Simulation-Based Simplification of omega-Automata

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Carsten Fritz

Kiel
2005

1. Gutachter	Prof. Dr. Thomas Wilke
2. Gutachter	Prof. Dr. Willem-Paul de Roever
3. Gutachter	Prof. Dr. Moshe Y. Vardi
Datum der mündlichen Prüfung	10. Februar 2006

Danksagung

Von den vielen Menschen, die mich bei der Erstellung dieser Dissertation unterstützt haben, möchte ich Thomas Wilke besonders hervorheben. Thomas stand mir stets mit Rat und Hilfe zur Seite und hat mich als Doktorand vorbildlich betreut. Als begeisterter, sorgfältiger Forscher und liebenswerter Mensch ist er mir ein wichtiges Vorbild.

Gerne bedanke ich mich bei der Deutschen Forschungsgemeinschaft (DFG) für die finanzielle Unterstützung der Forschung, die zu dieser Arbeit geführt hat (Projektnummer 223228).

Sehr wichtig für meine Arbeit ist auch die gute Atmosphäre am Lehrstuhl für Theoretische Informatik. Hierfür bedanke mich sehr herzlich bei meinen Kollegen Detlef Kähler, Margrit Krause, Ralf Küsters, Brigitte Scheidemann, Erich Valkema und bei unserem Systemadministrator Thomas Heß (Herzlichen Glückwunsch zu den Zwillingen!).

Andreas Niemann und Björn Teegen haben durch Implementierungen für Experimente zu dieser Arbeit beigetragen. Kousha Etessami, damals Murray Hill, New Jersey, und David Janin, Bordeaux, haben mir ermöglicht, ihre Arbeitsgruppen zu besuchen.

Mein besonderer Dank gilt meinen Eltern für ihre stetige Liebe und Unterstützung.

Kiel, im November 2005
Carsten Fritz

Contents

Introduction

Finite automata on infinite words (ω-automata) were introduced in the early sixties of the last century, motivated by questions of logic [Büc60, Büc62] and network theory [Mul63]. They are important from a theoretical point of view, e. g., as a tool for decision procedures in logic. They are also important in system and program verification, where ω-automata are used to model non-terminating reactive systems (for example, operating systems) and their specifications. A prominent example for the use of ω-automata in verification is the automata-theoretic approach to model checking of Vardi and Wolper [VW86], where both the specification and the system to be checked are translated to or interpreted as ω-automata.

For such applications to be practicable, it is crucial that the considered automata, especially the automaton representing the specification, are small, but finding a minimal equivalent ω-automaton for a given automaton is a **PSPACE**-hard problem (this is even true for nondeterministic automata on finite words). Therefore, heuristics are used to minimize the state space of ω-automata. In this work, we study simulation relations [Mil71] as a tool for minimizing automata. Simulation relations capture the notion that the moves of one automaton can be mimicked by the moves of another automaton. That is, they can be used for checking language containment between automata, cf. [DHWT91]. This is of practical importance if, e. g., one automaton describes a system and another automaton describes the allowed computations of this system. If the language of the first automaton is contained in the language of the second automaton, then every computation of the system is an allowed computation and the system is correct w. r. t. this specification, and a sufficient condition for this to hold is that the second automaton simulates the first. But simulation relations can also be used for reducing the state space of automata and transition systems before space and time consuming algorithms are applied to them. For example, if two or more states mutually simulate each other, it may be possible to merge them into a single state and to continue the computation on a smaller quotient structure. This is done by, e. g., Etessami and Holzmann [EH00] and Somenzi and Bloem [SB00] for nondeterministic Büchi automata in the context of checking linear-time temporal properties.

In previous work, simulation relations have been introduced for ordinary and alternating transition systems, see, e. g., [Mil89, HRHK95, AHKV98], and simulation relations for nondeterministic Büchi automata are studied in , e. g., [HKR97, EH00, ESW01]. Different modes of simulation are also introduced and analyzed in, e. g., [Ete02, BG02] and recently in [JP06].

The focus of this work is on simulation relations for alternating automata. Alternation, introduced by Chandra, Kozen, and Stockmeyer [CKS81] for Turing machines and automata on finite words and applied to ω-automata by Muller, Schupp, and Saoudi [MSS86, MS87], formally describes the idea that a machine or an automaton splits into several copies in order to do computations in parallel; it is well-known that alternating automata are exponentially more succinct than nondeterministic automata. Especially, alternating automata are used to study modal and temporal logics from an automata-theoretic point of view. New automata-theoretic methods for verification based on alternating automata have been developed, see, e. g., [MSS88, Var94, KVW00], so that simulation relations for alternating automata are of practical interest.

Our approach to simulation relations is game-based and follows the ideas of [HKR97, AHKV98, ESW01]. That is, we define simulation via a game of two players, the Spoiler and the Duplicator. We say that an automaton simulates another automaton if the Duplicator, controlling the nondeterministic choices of the first automaton, has a winning strategy against the Spoiler who controls the nondeterministic choices of the second automaton.

In Chapter 1, we use this concept to extend the notion of simulation relations to alternating Büchi automata. We concentrate on three modes of simulation, called direct [DHWT91], delayed [ESW01], and fair simulation [HKR97]. We show that our simulation relations for alternating Büchi automata enjoy many of the properties known of the respective relations for nondeterministic Büchi automata, namely, they can be used for state space reduction and can be computed as fast as in the nondeterministic case. Also, we show that the standard construction from alternating to nondeterministic Büchi automata is compatible with simulation.

In Chapter 2, we apply our simulation relations to the problem of constructing an equivalent nondeterministic automaton for a formula of propositional linear-time temporal logic (LTL [Pnu77]). LTL is a popular specification language, and the step from an LTL formula to an equivalent nondeterministic automaton is important for verification, so that there are several implementations. These include an implementation as part of Holzmann's model checker *Spin* [Hol], Etessami's *Temporal Massage Parlor* (TMP) [Ete], *LTL2BA* [Odd] of Gastin and Oddoux, and *Wring* [Blo] by Somenzi and Bloem. We make use of the fact that an LTL formula can be seen as an alternating Büchi automaton, cf. [Var94]. We develop an algorithm that makes use of simulation-based simplifications on the level of this alternating automaton (that is, on the formula) as well as during the trans-

lation to a nondeterministic automaton, but we avoid a time-consuming computation of simulation relations for the nondeterministic automaton itself (the size of this automaton can be exponential in the length of the input formula). We compare our approach experimentally to other implementations, and we give a detailed analysis of our concept of translation versus the basic translation concept of [GPVW95, DGV99] used in the Spin implementation, in TMP, and in Wring.

In Chapter 3, we extend delayed simulation to alternating automata with parity acceptance conditions [Mos84]. Parity acceptance is important for its connection to the modal μ-calculus of Kozen [Koz83], cf. [EJ91], and, in general, as a powerful acceptance and winning condition for automata and games, respectively, cf. [Tho97]. We develop a polynomial-time simplification algorithm for parity automata based on adapted notions of delayed simulation and give a sketch of how to apply these ideas to the simplification of formulas of a fragment of the modal μ-calculus.

Pre-published results Most results of Chapter 1 were published in the journal article [FW05] which builds on the paper [FW02] and the technical report [FW01]. An earlier version of the simplification algorithm of Chapter 2 was presented in [Fri03]. Implementations of two versions of this algorithm (both based on delayed simulation) are accessible via CGI [FTb, FTa]. The analysis of tableau-based versus ABA-based translation from LTL in Chapter 2 is presented in [Fri05a] and available as a technical report [Fri05b].

Chapter 1

Simulation Relations for Alternating Büchi Automata

In the first chapter of this thesis, we combine what has been done for alternating transition systems [AHKV98] and nondeterministic Büchi automata (e. g., [DHWT91, ESW01]): We introduce and study simulation relations for alternating Büchi automata. Our definitions of the various simulation relations for alternating Büchi automata are game-based and follow the approach of [HKR97, AHKV98, ESW01]. The main technical difficulty to deal with are the two different types (existential and universal) of states present in alternating automata. Our definitions of the simulation relations are most general with respect to this distinction as we allow that a universal state simulates an existential state and vice versa. This yields smaller automata after quotienting, and, as we prove, does not increase the complexity of the algorithms.

Treating existential and universal states at the same time makes the situation complicated. The naive quotient construction, which was also used in [ESW01] for nondeterministic Büchi automata, does not work with alternating Büchi automata. For this reason, we introduce new quotients, which we call minimax and semi-elective quotients, and show that they can replace the naive quotient in the context of alternating Büchi automata. *Minimax quotients* with respect to direct simulation and *semi-elective quotients* with respect to direct as well as delayed simulation preserve the recognized languages. For nondeterministic Büchi automata, the minimax quotient corresponds to direct simulation with edge deletion, cf. [SB00], while the semi-elective quotient w. r. t. delayed simulation is the same as the quotient construction of [ESW01]. (Note that a quotient with respect to fair simulation usually cannot preserve the recognized language [ESW01], even in the absence of alternation, but see [GBS02] for a different minimization technique using fair simulation.) We also show that all three types of simulation relations can be used for checking language containment.

Most of our results, especially the more complicated ones, rely on a specific construction to compose strategies in simulation games, which is reminiscent of intruder-in-the-middle attacks known from cryptography. Most of the technical work goes into analyzing this strategy composition method.

This chapter is organized as follows. In Section 1.1, we review the basic definitions on alternating automata and two-player games on graphs, which are our main tool. We also introduce some basic definitions and notations that are used throughout this thesis in the first section. In Section 1.2, we present our definitions of the various simulation relations and prove that simulation implies language containment. Section 1.3 is the technical core of the chapter and lays the ground for proving that direct and delayed quotients preserve the language recognized. In Sections 1.4 and 1.5 the definitions of minimax and semi-elective quotient are presented and it is shown that these quotients preserve the language recognized. In Section 1.6, we show that our simulation relations are compatible with the standard translation of alternating Büchi automata to nondeterministic Büchi automata. Section 1.7 presents efficient algorithms for computing the simulation relations introduced, building on the ideas of [ESW01] who improved on the algorithms of [HKR97, HR00] by making use of Jurdziński's algorithm [Jur00] for solving parity games.

1.1 Notation and Basic Definitions

In this section, we fix basic notation and definitions. We describe the games which all our simulation relations for alternating Büchi automata are based on, and we review the definition of alternating Büchi automata used in this chapter.

The set of natural numbers is denoted ω. As usual, given a set Σ, we denote the set of finite, finite but nonempty, and infinite sequences over Σ by Σ^*, Σ^+, and Σ^ω, respectively. We set $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. Words over Σ are viewed as functions from an initial segment of ω or ω itself to Σ, so when w is a word, then $w(i)$ denotes the letter at its ith position where the first letter is in position 0, and $w[i..j]$ denotes the substring extending from position i through position j.

When R is a binary relation, then uR denotes $\{v \mid (u,v) \in R\}$; similarly, $Rv = \{u \mid (u,v) \in R\}$. For ternary relations R, we will also write $R(u)$ for $\{(v,w) \mid (u,v,w) \in R\}$ and $R(u,v)$ for $\{w \mid (u,v,w) \in R\}$. When t is an n-tuple, $\mathrm{pr}_i(t)$ is the ith component of t (for $1 \leq i \leq n$).

1.1.1 Games

For our purposes, a *game* is a tuple

$$G = (P, P_0, P_1, p_I, Z, W) \tag{1.1}$$

where P is the set of all *positions* of G, $\{P_0, P_1\}$ is a partition of P into the positions of Player 0 and Player 1, respectively, where $P_0 = \emptyset$ or $P_1 = \emptyset$ are allowed, $p_I \in P$ is the *initial position* of G, $Z \subseteq P \times P$ is the set of *moves* of G, and $W \subseteq P^\omega$ is the *winning set* of G. The directed graph (P, Z) is called the *game graph* of G and also denoted by G (with no danger of confusion).

A *play* in G is a maximal path through G starting in p_I; a *partial play* is any path through G starting in p_I. A play $\pi = p_0 p_1 p_2 \ldots$ is winning for Player 1 if π is infinite and $\pi \in W$, or if π is finite and the last position of π belongs to Player 0 (it is her turn, but she cannot move). In all other cases, Player 0 wins the play.

A *strategy for Player 0* is a partial function $\sigma\colon P^*P_0 \to P$ satisfying the following condition for every $\pi \in P^*$ and $p \in P_0$. If $pZ \neq \emptyset$, then $\sigma(\pi p) \in pZ$, else $\sigma(\pi p)$ is undefined. A partial play π is *conform with* σ *(σ-conform)* if for every i such that $i+1 < |\pi|$ and $\pi(i) \in P_0$, we have $\pi(i+1) = \sigma(\pi[0..i])$. The strategy σ is a *winning strategy* for Player 0 if every σ-conform play is winning for Player 0. Player 0 *wins* G if he has a winning strategy.—For Player 1, the same notions are defined by exchanging 0 with 1.

Note that if Player 0 plays according to a strategy τ and Player 1 plays according to a strategy σ, the resulting play is completely determined. This play is called the (τ, σ)-*conform* play.

In general, when σ is a strategy, not all partial plays are σ-conform, which means strategies need not be total functions. In fact, it is usually enough to require that a strategy for Player 0 is defined for all σ-conform partial plays $\pi \in P^*P_0$.

1.1.2 Alternating Büchi automata

For our purposes, an alternating Büchi automaton, ABA, for short, is a tuple

$$\mathbf{Q} = (Q, \Sigma, q_I, \Delta^q, E^q, U^q, F^q) \tag{1.2}$$

where Q is a finite *set of states,* Σ is a finite *alphabet,* $q_I \in Q$ is the *initial state,* $\{E^q, U^q\}$ is a partition of Q in *existential* and *universal states,* where $E^q = \emptyset$ or $U^q = \emptyset$ are allowed, $\Delta^q \subseteq Q \times \Sigma \times Q$ is the *transition relation*, and $F^q \subseteq Q$ is the set of *accepting states.* Automata will also be named $\mathbf{R}$ and $\mathbf{S}$, with their components named accordingly; we will always assume a common alphabet Σ. We will omit the superscripts when no confusion can arise.

Acceptance of alternating Büchi automata is best defined via games. For an alternating Büchi automaton $\mathbf{Q}$ as above and an ω-word $w \in \Sigma^\omega$, the *word game* $G(\mathbf{Q}, w)$ is a game where $P = Q \times \omega$ is the set of positions with $P_0 = U^q \times \omega$, $P_1 = E^q \times \omega$, $p_I = (q_I, 0)$, $Z = \{((q,i),(q',i+1)) \mid (q, w(i), q') \in \Delta\}$, and $W = (P^*(F^q \times \omega))^\omega$.

Following [GH82], in the above game, Player 1 is called *Automaton* while Player 0 is called *Pathfinder*. Acceptance is now defined as follows. The word

w is *accepted* by the automaton $\mathbf{Q}$ if Automaton wins the game $G(\mathbf{Q},w)$. The language *recognized* by $\mathbf{Q}$ is

$$L(\mathbf{Q}) = \{w \in \Sigma^\omega \mid \text{Automaton wins } G(\mathbf{Q},w)\} \ . \tag{1.3}$$

A *nondeterministic*, i. e., non-alternating Büchi automaton, is an automaton as in (1.2) with $U^q = \emptyset$.

For $q \in Q$, we will write $\mathbf{Q}(q)$ for the automaton which is obtained from $\mathbf{Q}$ by setting q as the new initial state, i. e., $\mathbf{Q}(q) = (Q,\Sigma,q,\Delta^q,E^q,U^q,F^q)$.

In figures, existential states are shown as diamonds and universal states as squares; accepting states have double lines, see, e. g., Figure 1.1.

1.2 Simulation Relations for Alternating Büchi Automata

In this section, we define three types of simulation relations for alternating Büchi automata, namely direct, delayed, and fair simulation, which are all based on the same simple game, only the winning condition varies. We show that all these simulations have the property that if an automaton simulates another automaton the language recognized by the latter is contained in the language recognized by the former—we say simulation implies language containment.

1.2.1 Direct, delayed, and fair simulation

Let $\mathbf{Q} = (Q,\Sigma,q_I,\Delta^q,E^q,U^q,F^q)$ and $\mathbf{S} = (S,\Sigma,s_I,\Delta^s,E^s,U^s,F^s)$ be alternating Büchi automata. The *basic simulation game* $G(\mathbf{Q},\mathbf{S})$ is played by two players, *Spoiler* and *Duplicator,* who play the game in rounds. At the beginning of each round, a pair (q,s) of states $q \in Q$ and $s \in S$ is given, and the players play as follows.

1. Spoiler chooses a letter $a \in \Sigma$.

2. The next step depends on the modes (existential or universal) of q and s.

— If $(q,s) \in E^q \times E^s$, then Spoiler chooses a transition $(q,a,q') \in \Delta^q$ and after that Duplicator chooses a transition $(s,a,s') \in \Delta^s$.

— If $(q,s) \in U^q \times U^s$, then Spoiler chooses a transition $(s,a,s') \in \Delta^s$ and after that Duplicator chooses a transition $(q,a,q') \in \Delta^q$.

— If $(q,s) \in E^q \times U^s$, then Spoiler chooses transitions $(q,a,q') \in \Delta^q$ and $(s,a,s') \in \Delta^s$.

— If $(q,s) \in U^q \times E^s$, then Duplicator chooses transitions $(q,a,q') \in \Delta^q$ and $(s,a,s') \in \Delta^s$.

3. The starting pair for the next round is (q',s').

Intuitively, Spoiler produces, letter by letter, an ω-word as simultaneous input for the automata $\mathbf{Q}$ and $\mathbf{S}$. Spoiler controls the nondeterministic choices of $\mathbf{Q}$ while Duplicator controls the nondeterministic choices of $\mathbf{S}$. This is reversed at universal states: A player loses control of "his" automaton, and the adversary gets to choose a successor state.

The first round begins with the pair (q_I, s_I). If, at any point during the course of the game, a player cannot proceed any more, he or she looses (early). When the players proceed as above and no player looses early, they construct an infinite sequence $(q_0,s_0)(q_1,s_1)\ldots$ of pairs of states (with $q_0 = q_I$ and $s_0 = s_I$), and this sequence determines the winner, depending on the type of simulation relation we are interested in:

Direct simulation (di): Duplicator wins if for every i with $q_i \in F^q$ we have $s_i \in F^s$.

Delayed simulation (de): Duplicator wins if for every i with $q_i \in F^q$ there exists $j \geq i$ such that $s_j \in F^s$.

Fair simulation (f): Duplicator wins if there are only finitely many i with $q_i \in F^q$ or infinitely many j with $s_j \in F^s$.

In all other cases, Spoiler wins. This completes the description of the games.

The games above can formally be described in the following way, using the game notion of the previous section. Spoiler takes over the role of Player 0, while Duplicator takes over the role of Player 1. The positions in the game reflect the status of a round. We have positions of the form (q,s) for the starting point of a round, and positions of the form (q,s,a,A,b,A',b') which represent the fact that the round started out in (q,s), Spoiler chose the letter a, player A (Spoiler or Duplicator) first has to pick a transition in $\mathbf{Q}$ if $b = 0$ or in $\mathbf{S}$ if $b = 1$, and after that player A' has to pick a position in $\mathbf{Q}$ or $\mathbf{S}$ (depending on b').[1] Finally, we have positions of the form (q,s,a,A',b') which represent the fact that Spoiler chose the letter a, and player A' still has to pick a transition in $\mathbf{Q}$ ($b' = 0$) or $\mathbf{S}$ ($b' = 1$). That is, in the formal definition of the game, we use

$$U_{sp} = Q \times S \times \Sigma \times \{sp\} \times \{0,1\} \times \{sp,du\} \times \{0,1\} \ , \tag{1.4}$$

$$U_{du} = Q \times S \times \Sigma \times \{du\} \times \{0,1\} \times \{sp,du\} \times \{0,1\} \ , \tag{1.5}$$

$$V_{sp} = Q \times S \times \Sigma \times \{sp\} \times \{0,1\} \ , \tag{1.6}$$

$$V_{du} = Q \times S \times \Sigma \times \{du\} \times \{0,1\} \ . \tag{1.7}$$

Given a game type $x \in \{di, de, f\}$, the game $G^x(\mathbf{Q},\mathbf{S})$ is defined by

$$G^x(\mathbf{Q},\mathbf{S}) = (P, P_0, P_1, (q_I, s_I), Z, W^x) \tag{1.8}$$

[1] The reader may have observed that in a position of the form (q,s,a,A,b,A',b'), the last four components, A, b, A' and b', are redundant, as they can be inferred from q and s. But our definition will facilitate reading the proofs later.

where

$$P = (Q \times S) \cup U_{sp} \cup U_{du} \cup V_{sp} \cup V_{du} \ , \tag{1.9}$$
$$P_0 = (Q \times S) \cup U_{sp} \cup V_{sp} \ , \tag{1.10}$$
$$P_1 = U_{du} \cup V_{du} \ , \tag{1.11}$$

and the set $Z \subseteq P \times P$ contains all moves of the form

$$((q,s),(q,s,a,sp,0,du,1)) \ , \quad \text{for } q \in E^q, s \in E^s, a \in \Sigma \ , \tag{1.12}$$
$$((q,s),(q,s,a,sp,0,sp,1)) \ , \quad \text{for } q \in E^q, s \in U^s, a \in \Sigma \ , \tag{1.13}$$
$$((q,s),(q,s,a,du,0,du,1)) \ , \quad \text{for } q \in U^q, s \in E^s, a \in \Sigma \ , \tag{1.14}$$
$$((q,s),(q,s,a,sp,1,du,0)) \ , \quad \text{for } q \in U^q, s \in U^s, a \in \Sigma \ , \tag{1.15}$$
$$((q,s,a,x,0,y,1),(q',s,a,y,1)) \ , \quad \text{for } (q,a,q') \in \Delta^q, x,y \in \{sp,du\} \ , \tag{1.16}$$
$$((q,s,a,sp,1,du,0),(q,s',a,du,0)) \ , \quad \text{for } (s,a,s') \in \Delta^s \ , \tag{1.17}$$
$$((q,s,a,du,0),(q',s)) \ , \quad \text{for } (q,a,q') \in \Delta^q \ , \tag{1.18}$$
$$((q,s,a,x,1),(q,s')) \ , \quad \text{for } (s,a,s') \in \Delta^s, x \in \{sp,du\} \ . \tag{1.19}$$

Note that not all positions are reachable from the initial position of the game or from any position in $\mathbf{Q} \times \mathbf{S}$. These unreachable positions can be removed (cf. Section 1.7), but if we did this here, this would make the proofs somewhat more complicated, so we keep them.

The winning condition depends on the type of simulation relation (see above). To phrase it concisely, we will use the following notation. We will write $\hat{F}^q$ for the set of all positions with an element from F^q in the first component and $\hat{F}^s$ for the set of all positions with an element from F^s in the second component. Also, we will write $\bar{F}^q$ and $\bar{F}^s$ for $P \setminus \hat{F}^q$ and $P \setminus \hat{F}^s$, respectively. Now we can state the winning conditions formally:

The *direct winning condition* is $W^{di} = ((\bar{F}^q \cup \hat{F}^s) \cap (Q \times S))^\omega$. (1.20)
Always at the beginning of a round, it must be the case that the first component is not accepting or the second component is accepting.

The *delayed winning condition* is $W^{de} = P^\omega \setminus P^*(\hat{F}^q \cap \bar{F}^s)(\bar{F}^s)^\omega$. (1.21)
It must not be the case that eventually the first component is accepting while the second component is not accepting and remains not accepting forever.

The *fair winning condition* is $W^f = P^\omega \setminus P^*((\hat{F}^q \cap \bar{F}^s)(\bar{F}^s)^*)^\omega$. (1.22)
It must not be the case that eventually the second component is never accepting while the first component is accepting infinitely often.

For $x \in \{di, de, f\}$, we define a relation $\leq_x$ on alternating Büchi automata. We write

$$\mathbf{Q} \leq_x \mathbf{S} \text{ when Duplicator has a winning strategy in } G^x(\mathbf{Q}, \mathbf{S}) \tag{1.23}$$

and say that $\mathbf{S}$ *x-simulates* $\mathbf{Q}$. For states q of $\mathbf{Q}$, s of $\mathbf{S}$, we write $q \leq_x s$ to indicate that $\mathbf{S}(s)$ x-simulates $\mathbf{Q}(q)$. We write $G^x(q,s)$ instead of $G^x(\mathbf{Q}(q), \mathbf{S}(s))$ if $\mathbf{Q}$ and $\mathbf{S}$ are obvious from the context.

As an example for a simulation game, consider the automaton $\mathbf{Q}$ given in Figure 1.1, which we view as an automaton over the alphabet $\{a, b\}$.

