 that the automaton $\mathbb{A}^\sharp$ defined in the previous section is a regular automaton. Hence we have proved the main result of this section if we can show that every nondeterministic regular tree automaton can be replaced with one with a parity acceptance condition. This is what we will focus on now. In fact, we will effectively transform a nondeterministic, regular tree automaton into an equivalent parity automaton, at least if we are also given a word automaton recognizing the ω-regular language which forms the acceptance condition of the tree automaton.

Definition 5.31 Let $\mathbb{Z} = \langle Z, z_I, \delta, \Omega\rangle$ be a deterministic parity A-stream automaton, and let $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ be a nondeterministic tree automaton. Then $\mathbb{A} \odot \mathbb{Z}$ is the nondeterministic parity tree automaton given as $\mathbb{A}\odot\mathbb{Z} = \langle A \times Z, (a_I, z_I), \Delta^\delta, \Psi\rangle$, where $\Delta^\delta : A \times Z \to \wp(\mathsf{B}_C(A \times Z))$ is given by

$$\Delta^\delta(a, z) := \{(c, (a_0, \delta(z, a_0)), (a_1, \delta(z, a_1))) \in \mathsf{B}_C(A \times Z) \mid (c, a_0, a_1) \in \Delta(a)\},$$

and $\Psi : A \times Z \to \omega$ by

$$\Psi(a, z) := \Omega(z).$$

$\lhd$

Theorem 5.32 *Let $\mathbb{Z} = \langle Z, z_I, \delta, \Omega\rangle$ be a deterministic parity stream automaton, and let $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ be a nondeterministic tree automaton such that $Acc = L_\omega(\mathbb{Z})$. Then $\mathbb{A}$ and $\mathbb{A} \odot \mathbb{Z}$ are equivalent.*

▶ `Proof to be supplied`

Proof of Theorem 5.24

For a proof of Theorem 5.24, it suffices to combine the results of the previous two subsections.

▶ `Details to be supplied`

5.5 Notes

Tree automata, as finite devices operating on infinite trees, were introduced by Rabin in his seminal paper [27]. The concept of alternation, introduced by Chandra, Kozen & Stockmeyer [6], was brought to tree automata by Muller & Schupp [20]. The result that alternating automata can be simulated by nondeterministic ones is due to Muller & Schupp [21].

5.6 Exercises

Exercise 5.1 Consider the alphabet $C = \{b, g, y\}$.

(a) Construct a C-tree parity automaton $\mathbb{A}_1$ that accepts the class of pointed C-biflows $(\mathbb{S}, s)$ with the property that every path starting from s contains at most finitely many 'green' points (i.e., points coloured with 'g').

(b) Construct a C-tree Muller automaton $\mathbb{A}_2$ such that $\mathbb{A}_2$ accepts the class of pointed C-biflows $(\mathbb{S}, s)$ with the property that there exists a path starting from s such that every green node on this path has exactly one yellow son.

Exercise 5.2 (a) Let $\mathbb{A} = (A, \Delta, Acc, a_I)$ be a non-deterministic C-tree automaton for some finite alphabet C and let $(\mathbb{S}_1, s_1)$ and $(\mathbb{S}_2, s_2)$ be pointed C-biflows such that $(\mathbb{S}_1, s_1) \leftrightarrows (\mathbb{S}_2, s_2)$. Show that $\mathbb{A}$ accepts $(\mathbb{S}_1, s_1)$ iff $\mathbb{A}$ accepts $(\mathbb{S}_2, s_2)$.

(b) Let L_1 and L_2 be C-biflow languages and suppose that there are C-tree automata $\mathbb{A}_1$ and $\mathbb{A}_2$ such that $\mathbb{A}_i$ recognizes the language L_i for $i \in \{1, 2\}$. Show that there is a C-tree automata $\mathbb{A}_\cap$ such that $\mathbb{A}_\cap$ accepts $L_1 \cap L_2$.

6 Logic and Automata 1

▶ `introduction: logic and automata`

Throughout this chapter we will be dealing with a set P of proposition letters. Recall that the set $\mu\mathrm{MFL}_2(\mathsf{P})$ of modal fixpoint formulas for binary flows, over P, is given as follows:

$$\varphi \ ::= \ p \mid \neg p \mid \bot \mid \top \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc_i \varphi \mid \mu x.\varphi \mid \nu x.\varphi,$$

where $p, x \in \mathsf{P}$, and $i \in \{0, 1\}$, and all occurrences of x in $\mu x.\varphi$ and $\nu x.\varphi$ must be positive. In order to compare formulas of this language to tree automata, we will fix the set $C := \wp\mathsf{P}$ as the alphabet or colour set of the automata under consideration. Observe that with this definition, we may indeed identify C-biflows with Kripke models (based on biflows) for the language P.

▶ `overview of chapter`

6.1 Logical presentations of automata

In the previous chapter, we presented the choice of a player as a *set* of options. This set-theoretic presentation is fairly rigid in that it fixed the order and the role of the two players in the acceptance game — see our discussion in and preceding Convention 5.21. An alternative, *logic-based* approach, uses *formulas* to guide the dynamics of the acceptance games associated with the automaton. In particular, in order to represent players' choices, we employ lattice *connectives*: disjunctions for $\exists$, conjunctions for $\forall$. For instance, we might have chosen to define nondeterministic tree automata as structures with a transition function mapping a state a of the automaton to a *disjunction* of pairs in $\mathsf{B}_C A$. In this set-up we would represent a set $\Delta(a) = \{\alpha_1, \ldots, \alpha_k\} \subseteq \mathsf{B}_C A$ by the term $\bigvee_{1 \leq i \leq k} \alpha_i$. In case $\Delta(a) = \varnothing$ this would yield the term $\bigvee \varnothing = \bot$. To formalize this approach, we introduce the following syntax.

Definition 6.1 Given a set X, let *SLatt*(X) denote the set of finite disjunctions (semilattice terms) of elements of X:

$$\varphi ::= x \in X \mid \bigvee \Phi,$$

whereas *Latt*(X) denotes the set of all finite lattice terms of elements of X:

$$\varphi ::= x \in X \mid \bigvee \Phi \mid \bigwedge \Phi.$$

Here Φ denotes a finite set of semilattice terms (lattice terms, respectively). $\lhd$

Given this definition, we could now present alternating tree automata as structures $\mathbb{A} = \langle A, \Delta, Acc, q_I \rangle$, where the transition map is given as a function $\Delta : A \to \mathit{Latt}(\mathsf{B}_C A)$, and the acceptance game is given by the table below. Since it is not a priori known whether $\Delta(a)$ is a conjunction or a disjunction, we cannot assign a position of the form (a, s) to one of the players. This explains the 'automatic' move at basic positions.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$-$	$\{(\Delta(a), s)\}$
$(\bigvee \Phi, s) \in \mathit{Latt}(\mathsf{B}_C A) \times S$	$\exists$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
$(\bigwedge \Phi, s) \in \mathit{Latt}(\mathsf{B}_C A) \times S$	$\forall$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
$((c, a_0, a_1), s) \in \mathsf{B}_C A \times S$	$\exists$	$\{\{(a_0, \sigma_0(s)), (a_1, \sigma_1(s))\} \mid c = \sigma_C(s)\}$
$\{(a_0, s_0), (a_1, s_1)\} \subseteq A \times S$	$\forall$	$\{(a_0, s_0), (a_1, s_1)\}$

However, we may (and will) push the logical approach a bit further, and create even more symmetry between the two players by giving a 'logical deconstruction' of elements $(c, a_0, a_1) \in \mathsf{B}_C A$. Since $\exists$ loses a match at position $\big((c, a_0, a_1), s\big)$ if the color of s is distinct from c, and $\forall$ may choose a successor, a position of the form $((c, a_0, a_1), s)$ can be identified with a position $((c \wedge \bigcirc_0 a_0 \wedge \bigcirc_1 a_1), s)$, where the 'formulas' c, $\bigcirc_0 a_0$ and $\bigcirc_1 a_1$ get their obvious meaning (see Table 11 below). Given the connection between colors and sets of proposition letters, there is in fact no need to stop here: we may continue and 'deconstruct' any 'formula' c into its description in terms of proposition letters: $\bigwedge_{p \in c} p \wedge \bigwedge_{p \notin c} \neg p$.

In other words, there are many ways to define the notion of a 'logical automaton', and we will consider a number of these.

Definition 6.2 Given sets D and X, we let $\mathit{TLatt}(D, X) := \mathit{Latt}\big(D \cup \{\bigcirc_i x \mid x \in X\}\big)$ denote the set of *tree lattice terms over D and X*. Given the set P of proposition letters, we define $\pm\mathsf{P} := \{p, \neg p \mid p \in \mathsf{P}\}$. $\lhd$

Note that the set $\mathit{TLatt}(\pm\mathsf{P}, X)$ consists of modal formulas where elements from P occur at modal depth zero, and all elements of X occur only positively, and at modal depth one.

Definition 6.3 A *logical tree automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, Acc, q_I \rangle$ where q_I and Acc are as usual, and the transition function Δ has one of the following three shapes: $\Delta : A \to \mathit{Latt}(\mathsf{B}_C A)$, $\Delta : A \to \mathit{TLatt}(C, A)$, or $\Delta : A \to \mathit{TLatt}(\pm\mathsf{P}, A)$.

Given a logical C-tree automaton $\mathbb{A} = \langle A, q_I, \Delta, Acc \rangle$ and a C-flow $\mathbb{S} = \langle S, \sigma \rangle$, the *acceptance game* $\mathcal{A}(\mathbb{A}, \mathbb{S})$ is given by the rules of Table 11 (or the table given earlier in this section), together with the winning conditions that we saw for tree automata. A logical C-tree automaton $\mathbb{A} = \langle A, q_I, \Delta, Acc \rangle$ *accepts* a pointed binary C-flow $\mathbb{S} = \langle S, s_0, \sigma \rangle$, if the pair (q_I, s_0) is a winning position for $\exists$ in the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$. $\lhd$

Position	Player	Admissible moves
$(a, s) \in A \times S$	$-$	$\{(\Delta(a), s)\}$
$(\bigvee \Phi, s)$	$\exists$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
$(\bigwedge \Phi, s)$	$\forall$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
$(\bigcirc_i a, s)$	$-$	$\{(a, \sigma_i(s))\}$
$(c, s) \in C \times S, c \neq \sigma_C(s)$	$\exists$	$\varnothing$
$(c, s) \in C \times S, c = \sigma_C(s)$	$\forall$	$\varnothing$
$(p, s) \in \mathsf{P} \times S, p \in \sigma_C(s)$	$\forall$	$\varnothing$
$(p, s) \in \mathsf{P} \times S, p \notin \sigma_C(s)$	$\exists$	$\varnothing$
$(\neg p, s) \in \mathsf{P} \times S, p \in \sigma_C(s)$	$\exists$	$\varnothing$
$(\neg p, s) \in \mathsf{P} \times S, p \notin \sigma_C(s)$	$\forall$	$\varnothing$

Table 11: Acceptance game for logical tree automaton

As we will see now, the set-theoretic and three logical presentations of alternating tree automata are in fact equivalent, in the sense that there are effective transformations in all directions.

Proposition 6.4 *Fix a set* P *of proposition letters, and let* $C := \wp\mathsf{P}$*. Consider tree automata of the form* $\mathbb{A} = (A, a_I, \Delta, Acc)$*, where the transition map* Δ *has one of the following four formats:*
(1) $\Delta : A \to \wp_\exists\wp_\forall\mathsf{B}_C A$*,*
(2) $\Delta : A \to Latt(\mathsf{B}_C A)$*,*
(3) $\Delta : A \to TLatt(C, A)$*,*
(4) $\Delta : A \to TLatt(\pm\mathsf{P}, A)$*.*
Then there are effective transformations transforming an automaton of any one kind above to an equivalent automaton of any other kind.

From our discussion above it is easy to derive the transformations $1 \to 2 \to 3 \to 4$. For the transformations in the opposite direction, two new ideas are needed that we will formulate independently for future reference. First, in order to handle some of the atomic 'formulas' of logical automata, we add a new, so-called *true state* to the automaton, that is, a state from which $\exists$ has a guaranteed win.

Definition 6.5 A state a of an alternating automaton $\mathbb{A} = \langle A, q_I, \Delta, Acc\rangle$ is called a *true state* if $\Delta(a) = \{\varnothing\}$, and a *false state* if $\Delta(a) = \varnothing$. In case of a logical automaton, a *true state* has $\Delta(a) = \top$, and a *false* state has $\Delta(a) = \bot$. These states are usually denoted as $a_\top$ and $a_\bot$, respectively. $\lhd$

The second idea that we need to arrive at a set-theoretical representation of the automaton, is to bring this lattice term into a *distributive normal form.* The justification of this step is provided by the proposition below.

Proposition 6.6 *Let $\mathbb{A} = \langle A, q_I, \Delta, Acc\rangle$ and $\mathbb{A}' = \langle A, q_I, \Delta', Acc\rangle$ be two logical tree automata such that for all $a \in A$, $\Delta(a)$ and $\Delta'(a)$ are equivalent (as classical propositional formulas). Then $\mathbb{A}$ and $\mathbb{A}'$ are equivalent automata.*

▶ `Proof to be added`

Proof of Proposition 6.4. As mentioned, the transformations $1 \to 2 \to 3 \to 4$ can easily be defined on the basis of our discussion leading to Definition 6.3. We now discuss the opposite transformations.

$\boxed{4 \to 3}$ Given a logical automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ with $\Delta : A \to \mathit{TLatt}(\pm\mathsf{P}, A)$, replace, in each $\Delta(a)$, each occcurrence of p with the disjunction $\bigvee\{c \in C \mid p \in C\}$, and each occurrence of $\neg p$ with $\bigvee\{c \in C \mid p \notin C\}$. We leave it for the reader to verify that this transforms $\mathbb{A}$ into an equivalent automaton of the right kind.

$\boxed{3 \to 2}$ Starting from a logical automaton $\mathbb{A} = \langle A, a_I, \Delta, Acc\rangle$ with $\Delta : A \to \mathit{TLatt}(C, A)$, first add a true state $a_\top$ to A. The idea is then to replace, in each $\Delta(a)$ with $a \in A$, the atomic terms occurring in $\Delta(a)$, with the following expressions:

$$\begin{array}{lcl} c & \mapsto & c \wedge \bigcirc_0 a_\top \wedge \bigcirc_1 a_\top \\ \bigcirc_0 a_0 & \mapsto & \bigvee\{c \wedge \bigcirc_0 a_0 \wedge \bigcirc_1 a_\top \mid c \in C\} \\ \bigcirc_1 a_1 & \mapsto & \bigvee\{c \wedge \bigcirc_0 a_\top \wedge \bigcirc_1 a_1 \mid c \in C\} \end{array}$$

We leave it for the reader to verify that the resulting automaton $\mathbb{A}'$ is equivalent to $\mathbb{A}$. Finally, since for each $a' \in A'$, each atomic term in $\Delta'(a')$ occurs in a subformula of the form $c \wedge \bigcirc_0 a_0 \wedge \bigcirc_1 a_1$, we may easily transform $\mathbb{A}'$ into an equivalent automaton of type 2 above.

$\boxed{2 \to 1}$ As a corollary of Proposition 6.6, every logical alternating automaton $\mathbb{A} = \langle A, q_I, \Delta, Acc\rangle$ with $\Delta : A \to \mathit{Latt}(\mathsf{B}_C A)$ can be brought into distributive normal form, namely, by rewriting every $\Delta(a)$ as an equivalent disjunction of conjunctions of atomic terms. But such a disjunction of conjunctions can also be represented as a set of sets:

$$\bigvee_{i\in I}\bigwedge_{j\in J_i}\varphi_{i_j} \;\cong\; \Big\{\{\varphi_{i_j} \mid j \in J_i\} \mid i \in I\Big\}. \tag{30}$$

Replacing, for each state $a \in A$, the left hand side of (30) by the right hand side, we obtain an equivalent alternating tree automaton $\mathbb{A}'$. We leave the final details to the reader. QED

6.2 From formulas to automata

▶ `introductory remarks`
`(now focus on parity automata)`

Theorem 6.7 *There is an effective procedure that, given a binary modal flow formula ξ, returns an alternating parity tree automaton $\mathbb{A}_\xi$ that is equivalent to ξ.*

- `Corollary: decidability of logic`
- `finite model property of` $\mu\mathrm{MFL}_2$

By the results in the previous section it suffices to find an equivalent *logical* automaton for the formula ξ. In fact, if we are willing to stretch our definitions a little bit, allowing our automata to make so-called *silent moves*, then it is easy to find *some* kind of automaton for a formula. This first candidate will have the set of *subformulas* of ξ as its carrier set. The remainder of the section then consists in massaging this first candidate into the right shape of a logical tree automaton.

Definition 6.8 A *silent-step* tree automaton over P is an automaton $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$, with $\Delta : A \to \mathit{TLatt}(C \cup A, A)$ or $\Delta : A \to \mathit{TLatt}(\pm\mathsf{P} \cup A, A)$. The acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ associated with such an automaton $\mathbb{A}$ and a pointed binary flow $(\mathbb{S}, s)$ is determined by the rules given in Table 11. $\lhd$

The only difference with the logical automata of Definition 6.3 is that here, terms $\Delta(a)$ may contain *unguarded* occurrences of states in A, that is, occurrences of states $b \in A$ that are not in the scope of a modal operator. For instance, we may have $\Delta(a) = a$. In case $\Delta(a)$ indeed contains an unguarded occurrence of a state b, there may be rounds of the acceptance game in which the automaton changes state from a to b while play stays in the same position of the flow — this explains the name 'silent step'. We will come back to this issue after the next proposition, which states that any modal formula can be easily transformed into an equivalent silent-step automaton.

Proposition 6.9 *There is an effective procedure that, given a binary modal flow formula ξ, returns a silent-step parity automaton that is equivalent to ξ.*

Proof. The automaton $\mathbb{B}_\xi$ is directly based on the formula structure of ξ, that we may without loss of generality assume to be clean. As usual we let, for a bound variable x of ξ, $\eta_x x.\delta_x$ denote the unique subformula of ξ where x is bound.

For the states of $\mathbb{B}_\xi$ we could take the subformulas of ξ themselves, but the definition may be easier to understand if we make a formal distinction between formulas and states, by putting

$$B := \{\widehat{\varphi} \mid \varphi \in \mathit{Sfor}(\xi)\}.$$

The initial state b_I of $\mathbb{B}_\xi$ will clearly be the state $\widehat{\xi}$.

In order to define the transition function Δ we make a case distinction as to the kind of subformula that we are dealing with.

$$\begin{array}{lcl}
\Delta(\widehat{\varphi \vee \psi}) & := & \widehat{\varphi} \vee \widehat{\psi} \\
\Delta(\widehat{\varphi \wedge \psi}) & := & \widehat{\varphi} \wedge \widehat{\psi} \\
\Delta(\widehat{\bigcirc_i \varphi}) & := & \bigcirc_i \widehat{\varphi} \\
\Delta(\widehat{\eta x.\delta}) & := & \widehat{\delta} \\
\Delta(\widehat{x}) & := & \widehat{\delta_x},
\end{array}$$

whereas for the atomic formulas we take

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\Delta(\widehat{p}) & := & \bigvee\{c \in C \mid p \in c\} & \Delta(\widehat{\top}) & := & \top \\
\Delta(\widehat{\neg p}) & := & \bigvee\{c \in C \mid p \notin c\} & \Delta(\widehat{\bot}) & := & \bot.
\end{array}$$

We now turn to the parity function Ω. The only subformulas of which the parity will be of interest are the variables that may get unfolded in the acceptance game for ξ. That is, unless φ is a bound variable of ξ, we put $\Omega(\widehat{\varphi}) := 0$. This leaves the task of defining of $\Omega(\widehat{x})$ where $x \in BV(\xi)$. Recall that $\leq_\xi$ is the dependency order on these bound variables. It is in fact easy to define a function Ω that is compatible with this order, in the sense that

- $\Omega(\widehat{x})$ is odd if x is a μ-variable, and even if x is a ν-variable, and
- $\Omega(\widehat{x}) < \Omega(\widehat{y})$ if $x <_\xi y$.

The details of such a definition are left as an exercise to the reader.

