9 Alternating Tree Automata and Parity Games

Daniel Kirsten

Institut für Algebra
Technische Universität Dresden

9.1 Introduction

Since Büchi's work in 1960 [17], automata play an important role in logic. Numerous different notions of automata provide decision and complexity results in various kinds of logic. Often, one develops a method to translate some given formula φ into an appropriate finite automaton $\mathcal{A}$ such that $L(\varphi) = L(\mathcal{A})$. Such a translation reduces the model checking problem and the satisfiability problem in some logic to the word problem and the emptiness problem for finite automata. Moreover, such a translation provides algorithms to solve the model checking and the satisfiability problems on a computer. Consequently, one is interested in the decidability and the complexity of the word and emptiness problems of automata.

In this chapter, we introduce the notion of alternating tree automata. They have been introduced in [202] to get a better understanding of the modal μ-calculus. Thus, alternating tree automata work on transition systems (Kripke structures). We state complexity results for the word problem, the emptiness problem and complementation.

The notion of parity games and related results play a crucial role within the whole chapter. Parity games provide three advantages:

(1) We use parity games to define the semantics of alternating tree automata, i.e., we define whether an automaton accepts or rejects some transition system by the existence of a winning strategy for Player 0 in an appropriate parity game.
(2) Parity games provide a straightforward, convenient construction to complement a given alternating tree automaton; moreover, the fact that parity games are determined is used to prove the correctness of this construction.
(3) We use parity games to show the decidability of the word problem and the emptiness problem. By applying Jurdziński's result[93], we achieve strong complexity bounds.

The reader should be familiar with parity games as introduced in Chapter 2 and 5. To prove the decidability of the emptiness problem we use various notions of automata on infinite words such as Büchi automata, Rabin- and Streett-automata, parity automata, and transformations between them as introduced in Chapter 1 of this book. We also apply Safra's construction from Chapter 3.

The results from the present chapter will be used to examine the modal μ-calculus in Chapters 10 and 11. In particular, the complexity results of the

E. Grädel et al. (Eds.): Automata, Logics, and Infinite Games, LNCS 2500, pp. 153-167, 2002.

word problem and emptiness problem will be used to show complexity results for model checking and satisfiability in the modal μ-calculus.

This chapter is organized as follows: In Section 9.2, we introduce some basic notions. In Section 9.3, we introduce alternating tree automata and their semantics formally. Sections 9.4, 9.5, and 9.6 are devoted to the three main results: The decidability and complexity of the word problem, the complementation of alternating tree automata, and the decidability and complexity of the emptiness problem. Some remarks and exercises close this chapter.

The main ideas presented here are due to [202]. Our complementation of alternating tree automata is based on an idea from [137] with some extensions to our concept of alternating tree automata.

9.2 Preliminaries

We fix a set of **propositional variables** P during this chapter. A **transition system** is a triple $\mathcal{S} = (S, R, \lambda)$ where

- S is a set called **states**,
- $R \subseteq S \times S$ is a relation, and
- $\lambda : P \to \mathscr{P}(S)$ is a mapping which assigns a set of states to every propositional variable.

Transition systems are also known as Kripke structures. If we consider the inverse mapping $\gamma^{-1} : S \to \mathscr{P}(P)$, then we can regard transition systems as labeled, directed graphs. For every variable $p \in P$ and every state $s \in \lambda(p)$, we say that p is true in s, and for $s \in S \setminus \lambda(p)$, we say that p is false in s.

For every $s \in S$, we denote

$$sR = \{s' \in S \,|\, (s, s') \in R\} \qquad \text{and} \qquad Rs = \{s' \in S \,|\, (s', s) \in R\}.$$

A **pointed transition system** $(\mathcal{S}, s_\mathrm{I})$ is a transition system $\mathcal{S} = (S, R, \lambda)$ with an initial state $s_\mathrm{I} \in S$. We call a transition system $\mathcal{S} = (S, R, \lambda)$ (resp. $(\mathcal{S}, s_\mathrm{I})$) **finite** iff S is finite and $\lambda(p) \neq \emptyset$ for just finitely many $p \in P$.

9.3 Alternating Tree Automata

Our notion of an alternating tree automaton originates from [202]. An alternating tree automaton is a device which accepts or rejects pointed transition systems by parsing the paths.

In Subsection 9.3.1, we define alternating tree automata formally. In Subsection 9.3.2, we consider their semantics. At first, we discuss an ad hoc approach to define the behaviour of alternating tree automata on pointed transition systems. Then, we present two methods to define formally whether an alternating tree automaton accepts or rejects a pointed transition system. Both of these methods are based on parity games. The first method uses an infinite arena for almost

every transition system and is convenient in some proofs, for instance, to show the closure under complement. The second method uses a more compact arena, in particular, a finite one if the transition system in question is finite, and is used to examine the complexity of the word problem. In Proposition 9.2, we show that these two ways to define the semantics are equivalent.

In Section 9.3.3, we show a small lemma which is used to show the complexity of the emptiness problem.

In Section 9.3.4, we discuss a syntactic extension to our concept of alternating tree automata.

9.3.1 Formal Definition of Alternating Tree Automata

To define alternating tree automata formally, we need the notion of transition conditions. For now, let Q be some set of symbols. The **transition conditions** TC^Q over Q are defined as follows:

- The symbols 0 and 1 are transition conditions.
- For every $p \in P$, p and $\neg p$ are transition conditions.
- For every $q \in Q$, q, $\Box q$, and $\Diamond q$ are transition conditions.
- For every $q_1, q_2 \in Q$, $q_1 \wedge q_2$ and $q_1 \vee q_2$ are transition conditions.

Note that this definition does not allow transition conditions like $q_1 \wedge \Box q_2$ or $p \wedge q$ for $p \in P$ and $q \in Q$. Below, we will explain a method to allow these more complex transition conditions without violating our definition. An **alternating tree automaton** is a tuple $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ where

- Q is a finite set of **states** of the automaton,
- $q_\mathrm{I} \in Q$ is a state called the **initial state**,
- $\delta : Q \to \mathrm{TC}^Q$ is called **transition function**, and
- $\Omega : Q \to \omega$ is called **priority function**.

For convenience, we denote

$$Q_\Box := \big\{ q \in Q \,\big|\, \exists q' \in Q : \delta(q) = \Box q' \big\} \qquad \text{and}$$
$$Q_\Diamond := \big\{ q \in Q \,\big|\, \exists q' \in Q : \delta(q) = \Diamond q' \big\}.$$

For states $q \in Q$ we define $\overrightarrow{q} := q'$ if $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$. Otherwise, $\overrightarrow{q}$ is not defined. For subsets $V \subseteq Q$, we define $\overrightarrow{V} := \{\, \overrightarrow{q} \,|\, q \in V \,\}$.