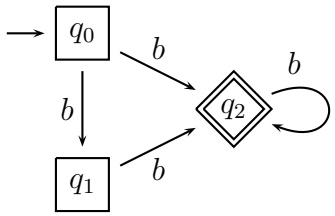

Figure 1.1: Alternating Büchi automaton

We argue that the games $G^{de}(q_0, q_1)$, $G^{de}(q_1, q_0)$, $G^f(q_0, q_1)$ and $G^f(q_1, q_0)$ are a win for Duplicator. To see this, consider the strategy σ defined by

$$\sigma(P^*(q_0, q_2, b, du, 0)) = (q_2, q_2) \ , \tag{1.24}$$

$$\sigma(P^*(q_1, y, b, du, 0)) = (q_2, y) \ , \qquad \text{for } y = q_1, q_2 \ , \tag{1.25}$$

$$\sigma(P^*(q_2, q_2, b, du, 1)) = (q_2, q_2) \ . \tag{1.26}$$

In a play starting in position (q_0, q_1), Spoiler has to choose the letter b, or he loses early, and he has to choose transition (q_1, b, q_2), i. e., the play reaches position $(q_0, q_2, b, du, 0)$ after his move. Playing according to σ, Duplicator now chooses the transition (q_0, b, q_2), and the next round starts in position (q_2, q_2). Now Spoiler always has to choose the letter b and the transition (q_2, b, q_2), but Duplicator (using σ) always chooses the same transition, so the play stays in (q_2, q_2) and thus is a win for Duplicator.

If the play starts in (q_1, q_0), the strategy σ also ensures that a play is either an early defeat for Spoiler or eventually stays in (q_2, q_2). That is, states q_0 and q_1 are equivalent w. r. t. delayed and fair simulation. Note that $q_2 \leq_x q_0$ and $q_2 \leq_x q_1$ for $x \in \{de, f\}$; the converse is false.

Lemma 1.1 (cf. [ESW01]) *For every alternating Büchi automaton, the following relations hold between the three types of simulation relations:*

$$\leq_{di} \subseteq \leq_{de} \subseteq \leq_{f} \ , \tag{1.27}$$

and these inclusions are strict for certain automata.

Proof. Since $W^{di} \subseteq W^{de} \subseteq W^{f}$, the inclusions follow immediately. It is easy to see that these inclusions are strict for the automata $\mathbf{Q}$ and $\mathbf{S}$ defined by

$$\mathbf{Q} = (\{q_0, q_1\}, \{a\}, q_0, \{(q_i, a, q_1) \mid i \in \{0,1\}\}, \{q_0, q_1\}, \emptyset, \{q_1\}) \ , \tag{1.28}$$
$$\mathbf{S} = (\{s_0, s_1\}, \{a\}, s_0, \{(s_i, a, s_1) \mid i \in \{0,1\}\}, \{s_0, s_1\}, \emptyset, \{s_0\}) \ . \tag{1.29}$$

In fact, we have $q_1 \leq_{de} q_0$, but $q_1 \not\leq_{di} q_0$, and $s_0 \leq_f s_1$, but $s_0 \not\leq_{de} s_1$. □

We say that an alternating Büchi automaton as in (1.2) is *complete* if for every $q \in Q$, $a \in \Sigma$, there is a state $q' \in Q$ such that $(q, a, q') \in \Delta^q$. Clearly, if we are given two alternating Büchi automata $\mathbf{Q}$ and $\mathbf{S}$ such that $\mathbf{Q} \leq_x \mathbf{S}$ for some $x \in \{di, de, f\}$, then, by adding at most two new states and at most $|\Sigma| \cdot (|Q| + 2)$ transitions, we can turn $\mathbf{Q}$ and $\mathbf{S}$ into equivalent complete automata $\mathbf{Q}'$ and $\mathbf{S}'$ such that $\mathbf{Q}' \leq_x \mathbf{S}'$ still holds. Therefore, we henceforth assume that all automata are complete; we allow incomplete automata only in Section 1.7, where we study algorithms for computing simulation relations, and in examples, which we want to keep small.

1.2.2 Simulation implies language containment

The first theorem states that all types of simulation imply language containment:

Theorem 1.1 *Let $x \in \{di, de, f\}$ and let $\mathbf{Q}$ and $\mathbf{S}$ be alternating Büchi automata. If $\mathbf{Q} \leq_x \mathbf{S}$, then $L(\mathbf{Q}) \subseteq L(\mathbf{S})$.*

Before we turn to the proof, we introduce useful conventions and notations concerning plays of simulation games.
Formally, a play of a simulation game is an infinite sequence $T = t_0 t_0^U t_0^V t_1 t_1^U t_1^V \dots$ where $t_i \in Q \times S$, $t_i^U \in U_{sp} \cup U_{du}$ and $t_i^V \in V_{sp} \cup V_{du}$. But the play T is obviously completely determined by the infinite sequence $t_0 t_1 \dots$ and the sequence of letters $w \in \Sigma^\omega$ in the third component of the elements of the sequence $t_0^U t_1^U t_2^U \dots$ (recall that each t_i^U is of the form (q, s, a, A, b, A', b') where a is a letter from Σ). A similar statement holds true for a partial play ending in a position in $Q \times S$. That is, there is a natural partial mapping

$$\xi \colon (Q \times S)^\infty \times \Sigma^\infty \to \text{set of partial or complete } G^x(\mathbf{Q}, \mathbf{S})\text{-plays} \ , \tag{1.30}$$

which maps $((q_i, s_i)_{i<n}, w)$ (where $n \in \omega \cup \{\omega\}$) to the corresponding partial play, provided there is such a play. This is the case if $|w| + 1 = n$ and, for all i with $i+1 < n$, $(q_i, w(i), q_{i+1}) \in \Delta^q$ and $(s_i, w(i), s_{i+1}) \in \Delta^s$.

An element of the domain of ξ will be called a *protoplay*.

Proof of Theorem 1.1. Let σ be a winning strategy for Duplicator in $G^x(\mathbf{Q}, \mathbf{S})$. Let $w \in L(\mathbf{Q})$, and let σ^q be a winning strategy for Automaton in $G(\mathbf{Q}, w)$. We have to show that Automaton has a winning strategy σ^s in $G(\mathbf{S}, w)$.

We first give an informal description of the way Automaton plays. While playing $G(\mathbf{S}, w)$, Automaton (in $G(\mathbf{S}, w)$) simultaneously plays the game $G^x(\mathbf{Q}, \mathbf{S})$ and the game $G(\mathbf{Q}, w)$. In the two plays he makes the moves for all players, Spoiler and Duplicator as well as Automaton and Pathfinder, and uses σ and σ^q to determine their moves. In other words, Automaton works as a puppeteer and moves four puppets at the same time. In this spirit, Automaton and Pathfinder in $G(\mathbf{Q}, w)$ and Spoiler and Duplicator in $G^x(\mathbf{Q}, \mathbf{S})$ will be called the automaton puppet, the pathfinder puppet, the spoiler puppet, and the duplicator puppet, respectively.

Automaton plays in such a way that after each round the state components in $G(\mathbf{Q}, w)$ and $G(\mathbf{S}, w)$ agree with the two state components of $G^x(\mathbf{Q}, \mathbf{S})$, and the partial games in $G^x(\mathbf{Q}, \mathbf{S})$ and $G(\mathbf{Q}, w)$ are conform with σ and σ^q. Then, clearly, since σ and σ^q are winning, in the emerging plays in $G(\mathbf{S}, w)$ infinitely many states will be in F^s, that is, Automaton will win $G(\mathbf{S}, w)$.

The above can be achieved when

- in $G^x(\mathbf{Q}, \mathbf{S})$, Automaton uses σ to determine the moves of the duplicator puppet, and,
- in $G(\mathbf{Q}, w)$, Automaton uses σ^q to determine the moves of the automaton puppet.

This is explained in more detail now.

Suppose that a play of $G(\mathbf{S}, w)$ is in a position (s, i) while $G(\mathbf{Q}, w)$ is in position (q, i). Consequently, $G^x(\mathbf{Q}, \mathbf{S})$ is in position (q, s). Automaton makes the spoiler puppet in the game $G^x(\mathbf{Q}, \mathbf{S})$ choose the letter $w(i)$. Automaton then proceeds as follows.

- If s is an existential state of $\mathbf{S}$, then Automaton has to move in $G(\mathbf{S}, w)$. Automaton proceeds according to the mode of q.
  - If q is an existential state of $\mathbf{Q}$, then Automaton makes the automaton puppet in the game $G(\mathbf{Q}, w)$ move according to the strategy σ^q. Automaton makes the spoiler puppet in $G^x(\mathbf{Q}, \mathbf{S})$ mimic this move and then makes the duplicator puppet in $G^x(\mathbf{Q}, \mathbf{S})$ react to this move by

choosing a successor state s' of s according to the strategy σ. This state s' is the successor state Automaton chooses as his move in $G(\mathbf{S}, w)$.

- If q is a universal state of $\mathbf{Q}$, then Automaton makes the duplicator puppet in game $G^x(\mathbf{Q},\mathbf{S})$ choose successor states q' of q and s' of s, according to σ. The pathfinder puppet mimics the choice of q' in $G(\mathbf{Q}, w)$ while Automaton moves to s' in $G(\mathbf{S}, w)$.

- If s is a universal state of $\mathbf{S}$, then Pathfinder has to move in $G(\mathbf{S}, w)$. Again, Automaton proceeds according to the mode of q.

  - If q is an existential state of $\mathbf{Q}$, then Automaton makes the automaton puppet in the game $G(\mathbf{Q}, w)$ move according to the strategy σ^q. He makes the spoiler puppet in $G^x(\mathbf{Q},\mathbf{S})$ mimic the automaton puppet's move in $G(\mathbf{Q}, w)$ and the Pathfinder's move in $G(\mathbf{S}, w)$.
  - If q is a universal state of $\mathbf{Q}$, then Automaton makes the spoiler puppet mimic the move of Pathfinder in $G^x(\mathbf{Q},\mathbf{S})$. Automaton then makes the duplicator puppet in game $G^x(\mathbf{Q},\mathbf{S})$ choose a successor state q' of q, according to σ. The pathfinder puppet in $G(\mathbf{Q}, w)$ mimics this choice.

We now proceed with a formal treatment. In order to define the winning strategy σ^s of Automaton in $G(\mathbf{S}, w)$, we first need a partial function pr^0 mapping partial $G^x(\mathbf{Q},\mathbf{S})$-protoplays to prefixes of $G(\mathbf{Q}, w)$-plays. For any partial $G^x(\mathbf{Q},\mathbf{S})$-protoplay $\pi = ((q_i, s_i)_{i\leq n}, w[0..n-1])$, we set

$$\mathrm{pr}^0(\pi) = (q_0, 0)\dots(q_n, n) \ . \tag{1.31}$$

As another auxiliary function, we define the partial function T by

$$T\colon (Q \times S \times \omega)^* \to (Q \times S)^* \times \Sigma^* \ , \tag{1.32}$$

$$(q_i, s_i, i)_{i\leq n} \mapsto ((q_i, s_i)_{i\leq n}, w[0..n-1]) \ . \tag{1.33}$$

Simultaneously, we define a partial function

$$\hat{\sigma}\colon (Q \times S \times \omega)^* \to Q \times S \times \omega \ , \tag{1.34}$$

describing the interplay of σ^q and σ for a given partial $G^x(\mathbf{Q},\mathbf{S})$-play, and a partial function

$$h\colon (S \times \omega)^* \to (Q \times S \times \omega)^* \ . \tag{1.35}$$

The function $\hat{\sigma}$ will be defined only for sequences $\rho = (q_i, s_i, i)_{i\leq n}$ where $s_n \in E^s$ and $T(\rho)$ is a partial $G^x(\mathbf{Q},\mathbf{S})$-protoplay; the function h assigns such sequences ρ to prefixes of $G(\mathbf{Q}, w)$-plays.

To define the value of the partial function $\hat{\sigma}$ for a sequence $\rho = (q_i, s_i, i)_{i \leq n}$ where $s_n \in E^s$, we first assume that $q_n \in E^q$.

We then define

$$\hat{\sigma} \colon \rho \mapsto (\sigma(\pi p p'), n+1) \ , \tag{1.36}$$

where π is the partial $G^x(\mathbf{Q}, \mathbf{S})$-play $\xi(T(\rho))$, $p = (q_n, s_n, w(n), sp, 0, du, 1)$ is the next position of the play, and $p' = (\mathrm{pr}_1(\sigma^q(\mathrm{pr}^0(T(\rho)))), s_n, w(n), du, 1)$ is the successor position chosen via σ^q.

For the case that $q_n \in U^q$, we define

$$\hat{\sigma} \colon \rho \mapsto (\sigma(\pi p p'), n+1) \ , \tag{1.37}$$

where again π is the partial $G^x(\mathbf{Q}, \mathbf{S})$-play $\xi(T(\rho))$. The following position now is $p = (q_n, s_n, w(n), du, 0, du, 1)$, and $p' = \sigma(\pi p)$ is the successor position chosen via σ.

The partial function h is inductively defined as follows. For the initial case, we set

$$h \colon (s_I, 0) \mapsto (q_I, s_I, 0) \ . \tag{1.38}$$

Now let $s_0 = s_I$ and $\rho = h((s_i, i)_{i \leq n})$, and assume that the last tuple in this sequence is (q_n, s_n, n). If $q_n \in E^q$, we define

$$h \colon (s_i, i)_{i \leq n+1} \mapsto \rho p \ , \tag{1.39}$$

where $p = (q_{n+1}, s_{n+1}, n+1)$ with $q_{n+1} = \mathrm{pr}_1(\sigma^q(\mathrm{pr}^0(T(\rho))))$.

For the case $q_n \in U^q$, we have to look at the sub-cases $s_n \in E^s$ and $s_n \in U^s$. If $s_n \in E^s$, we define

$$h \colon (s_i, i)_{i \leq n+1} \mapsto \rho p \ , \tag{1.40}$$

where $p = (\mathrm{pr}_1(\hat{\sigma}(\rho)), s_{n+1}, n+1)$, while for the sub-case $s_n \in U^s$, we define

$$h \colon (s_i, i)_{i \leq n+1} \mapsto \rho(q_{n+1}, s_{n+1}, n+1) \ , \tag{1.41}$$

where $(q_{n+1}, s_{n+1}) = \sigma(\xi(T(\rho)) p p')$ with $p = (q_n, s_n, w(n), sp, 1, du, 0)$ and $p' = (q_n, s_{n+1}, w(n), du, 0)$.

With these definitions, we can now define a Duplicator winning strategy σ^s for $G(\mathbf{S}, w)$ by

$$\sigma^s \colon (s_i, i)_{i \leq n} \mapsto (\mathrm{pr}_2(\hat{\sigma}(h((s_i, i)_{i \leq n+1}))), n+1) \ , \tag{1.42}$$

for $s_n \in E^s$.

With these definitions, it is tedious but routine to check the following.

1. The function σ^s is defined for $(s_I, 0)$ if $s_I \in E^s$, and if $(s_i, i)_{i<n}$ is a partial σ^s-conform $G(\mathbf{S}, w)$-play such that $s_{n-1} \in E^s$, then σ^s is defined for $(s_i, i)_{i<n}$.

   That is, σ^s is in fact a Duplicator strategy for $G(\mathbf{S}, w)$.

2. If $(s_i, i)_{i<\omega}$ is a σ^s-conform $G(\mathbf{S}, w)$-play, then $\mathrm{pr}^0(\xi(T(h((s_i, i)_{i<n}))))$ is a partial σ^q-conform $G(\mathbf{Q}, w)$-play, for all $n < \omega$.

   That is, since σ^q is a winning strategy, in the $G(\mathbf{Q}, w)$-play connected to $(s_i, i)_{i<\omega}$ via h there are infinitely many occurrences of accepting states.

3. If $(s_i, i)_{i<\omega}$ is a σ^s-conform $G(\mathbf{S}, w)$-play, then $\xi(T(h((s_i, i)_{i<n})))$ is a partial σ-conform $G^x(\mathbf{Q}, \mathbf{S})$-play, for all $n < \omega$.

From 2 and 3 and since σ also is a winning strategy, we conclude that there must be infinitely many occurrences of accepting states in $(s_i)_{i<\omega}$, that is, σ^s is a winning strategy. □

1.2.3 Positional strategies for simulation games

A strategy σ of Player i, $i \in \{0, 1\}$, for a game $G = (P, P_0, P_1, p_I, Z, W)$ as defined in Subsection 1.1.1 is called *positional* or *memoryless* if, for every $\pi, \pi' \in P^*$ and $p \in P_i$, either $\sigma(\pi p) = \sigma(\pi' p)$ or both $\sigma(\pi p)$ and $\sigma(\pi' p)$ are undefined. That is, a positional strategy σ only depends on the last position of a partial play and can hence be seen as a partial function $P_i \to P$.

It is well known that, if G is a so-called reachability game or a parity game and Player i wins G, then Player i has a positional winning strategy [EJ91, Mos91]. Corollary 1.1 follows immediately.

Corollary 1.1 *Let* $\mathbf{Q}$ *and* $\mathbf{S}$ *be two ABA. Spoiler (Duplicator) wins* $G^{di}(\mathbf{Q}, \mathbf{S})$ *or* $G^f(\mathbf{Q}, \mathbf{S})$ *if and only if there is a positional winning strategy of Spoiler (Duplicator) for* $G^{di}(\mathbf{Q}, \mathbf{S})$ *or* $G^f(\mathbf{Q}, \mathbf{S})$*, respectively.*

For delayed simulation games, this is only true for Duplicator.

Proposition 1.1

1. *For alternating Büchi automata* $\mathbf{Q}$ *and* $\mathbf{S}$*, Duplicator wins* $G^{de}(\mathbf{Q}, \mathbf{S})$ *if and only if there is a positional winning strategy of Duplicator for* $G^{de}(\mathbf{Q}, \mathbf{S})$.

2. *There are Büchi automata* $\mathbf{Q}$ *and* $\mathbf{S}$ *such that Spoiler wins* $G^{de}(\mathbf{Q}, \mathbf{S})$*, but no positional winning strategy is winning for Spoiler.*

We will only proof the second claim of the proposition here; the proof of the first claim needs some preparation and can be found in Subsection 1.7.2.

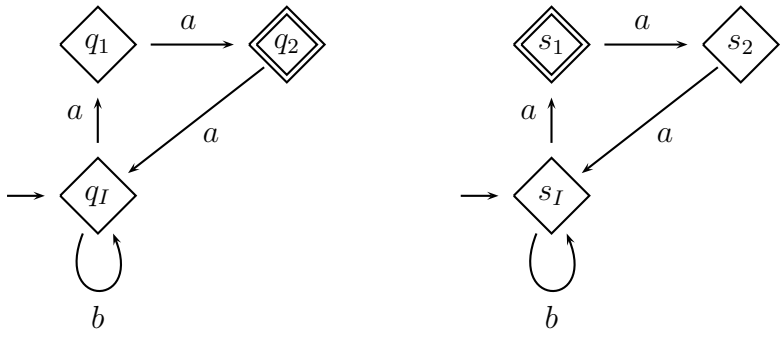

Figure 1.2: Spoiler needs memory to win $G^{de}(\mathbf{Q},\mathbf{S})$

Proof of Proposition 1.1, part 2. Consider the Büchi automata $\mathbf{Q}$ (on the left) and $\mathbf{S}$ (on the right) of Figure 1.2.

We claim that Spoiler wins $G^{de}(\mathbf{Q},\mathbf{S})$.

First, note that there are only existential states in the two automata. Therefore, in each round first Spoiler moves in $\mathbf{Q}$ and then Duplicator moves in $\mathbf{S}$. Next, note that the automata are deterministic. Thus, a play is completely determined by what Spoiler does and, moreover, the moves of Spoiler are completely determined by the letters he chooses at the beginning of each round. That is, a strategy of Spoiler can be denoted by an ω-word, for instance, $aaab^{\omega}$. It is easy to see that the set of all winning strategies for Spoiler can be denoted by elements from

$$(aaa+b)^{*}aaab^{\omega} \quad , \tag{1.43}$$

in particular, Spoiler wins $G^{de}(\mathbf{Q},\mathbf{S})$. However, the only two positional strategies for Spoiler which do not result in an early loss are a^{ω} and b^{ω}, and do not belong to the above set. That is, there is no positional winning strategy for Spoiler in $G^{de}(\mathbf{Q},\mathbf{S})$. □

1.3 Composing Simulation Strategies

In this section, let $x \in \{di, de, f\}$. We will introduce the join of two Duplicator strategies, a concept fundamental for the proofs of the results in Subsection 1.3.2 and Section 1.5. The idea is that two strategies for simulation games starting in positions (q,r) and (r,s), respectively, can be merged into a joint strategy for a game starting in (q,s); this joint strategy inherits crucial properties of the two original strategies (see Lemma 1.2 and Corollary 1.7), and will also be used to show that the relation $\leq_x$ is transitive.

1.3.1 Definition of the join of strategies

Let $q \in \mathbf{Q}$, $r \in \mathbf{R}$, $s \in \mathbf{S}$. Let σ_0 be a Duplicator strategy for the basic game $G(q,r)$, and let σ_1 be a Duplicator strategy for the basic game $G(r,s)$.

To describe the join of the strategies σ_0 and σ_1, denoted $\sigma_0 \bowtie \sigma_1$, informally, we can again use the puppeteering metaphor of the previous section: Duplicator, playing $G(q,s)$ using $\sigma_0 \bowtie \sigma_1$, simultaneously plays $G(q,r)$ and $G(r,s)$, using σ_0 and σ_1, respectively. His four puppets are Spoiler and Duplicator of these games. We will call Spoiler and Duplicator of $G(q,r)$ the left spoiler puppet and the left duplicator puppet, while Spoiler and Duplicator of $G(r,s)$ are the right spoiler puppet and the right duplicator puppet.

Duplicator (of $G(q,s)$, our puppeteer) plays in such a way that after each round the first state component of $G(q,r)$ and the second state component of $G(r,s)$ agree with the first and second state component of $G(q,s)$, respectively, and the second state component of $G(q,r)$ agrees with the first state component of $G(r,s)$, and the partial plays in $G(q,r)$ and $G(r,s)$ are conform with σ_0 and σ_1, respectively.

This can be achieved in the following way. In $G(q,r)$, Duplicator uses σ_0 to determine the moves of the left duplicator puppet, while in $G(r,s)$, he uses σ_1 to determine the moves of the right duplicator puppet. The spoiler puppets just mimic the moves of Spoiler and the duplicator puppets.

We will clarify this interplay by describing the course of two exemplary rounds.

Consider a position (q_i, s_i) of $G(q,s)$ where the simultaneous plays of $G(q,r)$ and $G(r,s)$ are in positions (q_i, r_i) and (r_i, s_i), respectively, such that $(q_i, r_i, s_i) \in E^q \times U^r \times E^s$. Let Spoiler choose a letter a in the $G(q,s)$-play.

At first, Duplicator makes the two spoiler puppets choose the same letter a.

Since q_i is existential, Spoiler has to choose an a-successor state q_{i+1} as his next move in $G(q,s)$. Duplicator makes the left spoiler puppet mimic this move in $G(q,r)$.

Since r_i is universal and s_i is existential, Duplicator proceeds as follows. He lets the right duplicator puppet choose a-successors r_{i+1} of r_i and s_{i+1} of s_i according to σ_1 in $G(r,s)$. The left spoiler puppet then mimics this and chooses r_{i+1} as its next move in $G(q,r)$; similarly, Duplicator chooses s_{i+1} in $G(q,s)$.

Now consider a situation where $(q_i, r_i, s_i) \in U^q \times E^r \times E^s$. After mimicking Spoiler's choice of a letter a by the two spoiler puppets, Duplicator makes the left duplicator puppet choose a-successors q_{i+1} of q_i and r_{i+1} of r_i in the $G(q,r)$-play according to σ_0. The choice of r_{i+1} is mimicked as its next move by the right spoiler puppet while the choice of q_{i+1} is used by Duplicator as his next move in $G(q,s)$. Duplicator then makes the right duplicator puppet react to the move of the right spoiler puppet by choosing an a-successor s_{i+1} of s_i according to σ_1 in the $G(r,s)$-play. Duplicator copies this choice of the right duplicator puppet as his

next move.

That is, first the spoiler puppets serve to mimic the moves of Spoiler. The moves of the left and right duplicator puppets are then guided by the two strategies σ_0 and σ_1, respectively. That is, the left duplicator puppet controls the choice of r_{i+1} if r_i is existential, and this choice is mimicked by the right spoiler puppet, which in turn allows the right duplicator puppet to react, if necessary. This situation is reversed if r_i is universal.

To define this strategy formally, we also have to keep track of the sequence of the **R**-states in the play of $G(q,r)$, which is identical to the sequence of **R**-states in the play of $G(r,s)$. We now continue with the formal definitions.

We simultaneously and inductively define the *joint strategy* $\sigma_0 \bowtie \sigma_1$, which is a Duplicator strategy for $G(q,s)$, and a sequence of **R**-states (starting with r) for partial $(\sigma_0 \bowtie \sigma_1)$-conform $G(q,s)$-plays, the so-called *intermediate sequence*.