With this definition it is easy to see that for any Kripke model $\mathbb{S}$, the acceptance game $\mathcal{A}$ for $\mathbb{B}_\xi$ and $\mathbb{S}$ on the one hand, and the evaluation game $\mathcal{E}$ for ξ and $\mathbb{S}$ on the other, are very similar. It is in fact not hard to prove that for any state s of $\mathbb{S}$, and for any subformula φ of ξ, $(\varphi, s) \in \mathrm{Win}_\exists(\mathcal{E})$ iff $(\widehat{\varphi}, s) \in \mathrm{Win}_\exists(\mathcal{A})$. QED

For many purposes it is no problem to work with silent-step autamata, but there are situations as well when we need to work with automata that are *guarded* in the sense that the terms $\Delta(a)$ only contain guarded occurrences of states of A. Fortunately, we may massage silent-step automata into the right guarded shape.

Proposition 6.10 *There is an effective procedure that, given a silent-step tree automaton, returns an equivalent logical automaton.*

Proof. The main idea of the proof is to use *semi-guarded* automata as an intermediate step. Formally, a silent-step tree automaton $\mathbb{A}$ is called *semiguarded* if $\Omega(b) > \Omega(a)$ whenever $b \lhd a$. Here we define the relation $\lhd \subseteq A \times A$ by putting $b \lhd a$ if $\Delta(a)$ contains an unguarded occurrence of b.

CLAIM 1 There is an effective procedure that, given a silent-step tree automaton, returns an equivalent semiguarded one.

PROOF OF CLAIM Fix the silent-step automaton $\mathbb{A} = (A, a_I, \Delta, \Omega)$, where Δ is of the form $\Delta : A \to TLatt(\pm\mathsf{P} \cup A, A)$.

Without loss of generality, we may assume that Ω is injective. By induction we will show that for all $i \geq -1$ we may find an automaton $\mathbb{A}_i = (A, a_I, \Delta_i, \Omega)$ which is equivalent to $\mathbb{A}$ and satisfies

$$\Omega(b) > \min(\Omega(a), i) \text{ for all } a \in A \text{ and for all } b \lhd a. \tag{31}$$

Clearly then the proposition is proved once i takes the value of the maximum parity of all states in A.

Since the base case of the induction ($i = -1$) is immediate by the definition of parity functions, we move on to the inductive case, for $i + 1$. By the inductive hypothesis we have that $\Omega(b) > \min(\Omega(a), i)$ for all $a \in A$ and for all $b \lhd a$. Now distinguish cases. If there is *no* $a \in A$ such that $\Omega(a) = i + 1$, we simply put $\mathbb{A}_{i+1} := \mathbb{A}_i$, and we leave it to the reader to prove that this $\mathbb{A}_{i+1}$ satisfies the required constraints.

Now suppose that, on the other hand, $i + 1$ *does* belong to the range of Ω. We only consider the case that $i + 1$ is odd — the case that it is even can be treated in a similar fashion. By our assumption on Ω there is in fact a *unique* state $b \in A$ such that $\Omega(b) = i + 1$. Define, for any natural number j, $A_j := \{a \in A \mid \Omega(a) \geq j\}$, then it easily follows from the induction hypothesis that $\Delta_i(b) \in TLatt(\pm\mathsf{P} \cup A_{i+1}, A)$. Due to the validity of the distributive laws in this context (see Proposition 6.6), without loss of generality we may assume that $\Delta_i(b)$ is of the form $(b \vee \delta_1) \wedge \delta_2$, where b does not appear in δ_1 or δ_2.

Let $\theta := \delta_1 \wedge \delta_2$, and let, for any $\varphi \in TLatt(\pm\mathsf{P} \cup A, A))$, $\varphi[\theta/c]$ denote the result of uniformly substituting θ for unguarded occurrences of b in the lattice term φ (guarded occurrences of b remain untouched under this operation). Now define Δ_{i+1} as follows:

$$\Delta_{i+1}(a) := \begin{cases} \Delta_i(a) & \text{if } \Omega(a) < i + 1, \\ \theta & \text{if } \Omega(a) = i + 1 \text{ (i.e., } a = b), \\ \Delta_i(a)[\theta/b] & \text{if } \Omega(a) > i + 1. \end{cases}$$

From the fact that $\Delta_i(b) \in \Delta_i(b) \in TLatt(\pm\mathsf{P} \cup A_{i+1}, A)$, and the assumption that b does not occur unguarded in δ_1 and δ_2, it follows that $\theta \in Latt(A_{i+2} \cup \Omega A)$. Using the induction hypothesis, it is straighforward to derive from this that Δ_{i+1} satisfies (31) for $i + 1$, whence $\mathbb{A}_{i+1}$ is at least of the right format.

It is thus left to prove that $\mathbb{A}_{i+1}$ is equivalent to $\mathbb{A}$, so by the induction hypothesis it suffices to show that $\mathbb{A}_{i+1}$ is equivalent to $\mathbb{A}_i$. Fix some biflow $\mathbb{S} = (S, \sigma)$, then we must show, for all points $s_0 \in S$, that

$$(a_I, s_0) \in \text{Win}_\exists(\mathcal{A}(\mathbb{A}_i, \mathbb{S})) \text{ iff } (a_I, s_0) \in \text{Win}_\exists(\mathcal{A}(\mathbb{A}_{i+1}, \mathbb{S})). \tag{32}$$

For the direction ($\Leftarrow$) of (32), let f be a history free winning strategy for $\exists$ in $\mathcal{A}_{i+1} := \mathcal{A}(\mathbb{A}_{i+1}, \mathbb{S})$. In order to define a strategy f' for her in $\mathcal{A}_i := \mathcal{A}(\mathbb{A}_i, \mathbb{S})$, suppose that play of $\mathcal{A}_i$ has arrived at a position (φ, s) with $\varphi \in \mathit{TLatt}(\pm\mathsf{P} \cup A, A)$. Let (a, s) be the last basic position that we passed in this match, and distinguish cases:

If $\Omega(a) \leq i+1$ and $\varphi \neq b \vee \delta_1$, then it is easy to see that (φ, s) must be a position of the game $\mathcal{A}_{i+1}$ as well, so that we may simply define $f'(\varphi, s) := f(\varphi, s)$. In this case we say that (φ, s) is its own corresponding position.

If $\Omega(a) = i+1$ and $\varphi = b \vee \delta_1$, then $\exists$ chooses $f'(b \vee \delta_1, s) := (\delta_1, s)$. In this case, (φ, s) has no corresponding position.

If $\Omega(a) > i+1$, then the pair $(\varphi[\theta/b], s)$ must be a position of $\mathcal{A}_{i+1}$; since we are dealing with positions for $\exists$, φ must be a disjunction. Now define $f'(\varphi, s) := (\psi[\theta/b], s)$, where ψ is the disjunct of φ given by $f(\varphi, s) = (\psi, s)$. Also, call $(\varphi[\theta/b], s)$ the corresponding position, in $\mathcal{A}_{i+1}$, of (φ, s).

Now consider an arbitrary match β of $\mathcal{A}_i$, starting from (a_I, s_0), and such that $\exists$ plays according to the strategy described above. It is easy to see that if we (i) replace every $\mathcal{A}_i$-position in β with its corresponding $\mathcal{A}_{i+1}$-position, (ii) erase all positions of the form $(b \vee \delta_1, s)$, and (iii) leave positions of the form $Z \subseteq A \times S$ untouched, then we obtain an f-conform match β' of $\mathcal{A}_{i+1}$. It follows by the assumption on f that β' is won by $\exists$.

Now let k be the highest parity occurring infinitely often in β. If $k < i+1$, then from a certain moment on, the matches β and β' are *identical*; clearly then, β is also won by $\exists$. If $k > i+1$, then it is not hard to see that k must also be the highest priority occurring infinitely often in β', so that again, $\exists$ is the winner of β. Now suppose for contradiction that $k = i+1$. Since $\exists$ never chooses b in a position of the form $(b \vee \delta_1, s)$, we may infer that positions of the form (b, s) occur infinitely often in β because they come about in a different way. Note as well that from a certain moment on, these positions are the ones with the highest parity. But then by definition this must apply to the match β' as well. From this it is easy to derive that $i+1$ is the highest parity occurring infinitely often in β'. This provides the desired contradiction with the assumption on f, and thus proves that $k \neq i+1$.

It follows that in all cases, $\exists$ wins β, and since β was arbitrary, we have proved that (a_I, s_0) is a winning position for $\exists$. This proves the direction ($\Leftarrow$) of (32); we omit the proof for the other direction, which is similar. ◀

CLAIM 2 There is an effective procedure that, given a semiguarded modal tree automaton, returns an equivalent guarded one.

PROOF OF CLAIM Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be a semiguarded modal tree automaton. For the definition of its guarded equivalent $\mathbb{A}'$, we need some preparations.

For each state $a \in A$, we will construct, in finitely many steps, a tree $T(a)$, together with a *labelling* and a (partial) *marking* of the nodes of the tree. To set up the construction, we start with the construction tree of the $TLatt(A)$-term $\Delta(a)$, seen as a lattice term. The inner nodes of this tree are *labelled* with a connective ($\vee$ or $\wedge$), and the leaves with a term of the form $\top$, $\bot$, p, $\neg p$, $\bigcirc_i b$ or b. Furthermore, the root of this tree is *marked* 'a', while all other nodes of the initial tree are marked with the empty list.

Now recursively, replace each leaf labelled $b \in A$ with its construction tree, and mark the leaf, which ahs become an inner node of the tree, with 'b'. Repeat the process until no leaves are left that are labelled with elements of A, and (thus) all leaves are labelled with terms of the form $\top$, $\bot$, p, $\neg p$, or $\bigcirc_i b$

It is not difficult to see that this process must terminate after finitely many steps. The key observation here is that for any two states $a, b \in A$, we find b as the label of some leaf of the construction tree of $\Delta(a)$ if and only if $b \lhd a$. And since $\mathbb{A}$ is semiguarded we have $\Omega(b) > \Omega(a)$ if $b \lhd a$. From this it follows that there are no infinite sequences $a_0 \rhd a_1 \rhd a_2 \rhd \ldots$, and so the algorithm must terminate.

We define $T(a)$ as the tree that is constructed by the algorithm that we just described. Clearly, $T(a)$ can be seen as the construction tree of some *guarded* $TLatt(A)$-term $\underline{\Delta}(a)$. Now consider a match of the game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ which has arrived at a basic position $(a_0, s) \in A \times S$. The key observation is that $T(a)$ represents the *static* stage of the continuation of this match, that is, the part that is played until the automaton moves to a successor of s. The point is that this part of the game is completely determined by the disjuncts and conjuncts chosen by $\exists$ and $\forall$, respectively. More specifically, a maximal path through $T(a)$ corresponds to a partial match of $\mathcal{A}(\mathbb{A}, \mathbb{S})@(a_0, s)$ that is *maximal* in the sense that it either ends in a win for one of the players, or else in a position of the form $(\bigcirc_i b, s)$, where $\bigcirc_i b$ is the label of the leaf corresponding to the last element of the maximal path through $T(a)$.

Now in principle, as the guarded equivalent of $\mathbb{A}$ we would like to take the structure $\underline{\mathbb{A}} := \langle A, a_I, \underline{\Delta}, \Omega \rangle$. Unfortunately, while it would not be hard to see that for any pointed binary flow $(\mathbb{S}, s)$, the *boards* of the two games $\mathcal{A}(\mathbb{A}, \mathbb{S})$ and $\mathcal{A}(\underline{\mathbb{A}}, \mathbb{S})$ are virtually identical (isomorphic modulo some automatic moves), they are rather different when we look at *parities*. The problem is that in a term $\underline{\Delta}(a)$ many original states of $\mathbb{A}$ are 'hidden', with the effect that their parities go unnoticed when playing the acceptance game for $\underline{\mathbb{A}}$ rather than for $\mathbb{A}$.

To take care of this, with each leaf l of the tree $T(a_0)$ associate, apart from its label $\bigcirc_{i_l} b_l$, also a unique sequence $a_n \lhd a_{n-1} \lhd \cdots \lhd a_0$, consisting of the marks encountered on the path leading from the root to the leaf l. Since the original automaton $\mathbb{A}$ is *semiguarded*, we find that $\Omega(a_n) > \Omega(a_{n-1}) > \cdots > \Omega(a_0)$. So $\Pi(a_0, l) := \Omega(a_n)$ is the *highest* parity of these a_i — corresponding to the highest parity encountered in the static partial match of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ from basic position (a_0, s) to the non-basic position $(\bigcirc_{i_l} b_l, s)$. The key idea in the definition of $\mathbb{A}'$ is to assign this number $\Pi(a_0, l)$ as priority to the state b_l — but then $\mathbb{A}'$ needs various *copies* of the element b. Formulated more precisely, we take a copy (a, n) of a for each n in the range $\Omega[A]$ of Ω. Thus we arrive at the following definition.

Given the semiguarded automaton $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$, we define the automaton $\mathbb{A}' = \langle A', a'_I, \Delta', \Omega'\rangle$ as follows:

$$\begin{array}{rcl} A' & := & A \times \Omega[A], \\ a'_I & := & (a_I, \Omega(a_I)), \\ \Omega'(a, n) & := & n, \end{array}$$

while $\Delta'(a, n)$ is the term $\underline{\Delta}(a)$ where, for each leaf l of $T(a)$ that has a label of the form $\bigcirc_i b$, this label is replaced with $\bigcirc_i(b, \Pi(a, l))$.

On the basis of the earlier given motivation for this definition, the reader should be able to prove that $\mathbb{A}$ and $\mathbb{A}'$ are indeed equivalent. Furthermore, it is obvious that $\mathbb{A}'$ is a guarded automaton. This suffices to prove the Proposition. ◀

The proof of Proposition 6.10 is immediate by the Claims 1 and 2. QED

Proof of Theorem 6.7. The proof of the Theorem now consists simply of glueing together the Propositions 6.9 and 6.10. QED

6.3 From automata to formulas

▶ `in other direction`

Theorem 6.11 *There is an effective procedure that, given an alternating tree automaton $\mathbb{A}$, returns a modal fixpoint formula $\xi_{\mathbb{A}}$ that is equivalent to $\mathbb{A}$.*

The key idea underlying the proof of this theorem is to view an automaton as a *system of equations*, of which the variables correspond to the states of the automaton. The proof of this theorem will be by induction on a notion of 'complexity' of the automaton which is called *index*.

Definition 6.12 Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$ be a logical automaton. The pair $\langle A, E_{\mathbb{A}}\rangle$ is called the *graph* of $\mathbb{A}$, where $E_{\mathbb{A}} \subseteq A \times A$ is defined by putting $E_{\mathbb{A}}ab$ holds if b *occurs* in $\Delta(a)$. The *index* of $\mathbb{A}$ is defined as follows:

$$ind(\mathbb{A}) := \begin{cases} -1 & \text{if } \langle A, E_{\mathbb{A}}\rangle \text{ has no cycles} \\ \max\{\Omega(c) \mid c \text{ lies on a cycle of } \langle A, E_{\mathbb{A}}\rangle\} & \text{otherwise,} \end{cases}$$

where SCC($\mathbb{A}$) denotes the set of strongly connected components of $\mathbb{A}$.

◁

Remark 6.13 In fact, the definition above is a simplified version of the proper definition of index, but it suffices for the present purposes. We will come back to this issue in a later version of these notes. $\triangleleft$

Proof of Theorem 6.11. For a proper formulation of the inductive hypothesis we introduce yet another type of automaton, the so-called (P, X)-automaton, which differs from automata over the set of variables $\mathsf{P}\cup X$ in that variables in X may occur within the scope of a modality, and only there. Formally, given sets P and X, a (P, X)-automaton is a structure $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$, where $\Delta : A \to \mathit{TLatt}(\pm\mathsf{P}, X \cup A)$. Note that (P, X)-automata operate on $\mathsf{P} \cup X$-biflow models, and that ordinary logical automata can be seen as $(\mathsf{P}, \varnothing)$-automata. The notion of acceptance for (P, X)-automata is defined as expected.

We extend the definition of *index* to (P, X)-automata in the obvious way, so that we may indeed use induction on the index in order to prove the following key claim.

Claim 1 There is an effective procedure that, given a (P, X)-automaton $\mathbb{A}$, returns a modal fixpoint formula $\xi_{\mathbb{A}}$ that is equivalent to $\mathbb{A}$, and in which all occurrences of variables in X are positive.

Proof of Claim As announced, we will prove the proposition by induction on the index $m := \mathit{ind}(\mathbb{A})$ of $\mathbb{A}$.

In the base case of the induction, where $\mathit{ind}(\mathbb{A}) = -1$, we are dealing with an automaton without any strongly connected components. In this case we can easily obtain the formula $\xi_{\mathbb{A}}$ as a *basic* modal formula (that is, no fixpoints involved), as follows. For $a \in \mathbb{A}$, let $\mathbb{A}_a$ be the automaton $\langle A, a, \Delta, \Omega\rangle$, i.e., the same as $\mathbb{A}$, but with a as its starting state. Define the *height* $h(a)$ of a state a as the length of the longest $E_{\mathbb{A}}$-path starting in a. Clearly every state has a finite height, since the automaton has no SCCs. Hence we may use a subinduction on the height of states to define a map ξ that associates, with each state a, a basic modal formula ξ_a which is equivalent to $\mathbb{A}_a$. In the base case of this subinduction, we are dealing with states of height zero. Then, by definition, in the term $\Delta(a)$ of such a state a there are no occurrences of states. In other words, $\Delta(a)$ is a basic modal logic formula and so we may put $\xi_a := \Delta(a)$. The equivalence of $\mathbb{A}_a$ and ξ_a is immediate. For the inductive case we consider a state a with $h(a) > 0$. Every b occurring in $\Delta(a)$ has lesser height than a, and so by the inductive hypothesis for each such b there is a formula ξ_b equivalent to the automaton $\mathbb{A}_b$. We define ξ_a as the formula we obtain from $\Delta(a)$ by substituting each b with ξ_b. We leave it as an (easy) exercise for the reader to check that indeed, the resulting formula is equivalent to the automaton $\mathbb{A}_a$.

Now we turn to the inductive case, where $\mathit{ind}(\mathbb{A}) \geq 0$. Let $M \subseteq A$ be the set of states that belong to some strongly connected component of $\mathbb{A}$, and actually have parity $\mathit{ind}(\mathbb{A})$. Then M is nonempty, say $M = \{a_1, \ldots, a_k\}$. Without loss of generality we may assume that the initial state a_I of $\mathbb{A}$ does not belong to M.

The main idea of the proof consists of removing the elements of M as *states* from $\mathbb{A}$, but at the same time adding them as *variables*. Formally, every term $\Delta(a) \in \mathit{TLatt}(\mathsf{P}, X \cup A)$ is also a well-formed $\mathit{MLatt}(\mathsf{P}, (X \cup M) \cup (A \setminus M))$-term, so that the structure

$$\mathbb{A}_M := \langle A \setminus M, a_I, \Delta\restriction_{A\setminus M}, \Omega\restriction_{A\setminus M} \rangle$$

is a $(\mathsf{P}, X \cup M)$-automaton. Furthermore, for each $i \in \{1, \dots, k\}$, we will consider the automaton $\mathbb{A}_i$, which is like the automaton $\mathbb{A}_M$, but with a copy of a_i added as starting state. (Note that we cannot take a_i itself as the starting state since we need a_i as a variable.) Formally, put $\mathbb{A}_i := \langle A_i, a_i, \Delta i, \Omega i\rangle$, where A_i is the set $(A \setminus M) \cup \{\bar{a}_i\}$; Δ_i is given by putting $\Delta_i(a) := \Delta(a)$ for $a \neq \bar{a}_i$, and $\Delta_i(\bar{a}_i) := \Delta(a_i)$; for Ω_i, we put $\Omega_i(a) := \Omega(a)$ for $a \neq \bar{a}_i$, and $\Omega_i(\bar{a}_i) := 0$.

The inductive hypothesis applies to each of these automata. Thus we obtain fixed point formulas $\varphi_M, \varphi_1, \dots, \varphi_k$, all taking free variables from the set $\mathsf{P} \cup X \cup M$, and such that for any $\mathsf{P} \cup X \cup M$-model $\mathbb{S}$ and any point s in $\mathbb{S}$, we have that $\mathbb{A}_M$ accepts $(\mathbb{S}, s)$ iff $\mathbb{S}, s \Vdash \varphi_M$, and, for each i, $\mathbb{A}_i$ accepts $(\mathbb{S}, s)$ iff $\mathbb{S}, s \Vdash \varphi_i$.