9.3.2 The Behavior of Alternating Tree Automata

Informal Explanation. We examine the behavior of an alternating tree automaton $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ on a pointed transition system $(\mathcal{S}, s_\mathrm{I})$. At first, we follow a straightforward approach without using parity games:

In every step, the automaton is in some state $q \in Q$, and it inspects some state $s \in S$ of the transition system. We can describe the situation by a pair $(q, s) \in Q \times S$. We call the pairs in $Q \times S$ instances.

In the beginning, the automaton is in the initial state q_I and inspects the state s_I of the alternating tree automaton.

Now, assume that the automaton is in the state q and it inspects the state s, i.e., the current instance is (q, s). The automaton tries to execute the transition condition $\delta(q)$. If $\delta(q) \in \{0, 1\}$, $\delta(q) = p$, or $\delta(q) = \neg p$ for some $p \in P$, then the automaton needs not to take any action.

If $\delta(q) = q' \in Q$ then the automaton changes into the state q', but it does not move to another state of the transition system, i.e., the new situation is (q', s). If $\delta(q) = q_1 \wedge q_2$ or $\delta(q) = q_1 \vee q_2$, then the automaton splits itself into two instances (q_1, s) and (q_2, s). If $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$, then the automaton parses the relation R of $\mathcal{S}$. The automaton splits into several instances. These instances are in state q' and inspect the successors of s in $\mathcal{S}$, i.e., for every $(s, s') \in R$ we get an instance (q', s'). Thus, the set of new instances is $\{q'\} \times sR$.

The result of this process is a possibly infinite parse tree with instances as nodes. The main question is how does this tree determine whether $\mathcal{A}$ accepts or rejects the pointed transition system $(\mathcal{S}, s_\mathrm{I})$. To answer this question, we try to develop a notion of a "successful instance". If $\delta(q)$ is a propositional variable p and p is true in the state s, then the instance (q, s) is successful. Similarly, if $\delta(q) = \neg p$ and $s \notin \lambda(p)$, then the instance is successful. Conversely, if $\delta(q) = p$ but $s \notin \lambda(p)$ (or $\delta(q) = \neg p$ but $s \in \lambda(p)$), then the instance is not successful. If $\delta(q) = 1$, then the instance succeeds, but if $\delta(q) = 0$, then it does not succeed.

If $\delta(q) = q'$, then we have seen above that the automaton changes its state to q', i.e., the new situation is (q', s). Straightforwardly, we simply say that the instance (q, s) is successful iff (q', s) is successful.

If $\delta(q) = q_1 \wedge q_2$, then the instance (q, s) succeeds iff both the instances (q_1, s) and (q_2, s) succeed. If $\delta(q) = q_1 \vee q_2$, then the instance succeeds iff at least one of the instances (q_1, s) and (q_2, s) succeeds.

The case $\delta(q) = \Box q'$ is very similar to the case $\delta(q) = q_1 \wedge q_2$, above. If $\delta(q) = \Box q'$, then the instance (q, s) succeeds iff for *every* $s' \in sR$ the instance (q', s') succeeds. Finally, if $\delta(q) = \Diamond q'$, then the instance (q, s) succeeds iff there is *at least one* $s' \in sR$ such that (q', s') succeeds.

The automaton accepts the transition system $(\mathcal{S}, s_\mathrm{I})$ iff the initial instance $(q_\mathrm{I}, s_\mathrm{I})$ succeeds.

If we try to formalize this idea of the notion of a "successful instance" then we will encounter problems:

- If the parse tree is infinite, then successful instances cannot be determined in a bottom-up-fashion.
- If $\delta(q) = q'$, the we simply said that the instance (q, s) is successful iff (q', s) is successful. However, if $\delta(q) = q$, then we end up in an infinite loop.

We resolve these problems by viewing the "evaluation problem" as solving a certain game where infinite plays—that is where we run into problems—are decided according to some acceptance (winning) condition that we have seen in earlier chapters.

Formal Definition. Now, we solve these problems by defining the acceptance of alternating tree automata using parity games.

Let $(\mathcal{S}, s_\mathrm{I})$ be a pointed transition system, and let $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ be an alternating tree automaton. To define the behavior of a $\mathcal{A}$ on $(\mathcal{S}, s_\mathrm{I})$, we consider sequences of pairs from $Q \times S$, i.e., we consider words over the alphabet $Q \times S$.

For a word $v \in (Q \times S)^*$ and a letter $(q, s) \in Q \times S$, the notation $v(q, s)$ denotes the concatenation of v and (q, s).

The **behavior** of $\mathcal{A}$ on $(\mathcal{S}, s_\mathrm{I})$ is the least language $V \subseteq (Q \times S)^*$ with $(q_\mathrm{I}, s_\mathrm{I}) \in V$ such that for every word $v(q, s) \in V$ we have:

- If $\delta(q) = q'$ for some $q' \in Q$, then $v(q, s)(q', s) \in V$.
- If $\delta(q) = q_1 \wedge q_2$ or $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in Q$, then $v(q, s)(q_1, s) \in V$ and $v(q, s)(q_2, s) \in V$.
- If $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$ for some $q' \in Q$, then $v(q, s)(q', s') \in V$ for every $s' \in sR$.

We use parity games to define acceptance. At first, we define an arena (V_0, V_1, E). We split the behavior V into V_0 and V_1 to define the locations of Player 0 and Player 1. Some word $v(q, s) \in V$ belongs to V_0 iff one of the following conditions holds:

- $\delta(q) = 0$,
- $\delta(q) = p$ and $s \notin \lambda(p)$,
- $\delta(q) = \neg p$ and $s \in \lambda(p)$,
- $\delta(q) = q'$,
- $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in Q$, or
- $\delta(q) = \Diamond q'$.

Conversely, some word $v(q, s) \in V$ belongs to V_1 iff one of the following conditions holds:

- $\delta(q) = 1$,
- $\delta(q) = p$ and $s \in \lambda(p)$,
- $\delta(q) = \neg p$ and $s \notin \lambda(p)$,
- $\delta(q) = q_1 \wedge q_2$ for some $q_1, q_2 \in Q$, or
- $\delta(q) = \Box q'$.