The definition (construction) of the joint strategy $\sigma_0 \bowtie \sigma_1$ for the prefix of a play that has lasted for n rounds uses the intermediate sequence of length $n+1$ for this prefix, and in turn the $(n+1)$th $(\sigma_0 \bowtie \sigma_1)$-conform round defines the $(n+2)$th element of the intermediate sequence for the prolonged prefix.

The joint strategy and the intermediate sequence will have the following property.

Property 1.1 *If $((q_j,s_j)_{j<n+1},w)$ is a partial $(\sigma_0 \bowtie \sigma_1)$-conform protoplay and $(r_j)_{j<n+1}$ is the intermediate sequence for this protoplay, then $((q_j,r_j)_{j<n+1},w)$ is a partial σ_0-conform $G(q,r)$-protoplay and $((r_j,s_j)_{j<n+1},w)$ is a partial σ_1-conform protoplay.*

Initially, for the $G(q,s)$-protoplay $((q,s),\varepsilon)$ (i.e., for the prefix of the play where no moves have been played), the intermediate sequence is q. Note that Property 1.1 holds.

Now assume that for a $(\sigma_0 \bowtie \sigma_1)$-conform protoplay $T = ((q_i,s_i)_{i<n+1},w)$, the intermediate sequence is given by $(r_i)_{i<n+1}$ (and $q_0 = q, r_0 = r, s_0 = s$). In particular, T and $(r_i)_{i<n+1}$ have Property 1.1. Let $T^0 = ((q_i,r_i)_{i<n+1},w)$ and $T^1 = ((r_i,s_i)_{i<n+1},w)$. Recall that the last position of $\xi(T)$ is (q_n,s_n).

In order to define $\sigma_0 \bowtie \sigma_1$ and r_{n+1} for the round following T, we distinguish eight cases depending on the modes of q_n, r_n, and s_n.

Case EEE, $(q_n,r_n,s_n) \in E^q \times E^r \times E^s$. Assume Spoiler chooses the $G(q,s)$-positions $t_n^U = (q_n,s_n,a,sp,0,du,1)$ and $t_n^V = (q_{n+1},s_n,a,du,1)$. Let

$$\sigma_0(\xi(T^0)(q_n,r_n,a,sp,0,du,1)(q_{n+1},r_n,a,du,1)) = (q_{n+1},r_{n+1}) \ , \qquad (1.44)$$

$$\sigma_1(\xi(T^1)(r_n,s_n,a,sp,0,du,1)(r_{n+1},s_n,a,du,0)) = (r_{n+1},s_{n+1}) \ . \qquad (1.45)$$

We define

$$\sigma_0 \bowtie \sigma_1(\xi(T)t_n^U t_n^V) = (q_{n+1},s_{n+1}) \qquad (1.46)$$

and define $(r_i)_{i\leq n+1}$ to be the intermediate sequence for the partial protoplay $((q_i,s_i)_{i\leq n+1},wa)$; note that the two have Property 1.1.

Case EUE, $(q_n,r_n,s_n)\in E^q\times U^r\times E^s$. Assume Spoiler chooses the $G(q,s)$-positions $t_n^U=(q_n,s_n,a,sp,0,du,1)$ and $t_n^V=(q_{n+1},s_n,a,du,1)$. Let

$$\sigma_1(\xi(T^1)(r_n,s_n,a,du,0,du,1))=(r_{n+1},s_n,a,du,1)\ , \quad (1.47)$$

$$\sigma_1(\xi(T^1)(r_n,s_n,a,du,0,du,1)(r_{n+1},s_n,a,du,1))=(r_{n+1},s_{n+1})\ . \quad (1.48)$$

We define

$$\sigma_0\bowtie\sigma_1(\xi(T)t_n^Ut_n^V)=(q_{n+1},s_{n+1}) \quad (1.49)$$

and $(r_i)_{i\leq n+1}$ as the corresponding intermediate sequence.

Case UEU, $(q_n,r_n,s_n)\in U^q\times E^r\times U^s$. Assume Spoiler chooses the $G(q,s)$-positions $t_n^U=(q_n,s_n,a,sp,1,du,0)$ and $t_n^V=(q_n,s_{n+1},a,du,0)$. Let

$$\sigma_0(\xi(T^0)(q_n,r_n,a,du,1,du,0))=(q_{n+1},r_n,a,du,1)\ , \quad (1.50)$$

$$\sigma_0(\xi(T^0)(q_n,r_n,a,du,1,du,0)(q_{n+1},r_n,a,du,1))=(q_{n+1},r_{n+1})\ . \quad (1.51)$$

We define

$$\sigma_0\bowtie\sigma_1(\xi(T)t_n^Ut_n^V)=(q_{n+1},s_{n+1}) \quad (1.52)$$

and $(r_i)_{i\leq n+1}$ as the corresponding intermediate sequence.

Case UUU, $(q_n,r_n,s_n)\in U^q\times U^r\times U^s$. Assume Spoiler chooses the $G(q,s)$-positions $t_n^U=(q_n,s_n,a,sp,1,du,0)$ and $t_n^V=(q_n,s_{n+1},a,du,0)$. Let

$$\sigma_1(\xi(T^1)(r_n,s_n,a,sp,1,du,0)(r_n,s_{n+1},a,du,0))=(r_{n+1},s_{n+1})\ , \quad (1.53)$$

$$\sigma_0(\xi(T^0)(q_n,r_n,a,sp,1,du,0)(q_n,r_{n+1},a,du,0))=(q_{n+1},r_{n+1})\ . \quad (1.54)$$

We define

$$\sigma_0\bowtie\sigma_1(\xi(T)t_n^Ut_n^V)=(q_{n+1},s_{n+1}) \quad (1.55)$$

and $(r_i)_{i\leq n+1}$ as the next intermediate sequence.

Case UEE, $(q_n,r_n,s_n)\in U^q\times E^r\times E^s$. Assume Spoiler chooses the position $t_n^U=(q_n,s_n,a,du,0,du,1)$. Let

$$\sigma_0(\xi(T^0)(q_n,r_n,a,du,0,du,1))=(q_{n+1},r_n,a,du,1)\ , \quad (1.56)$$

$$\sigma_0(\xi(T^0)(q_n,r_n,a,du,0,du,1)(q_{n+1},r_n,a,du,1))=(q_{n+1},r_{n+1})\ , \quad (1.57)$$

$$\sigma_1(\xi(T^1)(r_n,s_n,a,sp,0,du,1)(r_{n+1},s_n,a,du,1))=(r_{n+1},s_{n+1})\ . \quad (1.58)$$

We define

$$\sigma_0\bowtie\sigma_1(\xi(T)t_n^U)=(q_{n+1},s_n,a,du,1)\ , \quad (1.59)$$

$$\sigma_0\bowtie\sigma_1(\xi(T)t_n^U(q_{n+1},s_n,a,du,1))=(q_{n+1},s_{n+1})\ , \quad (1.60)$$

and choose $(r_i)_{i\leq n+1}$ as the corresponding intermediate sequence.

Case UUE, $(q_n, r_n, s_n) \in U^q \times U^r \times E^s$, and the following Spoiler-chosen $G(q,s)$-position is $t_n^U = (q_n, s_n, a, du, 0, du, 1)$. Let

$$\sigma_1(\xi(T^1)(r_n, s_n, a, du, 0, du, 1)) = (r_{n+1}, s_n, a, du, 1) \ , \quad (1.61)$$

$$\sigma_1(\xi(T^1)(r_n, s_n, a, du, 0, du, 1)(r_{n+1}, s_n, a, du, 1)) = (r_{n+1}, s_{n+1}) \ , \quad (1.62)$$

$$\sigma_0(\xi(T^0)(q_n, r_n, a, sp, 1, du, 0)(q_n, r_{n+1}a, du, 0)) = (q_{n+1}, r_{n+1}) \ . \quad (1.63)$$

We define

$$\sigma_0 \bowtie \sigma_1(\xi(T)t_n^U) = (q_{n+1}, s_n, a, du, 1) \ , \quad (1.64)$$

$$\sigma_0 \bowtie \sigma_1(\xi(T)t_n^U(q_{n+1}, s_n, a, du, 1)) = (q_{n+1}, s_{n+1}) \ , \quad (1.65)$$

and choose $(r_i)_{i\leq n+1}$ as the corresponding intermediate sequence.

Case EEU, $(q_n, r_n, s_n) \in E^q \times E^r \times U^s$. Assume that Spoiler chooses the $G(q,s)$-positions $t_n^U = (q_n, s_n, a, sp, 0, sp, 1)$ and $t_n^V = (q_{n+1}, s_n, a, sp, 1)$ and $t_{n+1} = (q_{n+1}, s_{n+1})$. Let

$$\sigma_0(\xi(T^0)(q_n, r_n, a, sp, 0, du, 1)(q_{n+1}, r_n, a, du, 1)) = (q_{n+1}, r_{n+1}) \ . \quad (1.66)$$

We define $(r_i)_{i\leq n+1}$ as the corresponding intermediate sequence (the strategy $\sigma_0 \bowtie \sigma_1$ need not be defined in this case, since Duplicator cannot move in a turn starting with a $E^q \times U^s$-state).

Case EUU, $(q_n, r_n, s_n) \in E^q \times U^r \times U^s$, and the following Spoiler-chosen $G(q,s)$-positions are the three positions defined by $t_n^U = (q_n, s_n, a, sp, 0, sp, 1)$, $t_n^V = (q_{n+1}, s_n, a, sp, 1)$ and $t_{n+1} = (q_{n+1}, s_{n+1})$. Let

$$\sigma_1(\xi(T^1)(r_n, s_n, a, sp, 1, du, 0)(r_n, s_{n+1}, a, du, 0)) = (r_{n+1}, s_{n+1}) \ . \quad (1.67)$$

We define $(r_i)_{i\leq n+1}$ as the next intermediate sequence (again, $\sigma_0 \bowtie \sigma_1$ need not be defined).

This completes the description of $\sigma_0 \bowtie \sigma_1$. It will be thoroughly analyzed in the next subsection.

1.3.2 Fundamental properties of composed strategies and simulation relations

In this subsection, we will show crucial properties of the simulation relations $\leq_{di}$, $\leq_{de}$, $\leq_f$ (summarized as $\leq_x$) using the concept of a join of two Duplicator strategies, as defined above.

We first want to show that $\leq_x$ is reflexive and transitive, i. e., a preorder. Reflexivity is obvious: whenever in a play a position $(q,q) \in E \times E$ is reached, Duplicator can move in the second component to the state that Spoiler has chosen in the first component; for $(q,q) \in U \times U$, he does the same in the first component (Duplicator literally duplicates Spoiler's moves). Using this strategy, Duplicator wins the game in all three versions.

Transitivity needs some more care. Here, we will need the join of two Duplicator strategies, as defined in Subsection 1.3.1.

Lemma 1.2 (composing winning strategies) *Let $q \in Q$, $r \in R$, and $s \in S$ such that $q \leq_x r$ and $r \leq_x s$. Let σ_0 be a Duplicator strategy for $G^x(q,r)$, and let σ_1 be a Duplicator strategy for $G^x(r,s)$.*

If σ_0 and σ_1 are winning strategies, $\sigma_0 \bowtie \sigma_1$ is a winning strategy (i.e., $q \leq_x r$ and $r \leq_x s$ imply $q \leq_x s$).

Proof. Let σ_0, σ_1 be winning strategies, and let T be a $(\sigma_0 \bowtie \sigma_1)$-conform play with intermediate sequence $(r_i)_{i<\omega}$. Note that the plays T^0 and T^1 (as defined in Subsection 1.3.1) are σ_0-conform and σ_1-conform, respectively.

In the case of direct simulation, since T^0 is σ_0-conform, for every i such that $q_i \in F^q$, we have $r_i \in F^r$. And since T^1 is σ_1-conform, this implies $s_i \in F^s$, that is, T is a win for Duplicator.

In the case of delayed simulation, for every i such that $q_i \in F^q$, there is a $j_0 \geq i$ such that $r_{j_0} \in F^r$, since T^0 is σ_0-conform. In turn, by the σ_1-conformity of T^1, there is a $j_1 \geq j_0$ such that $s_{j_1} \in F^s$. Hence, T is a win for Duplicator.

Finally, for fair simulation, if there are infinitely many i such that $q_i \in F^q$, the σ_0-conformity of T^0 ensures that there are also infinitely many j such that $r_j \in F^r$, and the σ_1-conformity of T^1 then ensures that there are infinitely many l such that $s_l \in F^s$. So, again, T is a win for Duplicator. □

Corollary 1.2 *For $x \in \{di, de, f\}$, $\leq_x$ is a preorder, that is, $\leq_x$ is reflexive and transitive.*

Being a preorder, $\leq_x$ induces an equivalence relation $\equiv_x$ by virtue of

$$q \equiv_x s \qquad \text{iff} \qquad q \leq_x s \text{ and } s \leq_x q. \tag{1.68}$$

By Theorem 1.1, $q \equiv_x s$ implies $L(\mathbf{Q}(q)) = L(\mathbf{S}(s))$. The relations $\equiv_{di}$, $\equiv_{de}$, $\equiv_f$ are called *direct*, *delayed* and *fair simulation equivalence*, respectively.

While the join of two Duplicator winning strategies is again a winning strategy, the join of two memoryless Duplicator strategies need not be a memoryless strategy.

Lemma 1.3 *There are Büchi automata* $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$ *such that* $\mathbf{Q} \leq_x \mathbf{R} \leq_x \mathbf{S}$ *but, for all Duplicator winning strategies* σ_0 *for* $G^x(\mathbf{Q},\mathbf{R})$ *and* σ_1 *for* $G^x(\mathbf{R},\mathbf{S})$, $\sigma_0 \bowtie \sigma_1$ *is not a positional strategy, but, of course, a winning strategy.*

Proof. We give a simple example of such automata for $x \in \{de, f\}$; this example can be modified easily so as to work in the case $x = di$.

Consider the automata $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$ (from left to right) of Figure 1.3.

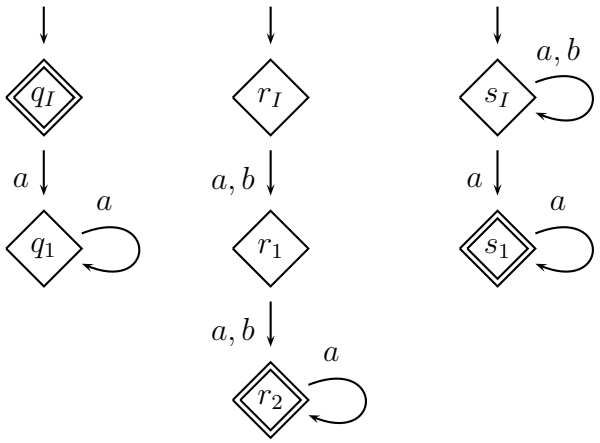

Figure 1.3: Joint strategies are not positional

The moves of Spoiler and Duplicator in $G^x(\mathbf{Q},\mathbf{R})$, and hence Duplicator's positional winning strategy σ_0, are fixed by the structure of the automata (if Spoiler does not want to lose early), i. e., there is only one infinite play of $G^x(\mathbf{Q},\mathbf{R})$. A positional winning strategy σ_1 for Duplicator in $G^x(\mathbf{R},\mathbf{S})$ has to satisfy $\sigma_1(r_1, s_I, a, du, 1) = (r_1, s_I)$ and $\sigma_1(r_2, s_I, a, du, 1) = (r_2, s_1)$.

Consequently, $\sigma_0 \bowtie \sigma_1$ maps the partial play $\pi = (q_I, s_I)\ (q_I, s_I, a, sp, 0, du, 1)\ (q_1, s_I, a, du, 1)$ to (q_1, s_I), but the partial play $\pi\ (q_1, s_I)\ (q_1, s_I, a, sp, 0, du, 1)\ (q_1, s_I, a, du, 1)$ is mapped to (q_1, s_1), i. e., $\sigma_0 \bowtie \sigma_1$ is not positional. □

Fundamental for the further study of $\leq_x$ is the following lemma, which is similar to [ESW01, Lemma 4.1].

Lemma 1.4 *Let* $\mathbf{Q}$, $\mathbf{S}$ *be alternating Büchi automata and let* q, s *be states of* $\mathbf{Q}$ *and* $\mathbf{S}$*, respectively, such that* $q \leq_x s$*. Let* $a \in \Sigma$*.*

1. *If* $(q,s) \in E^q \times E^s$*, there is, for every* $q' \in \Delta^q(q,a)$*, a state* $s' \in \Delta^s(s,a)$ *such that* $q' \leq_x s'$*.*

2. *If* $(q,s) \in E^q \times U^s$, *for all* $q' \in \Delta^q(q,a)$ *and for all* $s' \in \Delta^s(s,a)$ *we have* $q' \leq_x s'$.

3. *If* $(q,s) \in U^q \times E^s$, *there are* $q' \in \Delta^q(q,a)$ *and* $s' \in \Delta^s(s,a)$ *such that* $q' \leq_x s'$.

4. *If* $(q,s) \in U^q \times U^s$, *there is, for every state* $s' \in \Delta^s(s,a)$, *a* $q' \in \Delta^q(q,a)$ *such that* $q' \leq_x s'$.

Proof. First, let $(q,s) \in E^q \times E^s$. Since $q \leq_x s$, in a play T of $G^x(q,s)$ starting with $T_0 = (q,s)(q,s,a,sp,0,du,1)(q',s,a,du,1)$, i. e., $q' \in \Delta(q,a)$, Duplicator can use a winning strategy σ. Let $(q',s') = \sigma(T_0)$. Since σ is a winning strategy for Duplicator, there is a winning strategy of Duplicator for $G^x(q',s')$, thus $q' \leq_x s'$.

Similar arguments yield the claims for the other three cases, i. e., the case $(q,s) \in U^q \times U^s$ is symmetric, while the arguments for the other cases are as follows. Case $(q,s) \in E^q \times U^s$: If Duplicator cannot move in a round but has a winning strategy at the beginning of that round, he also has a winning strategy at the beginning of the next round, no matter what Spoiler does. Case $(q,s) \in U^q \times E^s$: If Duplicator has a winning strategy and can choose both transitions, he can choose the transitions using his winning strategy. Then he has a winning strategy at the beginning of the next round. □

In the sequel, we will call a Duplicator strategy σ for a game $G(q_0,s_0)$ $\leq_x$-*respecting* if $q \leq_x s$ holds true for every position (q,s) reachable in any play where Duplicator follows σ.

The following is easy to see:

Remark 1.1 *A winning strategy of Duplicator for an x-simulation game is* $\leq_x$-*respecting.*

The converse is false for $x \in \{de, f\}$, as we will see at the beginning of Section 1.5.

1.4 Quotienting Modulo Direct Simulation

In general, when $\equiv$ is an equivalence relation on the state space of an alternating Büchi automaton $\mathbf{Q}$, we call an alternating Büchi automaton a *quotient of* $\mathbf{Q}$ with respect to $\equiv$ if it is of the form

$$(Q/\equiv, \Sigma, [q_I], \Delta', E', U', F/\equiv) \tag{1.69}$$

where $[q] = \{q' \in Q \mid q \equiv q'\}$ for every $q \in Q$ and $M/\equiv = \{[q] \mid q \in M\}$ for every $M \subseteq Q$.

Furthermore, the following natural constraints must be satisfied:

1. If $([q], a, [q']) \in \Delta'$, then there exist $\hat{q} \equiv q$ and $\bar{q} \equiv q'$ such that $(\hat{q}, a, \bar{q}) \in \Delta$, that is, $\Delta' \subseteq \{([q], a, [q']) \mid (q, a, q') \in \Delta\}$,

2. if $[q] \subseteq E$, then $[q] \in E'$, and

3. if $[q] \subseteq U$, then $[q] \in U'$.

Note that 1–3 are minimal requirements so that the quotient really reflects the structure of $\mathbf{Q}$ and is not just any automaton on the equivalence classes of $\equiv$.

In the following, when the considered equivalence relation is direct or delayed simulation equivalence, we will, for instance, write Q_{de} instead of $Q/\equiv_{de}$ and F_{di} instead of $F/\equiv_{di}$.

A *naive quotient* is a quotient where the converse of the first constraint is true, that is, where transitions are representative-wise.

Direct simulation is particularly easy (compared to delayed or fair simulation), so one might expect that a naive definition of the quotient automaton modulo direct simulation should be equivalent to the original automaton. Problems arise for mixed equivalence classes, i. e., classes containing both existential and universal states. In the naive quotienting, these states can be made neither existential nor universal.

Consider Figure 1.4, where an alternating Büchi automaton $\mathbf{Q}$ over $\Sigma = \{a, b\}$ is shown on the left, and the naive x-quotient is shown on the right. For simplicity in notation, we denote the states in the quotients by representatives of the actual equivalence classes, for instance, q_0 on the right stands for $[q_0]$. Note that we have $q_3 \leq_x q_1 \leq_x q_0 \equiv_x q_2$, but $q_3 \not\equiv_x q_1$ and $q_1 \not\equiv_x q_0$ for $x \in \{di, de, f\}$.

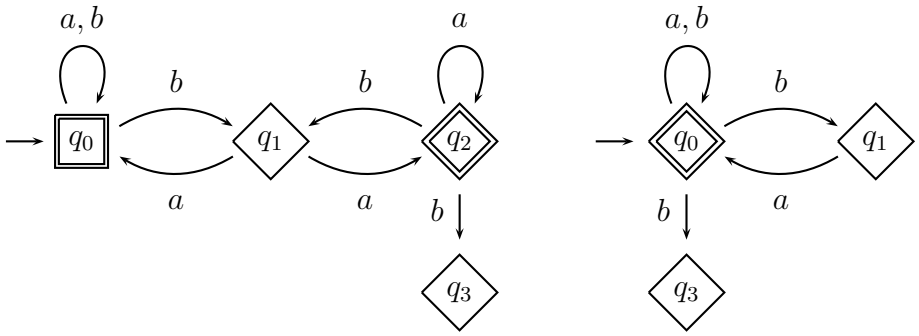

Figure 1.4: Naive quotients do not work

The language recognized by the original automaton is $(ba + a)^\omega$, while the naive quotient recognizes Σ^ω. The other possible naive quotient, where the state

$[q_0]$ is declared universal, is not equivalent to the original automaton either: that naive quotient only accepts the word a^ω.

We overcome these problems for direct simulation quotienting by using a more sophisticated transition relation for the quotient automaton, exploiting the simple structure of direct simulation games.

1.4.1 Minimal and maximal successors

To define quotient automata modulo $\equiv_x$ (in fact, for $x = di$ and $x = de$ only, since fair quotienting does not preserve the language, see [ESW01]), we will need the notion of maximal and minimal successors of states.

Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be an alternating Büchi automaton. Let $q \in Q$, and $a \in \Sigma$. A state $q' \in \Delta(q,a)$ is an *x-maximal a-successor of q* if and only if $q'' \leq_x q'$ holds for every $q'' \in \Delta(q,a)$ with $q' \leq_x q''$. We define

$$\max_a^x(q) = \{q' \in \Delta(q,a) \mid q' \text{ is an } x\text{-maximal } a\text{-successor of } q\} \quad . \tag{1.70}$$

A state $q' \in \Delta(q,a)$ is an *x-minimal a-successor of q* if and only if $q' \leq_x q''$ for every $q'' \in \Delta(q,a)$ with $q'' \leq_x q'$. We define

$$\min_a^x(q) = \{q' \in \Delta(q,a) \mid q' \text{ is an } x\text{-minimal } a\text{-successor of } q\} \quad . \tag{1.71}$$

We will also write $\min_a$ and $\max_a$ instead of $\min_a^x$ and $\max_a^x$, respectively, if the context determines the intended winning mode.

1.4.2 Minimax quotienting

We can now define a quotient that works for direct simulation, as follows. An *x-minimax quotient* of $\mathbf{Q}$ is a quotient where the transition relation is given by

$$\begin{aligned} \Delta_x^m = \{([q], a, [q']) \mid a \in \Sigma, q \in E, q' \in \max_a^x(q)\} \\ \cup \{([q], a, [q']) \mid a \in \Sigma, q \in U, q' \in \min_a^x(q)\} \quad . \end{aligned} \tag{1.72}$$

In particular, mixed classes can be declared existential or universal arbitrarily.

We now show that the di-minimax quotient and the original automaton recognize the same language.

We first need some additional insights about maximal successors and the associated strategies.

As a corollary of Lemma 1.4, we find:

Corollary 1.3 *Let $q \in Q, s \in S$ be states of alternating Büchi automata $\mathbf{Q}$ and $\mathbf{S}$ such that $q \equiv_x s$. Let $a \in \Sigma$.*

1. *If* $(q,s) \in E^q \times E^s$ *and* $q' \in \max_a^x(q)$*, then there is a state* $s' \in \max_a^x(s)$ *such that* $q' \equiv_x s'$.