Clearly then, for any $\mathsf{P} \cup X$-model $\mathbb{S}$, the k-tuple $\overline{\varphi}$ determines a monotone map $[\![\overline{\varphi}]\!]_{\mathbb{S}} : (\mathcal{P}(S))^k \to (\mathcal{P}(S))^k$ given by

$$[\![\overline{\varphi}]\!]_{\mathbb{S},V}(T_1, \dots, T_k) := ([\![\varphi_1]\!]_{\mathbb{S}[\overline{a}\mapsto\overline{T}]}, \dots, [\![\varphi_1]\!]_{\mathbb{S}[\overline{a}\mapsto\overline{T}]}).$$

Here $\mathbb{S}[\overline{a} \mapsto \overline{T}]$ denotes the variant of $\mathbb{S}$ with $\mathsf{P} \cup X \cup M$-valuation $V[\overline{a} \mapsto \overline{T}]$ given by

$$V[\overline{a} \mapsto \overline{T}](x) := \begin{cases} T_i & \text{if } x = a_i \in M, \\ V(x) & \text{if } x \in \mathsf{P} \cup X, \end{cases}$$

where V is the valuation of $\mathbb{S}$.

It follows from standard fixed point theory (cf. the discussion following Proposition 3.9, of the Gaussian elimination method), that the least and greatest fixed points of this map are given by μML-formulas. More precisely, there are formulas $\varphi^\mu_1, \dots, \varphi^\mu_k$ and $\varphi^\nu_1, \dots, \varphi^\nu_k$, all with free variables in $\mathsf{P} \cup X$, such that

$$([\![\varphi^\mu_1]\!]^{\mathbb{S}}, \dots, [\![\varphi^\mu_k]\!]^{\mathbb{S}}) \text{ is the } \textit{least} \text{ fixed point of } [\![\overline{\varphi}]\!]^{\mathbb{S}}$$

for every $\mathsf{P} \cup X$-model $\mathbb{S}$, and likewise for the greatest fixed point. Now define $\xi_{\mathbb{A}}$ as the formula

$$\xi_{\mathbb{A}} := \varphi_M[\overline{\varphi}^\eta/\overline{a}].$$

That is, we uniformly replace, in φ_M, each a_i with the formula φ^η_i, where η denotes μ if $\mathit{ind}(\mathbb{A})$ is odd, and ν if $\mathit{ind}(\mathbb{A})$ is even.

In order to show that $\xi_{\mathbb{A}}$ is indeed equivalent to the automaton $\mathbb{A}$, we need the following auxiliary result.

$$\mathbb{A}_{a_i} \text{ is equivalent to } \varphi^\eta_i, \text{ for each } i. \tag{33}$$

Here $\mathbb{A}_{a_i}$ is the (P, X)-automaton $\mathbb{A}$, but with a_i as its starting state.

The proof of (33) is fairly similar to the proof of the Adequacy Theorem, whence we omit further details for the moment.

► `Further details to be supplied`

Finally, in order to derive the equivalence of $\xi_{\mathbb{A}}$ and $\mathbb{A}$, observe that it follows from (33) and the inductive hypothesis on $\mathbb{A}_M$ that, for any $\mathsf{P} \cup X$-model $\mathbb{S}$, and any state s in $\mathbb{S}$:

$$\mathbb{S}, s \Vdash \xi_{\mathbb{A}} \text{ iff } \mathbb{A}_M \text{ accepts } (\mathbb{S}', s),$$

where $\mathbb{S}'$ is the $\mathsf{P} \cup X \cup M$-model which agrees with $\mathbb{S}$ on all variables in $\mathsf{P} \cup X$, and makes the variable a_i true at those states t such that $\mathbb{A}_{a_i}$ accepts $(\mathbb{S}, t)$. Hence it suffices to prove that

$$\mathbb{A} \text{ accepts } (\mathbb{S}, s) \text{ iff } \mathbb{A}_M \text{ accepts } (\mathbb{S}', s). \tag{34}$$

But this is in fact an easy exercise — further proof details are left to the reader. ◄

Since ordinary logical automata are $(\mathsf{P}, \varnothing)$-automata, Theorem 6.11 is an immediate consequence of the Claim. QED

6.4 Closure properties

The class of recognizable tree languages has some nice closure properties These have important applications, as we will see for instance in the next section, but they are of interest in their own right too. Hence we devote this (fairly brief) subsection to discussing and proving some of the more interesting closure properties.

Recall that a tree language L is called *recognizable* if there is a nondeterministic tree automaton $\mathbb{A}$ such that $L = L_t(\mathbb{A})$, and that the same applies to biflow languages. By Theorem 5.24, in order to prove that some language is recognizable, it suffices to find an *alternating* automaton for it. Throughout this section we will freely move from one presentation of automata to another.

Closure under union and intersection

We first turn to the Boolean operations of union and intersection. In the framework of alternating tree automata it is in fact not hard to show that the recognizable languages are closed under these operations.

Proposition 6.14 *Given two alternating parity tree automata $\mathbb{A}_1$ and $\mathbb{A}_2$ we can effectively construct alternating parity tree automata $\mathbb{A}_\cup$ and $\mathbb{A}_\cap$ such that $L(\mathbb{A}_\cup) = L(\mathbb{A}_1) \cup L(\mathbb{A}_2)$ and $L(\mathbb{A}_\cap) = L(\mathbb{A}_1) \cap L(\mathbb{A}_2)$. Moreover $\mathbb{A}_\cup$ is nondeterministic if $\mathbb{A}_1$ and $\mathbb{A}_2$ are so.*

Before we prove the proposition we define the automata $\mathbb{A}_\cup$ and $\mathbb{A}_\cap$.

Definition 6.15 Let $\mathbb{A}_1 = \langle A_1, a^1_I, \Delta_1, \Omega_1\rangle$ and $\mathbb{A}_2 = \langle A_2, a^2_I, \Delta_2, \Omega_2\rangle$ be two tree automata. We will define their *sum* $\mathbb{A}_\cup$ and *product* $\mathbb{A}_\cap$.

Both of these automata will have the *disjoint union* $A_{12} := \{*\} \uplus A_1 \uplus A_2$ as their collection of states. Also, the parity function Ω will be the same for both automata:

$$\Omega(a) := \begin{cases} 0 & \text{if } a = *, \\ \Omega_i(a) & \text{if } a \in A_i. \end{cases}$$

The only difference between the automata lies in the transition functions, which are defined as follows:

$$\Delta_\cup(a) := \begin{cases} \Delta_1(a^1_I) \cup \Delta_2(a^2_I) & \text{if } a = * \\ \Delta_i(a) & \text{if } a \in A_i, \end{cases}$$

$$\Delta_\cap(a) := \begin{cases} \{\Phi_1 \cup \Phi_2 \mid \Phi_i \in \Delta_i(q^i_I)\} & \text{if } a = * \\ \Delta_i(a) & \text{if } a \in A_i. \end{cases}$$

Finally, we put $\mathbb{A}_\cup := \langle A_{12}, a_I, \Delta_\cup, \Omega\rangle$ and $\mathbb{A}_\cap := \langle A_{12}, a_I, \Delta_\cap, \Omega\rangle$. $\lhd$

Let us now turn to the proof of Proposition 6.14.

Proof. The automata $\mathbb{A}_\cup$ and $\mathbb{A}_\cap$ are constructed as defined above in Definition 6.15. It is easy to see that $\mathbb{A}_\cup$ is nondeterministic if $\mathbb{A}_1$ and $\mathbb{A}_2$ are nondeterministic automata. We only show that

$$L(\mathbb{A}_\cup) = L(\mathbb{A}_1) \cup L(\mathbb{A}_2),$$

the other statements of the proposition admit similarly straightforward proofs. It suffice to show, for an arbitrary pointed biflow $(\mathbb{S}, s)$, that $\mathbb{A}_\cup$ accepts $(\mathbb{S}, s)$ iff $\mathbb{A}_1$ or $\mathbb{A}_2$ accepts $(\mathbb{S}, s)$.

First suppose that the automaton $\mathbb{A}_\cup$ accepts $(\mathbb{S}, s)$. Hence by definition, $\exists$ has a winning strategy f in the game $\mathcal{A} := \mathcal{A}(\mathbb{A}_\cup, \mathbb{S})$ starting from position $(*, s)$. Let i be such that $f(s, *) \in \Delta(q^i_I)$. It is then straightforward to verify that f, restricted to $\exists$'s positions in $\mathcal{A}(\mathbb{A}_i, \mathbb{S})$, is a winning strategy for $\exists$ from position (q^i_I, s). From this it is immediate that $\mathbb{A}_i$ accepts $(\mathbb{S}, s)$.

Conversely, suppose that $\mathbb{A}_i$ accepts $(\mathbb{S}, s)$, and let g be a winning strategy for $\exists$ in the game $\mathcal{A}(\mathbb{A}_i, \mathbb{S})$. Then in the game $\mathcal{A}(\mathbb{A}_\cup, \mathbb{S})$ starting at $(*, s)$, let $\exists$ start with playing $g(q^i_I, s) \in \Delta_\cup(*)$, and from then on, play her strategy g. It is again straightforward to check that this constitutes a winning strategy for $\exists$. QED

Closure under complementation

Closure under complementation is not easy to prove in the setting of nondeterministic automata, but once we turn to the framework of logical automata, where there is a strong symmetry between the two players, there is a rather intuitive construction based on a *role switch* between the two players.

Proposition 6.16 *Given an alternating parity tree automata $\mathbb{A}$ we can effectively construct an alternating parity tree automaton $\widetilde{\mathbb{A}}$ which recognizes the complement of $L(\mathbb{A})$.*

In the remainder we will assume that the alphabet C is of the form $\wp\mathsf{P}$ for some set P of proposition letters. See Remark 6.19 for the general case.

Definition 6.17 Given an element $\varphi \in \mathit{TLatt}(\mathsf{P}, X)$, let $\widetilde{\varphi}$ be the term obtained by simultaneously replacing all symbols $\bigwedge$ by $\bigvee$, p by $\neg p$, and vice versa. Given a logical parity automaton $\mathbb{A} = \langle A, q_I, \Delta, \Omega\rangle$ with $\Delta : A \to \mathit{TLatt}(\mathsf{P}, X)$, define the *complement of* $\mathbb{A}$ as the automaton $\widetilde{\mathbb{A}} := \langle A, q_I, \widetilde{\Delta}, \widetilde{\Omega}\rangle$, where $\widetilde{\Delta}$ is given by $\widetilde{\Delta}(a) := \widetilde{\Delta(a)}$, and $\widetilde{\Omega}$ by putting $\widetilde{\Omega}(a) := \Omega(a) + 1$, for all $a \in A$. $\lhd$

Proposition 6.18 *Let $(\mathbb{S}, s)$ be a pointed C-biflow. Then $(\mathbb{S}, s)$ is accepted by $\mathbb{A}$ iff it is rejected by $\widetilde{\mathbb{A}}$.*

Proof. The key observation underlying the proof (and the definition of $\widetilde{\mathbb{A}}$) is that, for any C-biflow $\mathbb{S}$, the two players 'switch roles' in the acceptance game $\mathcal{A}(\widetilde{\mathbb{A}}, \mathbb{S})$ as compared to $\mathcal{A}(\mathbb{A}, \mathbb{S})$. More precisely, consider the function

$$f : (\varphi, s) \mapsto (\widetilde{\varphi}, s)$$

which maps positions of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ to positions in $\mathcal{A}(\widetilde{\mathbb{A}}, \mathbb{S})$. A straightforward inspection shows that f is in fact an *isomorphism* between the game graphs of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ and $\mathcal{A}(\widetilde{\mathbb{A}}, \mathbb{S})$ — but with the twist that positions for $\exists$ are mapped to those of $\forall$, and vice versa. In addition, it follows from the definition of the parity maps, that any infinite match of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ is won by $\exists$ iff the (obviously defined) image of the match under f is won by $\forall$.

An immediate consequence of these observations is that for any position (φ, s) of $\mathcal{A}(\mathbb{A}, \mathbb{S})$ we have

$$(\varphi, s) \in \mathrm{Win}_{\exists}(\mathcal{A}(\mathbb{A}, \mathbb{S})) \text{ iff } (\widetilde{\varphi}, s) \in \mathrm{Win}_{\forall}(\mathcal{A}(\widetilde{\mathbb{A}}, \mathbb{S})).$$

In particular, we see that $(\mathbb{S}, s)$ is accepted by $\mathbb{A}$ iff it is rejected by $\widetilde{\mathbb{A}}$. QED

Remark 6.19 In case C is just an arbitrary set of colors, not necessarily linked to a set P of proposition letters, one can do complementation as follows. Without loss of generality (see Proposition 6.4) assume that Δ is a map from A to $TLatt(C, A)$. Now given a term $\Delta(a)\, TLatt(C, X)$, obtain $\widetilde{\Delta}(a)$ by swapping $\bigwedge$ with $\bigvee$ and replacing each occurrence of c with the term $\bigvee\{c' \in C \mid c' \neq c\}$. Apart from this, the definition of $\widetilde{\mathbb{A}}$ is as in Definition 6.17. $\lhd$

▶ `Easier in the case of stream automata`

▶ `size matters`

Closure under projection

An interesting operation on tree languages, corresponding to second-order quantification, involves *projection*. For its definition, assume that the color set C is really a cartesian product $C = C_1 \times C_2$. As a specific example, think of $C = \wp\mathsf{P}$ as a P-fold cartesian *power* of the set 2: $C = 2 \times 2 \cdots \times 2 = 2^{\mathsf{P}}$.

Definition 6.20 Assume that $C = C_1 \times C_2$, with associated projections $\pi_i : C \to C_i$. Then with any C-biflow $\mathbb{S} = (S, \sigma)$ we can naturally associate a C_i-projection $\mathbb{S}^{\pi_i} = \langle S, \sigma^{\pi_i}\rangle$, for each i. Here the transition map σ^{π_i} is given as $\sigma^{\pi_i}(s) := (\pi_i\sigma_C(s), \sigma_0(s), \sigma_1(s))$.

Given a C-biflow language L, we define its *C_i-projection* $L{\restriction}C_i$ as the class of C_i-projections of structures in L. $\lhd$

An obvious question is whether the class of recognizable biflow languages is closed under taking such projections. The answer to this question is *no*, as an easy example shows.

▶ `example to be supplied`

However, if we confine our attention to tree languages, then a positive result applies. Of course, the question here only makes sense once we have checked that the projection of a tree language is indeed a tree language, but that is almost immediate from the definitions. Contrary to the case of closure under complementation, here we need the automata to be *nondeterministic* rather than alternating.

Proposition 6.21 *Let $C = C_1 \times C_2$ be a set of colors. Given a nondeterministic parity C-tree automaton $\mathbb{A}$, for each i we can effectively construct an nondeterministic parity tree automaton $\mathbb{A}{\restriction}C_i$ which recognizes the i-th projection $L_t(\mathbb{A}){\restriction}C_i$ of the tree language $L_t(\mathbb{A})$.*

Definition 6.22 Let $C = C_1 \times C_2$ be a set of colors. Given a nondeterministic parity C-automaton $\mathbb{A} = \langle A, q_I, \Delta, \Omega\rangle$ with $\Delta : A \to \wp_\exists(\mathsf{B}_C A)$, define the C_i-automaton $\mathbb{A}{\restriction}C_i$ as the structure $\mathbb{A}{\restriction}C_i := \langle A, q_I, \Delta', \Omega\rangle$, where $\Delta' : A \to \wp_\exists(\mathsf{B}_{C_1} A)$ is given by

$$\Delta'(a) := \{(\pi_i(c), a_0, a_1) \in \mathsf{B}_{C_1} A \mid (c, a_0, a_1) \in \Delta(a)\}$$

for all $a \in A$. $\lhd$

Proof of Proposition 6.21. In this proof we will denote C-trees simply as maps $\tau : 2^* \to C$. Note that the C_i-projection of such a biflow can simply be denoted as the tree $\pi_i \circ \tau$. We need to show that for any C_i-tree ρ we have

$$\mathbb{A}{\restriction}C_i \text{ accepts } \rho \text{ iff } \rho = \pi_i \circ \tau \text{ for some } C\text{-tree } \tau \text{ accepted by } \mathbb{A}. \tag{35}$$

The direction from right to left is straightforward.

▶ `Proof details for the other direction to be supplied`

QED

Remark 6.23 For arbitrary biflows we can prove the following, for an arbitrary C_i-biflow $(\mathbb{S}, s)$:

$$\mathbb{A}{\restriction}C_i \text{ accepts } (\mathbb{S}, s) \text{ iff } \mathbb{A} \text{ accepts some } C\text{-biflow } (\mathbb{Q}, q) \text{ with } (\mathbb{Q}^{\pi_i}, q) \leftrightarroweq (\mathbb{S}, s).$$

Because of this we say that $L(\mathbb{A}{\restriction}C_i) = L(\mathbb{A}){\restriction}C_i$ *modulo bisimulation.* $\lhd$

6.5 Rabin's Theorem

▶ `intro: decidability of S2S (Rabin's Theorem)`

▶ `Seminal decidability result to which many others may be reduced`

Let us start with introducing and discussing the syntax and semantics of monadic second-order logic. It will be convenient for us to work with the following version of monadic second order logic in which *all* variables are set variables.

Definition 6.24 Given a set P of set variables, we define the language of *monadic second-order logic* S2S as follows:

$$\varphi \ ::= \ p \sqsubseteq q \ \mid \ S_i pq \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \exists p.\varphi$$

Here p and q are variables from P, and $i \in \{0, 1\}$. $\lhd$

Definition 6.25 Given a binary flow model $\mathbb{S} = \langle S, V, \sigma_0, \sigma_1 \rangle$, we define the semantics of S2S as follows:

$$\begin{array}{lll}
\mathbb{S} \models p \sqsubseteq q & \text{if} & V(p) \subseteq V(q) \\
\mathbb{S} \models S_i pq & \text{if} & \text{for all } s \in S:\ s \in V(p) \text{ implies } \sigma_i(s) \in V(q) \\
\mathbb{S} \models \neg\varphi & \text{if} & \mathbb{S} \not\models \varphi \\
\mathbb{S} \models \varphi \vee \psi & \text{if} & \mathbb{S} \models \varphi \text{ or } \mathbb{S} \models \psi \\
\mathbb{S} \models \exists p.\varphi & \text{if} & \mathbb{S}[p \mapsto X] \models \varphi \text{ for some } X \subseteq S.
\end{array}$$

An S2S-formula is *satisfiable* in a binary flow model $\mathbb{S}$ if $\mathbb{S} \models \varphi$. $\lhd$

Remark 6.26 Rather than Definition 6.24, the reader may have expected a language allowing *both* first- *and* second-order quantification. For instance, given a set X of individual variables and a set P of set variables, we define the language of *monadic second-order logic* MSOL as follows:

$$\varphi \ ::= \ R_i xy \ \mid \ p(x) \ \mid \ \neg\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \exists x.\varphi \ \mid \ \exists p.\varphi$$

Here x and y are variables from X, p is a variable from P, and R_i, for $i = 0, 1$ is the predicate denoting the i-th accessibility relation. This semantics of this language is completely standard, with $\exists x$ denoting first-order quantification (that is, quantification over individual states), and $\exists p$ denoting second-order quantification (that is, quantification over sets of states).

It is not too hard to see that the two languages are equivalent in a sense that we will not make precise. The key point is that S2S can *interpret* MSOL, by encoding individual variables as set variables denoting *singletons*. $\lhd$

Theorem 6.27 (Rabin) *It is decidable whether a given* S2S-*formula is satisfiable in a binary tree.*

The idea underlying the proof of Rabin's Theorem is that with each monadic second-order formula ξ we may associate an alternating automaton that is equivalent to ξ over the class of binary trees.

Proposition 6.28 *Let φ be some* S2S-*formula. Then for any set P of propositional variables containing the free variables of φ there is a nondeterministic or alternating automaton $\mathbb{B}_\varphi$ over the alphabet $\wp\mathsf{P}$, such that for any binary tree model $\mathbb{S}$ for P:*

$$\mathbb{S} \models \varphi \textit{ iff } \mathbb{B}_\varphi \textit{ accepts } (\mathbb{S}, \epsilon). \tag{36}$$

Proof. We prove (36) by induction on the complexity of φ. Throughout the proof we let C denote the set $\wp\mathsf{P}$, and the automata we consider will be of transition type $B \to \wp_\exists \mathsf{B}_C B$, or $B \to \wp_\exists \wp_\forall \mathsf{B}_C B$.