Clearly, V_0 and V_1 are a partition of V. We complete the definition of the parity game by defining the moves and the priority mapping:

- $E := \big\{ \big(v, v(q, s)\big) \;\big|\; v(q, s) \in V,\, v \neq \epsilon \big\}$
- $\Omega\big(v(q, s)\big) := \Omega(q)$ for every $v(q, s) \in V$

As explained above, $(q_\mathrm{I}, s_\mathrm{I})$ is the initial location.

The automaton $\mathcal{A}$ **accepts** the pointed transition system $(\mathcal{S}, s_\mathrm{I})$ iff there is a winning strategy for Player 0 in the parity game $\mathcal{G} = \big((V_0, V_1, E),\, \Omega,\, (q_\mathrm{I}, s_\mathrm{I})\big)$. The language of $\mathcal{A}$ consists of the pointed transition systems which $\mathcal{A}$ accepts and is denoted by $L(\mathcal{A})$.

Example 9.1. At first, we consider several very simple alternating tree automata with $Q = \{q_\mathrm{I}\}$.

(1) Let $\delta(q_\mathrm{I}) = \Box q_\mathrm{I}$ and $\Omega(q_\mathrm{I}) = 0$. Let $(\mathcal{S}, s_\mathrm{I})$ be any pointed transition system. Player 0 has not any location in $\mathcal{G}'$. However, Player 1 cannot win. He looses every finite play. He also looses every infinite play, because the only priority is 0. Hence, the automaton accepts every pointed transition system.
(2) Let $\delta(q_\mathrm{I}) = \Box q_\mathrm{I}$ and $\Omega(q_\mathrm{I}) = 1$. Again, Player 0 has no location. Let $(\mathcal{S}, s_\mathrm{I})$ be any pointed transition system with some infinite path starting at s_I. Player 1 can win the game by playing along the infinite path.
Conversely, let $(\mathcal{S}, s_\mathrm{I})$ be any pointed transition system in which every path starting from s_I is finite. There are just finite plays in $\mathcal{G}'$. Thus, Player 1 looses every play in $\mathcal{G}'$.
Consequently, the automaton accepts every pointed transition system $(\mathcal{S}, s_\mathrm{I})$ which has no infinite path starting at s_I.
(3) Let $\delta(q_\mathrm{I}) = \Diamond q_\mathrm{I}$ and $\Omega(q_\mathrm{I}) = 1$. This automaton accepts not any pointed transition system.

Exercise 9.1. Construct alternating tree automata for the following languages.

(1) The language of all pointed transition systems where p is true in the designated state.
(2) The language of all pointed transition systems that have an infinite path starting in the designated state.
(3) The language of all pointed transition systems where on each infinite path starting in the designated state p is true only finitely often.

Exercise 9.2. Let $(\mathcal{S}, \int_\mathrm{I})$ and $(\mathcal{S}', \int'_\mathrm{I})$ be two pointed transition systems and assume ρ is a bisimulation between the two systems, that is, $\rho \subseteq S \times S'$ such that the following holds true.

(1) $(s_\mathrm{I}, s'_\mathrm{I}) \in \rho$.
(2) For all $(s, s') \in \rho$ and $p \in P$, p holds in s iff p' holds in s'.
(3) For all $(s, s') \in \rho$ and $\hat{s} \in sR$ there exists $\hat{s}' \in s'R'$ such that $(\hat{s}, \hat{s}') \in \rho$.
(4) For all $(s, s') \in \rho$ and $\hat{s}' \in s'R'$ there exists $\hat{s} \in sR$ such that $(\hat{s}, \hat{s}') \in \rho$.

Show that for every alternating tree automaton $\mathcal{A}$ the following is true. $\mathcal{A}$ accepts $(\mathcal{S}, s_\mathrm{I})$ iff $\mathcal{A}$ accepts $(\mathcal{S}', s'_\mathrm{I})$.

An Alternative Formal Definition. A disadvantage of the parity game $\mathcal{G}$ defined is that its arena is possibly infinite, even if (S, s_I) is finite. Moreover, even if there is no infinite path in (S, s_I) the game $\mathcal{G}$ can be infinite. We need some more convenient way to define the behavior, in particular, to show the decidability of the word problem, below.

We still assume $(\mathcal{S}, s_\mathrm{I})$ and $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ from above. Let $[V] \subseteq Q \times S$ and $[E] \subseteq [V] \times [V]$ be the smallest graph with $(q_\mathrm{I}, s_\mathrm{I}) \in [V]$ such that for every $(q, s) \in [V]$ we have:

- If $\delta(q) = q'$ for some $q' \in Q$, then $(q', s) \in V$ and $\big((q, s), (q', s)\big) \in [E]$.
- If $\delta(q) = q_1 \wedge q_2$ or $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in Q$, then (q_1, s), $(q_2, s) \in [V]$ and $\big((q, s), (q_1, s)\big)$, $\big((q, s), (q_2, s)\big) \in [E]$.
- If $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$ for some $q' \in Q$, then $(q', s') \in [V]$ and $\big((q, s), (q', s')\big) \in [E]$ for every $s' \in sR$.

To define an arena from $[V]$ and $[E]$, we split $[V]$ into $[V_0]$ and $[V_1]$ as above. We simply use the priority mapping Ω and the initial location $(q_\mathrm{I}, s_\mathrm{I})$. We define a parity game by

$$\mathcal{G}' := \big(([V_0], [V_1], [E]),\ \Omega,\ (q_\mathrm{I}, s_\mathrm{I})\big).$$

Let $[\,] : (Q \times S)^+ \to (Q \times S)$ be the mapping which assigns every word in $(Q \times S)^+$ the last letter. Thus, $[\,]$ maps locations from V to $[V]$. Moreover, $[\,]$ preserves edges and priorities, and Player 0's and Player 1's locations are mapped to Player 0's and Player 1's locations, respectively. Consequently, $\mathcal{G}'$ can be obtained by applying $[\,]$ to $\mathcal{G}$.

Proposition 9.2. *Player 0 has a winning strategy in $\mathcal{G}$ iff Player 0 has a winning strategy in $\mathcal{G}'$.*

Proof. At first, we observe that the mapping $[\,]$ can be extended to plays by applying $[\,]$ to every location in the play. Thus, $[\,]$ transforms plays in $\mathcal{G}$ to plays in $\mathcal{G}'$. If π' is some play in $\mathcal{G}'$, then there is a unique play π in $\mathcal{G}$ such that $[\pi] = \pi'$. Note that π is simply the sequence of all prefixes of π'. A play π in $\mathcal{G}$ is won by Player 0 iff $[\pi]$ in $\mathcal{G}'$ is won by Player 0. Hence, $[\,]$ is a "winner-preserving" bijection between plays in $\mathcal{G}$ and plays in $\mathcal{G}'$.