2. *If* $(q,s) \in U^q \times U^s$ *and* $q' \in \min_a^x(q)$*, then there is a state* $s' \in \min_a^x(s)$ *such that* $q' \equiv_x s'$.

3. *If* $(q,s) \in E^q \times U^s$*, then all x-maximal a-successors of q and all x-minimal a-successors of s are x-equivalent.*

Proof. For the first part, let $(q,s) \in E^q \times E^s$ and $q' \in \max_a(q)$. By Lemma 1.4.1, we find an $s' \in \Delta(s,a)$ such that $q' \leq_x s'$. Let $s'' \in \Delta^s(s,a)$ such that $s' \leq_x s''$. Applying Lemma 1.4.1 again, there is a $q'' \in \Delta^q(q,a)$ such that $s'' \leq_x q''$, i.e., since q' is an x-maximal a-successor, $q' \leq_x s' \leq_x s'' \leq_x q'' \leq_x q' \leq_x s'$. Hence s' is an x-maximal a-successor of s and satisfies $q' \equiv_x s'$.

The second part is dual to the case $(q,s) \in E^q \times E^s$.

For the third part, let $(q,s) \in E^q \times U^s$, $q' \in \max_a^x(q)$, $s' \in \min_a^x(s)$. By Lemma 1.4.2, $q' \leq_x s'$. By Lemma 1.4.3, there is a state $q'' \in \Delta^q(q,a)$ and a state $s'' \in \Delta^s(s,a)$ such that $s'' \leq_x q''$. Lemma 1.4.2 shows $q' \leq_x s'' \leq_x q'' \leq_x s'$. But since q' is an x-maximal a-successor, $q'' \leq_x q'$ holds; since s' is an x-minimal a-successor, $s' \leq_x s''$ holds. Hence $q' \equiv_x s'$. So for every $r_0, r_1 \in \min_a^x(s) \cup \max_a^x(q)$, we have $r_0 \equiv_x r_1$, using the transitivity of $\equiv_x$. □

This is the reason why mixed classes can be declared existential or universal in the *di*-minimax quotient: From Corollary 1.3.3, we can conclude the following.

Remark 1.2 *For a mixed class* $M \in Q/\equiv_x$ *and* $a \in \Sigma$*,*

$$\begin{aligned}\{[q'] \mid \exists q(q \in M \cap E \wedge q' \in \max_a(q))\} \\ = \{[q'] \mid \exists q(q \in M \cap U \wedge q' \in \min_a(q))\} \ ,\end{aligned} \tag{1.73}$$

and the size of these sets is 1, i. e., mixed classes are deterministic states of minimax quotients.

By Corollary 1.3, we also have

$$\begin{aligned}\Delta_x^m = \{([q],a,[q']) \mid a \in \Sigma, q \in E, q' \in \max_a^x(q)\} \\ \cup \{([q],a,[q']) \mid a \in \Sigma, [q] \subseteq U, q' \in \min_a^x(q)\} \ .\end{aligned} \tag{1.74}$$

Given an alternating Büchi automaton $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,F)$, the two relations $\leq_{di} \subseteq Q \times Q$ and $\equiv_{di} \subseteq Q \times Q$ obviously have the following property.

Remark 1.3 1. *For all* $q,q' \in Q$*, if* $q \leq_{di} q'$ *and* $q \in F$*, then* $q' \in F$.

2. *For all $q, q' \in Q$, if $q \equiv_{di} q'$, then $q \in F$ iff $q' \in F$.*

Clearly, if $((q_i, s_i), w)$ is a protoplay in an x-game which is conform with a winning strategy for Duplicator, then $q_i \leq_x s_i$ holds for every $i \geq 0$. In the case of direct simulation, the converse is true as well:

Lemma 1.5 *Let $q_0 \leq_{di} s_0$. In the game $G^{di}(q_0, s_0)$, every $\leq_{di}$-respecting strategy for Duplicator is a winning strategy.*

Proof. Let $q_0 \leq_{di} s_0$, and let σ be a $\leq_{di}$-respecting strategy of Duplicator for $G^{di}(q_0, s_0)$. Let $T = ((q_i, s_i)_{i<\omega}, w)$ be a σ-conform $G^{di}(q_0, s_0)$-protoplay. By assumption, we have $q_i \leq_{di} s_i$ for every $i \geq 0$; by Remark 1.3, $s_i \in F^s$ whenever $q_i \in F^q$, for every $i \geq 0$. Hence T is a win for Duplicator and σ is a winning strategy for Duplicator. □

The $\leq_{di}$-respecting strategies are exactly the winning strategies. Of these winning strategies, some are optimal in the sense that they choose moves to maximal successors in the second component and to minimal successors in the first component.

Let σ be a Duplicator strategy for a game $G^x(q_0, s_0)$. We call σ a *minimax strategy* if, for every σ-conform protoplay $T = ((q_i, s_i)_{i<\omega}, w)$ and every $i < \omega$, if $(q_i, s_i) \in U^q \times S$, then $q_{i+1} \in \min^x_{w(i)}(q_i)$, and if $(q_i, s_i) \in Q \times E^s$, then $s_{i+1} \in \max^x_{w(i)}(s_i)$.

We note:

Lemma 1.6 *Let $\mathbf{Q}$, $\mathbf{S}$ be alternating Büchi automata. There is a positional strategy σ of Duplicator such that for all $q \in \mathbf{Q}$, $s \in \mathbf{S}$ where $q \leq_x s$, σ is a $\leq_x$-respecting minimax strategy for $G^x(q, s)$.*

Proof. Using Lemma 1.4, such a strategy can easily be defined. □

Now it is easy to show:

Theorem 1.2 (minimax quotients) *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be an alternating Büchi automaton and $\mathbf{Q}^m$ any di-minimax quotient of $\mathbf{Q}$.*

1. *For all $k_0, q_0 \in Q$ such that $k_0 \leq_{di} q_0$, $\mathbf{Q}(q_0)$ di-simulates $\mathbf{Q}^m([k_0])$ and $\mathbf{Q}^m([q_0])$ di-simulates $\mathbf{Q}(k_0)$, that is, $[k_0] \leq_{di} q_0$ and $k_0 \leq_{di} [q_0]$.*

2. *$\mathbf{Q}$ and $\mathbf{Q}^m$ di-simulate each other, that is, $\mathbf{Q} \equiv_{di} \mathbf{Q}^m$.*

3. *$\mathbf{Q}$ and $\mathbf{Q}^m$ are equivalent, that is, $L(\mathbf{Q}) = L(\mathbf{Q}^m)$.*

Proof. Mixed classes are deterministic states by Remark 1.2, so existential choice is the same as universal branching for these states. Hence, it suffices to consider a quotient $\mathbf{Q}^m$ where every mixed class is existential. Also, it is enough to show the first part, the other parts follow immediately from this.

Let $\mathbf{Q}^m = (Q_{di}, \Sigma, [q_I], \Delta^m, E^m, U^m, F_{di})$ such that $\Delta^m = \Delta^m_{di}$, $E^m = \{[q] \in Q_{di} \mid [q] \cap E \neq \emptyset\}$ and $U^m = Q_{di} \setminus E^m$. We first show that $\mathbf{Q}(q_0)$ *di*-simulates $\mathbf{Q}^m([k_0])$. To do so, we define a positional winning strategy σ of Duplicator for $G^{di}([k_0], q_0)$. First, let σ_{di} be a positional strategy of Duplicator such that σ_{di} is $\leq_{di}$-respecting and minimax for all games $G^{di}(q, q')$ where $q, q' \in Q$ and $q \leq_{di} q'$. Such a strategy exists by Lemma 1.6.

Further, for every class $[k] \in Q_{di}$, let $\mathrm{rep}([k])$ be a fixed representative of that class, i. e., $\mathrm{rep}([k]) \in [k]$. We also require that $\mathrm{rep}([k]) \in E$ if $[k] \in E^m$.

We now define σ as follows. For all $k, q \in Q$, $a \in \Sigma$, let

$$\sigma([k], q, a, du, 1) = ([k], \mathrm{pr}_2(\sigma_{di}(\mathrm{rep}([k]), q, a, du, 1))) \ , \tag{1.75}$$

$$\sigma([k], q, a, du, 0, du, 1) = ([\mathrm{pr}_1(\sigma_{di}(\mathrm{rep}([k]), q, a, du, 0, du, 1))], q, a, du, 1) \ , \tag{1.76}$$

$$\sigma([k], q, a, du, 0) = ([\mathrm{pr}_1(\sigma_{di}(\mathrm{rep}([k]), q, a, du, 0))], q) \ . \tag{1.77}$$

This function is well-defined because σ_{di} is minimax, i. e., the result of σ really is a successor position in $G^{di}(\mathbf{Q}^m, \mathbf{Q})$.

We now show that σ is, in fact, a winning strategy. Consider a round starting in a position $([k], q) \notin E^m \times U$ such that $k \leq_{di} q$. Since σ_{di} is a $\leq_{di}$-respecting minimax strategy, if Duplicator uses σ in this round, then the next round starts in a position $([k'], q')$ such that $k' \leq_{di} q'$.

We also consider the case of a round in which Spoiler acts alone, i. e., the round starts in a position of the form $([k], q) \in E^m \times U$ such that $k \leq_{di} q$. The round continues with the positions $([k], q, a, sp, 0, sp, 1)([k'], q, a, sp, 1)([k'], q')$, that is, there are $\hat{k} \in [k] \cap E$ and $\bar{k} \in [k']$ such that $(\hat{k}, a, \bar{k}) \in \Delta$. Now $k' \leq_{di} q'$ follows directly by Lemma 1.4.2.

This shows that σ is a winning strategy of Duplicator for $G^{di}([k_0], q_0)$, since $k \leq_{di} q$ holds for every position $([k], q)$ that occurs in a σ-conform play. Note that if we had a position $([k], q)$ occurring in such a play with $([k], q) \in F_{di} \times (Q \setminus F)$, then we would have $k \not\leq_{de} q$.

That $\mathbf{Q}^m([q])$ *di*-simulates $\mathbf{Q}(k)$ can be shown using a symmetrical construction and reasoning. To prove this, we now define a Duplicator winning strategy σ

for $G^{di}(k_0,[q_0])$ as follows. For all $k,q \in Q$, $a \in \Sigma$, let

$$\sigma(k,[q],a,du,1) = (k,[\mathrm{pr}_2(\sigma_{di}(k,\mathrm{rep}([q]),a,du,1))]) \ , \tag{1.78}$$

$$\sigma(k,[q],a,du,0,du,1) = (\mathrm{pr}_1(\sigma_{di}(k,\mathrm{rep}([q]),a,du,0,du,1)),[q],a,du,1) \ , \tag{1.79}$$

$$\sigma(k,[q],a,du,0) = (\mathrm{pr}_1(\sigma_{di}(k,\mathrm{rep}([q]),a,du,0)),[q]) \ . \tag{1.80}$$

Again, if a round starts in a position $(k,[q]) \notin E \times U^m$ such that $k \leq_{di} q$ and if Duplicator uses σ in this round, then the next round starts in a position $(k',[q'])$ such that $k' \leq_{di} q'$. This again follows since σ_{di} is a $\leq_{di}$-respecting minimax strategy.

Again, we finally consider the case of a round in which Spoiler acts alone, i. e., the last position is of the form $(k,[q]) \in E \times U^m$ such that $k \leq_{di} q$. The round continues with the positions $(k,[q],a,sp,0,sp,1)(k',[q],a,sp,1)(k',[q'])$, that is, there are $\hat{q} \in [q] \subseteq U$ and $\bar{q} \in [q']$ such that $(\hat{q},a,\bar{q}) \in \Delta$. Now $k' \leq_{di} q'$ follows directly by Lemma 1.4.2.

By an analogous argument as above, it follows that this σ is a Duplicator winning strategy for $G^{di}(k_0,[q_0])$ □

The above proof does not require the set of transitions to be minimal—we may allow more transitions, provided that mixed classes are existential in the quotient and no transitions induced by universal states to non-minimal successors are considered for mixed classes. That is, as a corollary of the proof of Theorem 1.2, we have:

Corollary 1.4 *Let* $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,F)$ *be an alternating Büchi automaton. Let* $\mathbf{Q}' = (Q_{di},\Sigma,[q_I],\Delta',E',U',F_{di})$ *be a quotient w. r. t. direct simulation of* $\mathbf{Q}$ *such that*

- $\Delta^m_{di} \subseteq \Delta'$,
- $[q] \cap E \neq \emptyset$ *implies* $[q] \in E'$, *and,*
- *for every* $q \in U$ *such that* $[q] \cap E \neq \emptyset$, *if* $([q],a,[q']) \in \Delta'$ *then there are* $\hat{q} \in [q] \cap E$, $\bar{q} \in [q']$ *such that* $(\hat{q},a,\bar{q}) \in \Delta$.

Then, $\mathbf{Q}$ *and* $\mathbf{Q}'$ *simulate each other.*

Theorem 1.2 is false for delayed simulation, as we will see in the next section.

1.4.3 Example: Minimax quotient

As an example, we reconsider the automaton of Figure 1.4. Remember that $q_3 \leq_{di} q_1 \leq_{di} q_0 \equiv_{di} q_2$, but $q_3 \not\equiv_{di} q_1$ and $q_1 \not\equiv_{di} q_0$ for this automaton. That is, $\min_b(q_0) = \{q_1\} = \max_b(q_2)$. Figure 1.5 shows the resulting *di*-minimax quotient where the state $[q_0] = [q_2]$ is declared universal.

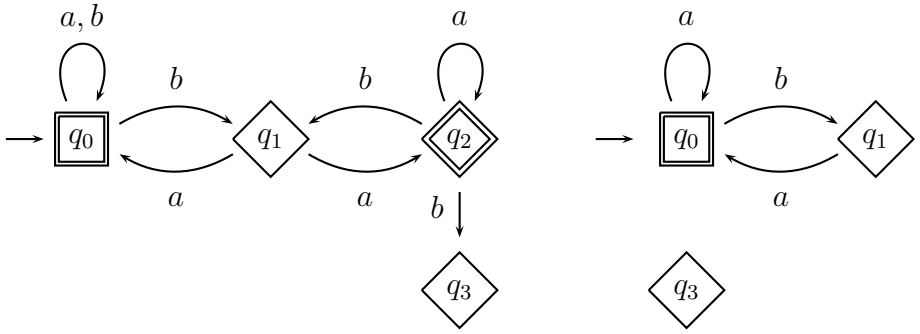

Figure 1.5: The automaton of Figure 1.4 (left) and its *di*-minimax quotient (right)

1.5 Quotienting Modulo Delayed Simulation

If there is a winning strategy for Duplicator in a game $G^{de}(q,s)$, there is also a $\leq_{de}$-respecting minimax strategy (cf. Lemma 1.6), but this may not necessarily be a winning strategy; it is possible that no minimax strategy is winning. Consider the automaton in Figure 1.6.

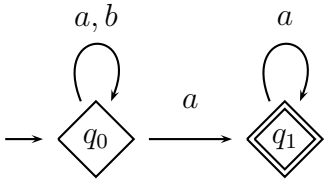

Figure 1.6: *De*-minimax quotients don't work

For $x \in \{de, f\}$, we have $q_0 \geq_x q_1$ but not $q_0 \equiv_x q_1$, i.e., $\max_a(q_0) = \{q_0\}$. That is, for a minimax strategy σ of Duplicator, $\sigma(P^*(q_1, q_0, a, du, 1)) = (q_1, q_0)$ holds.

Hence $((q_1,q_0)(q_1,q_0,a,sp,0,du,1)(q_1,q_0,a,du,1))^\omega$ is a σ-conform $G^x(q_1,q_0)$-play, but not a win for Duplicator. Consequently, the language of any minimax quotient is empty since Δ^m_{de} does not contain a transition from $[q_0]_{de}$ to $[q_1]_{de}$.

To circumvent this problem, we define semi-elective quotients.

1.5.1 Semi-elective quotienting

Let $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,F)$ be an alternating Büchi automaton. In the *semi-elective quotient* of $\mathbf{Q}$, denoted $\mathbf{Q}^{se}_x$, the transition relation is given by

$$\begin{aligned}\Delta^{se}_x = &\{([q],a,[q']) \mid (q,a,q') \in \Delta, q \in E\}\\ &\cup \{([q],a,[q']) \mid a \in \Sigma, [q] \subseteq U, q' \in \min_a(q)\},\end{aligned} \tag{1.81}$$

and every mixed class is declared existential, i. e., $E^{se}_x = \{[q] \in Q_x \mid [q] \cap E \neq \emptyset\}$.

That is, purely universal classes are treated like in the case of minimax quotienting while purely existential and mixed classes are existential states having all transitions induced by their existential states.

By Corollary 1.3.3, we have

$$\begin{aligned}\Delta^{se}_x = &\{([q],a,[q']) \mid (q,a,q') \in \Delta, q \in E\}\\ &\cup \{([q],a,[q']) \mid a \in \Sigma, q \in U, q' \in \min_a(q)\}\ ,\end{aligned} \tag{1.82}$$

where again mixed classes are existential states.

We will show that $\mathbf{Q}$ and $\mathbf{Q}^{se}_x$ simulate each other. For $x = di$, this follows immediately from Corollary 1.4, i. e.:

Corollary 1.5 *For every alternating Büchi automaton $\mathbf{Q}$, the automata $\mathbf{Q}$ and $\mathbf{Q}^{se}_{di}$ simulate each other, in particular, $L(\mathbf{Q}) = L(\mathbf{Q}^{se}_{di})$.*

Note that usually the minimax quotient has less transitions than the semi-elective quotient. That is, for direct simulation, the minimax quotient is the better choice, because it is advantageous to also minimize the number of transitions. This is especially important because states can become unreachable and can thus be deleted as a result of such a minimization.

The more complicated case, where $x = de$, is treated in the following subsections.

1.5.2 Q simulates $\mathbf{Q}^{se}_{de}$

Although a $\leq_{de}$-respecting minimax strategy σ of Duplicator is not necessarily a winning strategy, it is a $\leq_{de}$-respecting winning strategy for Duplicator in the basic simulation game $G(q,s)$; the winning condition is assumed to be trivial in

the sense that if no early loss occurs, Duplicator wins. That is, the basic simulation game is a simulation game in the sense of Subsection 1.2.1 with winning condition P^ω.

We may extend this observation to a basic simulation game $G(K_0, q_0)$ where K_0 is a state of the quotient automaton $\mathbf{Q}^{se}_{de}$ such that $k_0 \leq_{de} q_0$ holds for some $k_0 \in K_0$, which we write as $K_0 \sqsubseteq_{de} q_0$:

Corollary 1.6 *For all $K_0 \in Q_{de}$ and for all $q_0 \in Q$ such that $K_0 \sqsubseteq_{de} q_0$, there is a minimax strategy σ of Duplicator for $G(K_0, q_0)$ such that, for all Spoiler strategies τ for $G(K_0, q_0)$, the (τ, σ)-conform protoplay $((K_i, q_i)_{i<\omega}, w)$ satisfies $K_i \sqsubseteq_{de} q_i$ for every $i < \omega$.*

We then say that σ is a $\sqsubseteq_{de}$-respecting minimax strategy.

Proof. Let $K_0 \in Q_{de}$, $q_0 \in Q$. Let T_i be a prefix of a $G(K_0, q_0)$-play such that the last position of T_i is a P_1-position such that $K_i \sqsubseteq_{de} q_i$. Again, we make a case distinction.

In the first case, if $(K_i, q_i)(K_i, q_i, a, sp, 0, du, 1)(K_{i+1}, q_i, a, du, 1)$ is a suffix of T (hence $K_i \in E^{se}$), we find $k_i \in K_i \cap E$ and $k_{i+1} \in \Delta(k_i, a) \cap K_{i+1}$.

By Lemma 1.4.1, the set $\{q' \in \Delta(q_i, a) \mid k_{i+1} \leq_{de} q'\}$ is not empty. We choose a $\leq_{de}$-maximal element q_{i+1} of this set (which is an element of $\max^{de}_a(q_i)$) and define $\sigma(T) = (K_{i+1}, q_{i+1})$. Hence $K_{i+1} \sqsubseteq_{de} q_{i+1}$.

In the other cases, the suffixes are of the form $(K_i, q_i, a, du, 0, du, 1)$, of the form $(K_i, q_{i+1}, a, du, 0)$, or of the form $(K_i, q_i)(K_i, q_i, a, du, 0, du, 1)(K_{i+1}, q_i, a, du, 1)$ where K_{i+1} is chosen such that there is a $q' \in \Delta(q_i, a)$ satisfying $K_{i+1} \sqsubseteq_{de} q'$. These cases are also treated using Lemma 1.4, i. e., by Lemma 1.4.3 and 1.4.4, we can find a *de*-minimal a-successor K_{i+1} of K_i and use similar arguments if Duplicator has to move in the first component. Note that the case $(K_i, q_i) \in E^{se} \times U$, where Duplicator does not move in the following round, can again be treated by Lemma 1.4.2. □

Moreover, we can show that the join of such a $\sqsubseteq_{de}$-respecting minimax strategy and a Duplicator winning strategy is again $\sqsubseteq_{de}$-respecting.

Corollary 1.7 *Let $K_0 \in Q_{de}$, $q_0 \in Q$ such that $K_0 \sqsubseteq_{de} q_0$, and $s_0 \in S$ such that $q_0 \leq_{de} s_0$. Let σ be a $\sqsubseteq_{de}$-respecting minimax strategy for Duplicator in $G(K_0, q_0)$ and let σ^{de} be a Duplicator winning strategy for $G^{de}(q_0, s_0)$.*

Then $\sigma \bowtie \sigma^{de}$ is a $\sqsubseteq_{de}$-respecting strategy for $G(K_0, s_0)$.

Proof. Let τ be some Spoiler strategy for $G^{de}(K_0, s_0)$, and let $T = ((t_j)_{j<\omega}, w)$ be the $(\tau, \sigma \bowtie \sigma^{de})$-conform protoplay. Initially, we have $K_0 \sqsubseteq_{de} q_0 \leq_{de} s_0$, hence $K_0 \sqsubseteq_{de} s_0$.

Now let $i \in \omega$, and $T_i = ((t_j)_{j \leq i}, w[0..i-1])$ be the prefix of T of length $i+1$. Let $t_i = (K_i, s_i)$, and let $(q_j)_{j \leq i}$ be the intermediate sequence of T_i. Assume $K_i \sqsubseteq_{de} q_i \leq_{de} s_i$.

We show that $K_{i+1} \sqsubseteq_{de} q_{i+1} \leq_{de} s_{i+1}$ holds for the next $(Q_{de} \times S)$-position $t_{i+1} = (K_{i+1}, s_{i+1})$ of T and the next state of the intermediate sequence, distinguishing four cases.

In the first case, let $K_i \subseteq U^q$, $s_i \in U^s$. Let $t_i^U = \tau(\xi(T_i)) = (K_i, s_i, a, sp, 1, du, 0)$ and $t_i^V = \tau(\xi(T_i)t_i^U) = (K_i, s_{i+1}, a, du, 0)$. Let $\sigma \bowtie \sigma^{de}(\xi(T_i)t_i^U t_i^V) = (K_{i+1}, s_{i+1})$, and let q_{i+1} be the next state of the intermediate sequence according to Section 1.3.

If $q_i \in E^q$, the definition of $\sigma \bowtie \sigma^{de}$ implies $K_{i+1} \leq_{de} q_{i+1}$, since $\xi(T^0)$ is σ-conform (both K_{i+1} and q_{i+1} are chosen according to σ). And $q_{i+1} \leq_{de} s_{i+1}$ by Lemma 1.4, since $q_i \leq_{de} s_i$ and $(q_i, s_i) \in E^q \times U^s$. Hence $K_{i+1} \leq_{de} s_{i+1}$.

If $q_i \in U^q$, the definition of $\sigma \bowtie \sigma^{de}$ also implies $K_{i+1} \leq_{de} s_{i+1}$, since $\xi(T^1)$ is σ^{de}-conform (q_{i+1} is chosen according to σ^{de}, hence $q_{i+1} \leq_{de} s_{i+1}$). Because $\xi(T^1_{i+1})$ is σ-conform (i.e., K_{i+1} is chosen according to σ), we have $K_{i+1} \leq_{de} q_{i+1} \leq_{de} s_{i+1}$.

The other cases are shown analogously, i. e., the case $K_i \cap E^q \neq \emptyset$, $s_i \in E^s$ is symmetric to $K_i \subseteq U^q$, $s_i \in U^s$, and in the cases $K_i \cap E^q \neq \emptyset$, $s_i \in U^s$ and $K_i \subseteq U^q$, $s_i \in E^s$, the desired property also results from the definition of $\sigma \bowtie \sigma^{de}$ together with Lemma 1.4. □

And we can easily verify the following.