Regarding the base case, we only consider the case of the atomic formula $p \sqsubseteq q$. We define $\mathbb{B}_{p \sqsubseteq q}$ as the structure $\langle B, b_I, \Delta, \Omega \rangle$, where B contains only the state b_I, $\Delta : B \to \mathsf{B}_C B$ is given by $\Delta(b_I) := \{(c, b_I, b_I) \mid p \notin c \text{ or } q \in c\}$, and $\Omega(b_I) := 0$. In other words, $\exists$ wins any match in which she can stay alive. But as long as she does not encounter states where p is true and q is false, she can continue her match. This shows that $\mathbb{B}_{p \sqsubseteq q}$ indeed satisfies (36).

The proof of the inductive case is based on the closure properties of recognizable languages that we discussed in the previous section. We only consider the case where φ is the formula $\neg\psi$. Let P be a set of propositional variables containing the free variables of φ. Then obviously, P also contains all free variables of ψ, so that we may apply the inductive hypothesis to ψ. This provides us with a tree automaton $\mathbb{B}_\psi$ satisfying

$$\mathbb{S} \models \psi \text{ iff } \mathbb{B}_\psi \text{ accepts } (\mathbb{S}, \epsilon). \tag{37}$$

Now define $\mathbb{B}_\varphi$ as the *complement automaton* $\widetilde{\mathbb{B}}_\psi$ given by Definition 6.17. It immediately follows from Proposition 6.18 and (37) that $\mathbb{B}_\varphi$ has the required properties for (36). QED

Proof of Theorem 6.27. The proof of Rabin's Theorem is an immediate consequence of Proposition 6.28 and the decidability of the emptiness problem for tree automata. QED

Notes

The equivalence between tree automata and the modal μ-calculus (interpreted on trees) was established by Niwiński [22]. Rabin's landmark decidability result was published as [27].

Part IV
Games

7 Board games

Much of the work linking (fixpoint) logic to automata theory involves nontrivial concepts and results from the theory of infinite games. In this chapter we discuss some of the highlights of this theory in a fair amount of detail. This allows us to be rather informal about game-theoretic concepts in the rest of the notes.

7.1 Board games

The games that we are dealing with here can be classified as *board* or *graph games*. They are played by two agents, here to be called 0 and 1.

Definition 7.1 If $\sigma \in \{0, 1\}$ is a player, then $\bar{\sigma}$ denotes the *opponent* $1 - \sigma$ of σ. $\triangleleft$

A board game is played on a *board* or *arena*, which is nothing but a directed graph in which each node is marked with either 0 or 1. A *match* of the game consists of the two players moving a pebble or token across the board, following the edges of the graph. To regulate this, the collection of graph nodes, usually referred to as *positions* of the game, is partitioned into two sets, one for each player. Thus with each position we may associate a unique player whose turn it is to move when the token lies on position p.

Definition 7.2 A *board* is a structure $\mathbb{B} = \langle B_0, B_1, E\rangle$, such that B_0 and B_1 are disjunct, and $E \subseteq B^2$, where $B := B_0 \cup B_1$. We will make use of the notation $E[p]$ for the set of *admissible moves* from a board position $p \in B$, that is, $E[p] := \{q \in B \mid (p, q) \in E\}$. Positions not in $E[p]$ will sometimes be referred to as *illegitimate moves* with respect to p. A position $p \in B$ is a *dead end* if $E(p) = \varnothing$. If $p \in B$, we let P_p denote the (unique) player such that $p \in B_{P_p}$, and say that p *belongs to* P_p, or that it is P_p's *turn* to move at p. $\triangleleft$

A match of the game may in fact be identified with the sequence of positions visited during play, and thus corresponds to a *path* through the graph.

Definition 7.3 A *path* through a board $\mathbb{B} = \langle B_0, B_1, E\rangle$ is a (finite or infinite) sequence $\pi \in B^\infty$ such that $E\pi_i\pi_{i+1}$ whenever applicable. A *full* or *complete match* through $\mathbb{B}$ is either an infinite $\mathbb{B}$-path, or a finite $\mathbb{B}$-path π ending with a dead end (i.e. $E[\mathit{last}(\pi)] = \varnothing$).

A *partial match* is a finite path through $\mathbb{B}$ that is not a match; in other words, the last position of a partial match is not a dead end. We let PM_σ denote the set of partial matches such that σ is the player whose turn it is to move at the last position of the match. In the sequel, we will denote this player as P_π; that is, $P_\pi := P_{\mathit{last}(\pi)}$. $\triangleleft$

Each full or completed match is *won* by one of the players, and *lost* by their opponent; that is, there are no draws. A finite match ends if one of the players gets *stuck*, that is, is forced to move the token from a position without successors. Such a finite, completed, match is lost by the player who got stuck.

The importance of this explains the definition of the notion of a *subboard*. Note that any set of positions on a board naturally induces a board of its own, based on the restricted edge relation. We will only call this structure a subboard, however, if there is no disagreement between the two boards when it comes to players being stuck or not.

Definition 7.4 Given a board $\mathbb{B} = \langle B_0, B_1, E \rangle$, a subset $A \subseteq B$ determines the following board $\mathbb{B}_A := \langle A \cap B_0, A \cap B_1, E_{\upharpoonright A} \rangle$. This structure is called a *subboard* of $\mathbb{B}$ if for all $p \in A$ it holds that $E[p] \cap A = \varnothing$ iff $E_{\upharpoonright A}[p] \cap A = \varnothing$. ◁

If neither player ever gets stuck, an infinite match arises. The flavor of a board game is very much determined by the winning conditions of these infinite matches.

Definition 7.5 Given a board $\mathbb{B}$, a *winning condition* is a map $W : B^\omega \to \{0, 1\}$. An infinite match π is *won* by $W(\pi)$. A *board game* is a structure $\mathcal{G} = \langle B_0, B_1, E, W \rangle$ such that $\langle B_0, B_1, E \rangle$ is a board, and W is a winning condition on B. ◁

Although the winning condition given above applies to all infinite B-sequences, it will only make sense when applied to matches. We have chosen the above definition because it is usually much easier to formulate maps that are defined on all sequences.

Before players can actually start playing a game, they need a starting position. The following definition introduces some terminology and notation.

Definition 7.6 An *initialized board game* is a pair consisting of a board game $\mathcal{G}$ and a position q on the board of the game; such a pair is usually denoted $\mathcal{G}@q$.

Given a (partial) match π, its first element $\mathit{first}(\pi)$ is called the *starting position* of the match. We let $\mathrm{PM}_\sigma(q)$ denote the set of partial matches for σ that start at position q. ◁

Central in the theory of games is the notion of a *strategy*. Roughly, a strategy for a player is a method that the player uses to decide how to continue partial matches when it is their turn to move. More precisely, a strategy is a function mapping partial plays for the player to new positions, with the proviso that the new position must make a legitimate move.

Definition 7.7 Given a board game $\mathcal{G} = \langle B_0, B_1, E, W \rangle$ and a player σ, a *σ-strategy*, or a *strategy for* σ, is a map $f : \mathrm{PM}_\sigma \to B$ such that $E(\mathit{last}(\pi), f(\pi))$. In case we are dealing with an initialized game $\mathcal{G}@q$, then we may take a strategy to be a map $f : \mathrm{PM}_\sigma(q) \to B$ (satisfying the legitimacy condition).

A match π is *consistent* with a σ-strategy f if for any $\pi' \sqsubset \pi$ with $last(\pi') \in B_\sigma$, the next position on π (after π') is indeed the element $f(\pi')$.

A σ-strategy f is *winning for* σ if σ wins every play that is consistent with f. A position $q \in B$ is *winning for* σ if σ has a winning strategy for the game $\mathcal{G}$ starting at q; the collection of winning positions for σ in $\mathcal{G}@q$ is denoted as $\mathrm{Win}(\mathcal{G}@q)$. $\lhd$

Convention 7.8 In practice, when defining strategies, it will often be convenient to extend the definition of a strategy to include maps f that do not necessarily satisfy the condition that $E(last(\pi), f(\pi))$ for every partial play π. In such a case we will say that the map prescribes an *illegitimate* move in the partial play π. We will only permit ourselves such a sloppiness in a context where in fact the partial play π is not consistent with the pseudo-strategy f, and thus the situation where the pseudo-strategy would actually ask for an illegitimate move will not occur.

Definition 7.9 The game $\mathcal{G}$ on the board $\mathbb{B}$ is *determined* if $\mathrm{Win}_0(\mathcal{G}) \cup \mathrm{Win}_1(\mathcal{G}) = B$; that is, each position is winning for one of the players. $\lhd$

In principle, when deciding how to move in a match of a board game, players may use information about the entire history of the match played thus far. However, it will turn out to be advantageous to work with strategies that are simple to compute. This applies for instance to so-called finite-memory strategies, which can be computed using only a finite amount of information about the history of the match.

▶ `discuss finite-memory strategy`

Particularly nice are so-called *memory-free* or *history-free* strategies, which only depend on the current position (i.e., the final position of the partial play). These will be critically needed in the proofs of some of the most fundamental results in the area of logic and automata theory, such as the Theorems 8.11 and 9.6.

Definition 7.10 A strategy f is *memory-free* if $f(\pi) = f(\pi')$ for any π, π' with $last(\pi) = last(\pi')$. $\lhd$

7.2 Winning conditions

In case we are dealing with a *finite* board B, then we may nicely formulate winning conditions in terms of the set of positions that occur *infinitely often* in a given match. But in the case of an infinite board, there may be matches in which no position occurs infinitely often (or more than once, for that matter). Nevertheless, we may still define winning conditions in terms of objects that occur infinitely often, if we make use of *finite colorings* of the board. If we assign to each position $b \in B$ a *color*, taken from a finite set C of colors, then we may formulate winning conditions in terms of the *colors* that occur infinitely often in the match.

Definition 7.11 A *coloring* of B is a function Γ assigning to each position $p \in B$ a *color* $\Gamma(b)$ taken from some finite set C of colors. Such a coloring $\Gamma : B \to C$ naturally extends to a map $\Gamma : B^\omega \to C^\omega$ by putting $\Gamma(p_0 p_1 \dots) := \Gamma(p_0)\Gamma(p_1)\dots$. $\lhd$

Now if $\Gamma : B \to C$ is a coloring, for any infinite sequence $\pi \in B^\omega$, the map $\Gamma \circ \pi \in C^\omega$ forms the associated sequence of colors. But then since C is finite there must be some elements of C that occur infinitely often in this stream.

Definition 7.12 Let $\mathbb{B}$ be a board and $\Gamma : B \to C$ a coloring of B. Given an infinite sequence $\pi \in B^\omega$, we let $\mathit{Inf}_\Gamma(\pi)$ denote the set of colors that occur infinitely often in the sequence $\Gamma \circ \pi$.

A *Muller condition* is a collection $\mathcal{M} \subseteq \wp(C)$ of subsets of C. The corresponding winning condition is defined as the following map $W_\mathcal{M} : B^\omega \to \{0,1\}$:

$$W_\mathcal{M}(\pi) := \begin{cases} 0 & \text{if } \mathit{Inf}_\Gamma(\pi) \in \mathcal{M} \\ 1 & \text{otherwise.} \end{cases}$$

A *Muller game* is a board game of which the winning conditions are specified by a Muller condition. $\lhd$

In words, player 0 wins an infinite match $\pi = p_0 p_1 \dots$ if the set of colors one meets infinitely often on this path, belongs to the Muller collection $\mathcal{M}$.

▶ `Examples to be supplied.`

Muller games have two nice properties. First, they are determined. This follows from a well-known general game-theoretic result, but can also be proved directly. In addition, we may assume that the winning strategies of each player in a Muller game are finite-memory strategies.

▶ `Details to be supplied`

The latter property becomes even nicer if the Muller condition allows a formulation in terms of a *parity map*. In this case, as colors we take natural numbers. Note that by definition of a coloring, the range $\Omega[B]$ of the coloring function Ω is finite. This means that every subset of $\Omega[B]$ has a maximal element. Hence, every match determines a unique natural number, namely, the 'maximal color' that one meets infinitely often during the match. Now a parity winning condition states that the winner of an infinite match is 0 if this number is even, and 1 if it is odd. More succinctly, we formulate the following definition.

Definition 7.13 Let B be some set; a *parity map* on B is a coloring $\Omega : B \to \omega$, that is, a map of finite range. A *parity game* is a board game $\mathcal{G} = \langle B_0, B_1, E, W_\Omega \rangle$ in which the winning condition is given by

$$W_\Omega(\pi) \; := \; \max(\mathit{Inf}_\Omega(\pi)) \mod 2.$$

Such a parity game is usually denoted as $\mathcal{G} = \langle B_0, B_1, E, \Omega \rangle$. $\lhd$

The key property that makes parity games so interesting is that they enjoy memory-free determinacy. We will prove this in section 7.4. First we turn to a special case, viz., the reachability games.

7.3 Reachability Games

Reachability games are a special kind of board games. They are played on a board such as described in section 7.1, but now we also choose a subset $A \subseteq B$. The aim of the game is for the one player to move the pebble into A and for the other to avoid this to happen.

Definition 7.14 Fix a board $\mathbb{B}$ and a subset $A \subseteq B$. The reachability game $\mathcal{R}_\sigma(\mathbb{B}, A)$ is then defined as the game over $\mathbb{B}$ in which σ wins as soon as a position in A is reached or if $\bar{\sigma}$ gets stuck. On the other hand, $\bar{\sigma}$ wins if he can manage to keep the token outside of A infinitely long, or if σ gets stuck. $\lhd$

Remark 7.15 If we want to fit reachability games exactly in the format of a board game, we have to do the following. Given a reachability game $\mathcal{R}_\sigma(\mathbb{B}, A)$, define the board $\mathbb{B}' := \langle B_0, B_1, E' \rangle$ by putting:

$$E' := \{(p, q) \in E \mid p \notin B_{\bar{\sigma}} \cap A\}.$$

In other words, E' is like E except that player $\bar{\sigma}$ gets stuck in a position belonging to A. Furthermore, the winning conditions of such a game are very simple: simply define $W : B^\omega \to \{0, 1\}$ as the constant function mapping all infinite matches to $\bar{\sigma}$. This can be formulated as a parity condition: put $\Omega(p) := \bar{\sigma} \mod 2$ for every $p \in B$. $\lhd$

Since reachability games can thus be formulated as very simple parity games, the following theorem can be seen as a warming up exercise for the general case.

Theorem 7.16 *Reachability games enjoy Memory-Free Determinacy.*

Proof. We leave this proof as an exercise to the reader. QED

Definition 7.17 The winning region for σ in $\mathcal{R}_\sigma(\mathbb{B}, A)$ is called the *attractor set* of σ for A in $\mathbb{B}$, notation: $Attr_\sigma^{\mathbb{B}}(A)$. In the sequel we will fix a positional winning strategy for σ in $\mathcal{R}_\sigma(\mathbb{B}, A)$ and denote it as $attr_\sigma^{\mathbb{B}}(A)$. $\lhd$

Note that σ-attractor sets always contain all points from which σ can make sure that $\bar{\sigma}$ gets stuck. Furthermore, it is easy to see that in $attr_\sigma(A)$-conform matches the pebble never leaves $Attr_\sigma(A)$ (at least if the match starts inside $Attr_\sigma(A)$!).

Proposition 7.18 *$Attr_\sigma$ is a closure operation on $\mathcal{P}(B)$, i.e.*

1. *$A \subseteq A'$ implies $Attr_\sigma(A) \subseteq Attr_\sigma(A')$,*
2. *$A \subseteq Attr_\sigma(A)$,*
3. *$Attr_\sigma(Attr_\sigma(A)) = Attr_\sigma(A)$.*

A kind of counterpart to attractor sets are σ-traps. In words, a set A is a σ-trap if σ can't get the pebble out of A, while her opponent has the power to keep it inside A.

Definition 7.19 Given a board $\mathbb{B}$, we call a subset $A \subseteq B$ a *σ-trap* if $E[b] \subseteq A$ for all $b \in A \cap B_\sigma$, while $E[b] \cap A \neq \varnothing$ for all $b \in A \cap B_{\bar{\sigma}}$ ◁

Note that a σ-trap does not contain $\bar{\sigma}$-endpoints and that $\bar{\sigma}$ will therefore never get stuck in a σ-trap. We conclude this section with a useful proposition.

Proposition 7.20 *Let $\mathbb{B}$ be a board and $A \subseteq B$ an arbitrary subset of B. Then the following assertions hold.*

1. *If A is a σ-trap then A is a subboard of B.*
2. *The union $\bigcup\{A_i \mid i \in I\}$ of an arbitrary collection of σ-traps is again a σ-trap.*
3. *If A is a σ-trap then so is $Attr_{\bar{\sigma}}(A)$.*
4. *The complement of $Attr_\sigma(A)$ is a σ-trap.*
5. *If A is a σ-trap in $\mathbb{B}$ then any $C \subseteq A$ is a σ-trap in $\mathbb{B}$ iff C is a σ-trap in $\mathbb{B}_A$.*

Proof. All statements are easily verified and thus the proof is left to the reader. QED

7.4 Memory-Free Determinacy of Parity Games

Theorem 7.21 (Memory-Free Determinacy of Parity Games) *For any parity game $\mathcal{G}$ there are positional strategies f_0 and f_1 for 0 and 1, respectively, such that for every position q there is a player σ such that f_σ is a winning strategy for σ in $\mathcal{G}@q$.*

We start with the definition of players' paradises. In words, a subset $A \subseteq B$ is a σ-paradise if σ has a positional strategy f guarantees her both that she wins the game, and that the token stays in A.

Definition 7.22 Given a parity game $\mathbb{G}(\mathbb{B}, \Omega)$, we call a $\bar{\sigma}$-trap A a *σ-paradise* if there exists a positional winning strategy $f : A \cap B_\sigma \to A$. ◁

The following proposition establishes some basic facts about paradises.

Proposition 7.23 *Let $\mathbb{G}(\mathbb{B}, \Omega)$ be a parity game. Then the following assertions hold:*

1. *The union $\bigcup\{P_i \mid i \in I\}$ of an arbitrary set of σ-paradises is again a σ-paradise.*

2. *There exists a largest σ-paradise.*

3. *If P is a σ-paradise then so is $Attr_\sigma(P)$.*

Proof. The main point of the proof of part (1) is that we somehow have to uniformly choose a strategy on the intersection of paradises, such that we will end up following the strategy of only one paradise. For this purpose, we assume that we have a well-ordering on the index set I (i.e., for the general case we assume the Axiom of Choice).

For the details, assume that $\{P_i \mid i \in I\}$ is a family of paradises, and let f_i be the positional winning strategy for P_i. Note that $P := \bigcup\{P_i \mid i \in I\}$ is a trap for $\bar{\sigma}$ by Proposition 7.20. Assume that $<$ is a well-ordering of I, so that for each $q \in P$ there is a *minimal* index $\min(q)$ such that $p \in P_{\min(q)}$. Define a positional strategy on P by putting

$$f(q) := f_{\min(q)}(q).$$

This strategy ensures at all times that the pebble either stays in the current paradise, or else it moves to a paradise of lower index, and so, any match where σ plays according to f will proceed through a sequence of σ-paradises of decreasing index. Because of the well-ordering, this decreasing sequence of paradises cannot be strictly decreasing, and thus we know that after finitely many steps the pebble will remain in the paradise where it is, say, P_j. From that moment on, the match is continued as an f_j-conform match inside P_j, and since f_j is by assumption a winning strategy when played inside P_j, this match is won by σ.

Part (2) of the proposition should now be obvious: clearly the union of all σ-paradises is the greatest σ-paradise.

In order to proof part (3) we need to show that there exists a winning strategy for σ. The principal idea is to first move to P by $attr_\sigma(P)$ and once there to follow the winning strategy in P. Let f' be the winning strategy for P, we then define the following strategy f on $Attr_\sigma(P)$ by

$$f(p) := \begin{cases} f'(p) & \text{if } p \in P \\ attr_\sigma(p) & \text{otherwise.} \end{cases}$$

A match consistent with this strategy will stay in $Attr_\sigma(P)$ because it is a $\bar{\sigma}$-trap and $f(p) \in Attr_\sigma(P)$ for all $p \in Attr_\sigma(P)$. It is winning because if ever the match arrives at a point $p \in P$ then play continues as if the match were completely in P and since f' was supposed to be winning strategy for σ this play is won by σ. However if we start outside P we will at first follow the strategy $attr_\sigma(P)$ which will ensure that σ either wins or that the pebble ends up in P, in which case σ will also win. QED

Now we are ready to proof the main assertion from which Theorem 7.21 immediately follows.