To prove Proposition 9.2, we have to show that Player 0 (resp. 1) has a winning strategy in $\mathcal{G}$ if Player 0 (resp. 1) has a winning strategy in $\mathcal{G}'$.

Let $f'_0 : [V_0] \to [V]$ be a winning strategy for Player 0 in $\mathcal{G}'$. We define a mapping $f_0 : V_0 \to V$ by $f_0(v) := v f'_0([v])$. Let π be a play in $\mathcal{G}$ which is consistent with f_0. Then, $[\pi]$ is consistent with f'_0, and thus, $[\pi]$ and π are won by Player 0. Consequently, f_0 is a winning strategy for Player 0 in $\mathcal{G}$.

Clearly, we can apply a symmetric argument if $f'_1 : [V_1] \to [V]$ is a winning strategy for Player 1 in $\mathcal{G}'$. □

9.3.3 Inflated Transition Conditions

We call transition conditions of the form $q \wedge q$ and $q \vee q$ for $q \in Q$ **inflated**. The following lemma allows to simplify some technical details, later.

Lemma 9.3. *For every alternating tree automaton $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ there is an automaton $\mathcal{A}' = (Q, q_\mathrm{I}, \delta', \Omega)$ with $L(\mathcal{A}) = L(\mathcal{A}')$ such that for every $q \in Q$ $\delta'(q)$ is not inflated.*

Proof. We define $\delta' : Q \to \mathrm{TC}^Q$ for $q \in Q$ by

$$\delta'(q) := \begin{cases} q' & , \text{ if } \delta(q) = q' \wedge q' \text{ for some } q' \in Q \\ q' & , \text{ if } \delta(q) = q' \vee q' \text{ for some } q' \in Q \\ \delta(q) & , \text{ otherwise} \end{cases}$$

Clearly, $\delta'(q)$ is not inflated for $q \in Q$.

Let $(\mathcal{S}, s_\mathrm{I})$ be some pointed transition system. We want to show that $\mathcal{A}$ accepts $(\mathcal{S}, s_\mathrm{I})$ iff $\mathcal{A}'$ accepts $(\mathcal{S}, s_\mathrm{I})$. At first, we observe that $\mathcal{A}$ has the same behavior V on $(\mathcal{S}, s_\mathrm{I})$ as $\mathcal{A}'$. Let $\mathcal{G} = \big((V_0, V_1, E),\ \Omega,\ (q_\mathrm{I}, s_\mathrm{I})\big)$ be the parity game to determine whether $\mathcal{A}$ accepts $(\mathcal{S}, S_\mathrm{I})$.

Let $\hat{V} = \big\{\, v(q,s) \in V_1 \,\big|\, \delta(q) = q' \wedge q' \text{ for some } q' \in Q \,\big\}$. The locations in $\hat{V}$ have exactly one successor. The parity game

$$\mathcal{G}' = \big((V_0 \cup \hat{V},\ V_1 \setminus \hat{V},\ E),\ \Omega,\ (q_\mathrm{I}, s_\mathrm{I})\big)$$

determines whether $\mathcal{A}'$ accepts $(\mathcal{S}, s_\mathrm{I})$.

The plays in $\mathcal{G}$ are exactly the plays in $\mathcal{G}'$. The locations in $\hat{V}$ cannot be the last location in a play and the priority mappings in $\mathcal{G}$ and $\mathcal{G}'$ are the same. Thus, some play π in $\mathcal{G}$ is won by Player 0 iff π is won by Player 0 in $\mathcal{G}'$.

Let $f_0 : V_0 \to V$ be a winning strategy for Player 0 in $\mathcal{G}$. There is a unique extension $f_0' : V_0 \cup \hat{V} \to V$. Now, assume some play π in $\mathcal{G}'$ which is consistent with f_0'. Then, π in $\mathcal{G}'$ is consistent with f_0, and thus π is won by Player 0.

Conversely, let $f_1 : V_1 \to V$ be a winning strategy for Player 1 in $\mathcal{G}$. Clearly, the restriction of f_1 to $V_1 \setminus \hat{V}$ is a winning strategy for Player 1 in $\mathcal{G}'$.

Consequently, Player 0 has a winning strategy in $\mathcal{G}$ iff he has a winning strategy in $\mathcal{G}'$, i.e., $\mathcal{A}$ accepts $(\mathcal{S}, s_\mathrm{I})$ iff $\mathcal{A}'$ accepts $(\mathcal{S}, s_\mathrm{I})$. □

9.3.4 Complex Transition Conditions

Our definition of transition conditions TC^Q is restrictive. One could imagine more complex transition conditions. For instance, there are situations in which a condition φ like "Change the inner state to q_1 if p is true, otherwise change the inner state to q_2." or formally $\varphi = (q_1 \wedge p) \vee (q_2 \wedge \neg p)$ is convenient although such a condition does not belong to TC^Q.

To model such a condition, we introduce new states q_φ, $q_{(q_1 \wedge p)}$, $q_{(q_2 \wedge \neg p)}$, q_p, $q_{\neg p}$, and we define:

$$\begin{aligned}
\delta(q_\varphi) &:= q_{(q_1 \wedge p)} \vee q_{(q_2 \wedge \neg p)} \\
\delta(q_{(q_1 \wedge p)}) &:= q_1 \wedge q_p \\
\delta(q_{(q_2 \wedge \neg p)}) &:= q_2 \wedge q_{\neg p} \\
\delta(q_p) &:= p \\
\delta(q_{\neg p}) &:= \neg p
\end{aligned}$$

This can be easily generalized:

Remark 9.4. Alternating tree automata where transition conditions are built up from 0, 1, p, and $\neg p$ using $\vee$, $\wedge$, $\Diamond$, and $\Box$ in any way are no more powerful than ordinary alternating tree automata.

Example 9.5. We consider the states q_φ, $q_{(q_1 \wedge p)}$, $q_{(q_2 \wedge \neg p)}$, q_p, $q_{\neg p}$ from above, and we complete the definition of δ by $\delta(q_1) = \delta(q_2) = \Box q_\varphi$. We set $\Omega(q_1) = 2$ and we set the priorities of the other states to 1. Let q_φ be the initial state.

The automaton accepts some pointed transition system $(\mathcal{S}, s_\mathrm{I})$ iff every infinite path starting from s_I contains infinitely many states in which p is true.