Lemma 1.7 *Let $K_0, q_0, s_0, \sigma, \sigma^{de}$ be chosen like in Corollary 1.7.*

For every Spoiler strategy τ in $G^{de}(K_0, s_0)$, $q_0 \in F^q$ implies that the $(\tau, \sigma \bowtie \sigma^{de})$-conform play contains a position $(K_j, s_j) \in Q_{de} \times F^s$, i. e., $\sigma \bowtie \sigma^{de}$ is a winning strategy for Duplicator in $G(K_0, s_0)$ with winning set $\{u \in P^\omega \mid \exists i(u_i \in Q_{de} \times F^s)\}$.

Proof. Let τ be a Spoiler strategy for $G^{de}(K_0, s_0)$, and let $q_0 \in F^q$. Let $T = ((t_i)_{i<\omega}, w)$ be the $(\tau, \sigma \bowtie \sigma^{de})$-conform protoplay, and assume that there is no $i \in \omega$ such that $t_i = (K_i, s_i) \in Q_{de} \times F^s$. Since T is $\sigma \bowtie \sigma^{de}$-conform, the play T^1 (as defined in Section 1.3) is σ^{de}-conform. But T^1 is not a win for Duplicator, in contradiction to σ^{de} being a winning strategy for Duplicator. Hence there must be a position $t_i = (K_i, s_i)$ in T such that $s_i \in F^s$. □

We are now ready to show Theorem 1.3, stating that in fact an alternating Büchi automaton simulates its semi-elective quotient w. r. t. delayed simulation. The idea of the proof is that, in order to win the respective simulation game, Duplicator uses the join of a $\sqsubseteq_{de}$-respecting strategy and a winning strategy. But this joint strategy is only $\sqsubseteq_{de}$-respecting and not necessarily a winning strategy:

The intermediate sequence may miss the accepting representatives of the states of the quotient automaton, so that Duplicator may stick to a merely $\sqsubseteq_{de}$-respecting strategy.

As a remedy, we define the Duplicator strategy as a modified join of the two strategies such that Duplicator is forced to reach for accepting states when necessary.

Theorem 1.3 *Let* $\mathbf{Q}$ *be a Büchi automaton, and let* k, q *be states such that* $k \leq_{de} q$. $\mathbf{Q}(q)$ de-*simulates* $\mathbf{Q}^{se}_{de}([k])$*, i. e., there is a winning strategy for Duplicator in* $G^{de}([k], q)$.

Proof. To show that there is a winning strategy σ for Duplicator in $G^{de}([k], q)$, we fix

1. for every $K \in Q_{de}$, a representative $\mathrm{rep}(K) \in K$ such that if $K \cap F \neq \emptyset$ then $\mathrm{rep}(K) \in F$,
2. for every $(K, q) \in Q_{de} \times Q$ such that $K \sqsubseteq_{de} q$, a $\sqsubseteq_{de}$-respecting minimax strategy σ^{o}_{Kq} of Duplicator for $G(K, q)$ (by Corollary 1.6, there is such a strategy), and
3. for every $(k, q) \in Q \times Q$ such that $k \leq_{de} q$, a winning strategy σ^{de}_{kq} of Duplicator for $G^{de}(k, q)$.

For the prefix T_n of a $G^{de}([k], q)$-play T, let $(t_i)_{i \leq n} = (K_i, q_i)_{i \leq n}$ be the subsequence of the $(Q_{de} \times Q)$-positions in T_n. Let

$$j = \min\{i \leq n \mid (K_i, q_i) \in F_{de} \times (Q \setminus F) \ \wedge \forall i'(i \leq i' \leq n \rightarrow q_{i'} \notin F)\} \ , \quad (1.83)$$

or $j = 0$ if this set is empty. Let $T_{[j,i]}$ be the suffix of T_i starting with t_j, and define

$$\sigma(T_i) := \sigma^{o}_{K_j \mathrm{rep}(K_j)} \bowtie \sigma^{de}_{\mathrm{rep}(K_j) q_j}(T_{[j,i]}) \ . \quad (1.84)$$

By Corollary 1.7, σ is $\sqsubseteq_{de}$-respecting. Now if $t_i = (K_i, q_i)$ is the first $(F_{de} \times (Q \setminus F))$-position after the last $(Q_{de} \times F)$-position (or the first $(F_{de} \times (Q \setminus F))$-position at all), we have $K_i \sqsubseteq_{de} q_i$. The strategy σ is updated to $\sigma^{o}_{K_i \mathrm{rep}(K_i)} \bowtie \sigma^{de}_{\mathrm{rep}(K_i) q_i}$ where $\mathrm{rep}(K_i) \in K_i \cap F$, and only the suffix starting with (K_i, q_i) of the play is taken into account for the following moves of Duplicator. (Remember that a joint strategy cannot be assumed to be positional.)

By Lemma 1.7, Duplicator's use of σ forces the play to reach a position (K_j, q_j) in $Q_{de} \times F$ (and $K_j \sqsubseteq_{de} q_j$). Hence every position in $F_{de} \times (Q \setminus F)$ is followed by a position in $Q_{de} \times F$ in a σ-conform play. Thus σ is a winning strategy of Duplicator for $G^{de}([k], q)$. □

1.5.3 $\mathbf{Q}^{se}_{de}$ simulates $\mathbf{Q}$

Theorem 1.3 states that $\mathbf{Q}(q)$ *de*-simulates $\mathbf{Q}^{se}_{de}([k])$. We also want to show that $\mathbf{Q}^{se}_{de}([q])$ *de*-simulates $\mathbf{Q}(k)$. The main idea is quite similar to the previous proof: we will not join a $\leq_{de}$-respecting strategy with a winning strategy, but a winning strategy with a "$\equiv_{de}$-respecting" strategy, ensuring that the intermediate sequence is a path in the sequence of second state components in the plays of $G^{de}(k,[q])$.

We start with the following corollary, a direct consequence of the construction of $\mathbf{Q}^{se}_{de}$ together with Corollary 1.3.

Corollary 1.8 *Let $q'_0 \in [q_0]$. There is a Duplicator strategy $\sigma^{\equiv}$ for $G^{de}(q'_0,[q_0])$ such that, for every $Q \times Q_{de}$-position $(q'_i,[q_i])$ of a $\sigma^{\equiv}$-conform play, $q'_i \in [q_i]$ holds.*

We call such a strategy $\equiv_{de}$*-respecting.*

A $\equiv_{de}$-respecting strategy will replace the $\leq_{de}$-respecting minimax strategy of the previous proof. We will show that the join of a winning strategy for $G^{de}(k_0,q_0)$ and a $\equiv_{de}$-respecting strategy for $G^{de}(q_0,[q_0])$ is a winning strategy for $G^{de}(k_0,[q_0])$.

Theorem 1.4 *Let $\mathbf{Q}$ be a Büchi automaton with states k_0, q_0 such that $k_0 \leq_{de} q_0$. The automaton $\mathbf{Q}^{se}_{de}([q_0])$* de-*simulates $\mathbf{Q}(k_0)$, i. e., there is a winning strategy for Duplicator in $G^{de}(k_0,[q_0])$.*

Proof. Let σ^{de} be a winning strategy of Duplicator for $G^{de}(k_0,q_0)$, and let $\sigma^{\equiv}$ be a $\equiv_{de}$-respecting Duplicator strategy for $G^{de}(q_0,[q_0])$. We show that $\sigma^{de} \bowtie \sigma^{\equiv}$ is a Duplicator winning strategy for $G^{de}(k_0,[q_0])$.

Let τ be a Spoiler strategy for $G^{de}(k_0,[q_0])$. Let $T = (t_i)_{i<\omega}$ be the $(\tau, \sigma^{de} \bowtie \sigma^{\equiv})$-conform protoplay with the intermediate sequence $(q'_i)_{i<\omega}$.

Since $\xi(T^0)$ is σ^{de}-conform, there is, for every $i < \omega$ such that $\mathrm{pr}_1(t_i) \in F$, a $j \geq i$ such that $q'_j \in F$. Since $\xi(T^1)$ is $\sigma^{\equiv}$-conform, we have $q'_j \in \mathrm{pr}_2(t_j)$, hence $t_j \in Q \times F_{de}$. Consequently, $\sigma^{de} \bowtie \sigma^{\equiv}$ is a winning strategy. □

Theorems 1.3 and 1.4 yield:

Theorem 1.5 (semi-elective quotients) *For every alternating Büchi automaton $\mathbf{Q}$, the automata $\mathbf{Q}$ and $\mathbf{Q}^{se}_{de}$* de-*simulate each other, in particular, $L(\mathbf{Q}) = L(\mathbf{Q}^{se}_{de})$.*

1.5.4 Remarks and possible optimizations

In the construction of the quotient automaton, a transition $(q_u, a, q') \in \Delta$ with $q_u \in U$ only results in a transition $([q_u]_{de}, a, [q']_{de}) \in \Delta^{se}_{de}$ if $q' \in \min_a(q_u)$, even if $[q_u]_{de}$ is not a mixed but a purely universal class. This is not a technical trick to permit an easier proof, but a necessity, for without this restriction the resulting quotient automaton would not recognize the language of the original automaton.

Consider the automaton of Figure 1.1 again, and remember that the alphabet is $\{a, b\}$. We have $q_0 \equiv_{de} q_1 >_{de} q_2$. So a quotient construction preserving non-minimal successors of universal states would result in the automaton given in Figure 1.7. But the original automaton accepts b^ω whereas the quotient does not; in the semi-elective quotient w. r. t. delayed simulation, there is no edge from the state $[q_0]$ to itself.

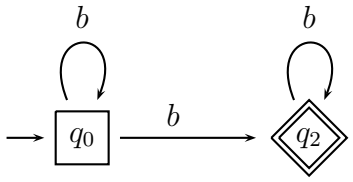

Figure 1.7: A quotient of the automaton in Figure 1.1

Above we saw that in some cases existential classes need transitions to non-maximal successors. In certain situations, not all such transitions are really necessary. For example, accepting classes only need maximal transitions:

Remark 1.4 *Let $\mathbf{Q}^{se'}_{de}$ be the quotient which is defined just as $\mathbf{Q}^{se}_{de}$ but with the transition relation given by*

$$\begin{aligned}\Delta^{se'}_{de} = \{([q], a, [q']) \mid (q, a, q') \in \Delta \wedge q \in E \wedge ([q] \cap F \neq \emptyset \rightarrow q' \in \max_a(q))\} \\ \cup \{([q], a, [q']) \mid a \in \Sigma, [q] \subseteq U, q' \in \min_a(q)\} \quad . \end{aligned} \tag{1.85}$$

Then $\mathbf{Q}^{se'}_{de}$ de-simulates $\mathbf{Q}$.

Proof. In a delayed simulation game, Duplicator can stick to a $\leq_{de}$-respecting minimax strategy until the play reaches an $(F \times (Q \setminus F))$-position, in which case he may be forced, in order to win, to switch to another strategy until the play reaches a $(Q \times F)$-position (cf. the proof of Theorem 1.3). That is, we may assume that a Duplicator winning strategy behaves like a $\leq_{de}$-respecting minimax strategy at all $(Q \times F)$-positions. Hence, only de-maximal successors are necessary at accepting existential states. □

In other words, if an existential state is *de*-equivalent to an accepting state, its transitions to non-*de*-maximal successor states are superfluous.

As a simple example, consider the automaton of Figures 1.4 and 1.5 once again. The semi-elective quotient of this automaton is shown on the left-hand side of Figure 1.8, while the quotient defined according to Remark 1.4 is shown on the right-hand side.

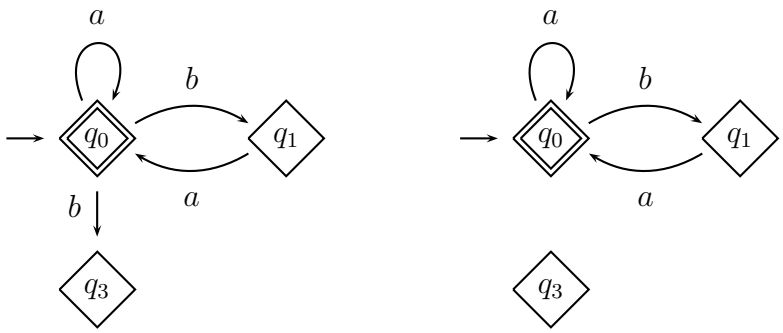

Figure 1.8: The semi-elective and optimized semi-elective quotients of the automaton of Figures 1.4 and 1.5

State $[q_3]$ is disconnected from state $[q_0]$ on the right-hand side because state $[q_0]$ is an accepting state.

In the same vein, a (non-accepting) existential state q does not need a transition (q,a,q') if there is another transition (q,a,q'') such that q'' is accepting and $q' \leq_{de} q''$.

Remark 1.5 *Let $\mathbf{Q}^{se''}_{de}$ be the quotient which is defined just as $\mathbf{Q}^{se'}_{de}$ of Remark 1.4 but with the transition relation given by*

$$\Delta^{se''}_{de} = \Delta^{se'}_{de} \setminus \{([q],a,[q']) \mid [q] \in E^{se}_{de}, \\ \exists([q],a,[q'']) \in \Delta^{se'}_{de} \colon [q''] \in F_{de} \wedge [q'] <_{de} [q'']\} \ . \tag{1.86}$$

Then $\mathbf{Q}^{se''}_{de}$ de-simulates $\mathbf{Q}^{se'}_{de}$ and thus also $\mathbf{Q}$.

Thus, a valid strategy for reducing the number of transitions is extending the set of accepting states without changing the simulation relation. One way how this can be carried out is explained in what follows. A necessary condition for the *de*-equivalence of a state to an accepting state is its *de*-equivalence to an accepting copy of itself, as defined below (without proof). By checking this equivalence to an accepting copy, we can also identify states which are not equivalent to an actual

accepting state in the original automaton, but which can be declared accepting without changing their status w. r. t. $\leq_{de}$.

Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be an ABA. Let $Q' = \{q' \mid q \in Q\}$ be a disjoint copy of Q (analogously $E', U' \subseteq Q'$), and let $\Delta' = \{(q', a, k) \mid (q, a, k) \in \Delta\}$. Let $\mathbf{Q}' = (Q \cup Q', \Sigma, q_I, \Delta \cup \Delta', E \cup E', U \cup U', F \cup Q')$.

We define

$$PF_{de} = \{q \in Q \mid q' \leq_{de} q\} \ . \tag{1.87}$$

The elements of PF_{de} are called *pseudo-accepting states*. Note that $F \subseteq PF_{de}$. We define

$$\mathbf{Q}_{PF} = (Q, \Sigma, q_I, \Delta, E, U, PF_{de}) \ . \tag{1.88}$$

Lemma 1.8 *For every alternating Büchi automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$, we have $\mathbf{Q} \equiv_{de} \mathbf{Q}_{PF}$.*

Proof. Obviously, $\mathbf{Q} \leq_{de} \mathbf{Q}_{PF}$. Conversely, let σ_q be a Duplicator winning strategy for $G^{de}(q', q)$ for every $q \in PF_{de} \setminus F$, where $q' \in PF_{de}$ is the copy of q in $\mathbf{Q}_{PF}$. Let $\sigma_{q\bar{q}}$ be a winning strategy for $G^{de}(q, \bar{q})$ for every pair of states $(q, \bar{q}) \in Q \times Q$ such that $q \leq_{de} \bar{q}$.

To win the game $G^{de}(\mathbf{Q}_{PF}, \mathbf{Q})$, Duplicator starts with the strategy $\sigma = \sigma_{q_I q_I}$, but whenever the play reaches a position (k, q) such that $k \in PF_{de} \setminus F$, $q \in Q \setminus F$, Duplicator switches to the strategy $\sigma_k \bowtie \sigma$. He then uses this strategy until the play reaches a position $(q, \bar{q}) \in Q \times F$; this is guaranteed to happen since both σ_k and σ are winning strategies, and it is also guaranteed that $q \leq_{de} \bar{q}$ holds. At this point, Duplicator changes his strategy to $\sigma = \sigma_{q\bar{q}}$ and continues with that strategy until the play reaches a position in $(PF_{de} \setminus F) \times (Q \setminus F)$ once again, which again forces him to switch his strategy as explained above. By Lemma 1.2, this strategy is winning. □

In summary, when computing the semi-elective quotient, we may treat pseudo-accepting states like accepting states for the purpose of deleting transitions according to Remarks 1.4 and 1.5.

1.5.5 Example: Semi-elective quotient

As an example of the construction of the semi-elective quotient automaton modulo delayed simulation, consider Figure 1.9.

For the automaton $\mathbf{Q}$ on the left, we have $q_2 <_{de} q_1 \equiv_{de} q_5 <_{de} q_0 \equiv_{de} q_3 <_{de} q_4$. Thus there are four states in the quotient automaton $\mathbf{Q}^{se}_{de}$ on the right. Since $\min_b(q_1) = \{q_2\}$, the edge $([q_1], b, [q_1])$ is not in Δ^{se}_{de} (cf. (1.82)); since $\min_a(q_0) =$

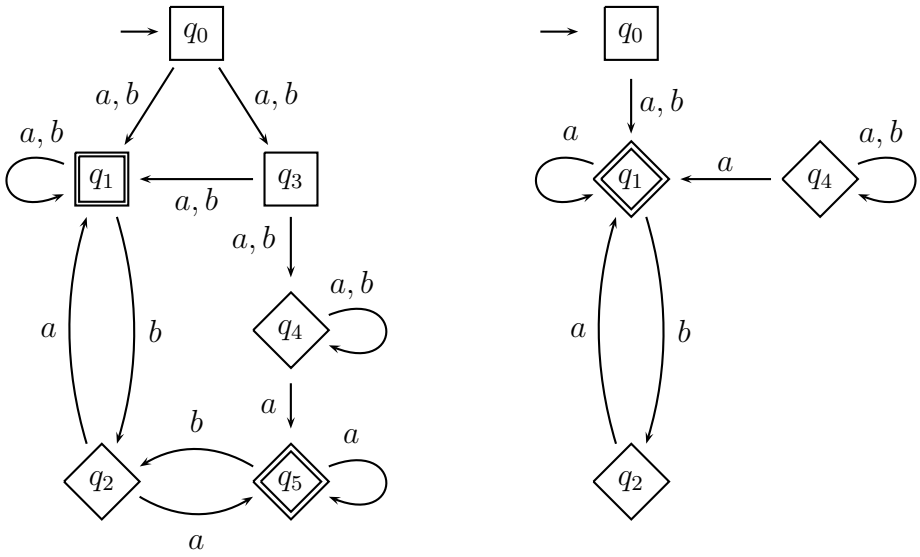

Figure 1.9: Automaton and de-semi-elective quotient

$\min_b(q_0) = \{q_1\}$, there is no edge $([q_0], c, [q_3])$ in Δ^{se}_{de} with $c \in \{a,b\}$. And since $\min_a(q_3) = \min_b(q_3) = \{q_1\}$, there is no edge $([q_3], c, [q_4])$ in Δ^{se}_{de} with $c \in \{a,b\}$. Consequently, the state $[q_4]$ is not reachable in $\mathbf{Q}^{se}_{de}$ and should be removed in a successive optimization of the quotient automaton.

1.6 From Alternating Büchi Automata to Nondeterministic Büchi Automata

Given an alternating Büchi automaton $\mathbf{Q}$, the standard approach for constructing an equivalent nondeterministic (i. e., non-alternating) Büchi automaton is the construction of Miyano and Hayashi [MH84].

In this section, we will show that our simulation relations are compatible with the Miyano–Hayashi construction. That is, if an ABA $\mathbf{Q}$ is simulated by an ABA $\mathbf{S}$, the same holds true for their nondeterministic versions resulting from the Miyano–Hayashi construction. We conclude that our simulation quotienting can be applied to the alternating automaton prior to the Miyano–Hayashi construction without changing its status w. r. t. the simulation relation. This is of practical importance since we can further conclude that our simulation relations can be used for on-the-fly simplifications during the Miyano–Hayashi construction. (However, a subsequent simulation quotienting usually will still improve the

result.) Traditionally, simulation quotienting and simulation-based simplifications are only applied to the nondeterministic automaton.

Figure 1.10 shows these two possible ways from an ABA to a nondeterministic automaton (NBA).

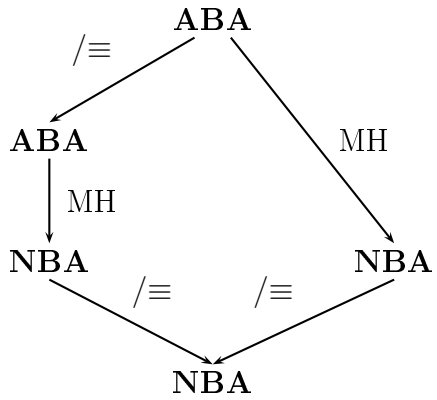

Figure 1.10: Two ways from alternating BA to nondeterministic BA (the symbol $/\equiv$ stands for quotienting)

From a practical point of view, applying simulation quotienting to the alternating automaton is relatively cheap compared to simulation quotienting for the nondeterministic automaton (cf. Sect. 1.7), since the Miyano–Hayashi construction (MH-construction, for short) incurs an exponential growth (see below). For this reason, a state space reduction of the alternating automaton often results in a substantial reduction of the size of the nondeterministic automaton. Aside from these savings in the state space, a smaller intermediate automaton speeds up a subsequent simulation quotienting.

We will now give a short summary of the MH-construction. The construction of Miyano and Hayashi for converting an ABA into a nondeterministic automaton is a subset construction modified for de-universalization instead of determinization. The states are *pairs* (M,N) of subsets of the state set Q. The first component is used in a similar fashion as in the normal subset construction, that is, if there is a universal state q_u in the first component M with a-successors q' and q'', and (M',N') is an a-successor state of (M,N) then $\{q',q''\} \subseteq M'$. The second component is used to keep track of computation branches with an obligation to reach an accepting state, i. e., $N \subseteq M$ and N is disjoint to the set of accepting states F, because a state is deleted from N as soon as its computation branch reaches an accepting state. Especially, (M,N) is accepting if the second component is the empty set, and if (M',N') is a successor state of $(M,\emptyset)$ then $N' = M' \setminus F$.

Formally, for an alternating Büchi automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$, the automaton resulting from the Miyano–Hayashi construction applied to $\mathbf{Q}$ (called the MH-automaton) is

$$\mathbf{Q}_{nd} = (Q_{nd}, \Sigma, q_I^{nd}, \Delta_{nd}, Q_{nd}, \emptyset, F_{nd}) \tag{1.89}$$

with

$$Q_{nd} \subseteq 2^Q \times 2^Q \ , \tag{1.90}$$

$$q_I^{nd} = (\{q_I\}, \{q_I \mid q_I \notin F\}) \ , \tag{1.91}$$

$$F_{nd} = \{(M,N) \in Q_{nd} \mid N = \emptyset\} \ . \tag{1.92}$$

The set of states Q_{nd} contains all pairs $(M,N) \in 2^Q \times 2^Q$ which are reachable from q_I^{nd} via Δ_{nd}. We have $((M,N), a, (M',N')) \in \Delta_{nd}$ if and only if there is a function $f\colon Q \to Q$ such that $f(q) \in \Delta(q,a)$ for all $q \in M \cap E$,

$$M' = \bigcup_{q \in M \cap U} \Delta(q,a) \ \cup \ \{f(q) \mid q \in M \cap E\} \ , \tag{1.93}$$

and $N' = M' \setminus F$ if $N = \emptyset$ or

$$N' = (\bigcup_{q \in N \cap U} \Delta(q,a) \ \cup \ \{f(q) \mid q \in N \cap E\}) \setminus F \tag{1.94}$$

if $N \neq \emptyset$.

Then $\mathbf{Q}_{nd}$ is a nondeterministic Büchi automaton such that $L(\mathbf{Q}) = L(\mathbf{Q}_{nd})$. Note that $\mathbf{Q}_{nd}$ is an exponential size automaton in the number of states of $\mathbf{Q}$ (and that this is necessarily so in the worst case).

Proposition 1.2 *Let* $\mathbf{Q} = (Q, \Sigma, q_I, \Delta^q, E^q, U^q, F^q)$ *and* $\mathbf{S} = (S, \Sigma, s_I, \Delta^s, E^s, U^s, F^s)$ *be alternating Büchi automata, and let* $x \in \{di, de, f\}$. *If* $\mathbf{Q} \leq_x \mathbf{S}$, *then* $\mathbf{Q}_{nd} \leq_x \mathbf{S}_{nd}$.

Proof. The proof is somewhat similar to the construction of a joint strategy (Section 1.3) and the proof of Lemma 1.2. Let σ be a Duplicator winning strategy for $G^x(\mathbf{Q}, \mathbf{S})$. We will now simultaneously and inductively construct a Duplicator strategy σ' for $G^x(\mathbf{Q}_{nd}, \mathbf{S}_{nd})$ and a set of σ-conform $G^x(\mathbf{Q}, \mathbf{S})$-protoplays L. This set is called the *logbook* of the partial σ'-conform play.