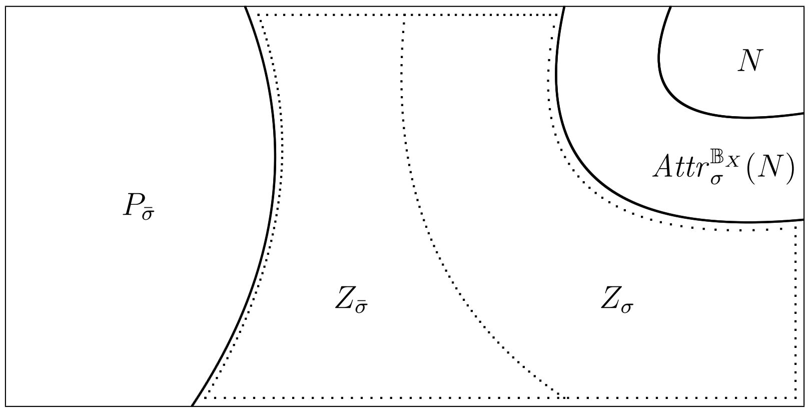
Proposition 7.24 *The board of a parity game $\mathbb{G}(\mathbb{B}, \Omega)$ can be partitioned into a 0-paradise and a 1-paradise.*

Proof. We will prove this proposition by induction on n, the maximal parity in the game (i.e. $n = \max(\Omega[B])$). If $n = 0$ we are dealing with a reachability game (namely $\mathcal{R}_1(\mathbb{B}, \varnothing)$), and from the results in section 7.3 we may derive that $Attr_1(\varnothing)$ is a 1-paradise and its complement is a 0-paradise. So the proposition holds in case $n = 0$.

Therefore in the remainder we can assume that $n \geq 1$. Let $\sigma := n \mod 2$ be the player that wins an infinite play π if $\max(Inf(\pi)) = \max(\Omega[B]) = n$. Let $P_{\bar{\sigma}}$ be the maximal $\bar{\sigma}$-paradise with associated positional strategy f. It now suffices to show that $X := B \setminus P_{\bar{\sigma}}$ is a σ-paradise.

First we shall show that X is a $\bar{\sigma}$-trap. By proposition 7.23(3) it follows that $Attr_{\bar{\sigma}}(P_{\bar{\sigma}})$ is itself also a $\bar{\sigma}$-paradise. By maximality of $P_{\bar{\sigma}}$ and the fact that $Attr_{\bar{\sigma}}$ is a closure operation, it follows that $P_{\bar{\sigma}} = Attr_{\bar{\sigma}}(P_{\bar{\sigma}})$. Thus by Proposition 7.20(4) we see that X, being the complement of a $\bar{\sigma}$-attractor set is a $\bar{\sigma}$-trap.

Consider $\mathbb{G}_X$, the subgame of $\mathbb{G}$ restricted to X. Note that by proposition 7.20(1), X is a subboard of $\mathbb{B}$, so the name 'subgame' is justified. Define $N := \{b \in X \mid \Omega(b) = n\}$ to be the set of all points in X with priority n and let $Z := X \setminus Attr_{\sigma}^{\mathbb{B}_X}(N)$. Since Z is the complement of a σ-attractor set in $\mathbb{B}_X$ it is a σ-trap in $\mathbb{B}_X$ and hence a σ-trap of $\mathbb{B}_X$, and so, a subboard of $\mathbb{B}$.



By the induction hypothesis we can split the subgame $\mathbb{G}_Z$ into a 0-paradise Z_0 and a 1-paradise Z_1, see the picture. The winning strategies in these paradises we call f_0 and f_1 respectively. (All notions are with regard to the game $\mathbb{G}_Z$.) We want to show that $Z_{\bar{\sigma}} = \varnothing$, so that $Z = Z_{\sigma}$.

To this aim, we claim that $P_{\bar{\sigma}} \cup Z_{\bar{\sigma}}$ is a $\bar{\sigma}$-paradise in $\mathbb{G}$, and in order to prove this, we consider the following strategy g of $\bar{\sigma}$:

$$g(b) := \begin{cases} f(b) & \text{if } b \in P_{\bar{\sigma}} \\ f_{\bar{\sigma}}(b) & \text{if } b \in Z_{\bar{\sigma}}. \end{cases}$$

It is left as an exercise for the reader to show that this is indeed a winning strategy for $\bar{\sigma}$ which keeps the pebble inside $P_{\bar{\sigma}} \cup Z_{\bar{\sigma}}$. (Here we need the fact that $\mathbb{B}_Z$ is a subboard of $\mathbb{B}$ — if this were not the case, then we could not rule out the existence of positions that are dead ends for σ in $\mathbb{B}_Z$, but not in $\mathbb{B}$.) By maximality of $P_{\bar{\sigma}}$ we see that $P_{\bar{\sigma}} = P_{\bar{\sigma}} \cup Z_{\bar{\sigma}}$ and since $P_{\bar{\sigma}}$ and $Z_{\bar{\sigma}}$ are disjoint we conclude that $Z_{\bar{\sigma}}$ is empty indeed.

This means we can write

$$X = Z_\sigma \cup Attr_\sigma^{\mathbb{B}_X}(N).$$

We are now almost ready to define the winning strategy for σ which keeps the token inside X. Recall that X is a $\bar{\sigma}$-trap, so that for each $b \in X \cap B_\sigma$, we may pick an element $k(b) \in E[b] \cap X$. Now define the following strategy h in $\mathbb{G}$ for σ on X.

$$h(b) := \begin{cases} k(b) & \text{if } b \in N \\ attr_\sigma(N)(b) & \text{if } b \in Attr_\sigma^{\mathbb{B}_X}(N) \setminus N \\ f_\sigma(b) & \text{if } b \in Z_\sigma = Z. \end{cases}$$

It is left as an exercise for the reader to show that h is indeed a winning strategy for σ in $\mathbb{G}$ and that it keeps the pebble in X. QED

Finally, the assertion made in Theorem 7.21 follows directly from this proposition because by definition of paradises there now exists for every point $b \in B$ a positional winning strategy for the game $\mathbb{G}(\mathbb{B}, \Omega)$.

Notes

The application of game-theoretic methods in the area of logic and automata theory goes back to work of Büchi. The memory-free determinacy of parity games was proved independently by Emerson & Jutla [9] and Mostowski in an unpublished technical report. Our proof of this result is based on Zielonka [33].

Part V
Transition systems

8 Graph automata

In this chapter we introduce and discuss the automata that we shall use for studying the modal μ-calculus. These *graph* automata all operate on the same type of structures, namely pointed Kripke models, or transition systems. Nevertheless, they come in a large variety of shapes, so it is good to observe that roughly speaking, every graph automaton belongs to either of the following two kinds.

1. *Modal automata* represent fairly straightforward generalizations of modal fixpoint formulas;

2. *Kripke automata* closely resemble the pointed Kripke structures on which they are supposed to operate.

Of these two kinds, the reader may at first sight find the modal automata more intuitive and easy to understand — this is also the reason why they are introduced first. However, working in the second format makes it easier to prove results about the modal μ-calculus. In particular, a key result, here given as Theorem 8.11, states that every alternating Kripke automaton can be effectively transformed into a nondeterministic equivalent. As we will see further on, many crucial results concerning the modal μ-calculus, such as its decidability and small model property, are direct consequences of this fundamental theorem.

Convention 8.1 Throughout this chapter we define automata that are supposed to accept or reject pointed Kripke models. In each case, such an automaton is of the form $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ where A is a set of states, $a_I \in A$ is the initial state, Δ is some kind of transition function on A, and $\Omega : A \to \omega$ is a parity condition.

Also in each case, the question whether such an automaton accepts or rejects a given pointed Kripke model $(\mathbb{S}, s)$ is determined by playing some kind of *acceptance game*. This game will always proceeds in *rounds*, from one basic position $(a, s) \in A \times S$ via some intermediate position(s) to a new basic position. The rules of this game are determined by the precise shape of the transition function Δ, and in each case will be given explicitly. However, the winning conditions are fixed. Finite matches, as always, are lost by the player who got stuck. The winner of an infinite match β is always determined by the infinite sequence $(a_I, s)(a_1, s_1)(a_2, s_2)\ldots$ of basic positions occurring in β, using the parity condition Ω on the sequence $a_I a_1 a_2 \ldots$ The definition of acceptance is also fixed: the automaton $\mathbb{A}$ *accepts* the pointed Kripke model $(\mathbb{S}, s)$ precisely if the pair (a_I, s) is a winning position for $\exists$ in the acceptance game.

Finally, throughout the chapter we will be working with a completely standard notion of equivalence between formulas and automata: We say that a modal μ-calculus formula ξ is *equivalent* to some automaton $\mathbb{A}$ operating on Kripke models if

$$\mathbb{S}, s \Vdash \xi \text{ iff } \mathbb{A} \text{ accepts } (\mathbb{S}, s)$$

for every pointed Kripke model $(\mathbb{S}, s)$.

Definition 8.2 Let $\mathbb{A}$ be some kind of automaton for pointed Kripke models. The class of pointed Kripke models that are accepted by a given automaton $\mathbb{A}$ is denoted as $\mathsf{L}(\mathbb{A})$. $\lhd$

Unless explicitly specified otherwise, throughout this chapter we work with a fixed set P of propositional variables, and a fixed set D of atomic actions. For notational convenience, we will usually abbreviate the associated Kripke functor (cf. Definition 1.4) $\mathsf{K}_{\mathsf{D},\mathsf{P}}$ by K, and as before we will think of the set $C := \wp\mathsf{P}$ as an alphabet or set of colors.

8.1 Modal automata

Formulas and automata are very much alike. As any reader going through the chapters 2 and 4 will have observed, there are many resemblances between the evaluation game of a modal (fixpoint) formula and the acceptance game of an automaton. States of the automata seem to have their counterpart in the *bound* variables of the formula, with greatest and least fixpoint operators corresponding to even and odd parity, respectively. In the case of tree automata, we have already made these resemblances explicit in Chapter 6, and we will now do something very similar for the case of arbitrary (not necessarily deterministic) transition systems.

Of course, an important difference with the case of deterministic models (i.e., the flows and biflows studied in the previous chapters) is that there, once the atomic action has been chosen, the next position of the game is completely determined. In the case of nondeterministic accessibility relations that we are dealing with here, successor states in Kripke models are not unique. The game reflects this by allowing one of the players to *choose* the next point in the Kripke model. But there is in fact no reason why we could not incorporate this kind of interaction in the definition of automata for Kripke models.

Definition 8.3 Given sets D (of atomic actions), C, and A, the collection $MLatt_{\mathsf{D}}(C, A)$ of *(poly-)modal lattice terms over C and A* is defined as the set $Latt(C \cup \{\Diamond_d a, \Box_d a \mid d \in \mathsf{D}, a \in A\})$. $\lhd$

In other words, the set $MLatt_{\mathsf{D}}(C, A)$ can be given as follows:

$$\varphi ::= c \mid \Diamond_d a \mid \Box_d a \mid \bigvee \Phi \mid \bigwedge \Phi$$

Here c, a and d refer to arbitrary elements of C, A, and D, respectively, and Φ denotes a finite set of modal lattice terms. Often we will consider modal lattice terms in which proposition letters and their negations may occur. The set of these terms is denoted by $MLatt_{\mathsf{D}}(\pm\mathsf{P}, A)$.

Position	Player	Admissible next moves
$(a, s) \in A \times S$	$-$	$\{(\Delta(a), s)\}$
$(\bigvee \Phi, s)$	$\exists$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
$(\bigwedge \Phi, s)$	$\forall$	$\{(\varphi, s) \mid \varphi \in \Phi\}$
(c, s), with $c = \sigma_C(s)$	$\forall$	$\varnothing$
(c, s), with $c \neq \sigma_C(s)$	$\exists$	$\varnothing$
(p, s), with $p \in \mathsf{P}$ and $s \in V(p)$	$\forall$	$\varnothing$
(p, s), with $p \in \mathsf{P}$ and $s \notin V(p)$	$\exists$	$\varnothing$
$(\neg p, s)$, with $p \in \mathsf{P}$ and $s \in V(p)$	$\exists$	$\varnothing$
$(\neg p, s)$, with $p \in \mathsf{P}$ and $s \notin V(p)$	$\forall$	$\varnothing$
$(\Diamond_d a, s)$	$\exists$	$\{(a, t) \mid t \in \sigma_d(s)\}$
$(\Box_d a, s)$	$\forall$	$\{(a, t) \mid t \in \sigma_d(s)\}$

Table 12: Acceptance game for modal automata

We are now ready for the definition of modal automata. As before, we allow some variation concerning the transition structure of these devices, and we also introduce the *silent-step* version allowing *unguarded* occurrences of states of the automaton in lattice terms.

Definition 8.4 A D-*modal automaton over* P is an automaton $\mathbb{A} = \langle A, \Delta, \Omega, a_I \rangle$ such that $\Delta : A \to \mathit{MLatt}_{\mathsf{D}}(C, A)$ or $\Delta : A \to \mathit{MLatt}_{\mathsf{D}}(\pm\mathsf{P}, A)$. A *silent-step* automaton is an automaton of which the transition map is of the form $\Delta : A \to \mathit{MLatt}_{\mathsf{D}}(C \cup A, A)$.

The acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ associated with such an automaton $\mathbb{A}$ and a pointed Kripke model $(\mathbb{S}, s)$ is determined by the rules given in Table 12 (given Convention 8.1). $\lhd$

Modal automata *generalize* the modal fixpoint formulas of Chapter 2. The difference between formulas and automata is that the latter allow more freedom in structure: while formulas by definition are required to have a tree structure (with back edges representing the unfolding relation between a fixpoint variable and its unfolding formula), for automata we accept structures that allow cyclicity. Nevertheless, there are effective procedures transforming fixpoint formulas into equivalent modal automata, and vice versa.

Proposition 8.5 *There are effective procedures that:*

1. *given a modal fixpoint formula $\xi \in \mu\mathrm{PML}(\mathsf{D}, \mathsf{P})$, return an equivalent D-modal automaton $\mathbb{A}_\xi$ over P;*

2. *given a D-modal automaton $\mathbb{A}$ over P, return an equivalent modal fixpoint formula $\xi_{\mathbb{A}} \in \mu\mathrm{PML}(\mathsf{D}, \mathsf{P})$.*

Proof sketch. In the same way as in Proposition 6.9, it is easy to show that a modal fixpoint formula can be turned into a silent-step automaton by taking the set of its subformulas as the collection of states. And similar to Proposition 6.10, we may effectively bring any silent-step automaton into a guarded normal form which then corresponds to a modal automaton.

Conversely, the procedure to transform an automaton into an equivalent modal fixpoint formula is exactly as described in the proof of Theorem 6.11. QED

8.2 Kripke automata

We have now arrived at the introduction of the second kind of automata for Kripke models: the ones that are similar to the Kripke models rather than to standard modal fixpoint formulas. These automata can be introduced in the same way as the nondeterministic or alternating stream and tree automata earlier on, namely, starting from the notion of a bisimulation. In this section we will work from the (coalgebraic) perspective on bisimulation, based on the notion of *relation lifting*, cf. the discussion following Definition 1.26.

So consider, for two Kripke models $\mathbb{A} = \langle A, \alpha \rangle$ and $\mathbb{S} := \langle S, \sigma \rangle$, the bisimulation game $\mathcal{B}(\mathbb{A}, \mathbb{S})$ of Definition 1.24. The main conceptual step is to think of $\mathbb{A}$ as a 'proto-automaton' that we use to *classify* $\mathbb{S}$ rather than as of a Kripke model that we are comparing with $\mathbb{S}$. In order to turn $\mathbb{A}$ into a proper Kripke automaton, four technical modifications have to be made:

1. A small change is that we require $\mathbb{A}$ (i.e., its carrier set A) to be finite — but in fact, it would be perfectly acceptable to allow for infinite automata as well.

2. Second, and equally undramatic, we add an initial state to the structure of $\mathbb{A}$.

3. Third, whereas the winner of an infinite match of a bisimulation game is always $\exists$, the winner of an infinite acceptance match will be determined by an explicit acceptance condition on A^ω — a parity condition, in our case.

4. The fourth and foremost modification is that we introduce *nondeterminism*, and even *alternation* to the transition structure of $\mathbb{A}$. Just as stream and tree automata, Kripke automata will harbour many 'realizations' of models — and in each round of the acceptance game, the actual local realization of a state is determined by the interaction of the two players.

Our presentation of alternating Kripke automata is set-theoretic in nature, and uses the $\wp_\exists/\wp_\forall$ notation of Convention 5.21.

Definition 8.6 Given a Kripke functor $\mathsf{K} = \mathsf{K}_{\mathsf{D},\mathsf{P}}$, an *(alternating) Kripke automaton* is a quadruple $\mathbb{A} = \langle A, \Delta, \Omega, a_I \rangle$ such that the transition function Δ is given as a map

$\Delta : A \to \wp_\exists\wp_\forall(\mathsf{K}A)$. Such a Kripke automaton is called *nondeterministic* if, for every $a \in A$, $|\Gamma| \leq 1$ for all $\Gamma \in \Delta(a)$ (that is, $\Delta(a)$ only contains singletons and possibly the empty set).

The *acceptance game* $\mathcal{A}(\mathbb{A}, \mathbb{S})$ associated with a Kripke automaton $\mathbb{A} = \langle A, \Delta, \Omega, a_I\rangle$ and a Kripke structure $\mathbb{S}$ is given by Table 13. $\lhd$

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\Gamma, s) \in \wp_\forall(\mathsf{K}(A)) \times S \mid \Gamma \in \Delta(a)\}$
$(\Gamma, s) \in \wp_\forall(\mathsf{K}(A)) \times S$	$\forall$	$\{(\gamma, s) \in \mathsf{K}(A) \times S \mid \gamma \in \Gamma\}$
$(\gamma, s) \in \mathsf{K}A \times S$	$\exists$	$\{Z \subseteq A \times S \mid (\gamma, \sigma(s)) \in \overline{\mathsf{K}}Z\}$
$Z \in \wp(A \times S)$	$\forall$	Z

Table 13: Acceptance game for alternating Kripke automata

For an informal description of the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$, the most important observation is that matches of this game proceed in rounds moving from one basic position to another. Think of a basic position (a, s) as a statement, defended by $\exists$ and attacked by $\forall$, that a and s 'fit well together', or that $\mathbb{A}$, 'as seen from a', is an adequate description of $\mathbb{S}$, 'as seen from s'.

Each round consists of exactly four moves, with interaction pattern $\exists\forall\exists\forall$:

- At a basic position (a, s), the 'K-successor' $\sigma(s) \in \mathsf{K}S$ of s is fixed, but $\exists$ and $\forall$ first have to play a finite (two-move) subgame in order to determine which element will temporarily act as the analogous 'K-successor' $\alpha \in \mathsf{K}A$ of a. More precisely, the rules of the game are such that first, $\exists$ chooses a subset $\Gamma \subseteq \mathsf{K}A$ from $\Delta(a)$, after which $\forall$ chooses, from the set Γ of options, an actual element $\gamma \in \mathsf{K}A$. Halfway the round then, play has arrived at a position of the form $(\gamma, s) \in \mathsf{K}A \times S$.

- The players now proceed as in the bisimilarity game for Kripke models. First, $\exists$ chooses a 'local bisimulation' linking γ and s (or rather: γ and $\sigma(s)$), that is, a relation $Z \subseteq A \times S$ such that $(\gamma, \sigma(s)) \in \overline{\mathsf{K}}Z$. Spelled out, this means that $\exists$ can only choose such a relation Z if γ is of the form $(c, B) \in \wp(\mathsf{P}) \times \wp(A)^{\mathsf{D}}$ with $c = \sigma_V(s)$, and that Z has to satisfy the back and forth conditions for each atomic action d, stating that for all $b \in B$ there is $t \in \sigma_d(s)$ with bZt, and vice versa. The second half of the round, ends with $\forall$ choosing an element from Z. This element is of the form $(b, t) \in A \times S$ and forms the new basic position.

As before, we refer to these two 'halves' of a round as the 'static' and the 'dynamic' stage, respectively. This terminology refers to the fact that in the static part of the round, the automaton remains in the same state of the transition system, whereas in the dynamic stage of the round, the successor state in the transition system is determined.