Exercise 9.3. Describe an alternating tree automaton which accepts a pointed transition system $(\mathcal{S}, s_{\mathrm{I}})$ iff for any two states $s_1, s_2 \in S$ with $s_1 R s_2$ the variable p is true in s_1 iff p is not true in s_2.

9.4 The Word Problem

In this section, we deal with the word problem, which means to decide whether a given alternating tree automaton $\mathcal{A}$ accepts a given finite pointed transition system $(\mathcal{S}, s_{\mathrm{I}})$. In Section 10, this result will be used to determine the complexity of the model checking problem for the modal μ-calculus.

We cannot solve the word problem by computing the whole behavior, because the behavior is possibly infinite, even if $(\mathcal{S}, s_{\mathrm{I}})$ is finite. However, we can reduce the parity game from the previous section to a finite parity game.

Theorem 9.6. *Let $\mathcal{A} = (Q, q_{\mathrm{I}}, \delta, \Omega)$ be an alternating tree automaton with d different non-zero priorities and let $(\mathcal{S}, s_{\mathrm{I}})$ be a finite pointed transition system.*

(1) *There is an algorithm which computes in time*

$$\mathcal{O}\left(d\,|Q|\,(|R|+1)\Big(\frac{|Q||S|+1}{\lceil d/2\rceil}\Big)^{\lceil d/2\rceil}\right)$$

and in space $\mathcal{O}\left(d\,|Q|\,|S|\,\log\left(|Q|\,|S|\right)\right)$ whether $\mathcal{A}$ accepts $(\mathcal{S}, s_{\mathrm{I}})$.

(2) *The problem whether $\mathcal{A}$ accepts $(\mathcal{S}, s_{\mathrm{I}})$ is in UP $\cap$ co-UP.*

Before we turn to the proof, let us understand the upper bound on the time complexity stated in the first part of the theorem. Let us consider the transition system complexity, i.e., we fix an automaton $\mathcal{A}$ and consider the complexity in dependence on the pointed transition system $(\mathcal{S}, s_{\mathrm{I}})$. Then, $d|Q|$ is a constant factor. Clearly, $|R|$ cannot exceed $|S|^2$. Hence, we roughly estimate $|R|+1$ by $|S|^2$ and simplify the above formula to $\mathcal{O}\left(|S|^2\,(|Q||S|)^{\lfloor d/2\rfloor}\right)$. Roughly spoken, the run-time of the algorithm is proportional to $|S|^{2+\lfloor d/2\rfloor}$. If for example $d = 2$, then the run-time is proportional to $|S|^3$, i.e., one can determine whether $\mathcal{A}$ accepts $(\mathcal{S}, s_{\mathrm{I}})$ in reasonable time. However, if $d = 20$ then the run-time of the algorithm is proportional to $|S|^{12}$. Then, the word problem will be practically unsolvable for reasonably big pointed transition systems.

Proof. The complexity of the word problem is in UP $\cap$ co-UP, because the problem to decide whether Player 0 has a winning strategy is in UP $\cap$ co-UP as explained in Chapter 6.

To prove the first part, we apply Jurdziński's result to the parity game $\mathcal{G}'$.

To prove the complexity bound of the word problem, we have to examine carefully the number of locations and moves, i.e., we have to estimate $|[V]|$ and $|[E]|$ (cf. [202]). The set of locations $[V]$ is a subset of $Q \times S$, i.e., there are at most $|Q||S|$ locations. Let $S' \subseteq S$ be the set of states in $(\mathcal{S}, s_{\mathrm{I}})$ which are reachable from q_{I}. Every state in S' except s_{I} has at least one predecessor. Hence, $|R| \geq |S'| - 1$.

To determine $|[E]|$, we count the number of successors of every location in $|[V]|$. The successors of a location $(q, s) \in [V]$ are $(q, s)[E]$. We have

$$|[E]| \;=\; \sum_{(q,s)\in[V]} |(q,s)[E]| \;=\; \sum_{q\in Q} \sum_{(q,s)\in[V]} |(q,s)[E]|.$$

Let $q \in Q$ be some state. We estimate the sum

$$\sum_{(q,s)\in[V]} |(q,s)[E]|.$$

If $\delta(q) \in \{0, 1\}$ or $\delta(q) \in \{p, \neg p\}$, then (q, s) has no successor, and we have

$$\sum_{(q,s)\in[V]} |(q,s)[E]| \;=\; 0.$$

If $\delta(q) \in Q$, then every location $(q, s) \in [V]$ has exactly one successor, i.e.,

$$\sum_{(q,s)\in[V]} |(q,s)[E]| \quad\leq\quad |S'| \quad\leq\quad |R| + 1.$$

If $\delta(q) = q_1 \wedge q_2$ or $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in Q$, then we have

$$\sum_{(q,s)\in[V]} |(q,s)[E]| \quad\leq\quad 2|S'| \quad\leq\quad 2(|R| + 1).$$

Now, assume $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$ for some $q' \in Q$. For every $(q, s) \in [V]$, we have $(q, s)[E] = \{q'\} \times sR$, i.e, $|(q, s)[E]| = |sR|$. We have

$$\sum_{(q,s)\in[V]} |(q,s)[E]| \quad=\quad \sum_{(q,s)\in[V]} |sR| \quad\leq\quad \sum_{s\in S} |sR| \quad=\quad |R|.$$

To sum up, we have $\sum_{(q,s)\in[V]} |(q,s)[E]| \leq 2(|R|+1)$ and $|[E]| \leq 2|Q|(|R|+1)$.

Now, we can apply Jurdziński's algorithm (Theorem 7.25, Section 7.5). □

9.5 Complementation

An advantage of alternating tree automata is the straightforward solution of the complementation problem: Given an alternating tree automaton $\mathcal{A}$, we can effectively construct an alternating tree automaton $\bar{\mathcal{A}}$ which accepts the complement of the language of $\mathcal{A}$. To prove the correctness of the construction we use the fact that parity games are determined in a crucial way. We follow ideas from [137].