Remember that, in a $G^x(\mathbf{Q}_{nd}, \mathbf{S}_{nd})$-protoplay $((P_i)_{i<n}, w)$, the positions P_i are pairs consisting of a state of $\mathbf{Q}_{nd}$ and a state of $\mathbf{S}_{nd}$, and these states in turn are pairs of subsets of Q and S, respectively. Hence, every such position P_i is of the form $((M_i^q, N_i^q), (M_i^s, N_i^s))$ where $N_i^q \subseteq M_i^q \subseteq Q$ and $N_i^s \subseteq M_i^s \subseteq S$.

For such a protoplay $((P_i)_{i<n}, w)$, the logbook L_{n-1} will have the following properties (for every $i < n$), called the logbook properties.

1. The elements of L_{n-1} are σ-conform $G^x(\mathbf{Q},\mathbf{S})$-protoplays over the word w.
2. For every $s \in M_i^s$, there is a $q \in M_i^q$ such that (q,s) is the $(i+1)$th position of an element of L_{n-1}, and, conversely,
3. if (q,s) is the $(i+1)$th position of an element of L_{n-1}, then $q \in M_i^q$, $s \in M_i^s$.

Initially, for the protoplay $((\{q_I\},\{q_I\} \setminus F^q),(\{s_I\},\{s_I\} \setminus F^s)),\varepsilon)$ of length 1, $L_0 = \{((q_I,s_I),\varepsilon)\}$ is a valid logbook.

Now let $T_n = ((P_i)_{i<n},w)$ be a partial σ'-conform $G^x(\mathbf{Q}_{nd},\mathbf{S}_{nd})$-protoplay with logbook L_{n-1}, and assume Spoiler chooses in $\xi(T_n)$ the position $t'_n = ((M_n^q,N_n^q),(M_{n-1}^s,N_{n-1}^s),a)$. (Since $\mathbf{Q}_{nd}$ and $\mathbf{S}_{nd}$ are nondeterministic automata, we may assume that Spoiler chooses a letter and a state simultaneously, cf. [ESW01].)

To define $\sigma'(\xi(T_n)t'_n)$, we only have to define the first component of the state Duplicator chooses, i. e., the set M_n^s, since N_n^s is determined by this choice. For every protoplay $K_{n-1} = ((q_i,s_i)_{i<n},w) \in L_{n-1}$, we distinguish the following four cases. Note that $q_{n-1} \in M_{n-1}^q$ and $s_{n-1} \in M_{n-1}^s$ by the logbook property.

- First case: $(q_{n-1},s_{n-1}) \in E^q \times E^s$. Then, there is a $q_n \in M_n^q$ such that $(q_{n-1},a,q_n) \in \Delta^q$. Let

$$\sigma(\xi(K_{n-1})(q_{n-1},s_{n-1},a,sp,0,du,1)(q_n,s_{n-1},a,du,1)) = (q_n,s_n) \ . \tag{1.95}$$

  We add s_n to M_n^s and $K_n = ((q_i,s_i)_{i<n+1},wa)$ to the logbook L_n.

- Second case: $(q_{n-1},s_{n-1}) \in U^q \times E^s$. Then, $\Delta^q(q_{n-1},a) \subseteq M_n^q$. Let

$$\sigma(\xi(K_{n-1})(q_{n-1},s_{n-1},a,du,0,du,1)) = (q_n,s_{n-1},a,du,1) \tag{1.96}$$

  and

$$\sigma(\xi(K_{n-1})(q_{n-1},s_{n-1},a,du,0,du,1)(q_n,s_{n-1},a,du,1)) = (q_n,s_n) \ . \tag{1.97}$$

  We add s_n to M_n^s and $K_n = ((q_i,s_i)_{i<n+1},wa)$ to the logbook L_n.

- Third case: $(q_{n-1},s_{n-1}) \in E^q \times U^s$. Then, there is a $q_n \in M_n^q$ such that $(q_{n-1},a,q_n) \in \Delta^q$, and it must be the case that $\Delta^s(s_{n-1},a) \subseteq M_n^s$. For every $s_n \in \Delta^s(s_{n-1},a)$, we add the protoplay $((q_i,s_i)_{i<n+1},wa)$ to the logbook L_n.

- Fourth case: $(q_{n-1},s_{n-1}) \in U^q \times U^s$. Then, $\Delta^q(q_{n-1},a) \subseteq M_n^q$, and it must be the case that $\Delta^s(s_{n-1},a) \subseteq M_n^s$. For every $s_n \in \Delta^s(s_{n-1},a)$, let

$$\sigma(\xi(K_{n-1})(q_{n-1},s_{n-1},a,sp,1,du,0)(q_{n-1},s_n,a,du,0)) = (q_n,s_n) \ ; \tag{1.98}$$

  we then add the protoplay $((q_i,s_i)_{i<n+1},wa)$ to the logbook L_n.

Finally, we define $\sigma'(\xi(T_n)t'_n) = ((M^q_n, N^q_n), (M^s_n, N^s_n))$, where the construction of M^s_n is determined by t'_n and L_{n-1} as defined above (and N^s_n in turn is determined by (M^s_n)).

It is easy to check that L_n again has the logbook property and that σ' is a Duplicator strategy for $G^x(\mathbf{Q}_{nd}, \mathbf{S}_{nd})$. We show that σ' is in fact a winning strategy. In the case $x = de$, suppose that Spoiler reaches an accepting state $(M^q_m, \emptyset)$ in the m-th turn of a $G^{de}(\mathbf{Q}_{nd}, \mathbf{S}_{nd})$-play π such that Duplicator is in a non-accepting state (M^s_m, N^s_m), i. e., $N^s_m \neq \emptyset$. Since $N^s_m \subseteq M^s_m$, by the logbook property there is, for every $s \in N^s_m$, a $q \in M^q_m$ such that (q,s) is the current position of a protoplay in the logbook L_m to π such that, in this protoplay, Duplicator has the obligation to reach an accepting state in the second component in order to win. Since the protoplays in the logbook proceed in a σ-conform way, there is a minimal $m' > m$ with the following property: For every protoplay P in the logbook $L_{m'}$, if (q,s) is the m-th position of P and Duplicator has to reach an accepting state in the second component in order to win P, then there is an $l \in \{m+1, \ldots, m'\}$ such that the l-th position of P is of the form (q', s') and $s' \in F^s$. By the above definition of σ', this implies that the m'-th Duplicator state in π is of the form $(M^s_{m'}, \emptyset)$, i. e., an accepting state.

For $x = di$ and $x = f$, analogous argumentations can be used. □

For sets of states $A, A' \subseteq Q$ of an ABA $\mathbf{Q}$, we say that A' is a *set of x-minimal representatives* of A if (1) $A' \subseteq A$ and (2) for every $q \in A \setminus A'$, there is a $q' \in A'$ such that $q' \leq_x q$ but not $q \leq_x q'$. For a state $(M,N) \in 2^Q \times 2^Q$ of $\mathbf{Q}_{nd}$, (M',N') is an *x-pruned state* for (M,N) if N' is a set of x-minimal representatives of N and M' is the union of N' and a set of x-minimal representatives of M. (That is, we have $N' \subseteq M'$ as well as $N \subseteq M$.)

The following corollary follows immediately from the proof of Proposition 1.2.

Corollary 1.9 (pruning of MH-automata) *Let $x \in \{di, de\}$. Let $\mathbf{Q}$ be an ABA, and let (M_0, N_0), (M_1, N_1) be two states of $\mathbf{Q}_{nd}$ such that (M_1, N_1) is an x-pruned state for (M_0, N_0). Let $\mathbf{Q}'_{nd}$ be the result of replacing every transition $((M,N), a, (M_0, N_0))$ of $\mathbf{Q}_{nd}$ by $((M,N), a, (M_1, N_1))$.*

Then $\mathbf{Q}_{nd} \equiv_x \mathbf{Q}'_{nd}$.

That is, non-x-minimal elements of the subsets in the states of $\mathbf{Q}_{nd}$ can be removed, but a state can only be removed from the first component if it is not in the second component or if it also is not x-minimal w. r. t. the second component.

Corollary 1.10 *For every ABA $\mathbf{Q}$ and $x \in \{di, de\}$, $((\mathbf{Q}^x)_{nd})^x \equiv_x (\mathbf{Q}_{nd})^x$ holds.*

Proof. We have $\mathbf{Q}^x \equiv_x \mathbf{Q}$, so by Proposition 1.2, $(\mathbf{Q}^x)_{nd} \equiv_x \mathbf{Q}_{nd}$ holds, so $((\mathbf{Q}^x)_{nd})^x \equiv_x (\mathbf{Q}_{nd})^x$ follows immediately. □

That is, the original alternating automaton, the intermediate automata of Figure 1.10 and the resulting nondeterministic Büchi automaton are all simulation equivalent.

Moreover, optimizations using Corollary 1.9 can be applied on-the-fly, that is, after the construction of every single state of the MH-automaton. These on-the-fly simplifications (for a specialized and optimized variant of the MH-construction) are discussed in detail in Chapter 2.

But note that the simulation quotients of simulation equivalent (alternating or nondeterministic) automata need not be isomorphic: Let $\mathbf{Q}^{de+}$ denote the *de*-semi-elective quotient of $\mathbf{Q}$ optimized using pseudo-accepting states as described in Subsection 1.5.4. Then, $((\mathbf{Q}^{de+})_{nd})^{de+}$, the result of taking the left-hand way in Fig. 1.10, is, for certain instances, smaller than $(\mathbf{Q}_{nd})^{de+}$, the result of the right-hand way. This is because a state (M,N) of $\mathbf{Q}_{nd}$ is pseudo-accepting only if all elements of N are pseudo-accepting. (Without additional optimizations, the left-hand quotient will not be smaller than the right-hand quotient.)

1.7 Efficient Algorithms

Efficient algorithms for computing simulation relations of nondeterministic Büchi automata are given in [ESW01]. We use the same ideas with minor modifications and adjustments for computing simulation relations of alternating Büchi automata. This is explained in the first and third subsection, while in the second subsection, we prove part 1 of Proposition 1.1, which we had postponed earlier. In the fourth subsection, we focus on weak alternating Büchi automata; we present a specific algorithm for computing simulation relations for weak alternating Büchi automata with a lower time complexity.

1.7.1 Modifications for the delayed simulation game

For direct and fair simulation, the winning conditions of the corresponding games can be phrased as parity or even simpler conditions. This is not true for delayed simulation. But a simple expansion of the game graph will achieve this, as pointed out in [ESW01]. The crucial information for the players of a delayed simulation game is whether the play has already visited a position in $\hat{F}^q \cap \bar{F}^s$ without having visited a $\hat{F}^s$-position since or not (cf. Subsection 1.2.1). Following [ESW01], we encode this information in the positions of the delayed simulation game. This yields a Büchi game.

For an alternating automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ and $k, q \in Q$, let

$$G(k,q) = (P, P_0, P_1, (k,q), Z) \tag{1.99}$$

be the basic simulation game according to Section 1.2. We define the game

$$G^{de2}(k,q) = (P^{de}, P_0^{de}, P_1^{de}, (k,q,b_{kq}), Z^{de}, W^{de2}) \tag{1.100}$$

by

$$P^{de} = P \times \{0,1\} \ , \tag{1.101}$$
$$P_0^{de} = P_0 \times \{0,1\} \ , \tag{1.102}$$
$$P_1^{de} = P_1 \times \{0,1\} \ , \tag{1.103}$$
$$W^{de2} = (P^{de*}(P \times \{0\}))^{\omega} \tag{1.104}$$

and

$$Z^{de} = \{((p,b),(p',b)) \in P^{de} \times P^{de} \mid (p,p') \in Z, p' \notin Q \times Q\} \tag{1.105}$$
$$\cup \{((p,b),(p',b)) \in P^{de} \times P^{de} \mid (p,p') \in Z, p' \in (Q \setminus F) \times (Q \setminus F)\} \tag{1.106}$$
$$\cup \{((p,b),(p',0)) \in P^{de} \times P^{de} \mid (p,p') \in Z, p' \in Q \times F\} \tag{1.107}$$
$$\cup \{((p,b),(p',1)) \in P^{de} \times P^{de} \mid (p,p') \in Z, p' \in F \times (Q \setminus F)\}. \tag{1.108}$$

with $b_{kq} = 1$ if $k \in F, q \notin F$ and else $b_{kq} = 0$. Observe that the parameters k and q influence the initial position only. The last component of these states will be called the *winning bit*.

Note that the set PF_{de} of pseudo-accepting states (see Subsection 1.5.4) can be computed together with the simulation relation $\leq_{de}$ without changing the automaton: A state q belongs to PF_{de} if and only if $q \in F$ or $(q,q,1)$ is a winning position of Duplicator in the above game graph.

We define that $k \leq_{de2} q$ holds if Duplicator has a winning strategy for G^{de2}.

Remark 1.6 *The game $G^{de}(k,q)$ is a win for Duplicator if and only if the game $G^{de2}(k,q)$ is a win for Duplicator, i. e., $\leq_{de2} = \leq_{de}$.*

So in the remainder it suffices to consider the games $G^{di}(k,q)$, $G^{de2}(k,q)$, and $G^{f}(k,q)$.

1.7.2 Proof of Proposition 1.1, part 1

From [EJ91, Mos91], it follows that the winner of a game $G^{de2}(q,s)$ always has a positional winning strategy. We can now show that, if Duplicator has a positional winning strategy for $G^{de2}(q,s)$, then he also has a positional winning strategy for $G^{de}(q,s)$. Together with Remark 1.6, this proofs the first part of Proposition 1.1.

The idea of the proof is that Duplicator, using a positional strategy for $G^{de}(\mathbf{Q},\mathbf{S})$, plays with a worst-case assumption: He does not "know" the current winning bit, but whenever he can win under the assumption that the winning bit is 1, he does assume that it is 1 indeed.

Let $\mathbf{Q} = (Q,\Sigma,q_I,\Delta^q,E^q,U^q,F^q)$ and $\mathbf{S} = (S,\Sigma,s_I,\Delta^s,E^s,U^s,F^s)$ be alternating Büchi automata such that Duplicator (Player 1) wins

$$G^{de}(\mathbf{Q},\mathbf{S}) = (P^{de},P_0^{de},P_1^{de},(q_I,s_I,b_I),Z^{de},W^{de2}) \ , \tag{1.109}$$

where $b_I = b_{q_I s_I}$.

Let $D \subseteq Q \times S \times \{0,1\}$ be the set of positions (q,s,b) in $G^{de2}(\mathbf{Q},\mathbf{S})$ such that Duplicator has a (positional) winning strategy for a game starting in (q,s,b); in particular, $(q_I,s_I,b_I) \in D$ by Remark 1.6. For every $(q,s,b) \in D$, let σ_{qsb} be a winning strategy for the game starting in (q,s,b).

We now define a game

$$G^D(\mathbf{Q},\mathbf{S}) = (P^{de},P_0^{de},P_1^{de},(q_I,s_I,b_D),Z^D,W^{de2}) \tag{1.110}$$

such that the set of moves Z^D equals Z^{de2}, but with the following changes. In Z^D, we replace every move $(p,(q,s,0))$ by a move $(p,(q,s,1))$ if $s \notin F^s$ and $(q,s,1) \in D$ (in this case, the winning bit of p is 0). We set $b_D = 1$ if $b_I = 1$, or if $s_I \notin F^s$ and $(q_I,s_I,1) \in D$, else $b_D = 0$. Note that $(q_I,s_I,b_D) \in D$.

Now Duplicator has a winning strategy for $G^D(\mathbf{Q},\mathbf{S})$. In a play of $G^D(\mathbf{Q},\mathbf{S})$, Duplicator starts with the strategy $\sigma_{q_I s_I b_D}$. Whenever a "new" move $(p,(q,s,1)) \in Z^D \setminus Z^{de2}$ is taken in this play, Duplicator switches his strategy to σ_{qs1} (remember that $(q,s,1) \in D$ by definition of Z^D).

This is a winning strategy, because whenever the winning bit switches from 0 to 1, Duplicator effectively plays in $G^{de2}(\mathbf{Q},\mathbf{S})$. Upon taking a move such that the winning bit switches from 0 to 1, no "new" transitions can be taken and Duplicator will not switch his strategy again until the winning bit switches back to 0. And this is guaranteed to happen since Duplicator uses a winning strategy.

That is, Duplicator wins $G^D(\mathbf{Q},\mathbf{S})$, and since $G^D(\mathbf{Q},\mathbf{S})$ is a Büchi game, there is a positional winning strategy σ for Duplicator. Now it easy to see that in a σ-conform play of $G^D(\mathbf{Q},\mathbf{S})$, for every $(q,s) \in Q \times S$, at most one of the positions $(q,s,0)$ and $(q,s,1)$ can be encountered. That is, we can assume that σ is defined for at most one value of the winning bit, for every pair of states, and every σ-conform play π can be mapped (by just deleting the winning bit) to a play π' of

$G^{de}(\mathbf{Q},\mathbf{S})$ such that Duplicator is the winner of π'. In other words, a positional Duplicator winning strategy $\sigma' : P_1 \to P$ for $G^{de}(\mathbf{Q},\mathbf{S})$ can be defined by

$$\sigma'(p) = \begin{cases} \sigma(p,0), & \text{if } \sigma(p,0) \text{ is defined,} \\ \sigma(p,1), & \text{if } \sigma(p,1) \text{ is defined,} \\ \text{undefined}, & \text{else,} \end{cases} \tag{1.111}$$

for all $p \in P$. □

1.7.3 Reduction of the game graphs

By definition and by Remark 1.6 it is clear that in order to determine whether $k \leq_{di} q$, $k \leq_{de} q$, or $k \leq_f q$ holds it is sufficient to determine the winner in the game $G^{di}(k,q)$, $G^{de2}(k,q)$, or $G^f(k,q)$, respectively. A priori, the size of these games can be reduced in order to reduce the complexity of determining whether one state simulates another state. (We can safely ignore the winning bit in the considerations of this subsection.)

We call a position *productive* if it is reachable in the game graph from a $(Q \times Q)$-position. A position $p \in P$ is a *dead end* if no $(Q \times Q)$-position is reachable from p and $p \notin Q \times Q$. Note that the game graph of a complete automaton does not have dead ends.

Remark 1.7 *1. A position $(k',q,a,A',1)$ is productive only if there is a $k \in Q$ such that $(k,a,k') \in \Delta$ and $(k,q) \notin U \times U$.*

2. A position $(k,q',a,A',0)$ is productive only if there is a $q \in U$ such that $(q,a,q') \in \Delta$ and $k \in U$.

3. A position (k,q,a,A,b,A',b') or (k,q,a,A,b) is a dead end if $\Delta(k,a) = \emptyset$ and $b = 0$, or $\Delta(q,a) = \emptyset$ and $b = 1$.

That is, in the game graph of an automaton with n states and m transitions, there are $O(n^2+nm)$ productive states that are not dead ends, and $O(n^2+nm)$ moves between them. Since we may remove all unproductive positions from the game graph we may assume that there are at most $O(|Q|^2+|Q|\cdot|\Delta|)$ positions and moves in the game graph. Since we also may assume that every state is reachable from the initial state, we have $|\Delta| \geq |Q|-1$. Note that the size of the alphabet is not a factor here. So we conclude:

Remark 1.8 *It can be assumed that the game graphs of $G^{di}(k,q)$, $G^{de2}(k,q)$, and $G^f(k,q)$ have $O(|Q|\cdot|\Delta|)$ positions and moves.*

We may now compute the winning sets and thus the relations $\leq_{di}$, $\leq_{de}$ and $\leq_f$ in the reduced game graph using the algorithms given in [ESW01]. This yields:

Theorem 1.6 (computing simulation relations) *Given an alternating Büchi automaton* $\mathbf{Q}$ *with n states and m transitions,* $\leq_{di}$ *can be computed in time* $O(nm)$. *The relations* $\leq_{de}$ *and* $\leq_f$ *can be computed in time* $O(n^3m)$ *and space* $O(nm)$.

The same complexity bounds hold for computing the respective quotients.

1.7.4 Computing simulation relations of weak alternating Büchi automata

A weak alternating Büchi automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ is an alternating Büchi automaton such that every strongly connected component (SCC for short) $C \subseteq Q$ of the transition graph satisfies $C \subseteq F$ or $C \subseteq Q \setminus F$. This strong requirement lets us design more efficient algorithms for computing simulation relations and quotients, similar to what was done in [KVW00] in the context of emptiness tests for weak alternating automata over one-letter alphabets.

The following is easy to see:

Remark 1.9 *If C is an SCC of the game graph of* $G^x(\mathbf{Q}, \mathbf{Q})$ *for* $x \in \{di, de2, f\}$, *there are SCCs* C_0, C_1 *of the transition graph of* $\mathbf{Q}$ *such that* $\{\mathrm{pr}_1(p) \mid p \in C\} \subseteq C_0$ *and* $\{\mathrm{pr}_2(p) \mid p \in C\} \subseteq C_1$.

As a result, if C is an SCC of the game graph of $G^{de2}(\mathbf{Q}, \mathbf{Q})$, precisely one of the following statements holds:

1. For all positions $(p,b) \in C$, $\mathrm{pr}_1(p) \in F$, $\mathrm{pr}_2(p) \in F$ and $b = 0$.
2. For all positions $(p,b) \in C$, $\mathrm{pr}_1(p) \notin F$, $\mathrm{pr}_2(p) \in F$ and $b = 0$.
3. For all positions $(p,b) \in C$, $\mathrm{pr}_1(p) \in F$, $\mathrm{pr}_2(p) \notin F$ and $b = 1$.
4. For all positions $(p,b) \in C$, $\mathrm{pr}_1(p) \notin F$, $\mathrm{pr}_2(p) \notin F$ and $b = 0$.
5. For all positions $(p,b) \in C$, $\mathrm{pr}_1(p) \notin F$, $\mathrm{pr}_2(p) \notin F$ and $b = 1$.

For a game $G^f(\mathbf{Q}, \mathbf{Q})$ the situation is similar but simpler, because there is no winning bit.

That is, for an SCC of the game graph of $G^{de2}(\mathbf{Q}, \mathbf{Q})$ or $G^f(\mathbf{Q}, \mathbf{Q})$ from which no other SCC is reachable the winning positions can be determined just as in an ordinary game: Duplicator wins the delayed game starting in any position of C if and only if the winning bit is 0. For the fair simulation game, types 4 and 5 collapse to a single type of SCC which is a win for Duplicator, and Duplicator

also wins in SCC types 1 and 2. In all other cases Spoiler wins, except for the cases where the SCC consists of a single dead end, but these cases are easy to handle.

Now assume that for an SCC C, the winning positions of all topologically smaller SCCs have already been computed, i. e., for all positions $p \in C$ such that $(p, p') \in Z$ for a $p' \notin C$, we already know whether p' is a winning position either for Spoiler or for Duplicator. If $p \in P_0$ and p' is a win for Spoiler, p also is a win for Spoiler; else if p' is a win for Duplicator, we may simply ignore the move (p, p') in the computation of the winning positions of C (symmetrically for $p \in P_1$). That is, the treatment of C reduces to a game of accessibility in a boolean graph, and can be carried out in linear time, see [And94].

This suggests the following algorithm to compute the winning positions of Duplicator in $G^{de2}(\mathbf{Q}, \mathbf{Q})$ and $G^{f}(\mathbf{Q}, \mathbf{Q})$:

1. Compute the SCCs $C_0, \ldots, C_{n-1}$ of the game graph (the time expense is linear in the number of positions and moves [Tar72], that is, the SCCs can be computed in time $O(nm)$).

2. Compute a topological sorting $C_{i_0} \leq_T C_{i_1} \leq_T \ldots \leq_T C_{i_{n-1}}$ of the SCCs of the game graph (linear in the number of positions and moves [Knu68]).

3. Compute in the order $C_{i_{n-1}}, C_{i_{n-2}}, \ldots, C_{i_0}$ the winning positions for the separate SCCs. Since these are in fact winning positions of reachability games, this can be done in time linear in the number of positions and moves, see [And94].

Using Remark 1.8 and Theorem 1.6, we conclude:

Theorem 1.7 (weak alternating automata) *Given a weak alternating Büchi automaton with n states and m transitions, $\leq_{di}$, $\leq_{de}$ and $\leq_f$ can be computed in time $O(nm)$.*

The same time bound holds for computing the respective quotients.

1.8 Conclusion of Chapter 1

In Chapter 1, we have adapted direct, delayed, and fair simulation relations to alternating Büchi automata, introduced new methods for constructing simulation quotients, and analyzed the complexity of computing these relations and quotients. As a result we can state that even with alternating Büchi automata simulation relations are an appropriate, efficient means for checking language containment and state-space reduction. Weak alternating Büchi automata are closely related to linear temporal logic formulas, so the results open up new directions

for minimizing temporal formulas and automata constructed from these formulas. An application to linear temporal logic and associated automata constructions is discussed in the following chapter.

Chapter 2

Simulation Relations and Büchi Automata from LTL

Propositional linear-time temporal logic (LTL for short) [Pnu77] is a popular language for the specification of system properties. The standard way of model checking an LTL specification against a system is the *automata-theoretic approach to model checking* [VW86]: Translate the negation of the specification into an equivalent nondeterministic Büchi automaton (which incurs an exponential blow-up), build the product of this automaton with the system, and check this product for emptiness—it is empty if and only if the system satisfies the specification.