▶ `associate automaton with Kripke model: example to be supplied`

At first sight, these Kripke automata may look rather different from the modal automata of section 8.1, they are in fact very similar. The best way to link Kripke automata to modal logic is via the *coalgebraic* approach towards modal logic which involves the coalgebraic modalities ∇ and $\bullet$ of Chapter 1. Recall that on the one hand, the semantics of ∇ and $\bullet$ is expressed in terms of the same kind of *relation lifting* that appears in the rules of the acceptance games for Kripke automata, and that on the other hand, the coalgebraic modalities are equally expressive as the standard boxes and diamonds.

▶ `It will be convenient to introduce some intermediate kinds of aut'a`

Definition 8.7 Given a set D of atomic actions, and sets C and A, the set $\mathit{MLatt}^{\nabla}_{\mathsf{D}}(C, A)$ of *nabla lattice terms over C and A* is defined as the set $\mathit{Latt}(C \cup \{\nabla_d \alpha \mid d \in \mathsf{D}, \alpha \in \wp A\})$. ◁

In particular, the set $\mathit{MLatt}^{\nabla}_{\mathsf{D}}(\pm\mathsf{P}, A)$ can be given by the following induction:

$$\varphi ::= p \mid \neg p \mid \nabla_d \alpha \mid \bigvee \Phi \mid \bigwedge \Phi$$

where p, d and α are arbitrary elements of the sets P, D, and $\wp A$, respectively, and Φ is a finite subset of $\mathit{MLatt}^{\nabla}_{\mathsf{D}}(\pm\mathsf{P}, A)$.

As an intermediate step between modal and Kripke automata we will consider automata $\mathbb{A} = \langle A, \Delta, \Omega, a_I \rangle$ such that $\Delta : A \to \mathit{MLatt}^{\nabla}_{\mathsf{D}}(\pm\mathsf{P}, A)$. It should be fairly obvious how to define the acceptance game associated with such automata, with the possible exception of the rules for positions of the form $(\nabla_d \alpha, s) \in \wp A \times S$:

Position	Player	Admissible next moves
$(\nabla_d \alpha, s)$	$\exists$	$\{Z \subseteq A \times S \mid (\alpha, \sigma_d(s)) \in \overline{\wp} Z\}$
$Z \in \wp(A \times S)$	$\forall$	Z

The following proposition states that all types of automata we have met in this Chapter are in fact equivalent.

Proposition 8.8 *Fix a set D of atomic actions, a set P of proposition letters, and let $C := \wp\mathsf{P}$. Consider automata of the form $\mathbb{A} = (A, a_I, \Delta, Acc)$, where the transition map Δ has one of the following four formats:*
(1) $\Delta : A \to \wp_{\exists}\wp_{\forall}\mathsf{K}_{\mathsf{D},\mathsf{P}} A$,
(2) $\Delta : A \to \mathit{Latt}(\mathsf{K}_{\mathsf{D},\mathsf{P}} A)$,
(3) $\Delta : A \to \mathit{MLatt}^{\nabla}_{\mathsf{D}}(\pm\mathsf{P}, A)$,
(4) $\Delta : A \to \mathit{MLatt}_{\mathsf{D}}(\pm\mathsf{P}, A)$.
Then there are effective transformations transforming an automaton of any one kind above to an equivalent automaton of any other kind.

Proof. This proposition can be proved on the basis of ideas underlying the proofs of Proposition 6.4 ($1 \leftrightarrow 2$), Theorem 1.36 ($2 \leftrightarrow 3$), and Proposition 1.31 ($3 \leftrightarrow 4$). We leave the details for the reader. QED

8.3 A fundamental theorem

In many cases it will be convenient to work with nondeterministic Kripke automata.

Definition 8.9 Given a Kripke functor K, an alternating Kripke automaton $\mathbb{A} = \langle A, \Delta, \Omega, a_I \rangle$ is called *nondeterministic* if, for every $a \in A$, all elements of $\Delta(a)$ are *singletons.* $\lhd$

Nondeterminism eliminates $\forall$'s role in the static part of the acceptance game — but not from the dynamic part, where it is still he who chooses the next basic position of the match.

Remark 8.10 To facilitate the presentation, we will often represent the transition function of such a nondeterministic automaton $\mathbb{A}$ as a map $\delta : A \to \wp_{\exists}(\mathsf{K}A)$. That is, rather than working with a set $\Delta(a)$ consisting of singletons we flatten the transition function and deal with the elements of those singletons directly. The structure of the associated acceptance game then looks as in Table 14. $\lhd$

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{(\gamma, s) \in \mathsf{K}(A) \times S \mid \gamma \in \delta(a)\}$
$(\gamma, s) \in \mathsf{K}A \times S$	$\exists$	$\{Z \subseteq A \times S \mid (\gamma, \sigma(s)) \in \overline{\mathsf{K}}Z\}$
$Z \in \wp(A \times S)$	$\forall$	Z

Table 14: Acceptance game for nondeterministic Kripke automata

The next theorem is perhaps the most fundamental result concerning the automata-theoretic approach towards the modal μ-calculus.

Theorem 8.11 *There is an effective procedure that transforms a given alternating Kripke automaton into an equivalent nondeterministic one.*

```
▶ Definition to be supplied
```

8.4 Closure properties

In order to discuss the recognizing power of automata, we first need to introduce the notion of a recognizable language.

Definition 8.12 We will refer to a class of pointed Kripke models as a K-*language*. Such a class C is called *recognizable* if there is a modal or Kripke automaton $\mathbb{A}$ such that $\mathsf{C} = \mathsf{L}(\mathbb{A})$. $\lhd$

It immediately follows from earlier results that recognizable K-languages have (at least two) alternative characterizations.

Proposition 8.13 *The following are equivalent for any* K*-language* C*:*

1. C *is recognizable;*
2. C *is nondeterministically recognizable, that is,* $\mathsf{C} = \mathsf{L}(\mathbb{A})$ *for some nondeterministic automaton* $\mathbb{A}$*;*
3. C *is definable by some modal fixpoint formula.*

The notion of recognizability can be used to study the *recognizing power* of certain kinds of graph automata. In particular, it will be of interest to see which *closure properties* are satisfied by graph automata.

Definition 8.14 Let Op be some operation on K-languages. We say that a class of languages is closed under Op if we obtain a language from this class whenever we apply Op to a family of languages from the class. $\lhd$

Closure under union and intersection

It follows immediately from the equivalence $1 \Leftrightarrow 3$ of Proposition 8.13 that the class of recognizable K-languages is closed under taking union, intersection and complementation. However, it is of interest for future reference to have direct, automata-theoretic proofs of the results for union and intersection. For that purpose, we define the sum and product of two K-automata, and prove that they recognize, respectively, the union and the intersection of the languages associated with the original automata.

Definition 8.15 Let $\mathbb{A}_1 = (A_1, a^1_I, \Delta_1, \Omega_1)$ and $\mathbb{A}_2 = (A_2, a^2_I, \Delta_2, \Omega_2)$ be two Kripke automata. We will define their *sum* $\mathbb{A}_\cup$ and *product* $\mathbb{A}_\cap$.

Both of these automata will have the *disjoint union* $A_{12*} := \{*\} \uplus A_1 \uplus A_2$ as their collection of states. Also, the parity function Ω will be the same for both automata:

$$\Omega(a) := \begin{cases} 0 & \text{if } a = *, \\ \Omega_i(a) & \text{if } a \in A_i. \end{cases}$$

The only difference between the automata lies in the transition functions, which are defined as follows:

$$\Delta_{\cup}(a) := \begin{cases} \Delta_1(a^1_I) \cup \Delta_2(a^2_I) & \text{if } a = * \\ \Delta_i(a) & \text{if } a \in A_i, \end{cases}$$

$$\Delta_{\cap}(a) := \begin{cases} \{\Phi_1 \cup \Phi_2 \mid \Phi_i \in \Delta_i(a^i_I)\} & \text{if } a = * \\ \Delta_i(a) & \text{if } a \in A_i. \end{cases}$$

Finally, we put $\mathbb{A}_{\cup} := (A_{12}, *, \Delta_{\cup}, \Omega)$ and $\mathbb{A}_{\cap} := (A_{12}, *, \Delta_{\cap}, \Omega)$. ◁

Proposition 8.16 *Let $\mathbb{A}_1$ and $\mathbb{A}_2$ be two Kripke automata. Then for any pointed* K-*coalgebra* $(\mathbb{S}, s)$ *we have:*

1. $\mathbb{A}_{\cup}$ *accepts* $(\mathbb{S}, s)$ *iff* $\mathbb{A}_1$ *or* $\mathbb{A}_2$ *accepts* $(\mathbb{S}, s)$*,*

2. $\mathbb{A}_{\cap}$ *accepts* $(\mathbb{S}, s)$ *iff both* $\mathbb{A}_1$ *and* $\mathbb{A}_2$ *accept* $(\mathbb{S}, s)$*.*

3. $\mathbb{A}_{\cup}$ *is non-deterministic if* $\mathbb{A}_1$ *and* $\mathbb{A}_2$ *are so.*

Proof. First suppose that the automaton $\mathbb{A}_{\cup}$ accepts $(\mathbb{S}, s)$. Hence by definition, $\exists$ has a winning strategy f in the acceptance game $\mathcal{A}(\mathbb{A}_{\cup}, \mathbb{S})$ starting from position $(*, s)$. Let i be such that $f(*, s) \in \Delta(a^i_I)$. It is then straightforward to verify that f, restricted to $\exists$'s positions in $\mathcal{A}(\mathbb{A}_i, \mathbb{S})$, is a winning strategy for $\exists$ from position (s, a^i_I). From this it is immediate that $\mathbb{A}_i$ accepts $(\mathbb{S}, s)$.

The other statements of the proof admit similarly straightforward proofs. QED

Closure under complementation

▶ `Material to be supplied`

Closure under projection

Just as in the case of biflow languages, we will be interested in the projection operation on K-languages as well. Our formulation here will be in terms of proposition letters rather than colors.

Definition 8.17 Let $\mathsf{P} = \mathsf{P}_1 \uplus \mathsf{P}_2$ be the disjoint union of two sets of proposition letters, and let $\mathbb{S} = \langle S, \sigma_V, \sigma_R \rangle$ be a Kripke model over P. Then we let $\mathbb{S}{\upharpoonright}_{\mathsf{P}_1}$ denote the restriction of $\mathbb{S}$ to P_1, that is, the structure $\langle S, \sigma_{V,1}, \sigma_R \rangle$ with $\sigma_{V,1} : S \to \wp(\mathsf{P}_1)$ given by $\sigma_{V,1}(s) := \sigma_V(s) \cap \mathsf{P}_1$.

Given a K-language L, $\mathsf{L}{\upharpoonright}_{\mathsf{P}_1}$ denotes the class of restriction to P_1 of models in L. ◁

As an easy example (generalizing results in the previous chapter) shows, the projection of a recognizable language is not necessarily recognizable. The point is that while any recognizable class L is closed under bisimilarity, this does not necessarily hold for the class $\mathsf{L}{\restriction_{\mathsf{P}_1}}$.

▶ `Example to be added`

On the other hand, this failure of bisimilarity invariance is the only thing that can go wrong. That is, we can show that for any recognizable language L, the 'bisimulation closure' of $\mathsf{L}{\restriction_{\mathsf{P}_1}}$, that is, the class of Kripke models that are *bisimilar* to a model in $\mathsf{L}{\restriction_{\mathsf{P}_1}}$, *is* recognizable. In brief, the recognizable Kripke languages are closed under projection *modulo bisimilarity.*

In order to prove this result, we need to work with nondeterministic automata. As discussed in Remark 8.10, we will represent the transition function of such a device $\mathbb{A}$ as a map $\delta : A \to \wp_{\exists}(\mathsf{K}A)$.

Definition 8.18 Let $\mathsf{P} = \mathsf{P}_1 \uplus \mathsf{P}_2$ be the disjoint union of two sets of proposition letters, and let $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$ be a $\mathsf{K}_{\mathsf{P},\mathsf{D}}$-automaton.

Then we define the $\mathsf{K}_{\mathsf{P}_1,\mathsf{D}}$-automaton $\mathbb{A}{\restriction_{\mathsf{P}_1}}$ as the structure $\langle A, a_I, \Delta', \Omega\rangle$, where

$$\Delta'(a) \ := \ \{(c \cap \mathsf{P}_1, B) \in \mathsf{K}_{\mathsf{P}_1,\mathsf{D}}(A) \mid (c, B) \in \Delta(a)\}.$$

defines the transition map Δ'. ◁

In words, Δ' is like Δ but keeps all options open for the proposition letters not in P_1. As an immediate corollary of the proposition below, the class of recognizable languages is closed under taking projections (modulo bisimilarity). The proof of the proposition uses a strong notion of *unravelling.*

Definition 8.19 A Kripke model $\mathbb{S}$ is *κ-unravelled*, where $\kappa \geq 1$ is some countable cardinal, if $\mathbb{S}$ is in tree-shape, and for each $s \in S$, each atomic action $d \in \mathsf{D}$, and each $t \in \sigma_d(s)$, there are at least κ may states $t' \in \sigma_d(s)$ that are bisimilar to t. ◁

Proposition 8.20 *Let $\mathsf{P} = \mathsf{P}_1 \uplus \mathsf{P}_2$ be the disjoint union of two sets of proposition letters, and let $\mathbb{A}$ be some $\mathsf{K}_{\mathsf{P},\mathsf{D}}$-automaton. Then*

1. *If $\mathbb{A}$ accepts some pointed Kripke model $(\mathbb{S}, s)$ over P, then $\mathbb{A}{\restriction_{\mathsf{P}_1}}$ accepts $(\mathbb{S}{\restriction_{\mathsf{P}_1}}, s)$.*
2. *If $\mathbb{A}{\restriction_{\mathsf{P}_1}}$ accepts $(\mathbb{S}', s')$ and $(\mathbb{S}', s')$ is $|A|$-unravelled, then $\mathbb{S}' = \mathbb{S}{\restriction_{\mathsf{P}_1}}$ for some $(\mathbb{S}, s')$ accepted by $\mathbb{A}$.*
3. *If $\mathbb{A}{\restriction_{\mathsf{P}_1}}$ accepts some Kripke model $(\mathbb{S}', s')$ over P_1, then $\mathbb{A}$ accepts some P-model $(\mathbb{S}, s)$ such that $\mathbb{S}', s \leftrightarroweq_{\mathsf{P}_1} \mathbb{S}{\restriction_{\mathsf{P}_1}}, s$.*

Finally, we summarize our findings.

Theorem 8.21 *For any Kripke functor K, the class of recognizable K-languages is closed under taking unions, intersections, complementation, and projections modulo bisimulation.*

Position	Player	Admissible next moves
$(a,s) \in A \times S$	$-$	$\{(\Delta(a,\sigma_C(s)), s)\}$
$(\bigvee \Phi, s)$	$\exists$	$\{(\varphi,s) \mid \varphi \in \Phi\}$
$(\bigwedge \Phi, s)$	$\forall$	$\{(\varphi,s) \mid \varphi \in \Phi\}$
$(\Diamond_d a, s)$	$\exists$	$\{(a,t) \mid t \in \sigma_d(s)\}$
$(\Box_d a, s)$	$\forall$	$\{(a,t) \mid t \in \sigma_d(s)\}$
$(\nabla_d \alpha, s)$	$\exists$	$\{Z \mid (\alpha, \sigma_d(s)) \in \overline{\wp}(Z)\}$
$Z \subseteq A \times S$	$\forall$	Z

Table 15: Acceptance game for chromatic automata

Notes

While Emerson & Jutla [9] already used automata-theoretic tools to prove results about versions of the modal μ-calculus that are interpreted over binary trees, Janin & Walukiewicz [10] were the first to introduce the (nondeterministic) Kripke automata discussed in this chapter, under the name of *μ-automata*. The fundamental equivalence between alternating and nondeterministic Kripke automata essentially goes back to this paper. A very general approach connecting automata and fixpoint logics was taken by Niwiński [23]. The automata we call *modal* were introduced in Wilke [32].

Exercises

Exercise 8.1 Fix a set P of proposition letters and let $C := \wp(\mathsf{P})$ be the corresponding alphabet. A *chromatic* automaton is a structure $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ with $\Delta : A \times C \to \mathit{MLatt}_{\mathsf{D}}(\varnothing, A)$ or $\Delta : A \times C \to \mathit{MLatt}^{\nabla}_{\mathsf{D}}(\varnothing, A)$ (In other words, the information about proposition letters and colors has been moved to the antecedent of the transition function.) Given a Kripke model $\mathbb{S}$, the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ is defined in the obvious way, see Table 15.

Provide effective transformations transformig ordinary automata into chromatic ones, and vice versa.

9 Results on the modal μ-calculus

In the previous chapter we introduced various kinds of *graph automata*, that is, automata operating on pointed Kripke models, or labelled transition systems. The main point about these automata as tools for studying the modal μ-calculus is that we may effectively identify modal fixpoint formulas with these graph automata. Taking these observations as our starting point, in this chapter we gather some of the most important results concerning the modal μ-calculus.

▶ `Summary of chapter.`

9.1 Decidability and small model property

In this section we will see a number of corollaries of the fundamental result on Kripke automata, Theorem 8.11, which states that every alternating Kripke automaton can be replaced with an equivalent nondeterministic one.

Disjunctive normal form

As an first consequence, we now see that every formula of the modal μ-calculus can be brought into so-called *disjunctive normal form.*

Definition 9.1 Given sets P of proposition letters, and D of atomic actions, respectively, the set $\mu\mathrm{CML}^{-}_{\mathsf{D}}(\mathsf{P})$ of *disjunctive* formulas is given by the following recursive definition:

$$\varphi \ ::= \ x \ | \ \bot \ | \ \varphi \vee \varphi \ | \ \pi \bullet \Phi \ | \ \mu x.\varphi \ | \ \nu x.\varphi$$

Here π denotes a subset of P, and $\Phi = \{\Phi_d \mid d \in \mathsf{D}\}$ a D-indexed collection of sets of disjunctive formulas, and x a variable *not* in P. $\lhd$

These formula are called disjunctive because the only admissible conjunctions are the special ones of the form $\pi \bullet \Phi$.

Theorem 9.2 *There is an effective algorithm that rewrites a modal fixpoint formula $\xi \in \mu\mathrm{PML}_{\mathsf{D}}(\mathsf{P})$ into an equivalent disjunctive formula ξ^d of length exponential in $|\xi|$.*

▶ `Proof (based on Theorem 8.11) to be supplied.`

Decidability

▶ `Intro`

Theorem 9.3 *There is an algorithm that decides in linear time whether a given disjunctive formula ξ is satisfiable or not.*

Proof. It is easy to see that the proof of this proposition is a direct consequence of the following observations:

1. $\top$ is satisfiable;
2. $\bot$ is not satisfiable;
3. $\varphi_1 \vee \varphi_2$ is satisfiable iff φ_1 or φ_2 is satisfiable;
4. $\pi \bullet \Phi$ is satisfiable iff both π and each $\varphi \in \bigcup_{d \in \mathsf{D}} \Phi_d$ is satisfiable;
5. $\mu x.\varphi$ is satisfiable iff $\varphi[\bot/x]$ is satisfiable;
6. $\nu x.\varphi$ is satisfiable iff $\varphi[\top/x]$ is satisfiable.

The proof of these claims is left as an exercise for the reader. QED

Decidability of the satisfiability problem for modal fixpoint formulas is then an immediate consequence of the previous two results.

Theorem 9.4 *There is an algorithm that decides in exponential time whether a given modal fixpoint formula ξ is satisfiable or not.*

Small model property

As a third immediate consequence of Theorem 8.11, we now show that any satisfiable modal fixpoint formula can in fact already be satisfied in a model of size at most exponential in the size of the formula.

For convenience here we will denote the transition function of a nondeterministic Kripke automaton $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ as a map $\Delta : A \to \wp_{\exists}\mathsf{K}A$, cf. Remark 8.10.

Definition 9.5 Let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be a nondeterministic Kripke automaton, and let $\alpha : A' \to \mathsf{K}A'$ be such that $\alpha(a) \in \Delta(a)$ for all $a \in A'$, where A' is some subset of A. Then we say that the pointed Kripke model $\langle A', \alpha, a_I \rangle$ is a *realization* of $\mathbb{A}$. $\lhd$

Theorem 9.6 *Let $\mathbb{A} = \langle A, a_0, \Delta, \Omega \rangle$ be a nondeterministic Kripke automaton. If $\mathbb{A}$ accepts some pointed Kripke model, then it accepts one that is a realization of $\mathbb{A}$.*

Proof. Defining a *nonemptiness game* on $\mathbb{A}$, one may give a proof of this theorem which is completely analogous to that of Theorem 5.19. QED

Corollary 9.7 *Let $\xi \in \mu\mathrm{ML}(\mathsf{P})$ be a modal fixpoint formula. Then ξ is satisfiable iff ξ is satisfiable in a model of size exponential in $|\xi|$.*

Proof.