Theorem 9.7. *Let $\mathcal{A} = (Q, q_\mathrm{I}, \delta, \Omega)$ be an alternating tree automaton. There is an alternating tree automaton $\bar{\mathcal{A}} = (Q, q_\mathrm{I}, \bar{\delta}, \bar{\Omega})$ such that $\bar{\mathcal{A}}$ accepts the complement of the language of $\mathcal{A}$.*

The definition of $\bar{\Omega} : Q \to \omega$ and $\bar{\delta} : Q \to \mathrm{TC}^Q$ is easy: We simply set for every $q \in Q$ the priority $\bar{\Omega}(q) = \Omega(q) + 1$ and

$$\bar{\delta}(q) := \begin{cases} 0 & , \text{ if } \delta(q) = 1 \\ 1 & , \text{ if } \delta(q) = 0 \\ \neg p & , \text{ if } \delta(q) = p \text{ for some } p \in P \\ p & , \text{ if } \delta(q) = \neg p \text{ for some } p \in P \\ q' & , \text{ if } \delta(q) = q' \text{ for some } q' \in Q \\ q_1 \wedge q_2 & , \text{ if } \delta(q) = q_1 \vee q_2 \text{ for some } q_1, q_2 \in Q \\ q_1 \vee q_2 & , \text{ if } \delta(q) = q_1 \wedge q_2 \text{ for some } q_1, q_2 \in Q \\ \Diamond q' & , \text{ if } \delta(q) = \Box q' \text{ for some } q' \in Q \\ \Box q' & , \text{ if } \delta(q) = \Diamond q' \text{ for some } q' \in Q \end{cases}$$

Proof. Let $(\mathcal{S}, s_\mathrm{I})$ be a pointed transition system and $\mathcal{G} = \big((V_0, V_1, E),\ \Omega, (q_\mathrm{I}, s_\mathrm{I})\big)$ be the parity game from Section 9.3.2 which determines whether $\mathcal{A}$ accepts $(\mathcal{S}, s_\mathrm{I})$. We show that $\mathcal{A}$ accepts $(\mathcal{S}, s_\mathrm{I})$ iff $\bar{\mathcal{A}}$ does not accept $(\mathcal{S}, s_\mathrm{I})$.

We examine the parity game $\bar{\mathcal{G}}$ which determines whether $\bar{\mathcal{A}}$ accepts $(\mathcal{S}, s_\mathrm{I})$. Intuitively, we simply change the ownership of every location, and we increase every priority by 1. Let $V = V_0 \cup V_1$ be the locations of $\mathcal{G}$ and $\mathcal{G}'$. Let $V' \subseteq V$ be the locations $v(q, s) \in V$ with $\delta(q) = q'$ for some $q' \in Q$. We do not change the ownership of locations in V'. The automaton $\bar{\mathcal{A}}$ accepts $(\mathcal{S}, s_\mathrm{I})$ iff there is winning strategy for Player 0 in the parity game

$$\bar{\mathcal{G}} = \big((V_1 \cup V',\ V_0 \setminus V',\ E),\ \bar{\Omega},\ (q_\mathrm{I}, s_\mathrm{I})\big).$$

Because parity games are determined (cf. Section 6.3), we have to show that there is a winning strategy for Player 0 in $\mathcal{G}$ iff there is no winning strategy for Player 1 in $\bar{\mathcal{G}}$. The argument is very similar to in the proof of Lemma 9.3. Therefore, it is left as Exercise 9.4. □

Exercise 9.4. Complete the proof of Theorem 9.7:

(1) Assume a winning strategy for Player 0 in $\mathcal{G}$ and construct a winning strategy for Player 1 in $\bar{\mathcal{G}}$.
(2) Assume a winning strategy for Player 1 in $\bar{\mathcal{G}}$ and construct a winning strategy for Player 0 in $\mathcal{G}$.

Exercise 9.5. Theorem 9.7 tells us that the languages recognizing by alternating tree automata are closed under complementation. Show that they are closed under intersection and union as well.

9.6 The Emptiness Problem

In this section, we show the decidability of the emptiness problem for alternating tree automata. As a byproduct, we show that an alternating tree automaton $\mathcal{A}$ accepts a finite pointed transition system if $\mathcal{A}$ accepts at least one transition

system. This result is used in Chapter 10 to show that the modal μ-calculus has the finite model property which means that every satisfyable formula in the modal μ-calculus has a finite model.

We fix some alternating tree automaton $\mathcal{A} = (Q, q_I, \delta, \Omega)$. By Lemma 9.3, we can assume that for every $q \in Q$ the transition condition $\delta(q)$ is not inflated.

At first, we give the notion of a tile, which is a graph consisting of states from $\mathcal{A}$ with various properties. We construct a parity game $\mathcal{T}$ from these tiles. In the parity game $\mathcal{T}$, Player 0 can use some arbitrary pointed transition system in $L(\mathcal{A})$ to construct a winning strategy. Conversely, if we assume some winning strategy for Player 0 in $\mathcal{T}$, we can construct some pointed transition system which $\mathcal{A}$ accepts.

9.6.1 Tiles

A **tile over Q** is a graph $\vartheta = (V_\vartheta, E_\vartheta)$ where $V_\vartheta \subseteq Q$, $E \subseteq V_\vartheta \times V_\vartheta$ and

(1) $\forall q \in V_\vartheta : \delta(q) \neq 0$
(2) $\neg \exists q_1, q_2 \in V_\vartheta \, \exists p \in P : \big(\delta(q_1) = p \wedge \delta(q_2) = \neg p\big)$
(3) $\forall q \in V_\vartheta : \quad \delta(q) = q_1 \longrightarrow (q, q_1) \in E_\vartheta$
(4) $\forall q \in V_\vartheta : \quad \delta(q) = q_1 \wedge q_2 \longrightarrow \big((q, q_1) \in E_\vartheta \wedge (q, q_2) \in E_\vartheta\big)$
(5) $\forall q \in V_\vartheta : \quad \delta(q) = q_1 \vee q_2 \longrightarrow \big((q, q_1) \in E_\vartheta \leftrightarrow (q, q_2) \notin E_\vartheta\big)$
(6) For every cycle in $(V_\vartheta, E_\vartheta)$ the maximal priority of its states is even.

Note that $(q, q_1) \in E_\vartheta$ in (3) (and similarly in (4) and (5)) implies $q_1 \in V_\vartheta$. Further, note that in condition (5) it is possible that both q_1 and q_2 belong to V_ϑ as long as exactly one of the pairs (q, q_1) or (q, q_2) belongs to E_ϑ. For condition (5), it is useful that there are no inflated transition conditions in $\mathcal{A}$.

A **tile with port** is a tuple (ϑ, q) where $\vartheta = (V_\vartheta, E_\vartheta)$ is some tile and $q \in V_\vartheta \cap Q_\Diamond$. We denote the set of all tiles and all tiles with port by Θ and Θ_p, respectively.