Obviously, the size of the Büchi automaton for the LTL formula is crucial for the efficiency of the above procedure. But minimizing Büchi automata is computationally difficult: Even testing universality for nondeterministic finite automata on finite strings is **PSPACE**-hard [GJ79]. This implies that approximating a minimum-size ω-automaton (up to a constant factor) is impossible in polynomial time unless **P** = **PSPACE**. Recent results [GS05] show that even approximation up to a polylogarithmic factor cannot be done in polynomial time if **P** $\neq$ **PSPACE**.

In practice, various heuristics are in use for state-space reduction of the resulting automata. Standard techniques are simplifications of the input formula using a set of rewrite rules [MP92], and modifications in the transition structure of the resulting Büchi automaton, cf. [EH00]. Quotienting with respect to simulation or bisimulation equivalences is a sophisticated example for the latter, cf. [DHWT91, ESW01, GBS02].

In this chapter, we discuss the approach of computing a simulation relation *before* the exponential blow-up (see above) occurs. That is, we compute simulation relations for an intermediate *alternating* Büchi automaton. The intermediate automaton can be interpreted as just another way of writing down the LTL formula—especially, the alternating automaton is only linear in the length of the

formula. Consequently, the computation of the relation is fast in comparison to other simulation-based approaches (in the best case, exponentially faster).

In the other approaches, the crucial step is to actually compute a simulation quotient; in the procedure presented here, an outright quotienting of the alternating automaton is not a necessary step of the construction. Instead, we use simulation relations for on-the-fly simplifications in the computation of the result, thus again speeding up the process and saving memory resources. The price of this is that the resulting automaton may still contain simulation equivalent states. Experimental data indicate that this drawback is compensated by the advantage of using alternating automata in an intermediate stage.

Our construction proceeds in three main steps. First, in Section 2.2, we give a very direct translation of the LTL formula to an alternating Büchi automaton in which every state has either a universal or existential modality and in which we allow transitions labeled with the empty word "ε". That is, we do not use alternating automata with transitions described as positive Boolean formulas, as in, e. g., [Var94]; our definition is better suited for simulation games.

In Section 2.3, we give a variant of the game rules of Chapter 1 that takes ε-transitions into account, in order to compute simulation relations on the states of this automaton. The game rules of Arnold and Santocanale [AS03] are quite similar to the rules we use here.

In the third step, we translate the alternating automaton to a nondeterministic (i. e., non-alternating) Büchi automaton, called the *top-down automaton*. In Section 2.4, we therefore introduce a variant of the method of [MH84] (cf. Section 1.6) which is specialized and optimized for the de-universalization of *very weak* alternating Büchi automata. Our approach is similar to [GO01] in its basic concept, but given as a pure translation of usual very weak alternating Büchi automata to nondeterministic automata without going through generalized Büchi automata. Also, we do not use acceptance conditions on automata transitions or co-Büchi acceptance. As in [GO01], the construction of the top-down automaton allows for a simple rule of identifying and deleting superfluous transitions, called *local optimization*. In Section 2.5, we analyze in detail how to use delayed and fair simulation for on-the-fly simplifications in concert with our de-universalization. In Section 2.6, we then give a step-by-step algorithm for the construction of an equivalent NBA for an input LTL formula. Simplifications based on both fair and delayed simulation are applied after the construction of every single state of the NBA. We also discuss possible post-processing steps and give two detailed examples for the results of the algorithm. Also in that section, we report experimental comparisons of a prototypical implementation [FTb] of an earlier version of our algorithm (based on delayed simulation only) to the programs LTL2BA [Odd] and TMP [Ete], using a tool of Tauriainen and Heljanko [TH02]. Our experiments show that the automata produced by our implementation are, on the aver-

age, as good (i. e., as small) as the automata of TMP (the automata of LTL2BA are larger). But with the complexity of the formulas increasing, our program becomes substantially faster than TMP, while LTL2BA is the fastest of the three programs.

The classical approach to LTL-to-NBA translation is tableau-based, as in [VW86] and refined in [GPVW95, DGV99]. For our approach and the refined versions of the tableau-based translation, the worst-case size of the resulting NBA is the same, but it is not clear how the two approaches relate for formulas which do not require NBA of worst-case size. To justify our choice of using a variant of the Miyano–Hayashi construction, we show in Section 2.7 that our de-universalization approach is, in its basic aspects, equivalent to the traditional tableau-based approach of automata construction from LTL: With the local optimization of Section 2.4, the [GPVW95] algorithm and our top-down approach yield the same automaton for input formulas in next normal form, provided that the format of the tableau automaton is adapted in a very straightforward way, similar to what is done in [GL02] and also implemented in TMP [Ete]. We further show that our local optimization covers the improvements based on syntactical implication introduced in [DGV99]; we conclude that the equality of automata in the above sense is also true for the results of the [DGV99] algorithm.

Finally, in Section 2.8, we discuss a variant of the optimized de-universalization, called *bottom-up de-universalization*. The basic idea of this bottom-up approach is to construct the NBA for an LTL formula inductively via nondeterministic subautomata for the subformulas. While the final result of this construction is the same as for the top-down approach of Section 2.4, the bottom-up construction allows to transfer simplifications of the subautomata into simplifications of the final automaton. For example, some or all of the subautomata may be quotient automata with respect to delayed simulation equivalence; these simplifications can then be copied into the final automaton without computing a simulation relation for it.

2.1 Basic Definitions

We identify a natural number n with the set $\{0,\dots,n-1\}$. We define $\max\emptyset = 0$, i. e., the maximum of the empty set is 0. For a sequence $(a_i)_{i<\omega}$, $\mathrm{Inf}((a_i)_{i<\omega})$ is the set of elements appearing infinitely often in $(a_i)_{i<\omega}$, i. e., $\mathrm{Inf}((a_i)_{i<\omega}) = \{a \mid \forall i < \omega \exists j \geq i\colon a = a_j\}$.

We use the notion of alternating and nondeterministic Büchi automata as in Subsection 1.1.2. We will regard a nondeterministic Büchi automaton (NBA) over an alphabet Σ as a 5-tuple

$$\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F) \tag{2.1}$$

where Q is a finite set of states, $q_I \in Q$ is an initial state, $\Delta \subseteq Q \times \Sigma \times Q$ a transition relation, and $F \subseteq Q$ a set of accepting states.

We will also use automata over finite, nonempty sets of propositions. If $\mathbf{Q}$ is an automaton over a set of propositions Σ, then the language of $\mathbf{Q}$ is a set of infinite sequences of subsets of Σ, i. e., an element of $(2^\Sigma)^\omega$. And, following [EH00], the transitions of automata over a set of propositions are labeled by so-called *terms* over the set of propositions. A term is the (possibly empty) conjunction of literals, i. e., positive and negative propositions. That is, the set of terms over Σ is

$$\mathrm{term}_\Sigma = \{ \bigwedge_{p \in M} p \wedge \bigwedge_{q \in N} \neg q \mid M, N \subseteq \Sigma \} \ . \tag{2.2}$$

We say that a set $M \subseteq \Sigma$ satisfies a term $t \in \mathrm{term}_\Sigma$ (written as $M \models t$) if

$$(\bigwedge_{a \in M} a \wedge \bigwedge_{b \in \Sigma \setminus M} \neg b) \rightarrow t \tag{2.3}$$

is a tautology. Note that $\mathtt{tt} \in \mathrm{term}_\Sigma$ (empty conjunction), and $M \models \mathtt{tt}$ for every $M \subseteq \Sigma$. On the other hand, if t is contradictory, i. e., $t \equiv \mathtt{ff}$, then $M \not\models t$ for all $M \subseteq \Sigma$.

For a set of literals M, we denote the term $\bigwedge_{l \in M} l$ as $\mathrm{term}(M)$. Conversely, for $t \in \mathrm{term}_\Sigma$, $\mathrm{lit}(t)$ is the set of literals such that $\mathrm{term}(\mathrm{lit}(t)) = t$ (modulo commutativity). Obviously, for non-contradictory terms $t, t' \in \mathrm{term}_\Sigma$, $t \rightarrow t'$ is equivalent to $\mathrm{lit}(t') \subseteq \mathrm{lit}(t)$.

An NBA $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ over a set of propositions Σ is defined like an NBA over an alphabet, with the difference that the transition relation Δ is a subset of $Q \times \mathrm{term}_\Sigma \times Q$.

Such an automaton $\mathbf{Q}$ over a set of propositions Σ accepts a word $w : \omega \rightarrow 2^\Sigma$ if and only if there is a sequence of states $(q_i)_{i<\omega}$ of $\mathbf{Q}$ such that $q_0 = q_I$ and for every $i < \omega$, there is a $t_i \in \mathrm{term}_\Sigma$ such that $(q_i, t_i, q_{i+1}) \in \Delta$ and $w(i) \models t_i$, and there are infinitely many $i < \omega$ such that $q_i \in F$. We call such a sequence of states an *accepting run* of $\mathbf{Q}$ on w; if we do not require that there are infinitely many $i < \omega$ such that $q_i \in F$, the sequence is just called a *run* of $\mathbf{Q}$ on w.

Alternating Büchi automata over a set of propositions are defined analogously. Especially, the set of moves of the word game $G(\mathbf{Q}, w)$ for an ABA $\mathbf{Q}$ over a set of propositions is

$$\{((q,i),(q',i+1)) \mid \exists t \in \mathrm{term}_\Sigma : w(i) \models t \text{ and } (q,t,q') \in \Delta\} \ . \tag{2.4}$$

LTL formulas over a set of propositions Σ are defined inductively by (1) $\mathtt{tt}$ and a are LTL formulas for every $a \in \Sigma$, and (2) if ψ and ρ are LTL formulas, then so are $\neg\psi$, $\psi \vee \rho$, $\mathsf{X}\psi$ and $\psi \,\mathsf{U}\, \rho$, in words "next ψ" and "ψ until ρ".

LTL formulas are interpreted over infinite sequences of subsets of Σ. For every such ω-word $w\colon \omega \to 2^\Sigma$, we define the relation $\models$ as follows.

$$
\begin{aligned}
w &\models \quad \mathsf{tt} \ , && (2.5)\\
w &\models \quad a \quad \text{iff } a \in w(0) \ , && (2.6)\\
w &\models \quad \neg\psi \quad \text{iff } w \not\models \psi \ , && (2.7)\\
w &\models \quad \psi \vee \rho \quad \text{iff } w \models \psi \text{ or } w \models \rho \ , && (2.8)\\
w &\models \quad \mathsf{X}\psi \quad \text{iff } w[1..] \models \psi \ , && (2.9)\\
w &\models \quad \psi \,\mathsf{U}\, \rho \quad \text{iff } \exists i(w[i..] \models \rho \wedge \forall j < i(w[j..] \models \psi)) \ , && (2.10)
\end{aligned}
$$

where $w[i..]$ is defined by $w[i..](n) = w(i+n)$ for every $n < \omega$. As usual, we will use derived logical operators like ff, $\wedge$, $\rightarrow$, and the temporal operators R, F, G ("releases", "finally" or "eventually", "globally" or "always") defined by

$$
\begin{aligned}
\psi \,\mathsf{R}\, \rho &= \neg(\neg\psi \,\mathsf{U}\, \neg\rho), && (2.11)\\
\mathsf{F}\psi &= \mathsf{tt} \,\mathsf{U}\, \psi, && (2.12)\\
\mathsf{G}\psi &= \mathsf{ff} \,\mathsf{R}\, \psi. && (2.13)
\end{aligned}
$$

That is, the semantics of these operators is

$$
\begin{aligned}
w &\not\models \quad \mathsf{ff} \ , && (2.14)\\
w &\models \quad \mathsf{F}\psi \quad \text{iff } \exists i(w[i..] \models \psi) \ , && (2.15)\\
w &\models \quad \psi \,\mathsf{R}\, \rho \quad \text{iff } \forall i(w[i..] \models \rho \vee \exists j < i(w[j..] \models \psi)) \ , && (2.16)\\
w &\models \quad \mathsf{G}\psi \quad \text{iff } \forall i(w[i..] \models \psi). && (2.17)
\end{aligned}
$$

The language of an LTL formula φ is

$$L(\varphi) = \{w \in (2^\Sigma)^\omega \mid w \models \varphi\} \ . \tag{2.18}$$

The set of subformulas of an LTL formula φ is denoted $\mathrm{sub}(\varphi)$. Literals are regarded as atomic subformulas, i. e., $\mathrm{sub}((\neg a) \,\mathsf{U}\, (\neg b)) = \{(\neg a) \,\mathsf{U}\, (\neg b), \neg a, \neg b\}$.

The length of an LTL formula φ, denoted $|\varphi|$, is the number of symbols, not counting negations and brackets. That is, the formula $\varphi = (\neg a) \,\mathsf{U}\, ((\neg a) \wedge b)$ has length 5.

Theorem 2.1 (Expressiveness of LTL) *For a language L over a set of propositions Σ, the following is equivalent.*

1. *There is an LTL formula φ over Σ such that $L = L(\varphi)$.*
2. *There is a very weak alternating Büchi automaton $\mathbf{Q}$ such that $L = L(\mathbf{Q})$.*

3. *There is a starfree* ω*-regular expression* r *over* Σ *such that* $L = L(r)$.

4. *There is a FO[suc,* $<$*]-formula* φ *over* Σ *such that* $L = L(\varphi)$.

See [Kam68, MP71, Tho90, Roh97, LT00] for more details.

An LTL formula φ is in *negation normal form* if every subformula $\neg\psi$ of φ is of the form $\neg a$ for some $a \in \Sigma$. The formula φ is in *next normal form* if every subformula $\mathsf{X}\psi$ is of the form $\mathsf{XX}\psi'$, $\mathsf{X}a$ or $\mathsf{X}\neg a$ for some $a \in \Sigma$. Using the operators ff, $\wedge$ and R, we can obviously compute an equivalent negation normal form for every LTL formula in linear time. Throughout this chapter, we will assume that LTL formulas are given in negation normal form, if not stated otherwise.

Next normal form can be computed in linear time by using the equivalences

$$\mathsf{Xtt} \equiv \mathsf{tt}\ , \tag{2.19}$$

$$\mathsf{Xff} \equiv \mathsf{ff}\ , \tag{2.20}$$

$$\mathsf{X}(\psi \vee \rho) \equiv (\mathsf{X}\psi) \vee (\mathsf{X}\rho)\ , \tag{2.21}$$

$$\mathsf{X}(\psi \wedge \rho) \equiv (\mathsf{X}\psi) \wedge (\mathsf{X}\rho)\ , \tag{2.22}$$

$$\mathsf{X}(\psi\ \mathsf{U}\ \rho) \equiv (\mathsf{X}\psi)\ \mathsf{U}\ (\mathsf{X}\rho)\ , \tag{2.23}$$

$$\mathsf{X}(\psi\ \mathsf{R}\ \rho) \equiv (\mathsf{X}\psi)\ \mathsf{R}\ (\mathsf{X}\rho)\ . \tag{2.24}$$

2.2 From LTL to Alternating Büchi Automata with ε-Transitions

In this section, we introduce the notion of an alternating Büchi automaton with ε-transitions, or ε-ABA for short, and we give a straightforward translation of LTL formulas into equivalent ε-ABA. This is our first step in the translation of LTL to nondeterministic Büchi automata. In Section 2.3, we introduce a basic simulation game for ε-ABA, generalizing the games defined in Chapter 1.

An ε-ABA is similar to an ABA over a set of propositions as defined in Section 2.1, with the difference that the transition relation Δ is a subset of $Q \times (\mathrm{term}_\Sigma \cup \{\varepsilon\}) \times Q$. That is, we also allow transitions labeled with the empty word.

We say that an ε-ABA $\mathbf{Q}$ is *legal* if there is no infinite sequence $(q_i)_{i<\omega}$ of states of $\mathbf{Q}$ such that $(q_i, \varepsilon, q_{i+1}) \in \Delta$ for every $i < \omega$. We will only consider legal ε-ABA.

We define the language of a legal ε-ABA $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ using a game-theoretic approach as in Section 1.1.2. Given an ω-word w, the *word game* $G(\mathbf{Q}, w)$ for $\mathbf{Q}$ and w is the Büchi game

$$(P, P_0, P_1, p_I, Z, F') \tag{2.25}$$

where

- $P = Q \times \omega \times \{0,1\}$ is the set of positions,
- $P_0 = U \times \omega \times \{0,1\}$ is the set of positions of Player 0, the Pathfinder,
- $P_1 = E \times \omega \times \{0,1\}$ is the set of positions of Player 1, the Automaton,
- $p_I = (q_I, 0, 1)$ is the initial position,
- $Z = \{((s,i,j),(s',i,0)) \mid (s,\varepsilon,s') \in \Delta\}$
  $\cup\ \{((s,i,j),(s',i+1,1)) \mid \exists t' \in \mathrm{term}_\Sigma\colon\ w(i) \models t'\ \wedge\ (s,t',s') \in \Delta\}$ is the set of moves, and
- $F' = F \times \omega \times \{1\}$ is the set of accepting positions.

That is, the word game can be viewed as being played in rounds, and a round ends if one of the players chooses a transition labeled by a term, in which case the third component of a position switches from 0 to 1 (for a legal ε-ABA, every round is finite). The winner is determined by the sequence of the initial states of the rounds: A main difference from the definition in Chapter 1 is that not all visited states are taken into account for acceptance. The language $L(\mathbf{Q})$ of an ε-ABA $\mathbf{Q}$ is

$$L(\mathbf{Q}) = \{w \in (2^\Sigma)^\omega \mid \text{Automaton wins } G(\mathbf{Q}, w)\}\ . \tag{2.26}$$

The translation from LTL to ε-ABA is straightforward using the well-known equivalences $\psi\,\mathsf{U}\,\rho \equiv \rho \vee (\psi \wedge \mathsf{X}(\psi\,\mathsf{U}\,\rho))$ and, dually, $\psi\,\mathsf{R}\,\rho \equiv \rho \wedge (\psi \vee \mathsf{X}(\psi\,\mathsf{R}\,\rho))$, see, e. g., [Var94]. That is, for an LTL formula φ_0, we define the ε-ABA $\mathbf{Q}(\varphi_0) = (Q, \Sigma, q_I, \Delta, E, U, F)$ inductively as follows.

1. The initial state is $q_I = \varphi_0 \in Q$.
2. If $\varphi \in Q$ then
   - if $\varphi \in \{\mathsf{tt}, \mathsf{ff}\}$, $(\varphi, \mathsf{tt}, \varphi) \in \Delta$ is a transition of $\mathbf{Q}(\varphi_0)$,
   - if $\varphi = a$ or $\varphi = \neg a$ for $a \in \Sigma$, then $(\varphi, \varphi, \mathsf{tt}), (\varphi, \mathsf{tt}, \mathsf{ff}) \in \Delta$ and $\mathsf{tt}, \mathsf{ff} \in Q$,[1]
   - if $\varphi = \psi \vee \rho$ or $\varphi = \psi \wedge \rho$, then $(\varphi, \varepsilon, \psi), (\varphi, \varepsilon, \rho) \in \Delta$ and $\psi, \rho \in Q$,
   - if $\varphi = \mathsf{X}\psi$, then $(\varphi, \mathsf{tt}, \psi) \in \Delta$ and $\psi \in Q$,
   - if $\varphi = \psi\,\mathsf{U}\,\rho$, then $(\varphi, \varepsilon, \rho), (\varphi, \varepsilon, \psi \wedge \mathsf{X}\varphi) \in \Delta$ and $\rho, \psi \wedge \mathsf{X}\varphi \in Q$,
   - if $\varphi = \psi\,\mathsf{R}\,\rho$, then $(\varphi, \varepsilon, \rho), (\varphi, \varepsilon, \psi \vee \mathsf{X}\varphi) \in \Delta$ and $\rho, \psi \vee \mathsf{X}\varphi \in Q$,

[1] For technical reasons, we add ff as a "sink state" to complete the game.

- if $\varphi = \mathsf{F}\psi$ or $\varphi = \mathsf{G}\psi$, then $(\varphi, \mathtt{tt}, \varphi), (\varphi, \varepsilon, \psi) \in \Delta$ and $\psi \in Q$.

That is, $\mathrm{sub}(\varphi_0) \subseteq Q$, and Q may also contain auxiliary formulas, e. g., if $\psi \mathsf{U} \rho \in \mathrm{sub}(\varphi_0)$ then $\psi \wedge \mathsf{X}(\psi \mathsf{U} \rho) \in Q$. The number of these auxiliary formulas is linear in $|\mathrm{sub}(\varphi_0)|$. States/formulas of the form $\psi \wedge \rho$, $\psi \mathsf{R} \rho$ and $\mathsf{G}\psi$ are elements of U (universal states), all other states are existential, i. e., elements of E. The set F of accepting states contains all states of the form $\mathtt{tt}$, $\psi \mathsf{R} \rho$, $\mathsf{G}\psi$, $\psi \vee \rho$ and $\mathsf{X}(\psi \mathsf{R} \rho)$.

Note that all nontrivial SCCs of the transition graph of $\mathbf{Q}(\varphi_0)$ either contain a formula $\psi \mathsf{U} \rho$ or a formula $\psi \mathsf{R} \rho$. In the case of an U-formula, all states in such an SCC are not accepting, while all states in the SCC are accepting if it contains an R-formula. That is, with this definition, $\mathbf{Q}(\varphi_0)$ is a *weak* alternating Büchi automaton. This property allows us to faster solve the simulation game (see Subsection 1.7.4). The structure of $\mathbf{Q}(\varphi_0)$ is even more special: In every SCC of the transition graph, there is at most one transition not labeled by ε. For example, if $\psi \mathsf{U} \rho$ is a state in an SCC of $\mathbf{Q}(\varphi_0)$, then the only transition not labeled by ε in that SCC is $(\mathsf{X}(\psi \mathsf{U} \rho), \mathtt{tt}, \psi \mathsf{U} \rho)$. This allows us to treat ε-ABA constructed from LTL as *very weak* alternating Büchi automata for the purpose of de-universalization, see Section 2.4.

Proposition 2.1 (cf. [Var94]) *For all LTL formulas* φ *over* Σ, $L(\varphi) = L(\mathbf{Q}(\varphi))$.

2.3 The Simulation Game for ε-ABA

Next, we define a simulation relation on the states of legal ε-ABA. This relation will be our main tool for on-the-fly simplifications.

As in Section 1.2, our definition of simulation relations for ε-ABA is based on a basic game for which different winning conditions can be defined.

Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be a legal ε-ABA and $p_0, q_0 \in Q$. The *basic simulation game* $G(p_0, q_0)$ is played in rounds by our two players Spoiler and Duplicator, who move two pebbles (a red and a green pebble) on the transition graph of $\mathbf{Q}$. A round ends if a term-labeled transition has been chosen for both pebbles; before that, arbitrarily many ε-labeled transitions can be chosen. We say that a pebble is *free* if no term-labeled transition has been chosen for it in the current round; else it is *locked*. At the beginning of a round, let the red pebble be placed on p and the green pebble on q. Then, the players play as follows.

1. Spoiler chooses a letter $\alpha \in 2^\Sigma$.

2. The progression of the round depends on the modes of p and q, and on the statuses of the pebbles (free or locked). Initially, both pebbles are free, and

the round ends if both pebbles are locked. A player moves a free pebble on a state s by choosing a transition $(s, y, s') \in \Delta$ such that $y = \varepsilon$ or $y \in \mathrm{term}_\Sigma$ and $\alpha \models y$. The round continues with the pebble on s'. If $y \in \mathrm{term}_\Sigma$, the moved pebble becomes locked for the remainder of the round. The following rules determine who of the players has to move which pebble.

- If $p \in E$ and $q \in U$ and both pebbles are free, then Spoiler moves one of the pebbles (he can choose which one).
- If $p \in E$ and the red pebble is free, but $q \in E$ or the green pebble is locked, then Spoiler has to move the red pebble.
- Conversely, if $p \in U$ or the red pebble is locked, but $q \in U$ and the green pebble is free, then Spoiler has to move the green pebble.

If these cases do not apply, Duplicator has to move a free pebble in a symmetric fashion, as follows.

- If $p \in U$ and $q \in E$ and both pebbles are free, then Duplicator chooses one of the pebbles and moves it.
- If $p \in U$ and the red pebble is free while the green pebble is locked, then Duplicator moves the red pebble.
- And if $q \in E$, the green pebble is free, and the red pebble is locked, then Duplicator moves the green pebble.

If a player has to move a pebble but cannot (because there is no appropriate transition), he loses early. Or else the sequence $(p_i, q_i)_{i<\omega}$ of the initial positions of the rounds determines the winner (cf. the rules of the word game in Section 2.2).