▶ `Immediate by earlier results, once size matters are taken into consideration`

QED

9.2 Uniform interpolation and bisimulation quantifiers

Definition 9.8 Given two modal fixpoint formulas φ and ψ, we say that ψ is a *(local) consequence* of φ, notation: $\varphi \models \psi$, if $\mathbb{S}, s \Vdash \varphi$ implies $\mathbb{S}, s \Vdash \psi$, for every pointed Kripke model $(\mathbb{S}, s)$. $\lhd$

A formalism has the *(Craig) interpolation property* if we can find an *interpolant* for every pair of formulas φ and ψ such that $\varphi \models \psi$. This interpolant is a formula θ such that $\varphi \models \theta$ and $\theta \models \psi$; but most importantly, the requirement on θ is that it may only use symbols that occur both in φ and ψ.

▶ `why this is an important property`

As we will see now, the modal μ-calculus has *uniform* interpolation. This is a very strong version of interpolation in which the interpolant θ does not depend on the shape of ψ, but only on the *language of* ψ. More precisely, we define the following.

Definition 9.9 Let φ be a modal fixpoint formula, and $\mathsf{Q} \subseteq FV(\varphi)$. Then a *uniform interpolant* of φ with respect to Q is a formula θ with $FV(\theta) \subseteq \mathsf{Q}$, such that

$$\varphi \models \psi \text{ iff } \theta \models \psi. \tag{38}$$

for all formulas ψ with $FV(\psi) \cap FV(\varphi) \subseteq \mathsf{Q}$. $\lhd$

Remark 9.10 Instead of (38) we could have required

$$\varphi \models \psi \text{ iff } \varphi \models \theta \text{ and } \theta \models \psi, \tag{39}$$

which perhaps shows more clearly that θ is indeed an interpolant.

These two definitions are in fact equivalent. The key observation to see this is that (38) implies that from $\theta \models \theta$ it follows that $\varphi \models \theta$. $\lhd$

Theorem 9.11 (Uniform Interpolation) *Every modal fixpoint formula has a uniform interpolant.*

The proof consists of showing that the modal μ-calculus can express *bisimulation quantifiers*.

Proof. Fix the formula φ and the set Q, and define $\mathsf{P} := FV(\varphi)$ and $\mathsf{R} := \mathsf{P} \setminus \mathsf{Q}$.

By the equivalence of modal fixpoint formulas and nondeterministic Kripke automata, it follows from Proposition 8.20 that the modal μ-calculus can express *bisimulation quantifiers*. That is, given φ and Q, there is a formula $I_{\mathsf{Q}}(\varphi)$ with $FV(I_{\mathsf{Q}}(\varphi)) \subseteq \mathsf{Q}$ such that, for all pointed Q-models $(\mathbb{S}, s)$:

$$\mathbb{S}, s \Vdash I_{\mathsf{Q}}(\varphi) \text{ iff } \mathbb{S}', s' \Vdash \varphi \text{ for some } \mathsf{P}\text{-model } \mathbb{S}' \text{ with } \mathbb{S}'{\restriction_{\mathsf{Q}}}, s' \leftrightarroweq_{\mathsf{Q}} \mathbb{S}, s.$$

We leave the fairly straightforward (but not completely trivial) task of verifying that this $I_{\mathsf{Q}}(\varphi)$ is a uniform interpolant of φ with respect to Q as an exercise for the reader. QED

▶ `introduce bisimulation quantifiers`

9.3 Expressive completeness

One of the key results in the theory of modal logic is that the basic modal language corresponds to the bisimulation invariant fragment of first-order logic. In this section we will prove an extension of this result stating that the model μ-calculus is the bisimulation invariant fragment of *second-order* logic.

Theorem 9.12 *Let φ be a monadic second order property of pointed transition systems which is bisimulation invariant. Then there is a μPML-formula $\widehat{\varphi}$, effectively obtainable from φ, which is equivalent to φ.*

Before giving the exact definition of our version MSOL of monadic second-order logic, we roughly sketch the idea for proving Theorem 9.12. The proof is based on an algorithm that transforms an MSOL-formula φ into a modal fixpoint formula $\widehat{\varphi}$ with the property that for all pointed Kripke models $(\mathbb{S}, s)$:

$$\mathbb{S}, s \Vdash \widehat{\varphi} \text{ iff } \mathbb{E}_\omega(\mathbb{S}, s), s \Vdash \varphi. \tag{40}$$

Here $\mathbb{E}_\omega(\mathbb{S}, s)$ denotes the *ω-expansion* of $\mathbb{S}$ around s, defined as follows.

Definition 9.13 Let κ be a countable cardinal with $1 \leq \kappa \leq \omega$, and let $(\mathbb{S}, s)$ be a pointed Kripke model of type (P, D). A *κ-path through $\mathbb{S}$ is a finite (non-empty) $(S \cup \mathsf{D} \cup \kappa)$-sequence of the form $s_0 d_1 k_1 s_1 \cdots s_{n-1} d_n k_n s_n$, such that $R_d s_i s_{i+1}$ for each $i < n$. The set of such paths through $\mathbb{S}$ is denoted as $Paths^\kappa(\mathbb{S})$; we use the notation $Paths^\kappa_s(\mathbb{S})$ for the set of paths starting at s.*

The κ-expansion *of* $\mathbb{S}$ around s *is the transition system* $\mathbb{E}_\kappa(\mathbb{S}, s) = \langle Paths^\kappa_s(\mathbb{S}), \sigma^\kappa \rangle$, *where*

$$\begin{array}{rcl} \sigma^\kappa_V(s_0 \cdots d_n k_n s_n) & := & \sigma_V(s_n), \\ \sigma^\kappa_d(s_0 \cdots d_n k_n s_n) & := & \{(s_0 \cdots d_n k_n s_n d k t) \in Paths_s(\mathbb{S}) \mid R_d s_n t, 0 < k < \kappa\}. \end{array}$$

defines the coalgebra map $\sigma^\kappa = (\sigma_V, (\sigma_d \mid d \in \mathsf{D}))$. $\lhd$

It is not hard to check that the *unravellings* of a model (Definition 8.19) can be identified with its 1-expansions. The notion of κ-expansion generalizes that of the unravelling in that we take κ many copies of each d-successor in the κ-expansion. It easily follows from the definitions that $\mathbb{S}, s \leftrightarroweq \mathbb{E}_\omega(\mathbb{S}, s), s$, for any pointed Kripke model $(\mathbb{S}, s)$, and that every κ-expansion is *κ-unravelled* (see Definition 8.19): it is a tree in which d-successor nodes come in packs, of size at least κ, consisting of bisimilar/isomorphic siblings.

Returning to the proof sketch of our main result, Theorem 9.12 is in fact a direct consequence of (40): if φ is invariant under bisimulation we have

$$\mathbb{S}, s \models \varphi \text{ iff } \mathbb{E}_\omega(\mathbb{S}, s), (s) \models \varphi,$$

and so the equivalence of φ and $\widehat{\varphi}$ is immediate by (40).

Monadic Second-Order Logic

Definition 9.14 Given a collection P of set variables, and a set D of atomic actions, we define the language of *monadic second-order logic* MSOL as follows:

$$\varphi ::= p \sqsubseteq q \mid R_d(p,q) \mid \Downarrow p \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists p.\varphi$$

Here p and q are variables from P. $\lhd$

Definition 9.15 Given a Kripke model $\mathbb{S} = \langle S, V, R\rangle$, and a designated point $s \in S$, we define the semantics of MSOL as follows:

$$\begin{array}{lll}
\mathbb{S}, s \models p \sqsubseteq q & \text{if} & V(p) \subseteq V(q) \\
\mathbb{S}, s \models R(p,q) & \text{if} & \text{for all } s \in V(p) \text{ there is a } t \in V(q) \text{ with } Rst \\
\mathbb{S}, s \models \Downarrow p & \text{if} & V(p) = \{s\} \\
\mathbb{S}, s \models \neg\varphi & \text{if} & \mathbb{S}, s \not\models \varphi \\
\mathbb{S}, s \models \varphi \vee \psi & \text{if} & \mathbb{S}, s \models \varphi \text{ or } \mathbb{S}, s \models \psi \\
\mathbb{S}, s \models \exists p.\varphi & \text{if} & \mathbb{S}, s[p \mapsto X] \models \varphi \text{ for some } X \subseteq S.
\end{array}$$

An MSOL-formula φ is *bisimulation invariant* if $\mathbb{S}, s \leftrightarroweq \mathbb{S}', s'$ implies that $\mathbb{S}, s \models \varphi \Leftrightarrow \mathbb{S}', s \models \varphi$. $\lhd$

Note that the connective $\Downarrow$ is there to encode the *actual* world. This is needed if we want to compare MSOL with the modal μ-calculus, since formulas of the latter formalism are always evaluated relative to a point in the model.

Remark 9.16 In our version of second-order logic, there are *only* second-order variables. (In fact, one may think of this formalism as a *first-order* logic of which the intended models are first-order structures of the form $\langle \wp(S), \subseteq, \vec{R}\rangle$, where $\vec{R}(Y,Z)$ iff for all $y \in Y$ there is a $z \in Z$ such that Ryz.)

Rather than Definition 9.14, the reader may have expected a language allowing *both* first- *and* second-order quantification. For instance, given a set X of individual variables and a set P of set variables, we define the language MSOL$'$ as follows:

$$\varphi ::= x \approx y \mid R_d xy \mid p(x) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists p.\varphi$$

Here x and y are variables from X, p is a variable from P, and $d \in \mathsf{D}$ is an atomic action. This semantics of this language is completely standard, with $\exists x$ denoting first-order quantification (that is, quantification over individual states), and $\exists p$ denoting second-order quantification (that is, quantification over sets of states). Formulas of this languages are interpreted over Kripke models $\mathbb{S}$ *with* an assignment, that is, a map $\alpha : \mathsf{X} \to S$ interpreting the variables as elements of S.

It is not too hard to see that the two languages are in some sense equivalent. The key point is that MSOL can *interpret* MSOL$'$, by encoding individual variables as set variables denoting *singletons*. To understand how this works, we start with encoding a MSOL$'$-mode $\mathbb{M}$ with assignment α, as the $\mathsf{P} \cup \mathsf{X}$-model $\mathbb{M}^{\alpha} = (M, R, V^{\alpha})$, where $V^{\alpha}(p) := V(p)$ if $p \in \mathsf{P}$, and $V^{\alpha}(x) := \{\alpha(x)\}$ if $x \in \mathsf{X}$. It is then not hard to give a translation $(\cdot)^t$ from MSOL$'$ to MSOL, such that

$$\mathbb{M} \models \varphi[\alpha] \text{ iff } \mathbb{M}^{\alpha} \models \varphi^t,$$

for all MSOL$'$-formulas φ, all Kripke models $\mathbb{S}$, and all assignments α. The translation $(\cdot)^t$ crucially involves the MSOL-formulas $\mathtt{empty}(p)$ and $\mathtt{sing}(p)$ given by

$$\begin{array}{lcl} \mathtt{empty}(p) & := & \forall q\, (p \sqsubseteq q) \\ \mathtt{sing}(p) & := & \forall q\, \big(q \sqsubseteq p \to (\mathtt{empty}(q) \lor p \sqsubseteq q)\big). \end{array}$$

It is not hard to prove that these formulas hold in $\mathbb{S}$ iff, respectively, $V(p)$ is empty and $V(p)$ is a singleton. $\lhd$

```
▶ Examples of what you can say in msol, and not in muML:
  - every point has exactly two d-successors
  - relation R has no cycles (check!)
```

Automata for Monadic Second-Order Logic

We will now introduce the key instruments for proving Theorem 9.12, viz., the MSO-automata that correspond to formulas of second-order logic. In order to simplify our presentation we restrict to the uni-modal case in the remainder of this section.

In order to introduce MSO-automata, let us take a slightly different perspective on the modal automata of the previous chapter. First of all, here we work with *chromatic* automata, that is, the transition map Δ of the automaton $\mathbb{A}$ is of the form $\Delta : A \times C \to MLatt(\varnothing, A)$ or $\Delta : A \times C \to MLatt^{\nabla}(\varnothing, A)$. (The reader has been asked to prove that these automata are equivalent to the ordinary ones in Exercise 8.1).

The most important change of perspective is to think of states of $\mathbb{A}$ as *monadic predicates* of some first-order language, and of each $\Delta(a)$ as a *first-order formula* in this language. For instance, the term $\Box a_1 \land (\Diamond a_2 \lor \Box a_3)$ corresponds to the formula $\forall x\, a_1(x) \land \big(\exists y\, a_2(y) \lor \forall z\, a_3(z)\big)$. If the acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ arrives at a basic position (a, s), it is the task of $\exists$ to find an *interpretation* m of the predicates a_i occurring in $\Delta(a)$ as *subsets* of the set $\sigma_R(s)$ of successors of s, in such a way that the formula $\Delta(a)$ becomes *true* in the resulting model $(\sigma_R(s), m)$.

That this new perspective corresponds to the old one is fairly easy to see for the nabla formulas. To see this, suppose that the acceptance game arrives at a basic position of the form (a, s), with $\Delta(a) = \nabla\alpha$. Then $\exists$ has to come up with a relation Z linking every $a \in \alpha$ to some $t \in \sigma_R(s)$, and vice versa. But using the correspondence between relations $Z \subseteq A \times \sigma_R(s)$ and maps $m : A \to \wp(\sigma_R(s))$, this is the same as finding an interpretation m_Z making the formula

$$\Delta'(a) = \bigwedge_{a\in\alpha} \exists y\, a(y) \wedge \forall z \bigvee_{a\in\alpha} a(z)$$

true in the model $(\sigma_R(s), m_Z)$.

The point of this *new* perspective is that we may now generalize this set-up to a *wider set* of first-order sentencess, that may also use $\approx$ and its negation $\not\approx$, and thus properly extend the ones that are obtained as translations of modal formulas.

Definition 9.17 Given a set A, consider the set of first-order formulas defined by the following grammar (where x and y range over some set X of variables):

$$\varphi \;::=\; x \approx y \;\mid\; a(x) \;\mid\; \neg\varphi \;\mid\; \varphi \vee \varphi \;\mid\; \exists x.\varphi$$

We let $FO(A)$ denote the set of *sentences* in this language.

A *structure* for this language is given as a pair (Q, m), where $m : A \to \wp(Q)$ is an *interpretation* assigning a subset of Q to each 'predicate' $a \in A$. $\lhd$

Given a $FO(A)$-formula φ and a structure (Q, m), we can use the standard semantics of first-order logic to determine whether the formula is true in the structure, notation: $(Q, m) \models \varphi$, or not.

Definition 9.18 Given a set P of proposition letters, an MSO-*automaton* is a structure $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$, where A, a_I and Ω are as usual, and Δ is a map $\Delta : A \times C \to FO(A)$.

Given a Kripke model $\mathbb{S}$, the acceptance game of such an automaton with respect to $\mathbb{S}$ is given in the table below.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{m : A \to \wp(\sigma_R(s)) \mid (\sigma_R(s), m) \models \Delta(a, \sigma_V(s))\}$
$m : A \to \wp(S)$	$\forall$	$\{(b, t) \mid t \in m(b)\}$

The winning conditions for both finite and infinite matches are as usual. $\lhd$

In words, the acceptance game proceeds as follows. At a basic position (a, s), $\exists$ chooses a so-called *marking* m interpreting each 'predicate' $a \in A$ as a subset $m(a)$ of the set $\sigma_R(s)$ of successors of s. In this choice, she is bound by the condition that the formula $\Delta(a, \sigma_V(s))$ must be *true* in the resulting $FO(A)$-model $(\sigma_R(s), m)$. Once chosen, this map m itself becomes the next position of the match. As such it belongs to $\forall$, and all he has to do is to choose a pair (b, t) such that t satisfies the 'predicate' b, or equivalently, $t \in m(b)$. This pair (b, t) is the next basic position of the match.

As it turns out, we may always transform an MSO-automaton into a special one, in which every formula $\Delta(a)$ has been brought into a special normal form.

Definition 9.19 A sentence $\varphi \in FO(A)$ is in *basic form* if it has the following shape:

$$\exists \vec{y} \Big(\mathtt{diff}(\vec{y}) \wedge \bigwedge_{1 \leq i \leq n} \bigwedge_{a \in \alpha_i} a(y_i) \wedge \forall z \left(\mathtt{diff}(\vec{y}, z) \to \bigvee_{\beta \in B} \bigwedge_{b \in \beta} b(z) \right) \Big).$$

Here each $\beta \in B$ and each α_i is a subset of A, and $\mathtt{diff}(y_1, \ldots, y_n)$ denotes the formula $\bigwedge\{y_i \neq y_j \mid 1 \leq i < j \leq n\}$. Such a formula is in *special* basic form if each $\beta \in B$ and each α_i is in fact a singleton. The sets of these sentences are denoted by $BF(A)$ and $SBF(A)$, respectively.

An MSO-automaton is called *nondeterministic* if the range of Δ is in $SLatt(SBF(A))$, that is, every formula $\Delta(a, c)$ is a disjunction of special basic formulas. $\lhd$

In the next proposition we first show that every MSO-automaton can be transformed into one where every formula $\Delta(a, c)$ is a *disjunction of special formulas*. We then move on to the much harder result, which can be seen as the analogon of Theorem 8.11, that every MSO-automaton has a nondeterministic equivalent.

Proposition 9.20 *Fix a set* P *of proposition letters, and let* $C := \wp\mathsf{P}$. *Consider MSO-automata of the form* $\mathbb{A} = (A, a_I, \Delta, Acc)$, *where the transition map* Δ *has one of the following four formats:*
(1) $\Delta : A \times C \to SLatt(FO(A))$,
(2) $\Delta : A \times C \to SLatt(BF(A))$,
(3) $\Delta : A \times C \to SLatt(SBF(A))$,
Then there are effective transformations transforming an automaton of any one kind above to an equivalent automaton of any other kind.

Proof. The equivalence $1 \leftrightarrow 2$ is based on a result in first-order model theory, namely, that every first-order sentence in $FO(A)$ can be rewritten into an equivalent normal form in $SLatt(FO(A))$.

For the hard direction of $1/2 \leftrightarrow 3$, let $\mathbb{A} = \langle A, a_I, \Delta, \Omega \rangle$ be an MSO-automaton, with $\Delta : A \to SLatt(BF(A))$.

▶ `Details to be supplied`

QED

Proposition 9.21 *Let* φ *be some* MSOL*-formula. Then for any set* P *of propositional variables containing the free variables of* φ *there is an effectively obtainable nondeterministic or alternating automaton* $\mathbb{B}_\varphi$ *over the alphabet* $\wp\mathsf{P}$, *such that for any* ω-*unravelled tree model* $\mathbb{S}$ *for* P, *with root* r:

$$\mathbb{S}, r \models \varphi \textit{ iff } \mathbb{B}_\varphi \textit{ accepts } (\mathbb{S}, r). \tag{41}$$

Proof. We prove the proposition by induction on the complexity of φ.

For the base case we only give the automaton characterizing the atomic formula $R(p,q)$. This automaton $\mathbb{B}_{R(p,q)}$ is given as the structure $\langle\{a_0,a_1\},a_0,\Delta,\Omega\rangle$, where Δ is given by putting:

$$\begin{array}{rcll} \Delta(a_0,c) & := & \left\{ \begin{array}{ll} \exists y\,\big(a_1(y) \wedge \forall z\,(z \neq y \rightarrow a(z))\big) & \text{if } p \in c \\ \forall z\, a_0(z) & \text{otherwise} \end{array} \right. \\ \Delta(a_1,c) & := & \left\{ \begin{array}{ll} \bot & \text{if } q \notin c \\ \exists y\,\big(a_1(y) \wedge \forall z\,(z \neq y \rightarrow a(z))\big) & \text{if } q \in c \text{ and } p \in c \\ \forall z\, a_0(z) & \text{otherwise} \end{array} \right. \end{array}$$

Furthermore, Ω is defined $\Omega(a_i) := 0$ for each a_i — as a consequence, $\exists$ wins all infinite games. We leave it for the reader to verify that this automaton is of the right shape, and that it is equivalent to the formula $R(p,q)$.