We call a tile with port $\vartheta_0 = (V_{\vartheta 0}, E_{\vartheta 0}, q_0)$ and a tile $\vartheta_1 = (V_{\vartheta 1}, E_{\vartheta 1})$ (similarly tile with port $\vartheta_1 = (V_{\vartheta 1}, E_{\vartheta 1}, q_{\vartheta 1})$) **concatenable** iff $\overrightarrow{q_0} \in V_{\vartheta 1}$ and $\overrightarrow{V_{\vartheta 0} \cap Q_\Box} \subseteq V_{\vartheta 1}$.

Let $g = (\vartheta_1, q_1), (\vartheta_2, q_2), \cdots \in \Theta^\omega$ be an infinite sequence of tiles with port where (ϑ_i, q_i) and $(\vartheta_{i+1}, q_{i+1})$ are concatenable for every $i \in \omega$. We define the **graph of g** in a usual way:

- $\mathcal{V} := \bigcup_{i \in \omega} \{i\} \times V_i$
- $\mathcal{E} := \bigcup_{i \in \omega} \big\{ \big((i, q'), (i, q'')\big) \mid (q', q'') \in E_i \big\} \cup \bigcup_{i \in \omega} \big\{ \big((i, q_i), (i+1, \overrightarrow{q_i})\big) \big\}$
  $\cup \bigcup_{i \in \omega} \big\{ \big((i, q), (i+1, \overrightarrow{q})\big) \mid q \in V_i \cap Q_\Box \big\}$

We call an infinite path π in $(\mathcal{V}, \mathcal{E})$ even iff the maximal priority which occurs in π infinitely often is even. We call the sequence g **even** iff every infinite path π in $(\mathcal{V}, \mathcal{E})$ is even.

There can be infinite paths π in $(\mathcal{V}, \mathcal{E})$ which get stuck in one tile, i.e., there is some integer i such that vertices (i', q) for any $i' > i$ and any $q \in Q$ do not occur in π. These paths π are even, because of (6) in the definition of a tile.

Proposition 9.8. *There is a deterministic parity ω-automaton $\mathcal{C}$ with $2^{\mathcal{O}(|Q|^4 \log |Q|)}$ states and priorities bounded by $\mathcal{O}(|Q|^4)$ which accepts a sequence of concatenable tiles $g \in \Theta^\omega$ iff g is even.*

Proof. At first, we construct a non-deterministic parity ω-automaton $\mathcal{B}$. Then, we construct $\mathcal{C}$ by a determinization and a complementation of $\mathcal{B}$.

The set of states of $\mathcal{B}$ are $Q \times \{0, \ldots, |Q|\}$. Thus, $\mathcal{B}$ has $m := |Q|(|Q|+1)$ states. The initial state of $\mathcal{B}$ is $(q_\mathrm{I}, 0)$.

We specify the transition function δ by a set of triples. Let (q_1, i_1), (q_2, i_2) be two states of $\mathcal{B}$, and let (V, E, q) be a tile with port. There is a transition $\big((q_1, i_1), (V, E, q), (q_2, i_2)\big)$ in $\mathcal{B}$ iff

- there is some state $q' \in V$ with $q' \in Q_\square$ or $q' = q$ and $\overrightarrow{q} = q_2$,
- there is some path in (V, E) which starts in q_1 and ends in q', and
- the maximal priority of the states in this path is i_2.

The priority of a state (q, i) is $i + 1$. Clearly, $\mathcal{B}$ accepts some infinite sequence of concatenable tiles iff this sequence is *not* even. Finally, we construct $\mathcal{C}$ in several steps:

(1) We convert $\mathcal{B}$ into a non-deterministic Büchi automaton $\mathcal{B}_1$ with $L(\mathcal{B}) = L(\mathcal{B}_1)$. This transformation is straightforward. The automaton $\mathcal{B}_1$ has $\mathcal{O}(m^2)$ states.
(2) We apply Safra's construction (see Chapter 4) and transform $\mathcal{B}_1$ into a deterministic Rabin-automaton $\mathcal{B}_2$. The automaton $\mathcal{B}_2$ has $2^{\mathcal{O}(m^2 \log m^2)}$ states and $\mathcal{O}(m^2)$ accepting pairs.
(3) We realize that $\mathcal{B}_2$ is a deterministic Streett automaton for the complement of the language of the Rabin-automaton $\mathcal{B}_2$ (see Chapter 1).
(4) We transform the Streett automaton $\mathcal{B}_2$ into a deterministic parity automaton $\mathcal{C}$ (see Chapter 1). The automaton $\mathcal{C}$ still has $2^{\mathcal{O}(m^2 \log m^2)}$ states and $\mathcal{O}(m^2)$ priorities. □

9.6.2 Parity Games over Tiles

We denote the set of states of the automaton $\mathcal{C}$ by $Q^\mathcal{C}$ an its initial state by $q_\mathrm{I}^\mathcal{C}$.

We construct a parity game $\mathcal{T}$ over tiles.

The locations are $\mathcal{V}_0 := Q^\mathcal{C} \times \Theta_p$ and $\mathcal{V}_1 := Q^\mathcal{C} \times \Theta$.

We define the set of moves $\mathcal{E}$. For every state $q^\mathcal{C} \in Q^\mathcal{C}$ and every tile with port $(\vartheta, q) \in \Theta_p$, there is a move from $(q^\mathcal{C}, \vartheta) \in \mathcal{V}_1$ to $(q^\mathcal{C}, \vartheta, q) \in \mathcal{V}_0$.

Let $(q^\mathcal{C}, \vartheta, q) \in \mathcal{V}_0$, and let $(q_1^\mathcal{C}, \vartheta_1) \in \mathcal{V}_1$. There is a move from $(q^\mathcal{C}, \vartheta, q)$ to $(q_1^\mathcal{C}, \vartheta_1)$ iff (ϑ, q) and ϑ_1 are concatenable and $\mathcal{C}$ admits a transition from $q^\mathcal{C}$ to $q_1^\mathcal{C}$ via (ϑ, q). Consequently, a move of Player 0 means to construct a tile, the state $q_1^\mathcal{C}$ is determined by the automaton $\mathcal{C}$. We can imagine Player 0 and 1 as "tile constructor" and "port selector", respectively.

We define the priority $\Omega_\mathcal{T}$ of a location $(q^\mathcal{C}, \vartheta)$ (resp. $(q^\mathcal{C}, \vartheta, q)$) as the priority of the state $q^\mathcal{C}$ in the parity automaton $\mathcal{C}$.