For the inductive step of the argument, there are three cases to consider. Leaving the other cases as exercises for the reader, we treat formulas of the form $\varphi = \exists p.\psi$. Inductively we may assume that there is a nondeterministic MSO-automaton $\mathbb{B}_\psi = \langle B, b_I, \Delta, \Omega\rangle$, with alphabet $C' = \wp(\mathsf{P} \cup \{p\})$ which is equivalent to ψ on the class of ω-unravelled trees.

We can define the automaton $\mathbb{B} = \langle B, b_I, \Delta_C, \Omega\rangle$, with alphabet $C = \wp(\mathsf{P})$, by putting

$$\Delta_C(a,c) := \Delta(a,c) \vee \Delta(a, c \cup \{p\}).$$

Clearly then $\mathbb{B}$ is a nondeterministic MSO-automaton, so it remains to prove that $\mathbb{B}$ is equivalent to φ. For the hard part of this proof, assume that $\mathbb{S}, r$ is an ω-unravelled tree model accepted by $\mathbb{B}$. We need to show that $\mathbb{S}, r \models \varphi$, that is, we need to find a subset $P \subseteq S$ such that $\mathbb{S}[p \mapsto P], r \models \psi$, or, equivalently, such that $\mathbb{B}_\psi$ accepts $(\mathbb{S}[p \mapsto P], r)$.

We leave it for the reader to verify that due to the fact that $\mathbb{S}$ is an ω-unravelled tree, we may without loss of generality assume that $\exists$ plays a *scattered* strategy. That is, we may assume that for every point $s \in S$ there is a unique automata state a_s such that $\exists$'s strategy guarantees that s will only be visited by $\mathbb{B}$ if $\mathbb{B}$ is in state a_s. Putting it differently, we may encode $\exists$'s winning strategy by a map $s \mapsto a_s$ such that $a_r = a_I$, and for each $s \in S$, the position (a_s, s) is a winning position for $\exists$, with her winning strategy telling her to pick the following marking m_s of the set $\sigma_R(s)$:

$$m_s(a) := \{t \in \sigma_R(s) \mid a_t = a\}.$$

Since this m_s is part of a winning strategy, it holds that $(\sigma_R(s), m_s) \models \Delta_C(a_s, \sigma_C(s))$. Thus by definition, for each $s \in S$ we have $(\sigma_R(s), m_s) \models \Delta(a_s, \sigma_C(s)) \vee \Delta(a_s, \sigma_C(s) \cup \{p\})$. Now we define

$$P := \big\{ s \in S \mid (\sigma_R(s), m_s) \models \Delta(a_s, \sigma_C(s) \cup \{p\}) \big\}.$$

From this definition it is not hard to prove that the very same strategy of $\exists$ that was winning in the acceptance game of $\mathbb{B}_\psi$ with respect to $(\mathbb{S}, r)$ is also winning in the acceptance game of $\mathbb{B}$ with respect to $(\mathbb{S}[p \mapsto P, r)$. QED

From monadic second-order logic to the modal μ-calculus

Definition 9.22 Consider an *SBF*(A)-sentence

$$\varphi \;=\; \exists y_1 \cdots y_n \Big(\mathtt{diff}(\vec{y}) \wedge \bigwedge_{1 \leq i \leq n} a_i(y_i) \wedge \forall z \left(\mathtt{diff}(\vec{y}, z) \to \bigvee_{b \in \beta} b(z)\right)\Big),$$

Abbreviate $\alpha := \{a_1, \ldots, a_n\}$, and define φ^- as the *FO*(A)-sentence

$$\varphi^- \;:=\; \bigwedge_{a \in \alpha} \exists y\, a(y) \wedge \forall z \big(\bigvee_{b \in \beta} b(z)\big).$$

Furthermore, let φ^μ denote the *MLatt*$(\varnothing, A)$-term

$$\varphi^\mu \;:=\; \bigwedge_{a \in \alpha} \Diamond a \wedge \Box \bigvee_{b \in \beta} b.$$

Given an MSO-automaton $\mathbb{A} = \langle A, a_I, \Delta, \Omega\rangle$, let $\mathbb{A}^-$ be the MSO-automaton $\langle A, a_I, \Delta^-, \Omega\rangle$, where, for each $a \in A$ and $c \in C$, the formula $\Delta^-(a, c)$ is obtained from $\Delta(a, c)$ by replacing each disjunct φ of $\Delta(a, c)$ with the formula φ^-.

The chromatic modal automaton $\mathbb{A}^\mu$ (see Exercise 8.1) is defined analogously. That is, we define $\Delta^\mu : A \times C \to \mathit{MLatt}^\nabla(\varnothing, A)$, by putting

$$\Delta^\mu(a, c) := \bigvee_{1 \leq i \leq n} \varphi_i^\mu$$

if $\Delta(a, c) = \bigvee_{1 \leq i \leq n} \varphi_i$ $\lhd$

Proposition 9.23 *Let $\mathbb{A}$ be a* MSO*-automaton, and let $(\mathbb{S}, s)$ be some pointed Kripke model. Then*
(1) $\mathbb{A}^-$ accepts $(\mathbb{S}, s)$ iff $\mathbb{A}$ accepts $(\mathbb{E}_\omega(\mathbb{S}, s), s)$.
(2) $\mathbb{A}^-$ accepts $(\mathbb{S}, s)$ iff $\mathbb{A}^\mu$ accepts $(\mathbb{S}, s)$.

Proof. We leave the proof of this Proposition as an exercise for the reader. Note that the proof of the second part is almost immediate. QED

Proof of Theorem 9.12. Let φ be a formula of monadic second-order logic, and let $(\mathbb{S}, s)$ be an arbitrary pointed Kripke model. By Proposition 9.21 there is an MSO-automaton $\mathbb{A}$ which is equivalent to φ on ω-unravelled models.

By the results in the previous chapter (including Exercise 8.1), we can effectively obtain a modal fixpoint formula $\widehat{\varphi}$ which is equivalent to the automaton $\mathbb{A}^\mu$. But then by Proposition 9.23 we may derive (40) above, and we already saw that this suffices to prove the theorem. QED

9.4 Preservation results

▶ `Details to be supplied`

9.5 Model checking

▶ `Details to be supplied`

Notes

The decidability of the satisfiability problem of the modal μ-calculus was first proved by Kozen and Parikh [15] via a reduction to SnS. Emerson & Jutla [8] established the EXPTIME-completeness of this problem. The finite model property was proved by Kozen [14].

Uniform interpolation of the modal μ-calculus was proved by D'Agostino & Hollenberg [7], who established other model-theoretic results as well. The result that the modal μ-calculus is the bisimulation-invariant fragment of monadic second-order logic is due to Janin & Walukiewicz [11].

Exercises

Exercise 9.1 Let γ be some disjunctive fixed point formula.

(a) Show that $\mu x.\gamma$ is satisfiable iff $\gamma[\bot/x]$ is satisfiable.

(b) Show that $\nu x.\gamma$ is satisfiable iff $\gamma[\top/x]$ is satisfiable.

(c) Do the above statements hold for arbitrary fixed point formulas as well?

Exercise 9.2 Provide the proof of Proposition 9.23, part (1).

10 Axiomatization

Definition 10.1 Let $\mathbf{K}\mu$ be the logic obtained by adding the following axiom scheme

(prefix) $\varphi[\mu x.\varphi/x] \to \mu x.\varphi$

and derivation rule:

(min) from $\vdash \varphi[\psi/x] \to \psi$ infer $\vdash \mu x.\varphi \to \psi$.

to the basic modal logic $\mathbf{K}$. $\lhd$

Clearly, (prefix) expresses that $\mu x.\varphi$ is a prefixpoint of the formula $\varphi(x)$, and (min) says that $\mu x.\varphi$ is in fact below any prefixpoint of $\varphi(x)$.

Theorem 10.2 $\mathbf{K}\mu$ *is sound and complete with respect to the standard semantics.*

Notes

D. Kozen proposed the axiomatization $\mathbf{K}\mu$ already in the original paper [13] where he introduced the formalism of the modal μ-calculus. He showed the axiomatization to be complete for the so-called *aconjunctive fragment*. The completeness for the full language turned out to be a hard nut to crack, but the problem was finally solved by I. Walukiewicz [30, 31].

Part VI
Appendix

A Mathematical preliminaries

Sets and functions We use standard notation for set-theoretic operations such as union, intersection, product, etc. The power set of a set S is denoted as $\wp(S)$ or $\wp S$, and we sometimes denote the relative complement operation as $\sim_S X := S \setminus X$. The size or cardinality of a set S is denoted as $|S|$.

Let $f : A \to B$ be a function from A to B. Given a set $X \subseteq A$, we let $f[X] := \{f(a) \in B \mid a \in X\}$ denote the image of X under f, and given $Y \subseteq B$, $f^{-1}[Y] := \{a \in A \mid f(a) \in Y\}$ denotes the preimage of Y. In case f is a bijection, we let f^{-1} denote its inverse. The composition of two functions $f : A \to B$ and $g : B \to A$ is denoted as $g \circ f$ or gf, and the set of function from A to B will be denoted as either B^A or $A \to B$.

It is well-known that there is a bijective correspondence

$$(A \times B) \to C \;\cong\; A \to (B \to C).$$

With a function $f : A \times B \to C$, we associate the map that, for each $a \in A$, yields the function $f_a : B \to C$ given by $f_a(b) := f(a, b)$.

Relations Given a relation $R \subseteq A \times B$, we introduce the following notation. $\mathsf{Dom}(R)$ and $\mathsf{Ran}(R)$ denote the domain and range of R, respectively. R^{-1} denotes the converse of R. For $R \subseteq S \times S$, R^* denotes the reflexive-transitive closure of R, and R^+ the transitive closure. For $X \subseteq A$, we put $R[X] := \{b \in B \mid (a, b) \in R \text{ for some } a \in X\}$; in case $X = \{s\}$ is a singleton, we write $R[s]$ instead of $R[\{s\}]$. For $Y \subseteq B$, we will write $\langle R\rangle Y$ rather than $R^{-1}[Y]$, while $[R]Y$ denotes the set $\{a \in A \mid b \in Y \text{ whenever } (a, b) \in R\}$. Note that $[R]Y = A \setminus \langle R\rangle(B \setminus Y)$. A relation R on S is *acyclic* if there are no elements s such that R^+ss.

Sequences, lists and streams Given a set C, we define C^* as the set of finite *lists*, *words* or *sequences* over C. We will write ε for the empty sequence, and define $C^+ := C^* \setminus \{\varepsilon\}$ as the set of nonempty words. An infinite word or sequence, of *stream* over C is a map $\gamma : \omega \to C$ mapping natural numbers to elements of C; the set of these maps is denoted by C^ω. We write $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$ for the set of all sequences over Σ.

We use $\sqsubseteq$ for the initial segment relation between sequences, and $\sqsubset$ for the proper (i.e., irreflexive) version of this relation. For a nonempty sequence π, *first*(π) denotes the first element of π. In the case that π is finite and nonempty we write *last*(π) for the last element of π. Given a stream $\gamma = c_0c_1\dots$ and two natural numbers $i < j$, we let $\gamma[i, j)$ denote the finite word $c_ic_{i+1}\dots c_{j-1}$.

Graphs and trees A *(directed) graph* is a pair $\mathbb{G} = \langle G, E\rangle$ consisting of a set G of *nodes* or *vertices* and a binary *edge* relation E on G. A *path* through such a graph is a sequence $(s_0, \ldots, s_n)$ in G^* such that Es_is_{i+1} for all $i < n$. A (proper) *cycle* is a path $(s_0, \ldots, s_n)$ such that $n > 0$ and $s_0 = s_n$ (and all $s_0, \ldots, s_{n-1}$ are all distinct). A graph is *acyclic* if it has no cycles. A *tree* is a graph $\mathbb{T} = (T, R)$ which contains a node r, called a *root* of $\mathbb{T}$, such that every element $t \in T$ is reachable by a *unique* path from r. (In particular, this means that R is acyclic, and that the root is unique.)

Fact A.1 (König's Lemma) *Let $\mathbb{G}$ be a finitely branching, acyclic graph. If $\mathbb{G}$ is infinite, then it has an infinite path.*

Order and lattices A *partial order* is a structure $\mathbb{P} = \langle P, \leq \rangle$ such that $\leq$ is a reflexive, transitive and antisymmetric relation on P. Given a partial order $\mathbb{P}$, an element $p \in P$ is an *upper bound* (*lower bound, respectively*) of a set $X \subseteq P$ if $p \geq x$ for all $x \in X$ ($p \leq x$ for all $x \in X$, respectively). If the set of upper bounds of X has a minimum, this element is called the *least upper bound*, *supremum*, or *join* of X, notation: $\bigvee X$. Dually, the *greatest lower bound*, *infimum*, or *meet* of X, if existing, is denoted as $\bigwedge X$.

A partial order $\mathbb{P}$ is called a *lattice* if every two-element subset of P has both an infimum and a supremum; in this case, the notation is as follows: $p \wedge q := \bigwedge\{p, q\}$, $p \vee q := \bigvee\{p, q\}$. Such a lattice is *bounded* if it has a minimum $\bot$ and a maximum $\top$. A partial order $\mathbb{P}$ is called a *complete lattice* if every subset of P has both an infimum and a supremum. In this case we abbreviate $\bot := \bigvee \varnothing$ and $\top := \bigwedge \varnothing$; these are the smallest and largest elements of $\mathbb{C}$, respectively. A complete lattice will usually be denoted as a structure $\mathbb{C} = \langle C, \bigvee, \bigwedge \rangle$. Key examples of complete lattices as full power set algebras: given a set S, it is easy to show that the structure $\langle \wp(S), \bigcup, \bigcap \rangle$ is a complete lattice.

Given a family $\{\mathbb{P}_i \mid i \in I\}$ of partial orders, we define the *product* order $\prod_{i \in I} \mathbb{P}_i$ as the structure $\langle \prod_{i \in I} P_i, \leq \rangle$ where $\prod_{i \in I} P_i$ denotes the cartesian product of the family $\{P_i \mid i \in I\}$, and $\leq$ is given by $\pi \leq \pi'$ iff $\pi(i) \leq_i \pi'(i)$ for all $i \in I$. It is not difficult to see that the product of a family of (complete) lattices is again a (complete) lattice, with meets and joins given coordinatewise. For instance, given a family $\{\mathbb{C}_i \mid i \in I\}$ of complete lattices, and a subset $\Gamma \subseteq \prod_{i \in I} C_i$, it is easy to see that Γ has a least upper bound $\bigvee \Gamma$ given by

$$\left(\bigvee \Gamma\right)(i) = \bigvee \{\gamma(i) \mid \gamma \in \Gamma\},$$

where the join on the right hand side is taken in $\mathbb{C}_i$.

Ordinals A set S is *transitive* if $S \subseteq \wp(S)$; that is, if every element of S is a subset of S, or, equivalently, if $S'' \in S' \in S$ implies that $S'' \in S$. An *ordinal* is a transitive set of which all elements are also transitive. From this definition it immediately follows that any element of an ordinal is again an ordinal. We let $\mathcal{O}$ denote the class of all ordinals, and use lower case Greek symbols (α, β, γ, $\ldots$, λ, $\ldots$) to refer to individual ordinals.

The smallest, *finite*, ordinals are

$$\begin{array}{lcll}
0 & := & \varnothing & \\
1 & := & \{0\} & (= \{\varnothing\}) \\
2 & := & \{0,1\} & (= \{\varnothing, \{\varnothing\}\}) \\
3 & := & \{0,1,2\} & (= \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}) \\
\vdots & & &
\end{array}$$

In general, the *successor* $\alpha+1$ of an ordinal α is the set $\alpha \cup \{\alpha\}$; it is easy to check that $\alpha + 1$ is again an ordinal. Ordinals that are not the successor of an ordinal are called *limit* ordinals. Thus the smallest limit ordinal is 0; the next one is the first infinite ordinal

$$\omega := \{0, 1, 2, 3, \ldots\}.$$

But it does not stop here: the successor of ω is the ordinal $\omega + 1$, etc. It is important to realize that there are in fact too many ordinals to form a set: $\mathcal{O}$ is a proper class. As a consequence, whenever we are dealing with a function $f : \mathcal{O} \to A$ from $\mathcal{O}$ into some set A, we can conclude that there exist distinct ordinals $\alpha \neq \beta$ with $f(\alpha) = f(\beta)$. (Such a function f will also be a class, not a set.)

We define an ordering relation $<$ on ordinals by:

$$\alpha < \beta \text{ if } \alpha \in \beta.$$

From this definition it follows that $\alpha = \{\beta \text{ in } \mathcal{O} \mid \beta < \alpha\}$ for every ordinal α. The relation $<$ is obviously transitive (if we permit ourselves to apply such notions to relations that are classes, not sets). It follows from the axioms of ZFC that $<$ is in fact *linear* (that is, for any two ordinals α and β, either $\alpha < \beta$, or $\alpha = \beta$, or $\beta < \alpha$) and *well-founded* (that is, every non-empty set of ordinals has a smallest element).

The fact that $<$ is well-founded allows us to generalize the principle of induction on the natural numbers to the transfinite case.

Transfinite Induction Principle In order to prove that all ordinals have a certain property, it suffices to show that the property is true of an arbitrary ordinal α whenever it is true of all ordinals $\beta < \alpha$.

A proof by transfinite induction typically contains two cases: one for successor ordinals and one for limit ordinals (the base case of the induction is then a special case of a limit ordinal). Analogous to the transfinite inductive proof principle there is a transfinite inductive way of defining functions on the class of ordinals.

References

[1] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.5, pages 739–782. North-Holland Publishing Co., Amsterdam, 1977.

[2] J. van Benthem. *Modal Correspondence Theory.* PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[4] J.R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel, editor, *Proceedings of the International Congress on Logic, Methodology and the Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[5] A. Chagrov and M. Zakharyaschev. *Modal Logic*, volume 35 of *Oxford Logic Guides.* Oxford University Press, 1997.

[6] A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[7] G. D'Agostino and M. Hollenberg. Logical questions concerning the μ-calculus. *Journal of Symbolic Logic*, 65:310–332, 2000.

[8] E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *Proceedings of the 29th Symposium on the Foundations of Computer Science*, pages 328–337. IEEE Computer Society Press, 1988.

[9] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proceedings of the 32nd Symposium on the Foundations of Computer Science*, pages 368–377. IEEE Computer Society Press, 1991.

[10] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science, MFCS'95*, volume 969 of *LNCS*, pages 552–562. Springer, 1995.

[11] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus w.r.t. monadic second-order logic. In *Proceedings of the Seventh International Conference on Concurrency Theory, CONCUR '96*, volume 1119 of *LNCS*, pages 263–277, 1996.

[12] B. Knaster. Un théorème sur les fonctions des ensembles. *Annales de la Societé Polonaise de Mathematique*, 6:133–134, 1928.

[13] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[14] D. Kozen. A finite model theorem for the propositional μ-calculus. *Studia Logica*, 47:233–241, 1988.

[15] D. Kozen and R. Parikh. A decision procedure for the propositional μ-calculus. In *Proceedings of the Workshop on Logics of Programs 1983*, LNCS, pages 313–325, 1983.

[16] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[17] L. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999. (Erratum published *Ann.P.Appl.Log.* 99:241–259, 1999).

[18] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In A. Skowron, editor, *Computation Theory*, LNCS, pages 157–168. Springer-Verlag, 1984.

[19] D.E. Muller. Infinite sequences and finite machines. In *Proceedings of the 4th IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 3–16, 1963.

[20] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[21] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata. *Theoretical Computer Science*, 141:69–107, 1995.

[22] D. Niwiński. Fixed points vs infinite generation. In *Proceedings of the 3rd Annual IEEE Symposium on Logic in Computer Science (LICS'88)*, pages 402–409. IEEE Computer Society Press, 1988.

[23] D. Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189:1–69, 1997.

[24] D. Park. Concurrency and automata on infinite sequences. In *Proceedings 5th GI Conference*, pages 167–183. Springer, 1981.

[25] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. Foundations of Computer Science*, pages 46–57, 1977.

[26] V.R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symposium on Computer Science*, pages 109–121, 1976.

[27] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[28] S. Safra. On the complexity of ω-automata. In *Proceedings of the 29th Symposium on the Foundations of Computer Science*, pages 319–327. IEEE Computer Society Press, 1988.

[29] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[30] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS'95)*, pages 14–24. IEEE Computer Society Press, 1995.

[31] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157:142–182, 2000.

[32] T. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8:359–391, 2001.

[33] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.