For convenience, we define a set of initial locations: Every location $(q_{\mathrm{I}}^{\mathcal{C}}, \vartheta)$ of Player 0 is an initial location iff $q_{\mathrm{I}} \in V_\vartheta$. As the very first action in a play Player 0 chooses one of these initial locations. A winning strategy for Player 0 has additionally to specify some initial location which Player 0 has to choose to start the game. To know whether Player 0 has a winning strategy in some parity game with multiple initial locations, we calculate Player 0's winning region by Jurdziński's algorithm and check whether an initial place belongs to Player 0's winning region.

Theorem 9.9. *The following three assertions are equivalent:*

(1) *The automaton $\mathcal{A}$ accepts at least one pointed transition system.*
(2) *There is a winning strategy for Player 0 in $\mathcal{T}$.*
(3) *The automaton $\mathcal{A}$ accepts some pointed transition system with at most $2^{\mathcal{O}(|Q|^4 \log |Q|)}$ states.*

Proof. (1) ⇒ (2) Let $(\mathcal{S}, s_{\mathrm{I}})$ be some pointed transition system which $\mathcal{A}$ accepts. We consider the parity game $\mathcal{G}'$ from the proof of Theorem 9.6. Let $f : [V_0] \to [V]$ be a memoryless winning strategy for Player 0 in $\mathcal{G}'$. We construct a winning strategy for Player 0 in $\mathcal{T}$. The winning strategy which we construct is not necessarily memoryless.

At first, we show a mechanism how Player 0 can construct tiles. He construct tiles outgoing from some set $V \subseteq Q$ w.r.t. some state $s \in S$. Player 0 starts his construction with $(V, \emptyset)$. He chooses some state $q \in V$, and adds new states and edges in order to satisfy the closure properties (3), (4), (5) in the definition of a tile. If for example $\delta(q) = q_1 \wedge q_2$, he adds two states q_1 and q_2 and two edges (q_I, q_1) and (q_I, q_2) to the tile. Then, he has to take care about both q_1 and q_2. For example, let $\delta(q_1) = q_3 \vee q_4$. To satisfy property (5), Player 0 has to choose between q_3 and q_4. He simply calculates $f(s, q_1)$. If $f(s, q_1) = (s, q_3)$, he adds state q_3 and the edge (q_1, q_3) to his tile. Conversely, if $f(s, q_1) = (s, q_4)$, he adds q_4 and (q_1, q_4) to his tile.

Now, we explain a winning strategy for Player 0. At the beginning, Player 0 constructs a tile outgoing from $\{q_{\mathrm{I}}\}$ w.r.t. s_{I}. Let us call this tile ϑ_1. Player 0 chooses $(q_{\mathrm{I}}^{\mathcal{C}}, \vartheta_1)$ as initial location. Next, Player 1 chooses some port, i.e., he chooses a state from $q \in V_{\vartheta 1} \cap Q_\diamond$ and moves to $(q_{\mathrm{I}}^{\mathcal{C}}, \vartheta_1, q)$.

Then, Player 0 has to move to a state/tile pair $(q_2^{\mathcal{C}}, \vartheta_2)$. It suffices to construct ϑ_2, because $q_2^{\mathcal{C}}$ is determined by $\mathcal{C}$. Let $f(s_{\mathrm{I}}, q) = (s', q')$. Player 0 constructs ϑ_2 outgoing from $\overrightarrow{V_{\vartheta 1} \cap Q_\square} \cup \{\overrightarrow{q}\}$ w.r.t. s'.

It is easy but technically involved to verify that this technique yields a winning strategy for Player 0.

(2) ⇒ (3) Let $f : \mathcal{V}_0 \to \mathcal{V}_1$ be a memoryless winning strategy for Player 0 in the parity game $\mathcal{T}$.

We construct a pointed transition system which $\mathcal{A}$ accepts. Its states are Player 1's locations $\mathcal{V}_1 = Q^{\mathcal{C}} \times \Theta$. We can estimate $|\mathcal{V}_1|$ by $|Q^{\mathcal{C}}|\,|\Theta|$, which is $2^{\mathcal{O}(|Q|^4 \log |Q|)} \cdot 2^{|Q|+|Q|^2}$, i.e., $2^{\mathcal{O}(|Q|^4 \log |Q|)}$.

To define $\lambda : P \to \wp(\mathcal{V}_0)$, condition (2) in the definition of a tile is crucial. For some $p \in P$ and some location $(q^{\mathcal{C}}, \vartheta) \in \mathcal{V}_0$, we let $(q^{\mathcal{C}}, \vartheta) \in \lambda(p)$ iff there is some state $q \in V_\vartheta$ with $\delta(q) = p$.

Let $(q_{\mathrm{I}}^{\mathcal{C}}, \vartheta_{\mathrm{I}}) \in \mathcal{V}_1$ be the location which Player 0 chooses as initial location. This location is the initial state of our pointed transition system. We define the accessibility relation: There is some edge from $(q_1^{\mathcal{C}}, \vartheta_1)$ to $(q_2^{\mathcal{C}}, \vartheta_2)$ iff there is some state $q \in V_{\vartheta 1} \cap Q_\Diamond$ such that $f(q_1^{\mathcal{C}}, \vartheta_1, q) = (q_2^{\mathcal{C}}, \vartheta_2)$, i.e., iff the winning strategy of Player 0 in $\mathcal{T}$ leads to $(q_2^{\mathcal{C}}, \vartheta_2)$.

It remains to show that $\mathcal{A}$ really accepts this pointed transition system. We consider the "small" parity game $\mathcal{G}'$ from the the proof of Theorem 9.6. Let $(q^{\mathcal{C}}, \vartheta, q)$ be some location of Player 0. If $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in Q$, then the winning strategy for Player 0 is determined within the tile ϑ itself. If $\delta(q) = \Diamond q_1$ for some $q_1 \in Q$, then Player 0 simply uses the winning strategy f from $\mathcal{T}$.

(3) $\Rightarrow$ (1) This is obvious. □

Corollary 9.10. *The problem whether some alternating tree automaton accepts at least one pointed transition system is decidable in* EXPTIME.

Exercise 9.6. Let $\mathcal{T}$ be a class of pointed transition systems and $P' \subseteq P$. The cylindrification of $\mathcal{T}$ with respect to P' consists of all pointed transition systems that coincide with some transition system from $\mathcal{T}$ on all propositions except the ones from P'. Show that if $\mathcal{T}$ is recognized by an alternating tree automaton, then so are its cylindrifications.

9.7 Acknowledgements

The author thanks Thomas Wilke for reading and improving a preliminary version of this chapter.