While these rules may seem to be intricate and confusing, they in fact result from the direct transformation of the rules of the word game to a simulation game with the proviso that, whenever possible, Spoiler is the first of the players to move a pebble, i. e., Duplicator really has the chance to "duplicate" Spoiler's moves. We illustrate the game rules with a short example in Subsection 2.6.2.

For implementation purposes, there is a problem with these rules, however: If Spoiler can choose an arbitrary element of 2^Σ at the beginning of a round, the size of the game graph obviously is exponential in $|\Sigma|$. To reduce the size of the game, we do not let Spoiler choose an element of 2^Σ in step 1 of a round of a simulation play, but an element of term_Σ instead. Also, we restrict the set of terms from which Spoiler can choose from, based on the position of the red pebble at the beginning of a round. We therefore introduce the mapping posTerms which assigns to every LTL formula φ over Σ the set of terms $\mathrm{posTerms}(\varphi)$ from which

Spoiler can choose from if the red pebble is on state φ. The function posTerms is defined inductively as follows.

$$\begin{aligned}
\text{posTerms}(\mathsf{ff}) &= \emptyset \ , && (2.27)\\
\text{posTerms}(\varphi) &= \{\varphi\}, \text{ if } \varphi \in \{\mathsf{tt}, a, \neg a \mid a \in \Sigma\} \ , && (2.28)\\
\text{posTerms}(\psi \vee \rho) &= \text{posTerms}(\psi) \cup \text{posTerms}(\rho) \ , && (2.29)\\
\text{posTerms}(\psi \wedge \rho) &= \{t \wedge t' \mid t \in \text{posTerms}(\psi),\ t' \in \text{posTerms}(\rho)\} \ , && (2.30)\\
\text{posTerms}(\mathsf{X}\psi) &= \{\mathsf{tt}\} \ , && (2.31)\\
\text{posTerms}(\mathsf{F}\psi) &= \{\mathsf{tt}\} \cup \text{posTerms}(\psi) \ , && (2.32)\\
\text{posTerms}(\mathsf{G}\psi) &= \text{posTerms}(\psi) \ , && (2.33)\\
\text{posTerms}(\psi \,\mathsf{U}\, \rho) &= \text{posTerms}(\psi) \cup \text{posTerms}(\rho) \ , && (2.34)\\
\text{posTerms}(\psi \,\mathsf{R}\, \rho) &= \text{posTerms}(\rho) \\
&\quad \cup \{t \wedge t' \mid t \in \text{posTerms}(\psi),\ t' \in \text{posTerms}(\rho)\} \ . && (2.35)
\end{aligned}$$

Step 1 of a round of the simulation game now is

If the red pebble is on state p, then Spoiler chooses a term $t \in \text{posTerms}(p)$.

In step 2, we now have

A player moves a free pebble on a state s by choosing a transition $(s, y, s') \in \Delta$ such that $y = \varepsilon$ or $y \in \text{term}_\Sigma$ and $t \to y$.

Spoiler loses early if $\text{posTerms}(p) = \emptyset$ or $t \equiv \mathsf{ff}$ for all $t \in \text{posTerms}(p)$.

Note that this definition does not completely prevent the size of posTerms(φ) to be exponential in the length of φ. For example, if $\varphi = \bigwedge_{i<n} a_i \,\mathsf{U}\, b_i$, then

$$\text{posTerms}(\varphi) = \{ \bigwedge_{i \in M} a_i \wedge \bigwedge_{j \in (n \setminus M)} b_j \mid M \in 2^n \} \ . \tag{2.36}$$

That is, $|\varphi| \in O(n)$ and $|\text{posTerms}(\varphi)| = 2^n$. For a formula of the form $a_0 \,\mathsf{R}\, (a_1 \,\mathsf{R}\, (a_2 \ldots \mathsf{R}\, a_{n-1}) \ldots)$, the set of possible terms for Spoiler also is exponential in n. That is, if n is the length of a formula φ and k is its maximal nesting depth of the operators $\wedge$ and R, then the simulation game graph of $G^x(\varphi, \varphi)$ has $2^{O(k \log n)}$ positions and moves, while the ε-ABA $\mathbf{Q}(\varphi)$ only has $O(n)$ states and $O(n)$ transitions. (For the two example formulas, the nesting depth k is $n-1$.)

The formal definition of the game graph is as follows. Let $\mathbf{Q} = \mathbf{Q}(\varphi_0) = (Q, \Sigma, q_I, \Delta^q, E^q, U^q, F^q)$ and $\mathbf{S} = \mathbf{Q}(\varphi_1) = (S, \Sigma, s_I, \Delta^s, E^s, U^s, F^s)$ be two legal ε-ABA constructed for LTL formulas φ_0, φ_1. Let $x \in \{di, de, f\}$. The game $G^x(\mathbf{Q}, \mathbf{S})$ is defined by

$$G^x(\mathbf{Q}, \mathbf{S}) = (P, P_0, P_1, (q_I, s_I), Z, W^x) \ , \tag{2.37}$$

where

$$P = (Q \times S) \cup (Q \times S \times \mathrm{term}_\Sigma \times \{0,1\}^2) \ . \tag{2.38}$$

Here, a state of the form (q,s,t,b,b') describes the following situation of a play: The red pebble is on q, the green pebble is on s, Spoiler has chosen the term t at the beginning of the round, the red pebble is free if $b = 0$ (else it is locked), and the green pebble is free if $b' = 0$ or it is locked if $b' = 1$. Consequently, a state belongs to P_0, the states where Spoiler has to move, if it is in $Q \times S$ or if it of the form (q,s,t,b,b') such that one of the following holds.

- $q \in E^q$, $s \in U^s$, $b = b' = 0$
- $q \in E^q$, $b = 0$, and $s \in E^s$ or $b' = 1$
- $s \in U^s$, $b' = 0$, and $q \in U^q$ or $b = 1$

In all other cases, a state belongs to P_1.

The set $Z \subseteq P \times P$ contains all moves of the form

$$((q,s),(q,s,t,0,0)) \ , \qquad \text{for } t \in \mathrm{posTerms}(q) \ , \tag{2.39}$$

$$((q,s,t,1,1),(q,s)) \ , \tag{2.40}$$

$$((q,s,t,0,b'),(q',s,t,0,b')) \ , \qquad \text{for } (q,s) \notin U^q \times U^s, \ (q,\varepsilon,q') \in \Delta^q \ , \tag{2.41}$$

$$((q,s,t,0,b'),(q',s,t,1,b')) \ , \qquad \text{for } (q,s) \notin U^q \times U^s, \ (q,t',q') \in \Delta^q, \ t \to t' \ , \tag{2.42}$$

$$((q,s,t,1,0),(q,s',t,1,0)) \ , \qquad \text{for } (s,\varepsilon,s') \in \Delta^s \ , \tag{2.43}$$

$$((q,s,t,1,0),(q,s',t,1,1)) \ , \qquad \text{for } (s,t',s') \in \Delta^s, \ t \to t' \ , \tag{2.44}$$

$$((q,s,t,0,0),(q,s',t,0,0)) \ , \qquad \text{for } (q,s) \notin E^q \times E^s, \ (s,\varepsilon,s') \in \Delta^s \ , \tag{2.45}$$

$$((q,s,t,0,0),(q,s',t,0,1)) \ , \qquad \text{for } (q,s) \notin E^q \times E^s, \ (s,t',s') \in \Delta^s, \ t \to t' \ . \tag{2.46}$$

We can use the same winning conditions as in Chapter 1: Direct (*di*), delayed (*de*) and fair (*f*) simulation. (In fact, the definition of the sets W^{di}, W^{de}, and W^f is the same as in Subsection 1.2.1.) Without danger of confusion, we also write $\leq_{di}$, $\leq_{de}$ and $\leq_f$ for the respective simulation relations for legal ε-ABA. That is, for $x \in \{di, de, f\}$ and legal ε-ABA $\mathbf{Q}$, $\mathbf{S}$, we have $\mathbf{Q} \leq_x \mathbf{S}$ if and only if Duplicator has a winning strategy for the simulation $G^x(\mathbf{Q},\mathbf{S})$ with winning condition x.

It is easy to see that these simulation relations share many properties with the simulation relations defined in Chapter 1. Namely, for every legal ε-ABA $\mathbf{Q}$ and for all states p,q of $\mathbf{Q}$, if $p \leq_x q$ then $L(\mathbf{Q}(p)) \subseteq L(\mathbf{Q}(q))$. Also, these relations are preorders (cf. Theorem 1.1 and Corollary 1.2). To show transitivity is even more tedious than in Chapter 1: An exhaustive definition of the join of two

Duplicator strategies for simulation games of legal ε-ABA, similar to what is done in Section 1.3, has to distinct 56 cases (2^3 combinations of modalities combined with $2^3 - 1$ combinations of statuses).

Also as in Chapter 1, $\leq_x$ induces an equivalence relation $\equiv_x$ defined by $p \equiv_x q$ if and only if $p \leq_x q$ and $q \leq_x p$.

Since the ε-ABA that we construct for LTL formulas are weak (see Section 2.2), the simulation games for delayed and fair simulation are in fact reachability games in an AND/OR-graph and can thus be solved asymptotically as fast as a direct simulation game, see Theorem 1.7.

2.4 Optimized De-Universalization of Very Weak Alternating Büchi Automata

The construction of Miyano and Hayashi [MH84] for the de-universalization of ABA can easily be adapted in a straightforward manner to the de-universalization of legal ε-ABA, thereby eliminating the ε-labeled transitions (cf. Section 1.6). However, for an input formula φ of length n with k subformulas of the form $\psi \mathsf{U} \rho$ and $\mathsf{F}\rho$, this standard de-universalization of the alternating Büchi automaton for φ yields an automaton with $\Omega(2^{n+k})$ states in the worst case, while the optimized de-universalization presented in this section yields an automaton of size $O(k2^n)$.

The basic concept of our de-universalization is similar to the construction of [GO01], though we do not use co-Büchi acceptance or acceptance conditions on automata transitions. Also, our construction avoids going through generalized Büchi automata.

We first give the basic concept of an optimized de-universalization for very weak Büchi automata over an alphabet and without ε-transitions. We then show how to apply this concept to ε-ABA constructed from LTL formulas. In Section 2.6, we show how our optimized de-universalization can be used in concert with on-the-fly simplifications based on the simulations relations computed according to Section 2.3.

2.4.1 Basic concept of optimized de-universalization

Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be an alternating Büchi automaton over an alphabet Σ such that there is an injective mapping $\nu : Q \to \omega$ such that $(q, a, q') \in \Delta$ implies $\nu(q) \leq \nu(q')$ for all $q, q' \in Q, a \in \Sigma$. In other words, all SCCs of the transition graph of $\mathbf{Q}$ consist of only one state. We say that $\mathbf{Q}$ is *very weak* (cf. [Roh97, KV97, GO01]).

Let

$$P = \{q \in Q \mid q \notin F \text{ and } \exists a \in \Sigma : (q,a,q) \in \Delta\} \tag{2.47}$$

and $k = |P|$. That is, P contains those states of $\mathbf{Q}$ in which a run on a given word can "get stuck" without accepting the word. Let z be a bijection $P \to \{1,\dots,k\}$.

For $\mathbf{Q}$, we construct an equivalent nondeterministic Büchi automaton as follows. Let

$$\mathbf{Q}_{nd} = (2^Q \times (k+1), \Sigma, (\{q_I\}, z(q_I)), \Delta_{nd}, 2^Q \times \{0\}) \ , \tag{2.48}$$

where $z(q_I) = 0$ if $q_I \notin P$. To define Δ_{nd}, we first define the notion of a *successor set* of a set $M \in 2^Q$ *for* $a \in \Sigma$. A set $M' \in 2^Q$ is a successor set of M for a if, for every $q \in M \cap U$, $\Delta(q,a)$ is a subset of M' and, for every $q \in M \cap E$, $M' \cap \Delta(q,a) \neq \emptyset$. For $M, M' \in 2^Q$ and $a \in \Sigma$, we will write $\mathrm{scs}(M,a,M')$ if M' is a successor set of M for a, i. e., scs is a relation over $2^Q \times \Sigma \times 2^Q$. We will also write, e. g., $\mathrm{scs}(M,a)$ instead of $\{M' \in 2^Q \mid \mathrm{scs}(M,a,M')\}$.

Note that every $M' \in 2^Q$ is a successor set of $\emptyset$ for every $a \in \Sigma$. Further note that if $M = M_0 \cup M_1$ and M'_0 and M'_1 are successor sets for a of M_0 and M_1, respectively, then $M'_0 \cup M'_1$ is a successor set of M for a.

We further define the function next: $2^Q \times (k+1) \to k+1$ by

$$\text{next}: (M,i) \mapsto \max\{z(p) \mid p \in M \cap P,\ z(p) < i\} \ . \tag{2.49}$$

(Remember that $\max \emptyset = 0$ in this chapter.)

In the following, we are only interested in states $(M,i) \in 2^Q \times (k+1)$ with $i = 0$ or $z^{-1}(i) \in M$. We say that these states are *consistent*.

The relation Δ_{nd} is the smallest set that contains the following transitions, for consistent states $(M,i) \in 2^Q \times (k+1)$.

- *Case* $i = 0$. If M' is a successor set of M for a then

$$((M,0), a, (M', \max\{z(p) \mid p \in M' \cap P\})) \in \Delta_{nd} \ . \tag{2.50}$$

- *1st case* $i \neq 0$. If M' is a successor set of $M \setminus \{z^{-1}(i)\}$ for a, N' is a successor set of $\{z^{-1}(i)\}$ for a and $z^{-1}(i) \notin N'$ then

$$((M,i), a, (M' \cup N', \text{next}(M' \cup N', i))) \in \Delta_{nd} \ . \tag{2.51}$$

- *2nd case* $i \neq 0$. If M' is a successor set of $M \setminus \{z^{-1}(i)\}$ for a, N' is a successor set of $\{z^{-1}(i)\}$ for a and $z^{-1}(i) \in N'$ then

$$((M,i), a, (M' \cup N', i)) \in \Delta_{nd} \ . \tag{2.52}$$

That is, every reachable state is consistent.

Intuitively, if a run π of $\mathbf{Q}_{nd}$ on a word w is in a state (M,i), then a play of the word game of $G(\mathbf{Q},w)$ which is compatible with the nondeterministic choices in π, can be in any of the states in M. The value i of the counter component indicates that, in order to get an accepting run, we currently have the obligation to show that if a play currently is in $z^{-1}(i)$, then it will not stay there forever, because in that case, Automaton loses the play.

We claim that $\mathbf{Q}$ and $\mathbf{Q}_{nd}$ are equivalent, i. e.,

Theorem 2.2 *Let $\mathbf{Q}$ be a very weak alternating Büchi automaton with n states, k of which belong to its set P. Then $L(\mathbf{Q}) = L(\mathbf{Q}_{nd})$, and $\mathbf{Q}_{nd}$ has at most $(k+1)2^n \in O(k2^n)$ states.*

Note that for a very weak ABA with n states, the usual de-universalization according to [MH84] results in a nondeterministic automaton with $\Omega(4^n)$ states in the worst case.

Proof. Without loss of generality, we assume that $\mathbf{Q}$ is *complete*, i. e., for every $q \in Q$, $a \in \Sigma$, there is a $q' \in Q$ such that $(q,a,q') \in \Delta$.

First, let $w \in L(\mathbf{Q})$. Since $w \in L(\mathbf{Q})$, there is a memoryless Automaton winning strategy σ for the word game $G(\mathbf{Q},w)$. That is, σ is a function $E \times \omega \to Q$ such that every sequence of states $(q_i)_{i<\omega}$ where

1. $q_0 = q_I$,
2. $(q_i, w(i), q_{i+1}) \in \Delta$ for every $i < \omega$ and
3. $q_{i+1} = \sigma(q_i, i)$ for every $i < \omega$ such that $q_i \in E$

contains infinitely many elements of F. Since $\mathbf{Q}$ is very weak, this is equivalent to the existence of an $n < \omega$ such that $q_n = q_{n+1} = \cdots \in F$.

We now inductively construct an accepting run of $\mathbf{Q}_{nd}$ on w as follows. The first state of the run is $(M_0, m_0) = (\{q_I\}, z(q_I))$. If (M_i, m_i) is the $(i+1)$th state of the run for an $i < \omega$, let

$$M_{i+1} = \bigcup_{q \in M_i \cap U} \Delta(q, w(i)) \cup \{\sigma(q,i) \mid q \in M_i \cap E\} \tag{2.53}$$

and

$$m_{i+1} = \begin{cases} \text{next}(M_{i+1}, k+1) & \text{if } m_i = 0, \\ m_i & \text{if } m_i \neq 0 \text{ and } z^{-1}(m_i) \in U \cap \Delta(z^{-1}(m_i), w(i)), \\ m_i & \text{if } m_i \neq 0,\ z^{-1}(m_i) \in E \\ & \text{and } \sigma(z^{-1}(m_i), i) = z^{-1}(m_i), \\ \text{next}(M_{i+1}, m_i) & \text{else,} \end{cases} \tag{2.54}$$

and let (M_{i+1}, m_{i+1}) be the $(i+2)$th state of the run. It is easy to see that $(M_i, m_i)_{i<\omega}$ is in fact a run of $\mathbf{Q}_{nd}$ on w.

Now assume that $(M_i, m_i)_{i<\omega}$ is not accepting, i. e., there are only finitely many $j < \omega$ s. t. $m_j = 0$. For a transition $((M_i, m_i), w(i), (M_{i+1}, m_{i+1}))$ in Δ_{nd}, we have $m_i \geq m_{i+1}$ or $m_i = 0$. That is, if $(M_i, m_i)_{i<\omega}$ is not an accepting run, there is an $n < \omega$ such that $0 \neq m_n = m_{n+1} = \cdots$. By construction of $(M_i, m_i)_{i<\omega}$, we can then find a sequence of states $(q_i)_{i<\omega}$ such that $q_0 = q_I$, $(q_i, w(i), q_{i+1}) \in \Delta$, $q_{i+1} = \sigma(q_i, w(i))$ if $q_i \in E$ and $q_l = z^{-1}(m_n)$ for all $l \geq n$. That is, $(q_i)_{i<\omega}$ is a σ-conform run of $\mathbf{Q}$ on w, but it is not an accepting run since $z^{-1}(m_n) \notin F$. This contradicts our choice of σ as a winning strategy. Hence $(M_i, m_i)_{i<\omega}$ must be an accepting run and thus $w \in L(\mathbf{Q}_{nd})$.

To show $L(\mathbf{Q}_{nd}) \subseteq L(\mathbf{Q})$, let $w \in L(\mathbf{Q}_{nd})$. Let $(M_i, m_i)_{i<\omega}$ be an accepting run of $\mathbf{Q}_{nd}$ on w. Without loss of generality, we can assume that the run is *minimal* in the following sense: For every $i < \omega$,

$$M_{i+1} = \bigcup_{q \in M_i \cap U} \Delta(q, w(i)) \cup \bigcup_{q \in M_i \cap E} \{q'\} \ , \tag{2.55}$$

where q' is some state in $\Delta(q, w(i))$, for $q \in M_i \cap E$. It is easy to see that, if there is an accepting run of $\mathbf{Q}_{nd}$, there also is a minimal accepting run. We further can assume that, for all M_i and $q \in M_i \cap E$ such that $z(q) = m_i > 0$, the state q' is different from q if and only if m_{i+1} is different from m_i.

Under these assumptions, we can find a mapping $\tau \colon E \times \omega \to Q$ such that, for every $i < \omega$,

$$M_{i+1} = \bigcup_{q \in M_i \cap U} \Delta(q, w(i)) \cup \{\tau(q, i) \mid q \in M_i \cap E\} \ , \tag{2.56}$$

and if $z(q) = m_i > 0$, then $\tau(q, i) \neq q$ if and only if $m_i \neq m_{i+1}$.

This mapping τ is an Automaton winning strategy for $G(\mathbf{Q}, w)$: For every given Pathfinder strategy ρ, we can inductively define a (ρ, τ)-conform play $(q_i)_{i<\omega}$ of $G(\mathbf{Q}, w)$. By construction, we have $q_i \in M_i$ for every $i < \omega$. Since $(M_i, m_i)_{i<\omega}$ is an accepting run, there are infinitely many $j < \omega$ such that $m_j = 0$. Hence it is not the case that there is an $n < \omega$ such that $q_n = q_{n+1} = \cdots \notin F$, because in that case, we have $q_n \in P$ and, if also $q_n \in E$, $\tau(q_n, w(l)) = q_n$ for all $l \geq n$. And this implies that $0 \neq m_p = m_{p+1} = \cdots$ for some $p \geq n$, in contradiction to our choice of $(M_i, m_i)_{i<\omega}$. That is, the play $(q_i)_{i<\omega}$ does not get stuck in some non-accepting state, and consequently, since $\mathbf{Q}$ is very weak, it gets stuck in some accepting state. Hence $(q_i)_{i<\omega}$ is winning for Automaton, which shows that τ is a winning strategy for $G(\mathbf{Q}, w)$, i. e., $w \in L(\mathbf{Q})$. □

2.4.2 Local optimization of Büchi automata from very weak ABA

We now give a simple rule for the deletion of superfluous transitions in nondeterministic Büchi automata constructed from very weak alternating Büchi automata as described above. In Section 2.7, we will see that this local optimization of Büchi automata plays a crucial role in comparing the automata of [GPVW95, DGV99] to our automaton.

Definition 2.1 (local optimization) *Let* $\mathbf{Q}$ *be a very weak alternating Büchi automaton. For two transitions* $((M,i),a,(N,j))$ *and* $((M,i),a,(N',j'))$ *of* $\mathbf{Q}_{nd}$*, we say that* $((M,i),a,(N',j'))$ *is a* better transition *than* $((M,i),a,(N,j))$ *if* $N' \subseteq N$*, and* $j' \leq j$*.*

The locally optimized automaton $\mathbf{Q}^{lo}_{nd}$ *is defined like* $\mathbf{Q}_{nd}$*, only the set of transitions is different. The set of transitions of* $\mathbf{Q}^{lo}_{nd}$ *is*

$$\begin{aligned}\Delta^{lo}_{nd} = \Delta_{nd} \setminus \{((M,i),a,(N,j)) \mid \exists((M,i),a,(N',j')) \in \Delta_{nd}: \\ ((M,i),a,(N',j')) \text{ is better than } ((M,i),a,(N,j))\}\ .\end{aligned} \tag{2.57}$$

That is, $\mathbf{Q}^{lo}_{nd}$ only contains "optimal" transitions. We now show, using the following two lemmas, that local optimization is correct in the sense that $L(\mathbf{Q}^{lo}_{nd}) = L(\mathbf{Q}_{nd})$.

Lemma 2.1 *Let* $\mathbf{Q}$ *be a very weak alternating Büchi automaton. Let* (M_0,i_0)*,* $(N_0,j_0) \in 2^Q \times (k+1)$ *be two states of* $\mathbf{Q}_{nd}$ *and* $\mathbf{Q}^{lo}_{nd}$*, respectively, such that* $N_0 \subseteq M_0$ *and* $j_0 \leq i_0$*. Let* $(M_l,i_l)_{l<\omega}$ *be a run of* $\mathbf{Q}_{nd}$ *on a word* $w \in \Sigma^\omega$ *starting from* (M_0,i_0) *such that for* $n < \omega$*,* $i_n = 0$ *and* $i_l \neq 0$ *for all* $l < n$*.*

Then there is a run $(N_l,j_l)_{l<\omega}$ *of* $\mathbf{Q}^{lo}_{nd}$ *on* w *starting from* (N_0,j_0) *such that there is an* $m \leq n$ *with* $j_m = 0$ *and* $N_l \subseteq M_l$ *for all* $l < \omega$*.*

Proof. We construct the run $(N_l,j_l)_{l<\omega}$ as follows. Assume that we have $M_0 = \{q_1,\ldots,q_p\}$. We find successor sets $M^1_1,\ldots,M^p_1$ of $\{q_1\},\ldots,\{q_p\}$ for $w(0)$ such that $M_1 = \bigcup_{r=1}^{p} M^r_1$ and, if $i_0 \neq 0$ and $z(q_r) = i_0$ for some $1 \leq r \leq p$, then $i_0 = i_1$ if and only if $q_r \in M^r_1$. Now assume that $N_0 = \{q_1,\ldots,q_s\}$ with $s \leq p$. We then find a transition $((N_0,j_0),w(0),(N_1,j_1))$ in $\mathbf{Q}^{lo}_{nd}$ which is at least as good as the transition from (N_0,j_0) via $w(0)$ to $(\bigcup_{r=1}^{s} M^r_1, j'_1)$, where the value j'_1 is equal to j_0 if $j_0 = z(q_r)$ for an $q_r \in N_0$ and $q_r \in M^r_1$; else $j'_1 = \text{next}(\bigcup_{r=1}^{s} M^r_1, j_0)$, or $j'_1 = \text{next}(\bigcup_{r=1}^{s} M^r_1, k+1)$ if $j_0 = 0$. We thus have $N_1 \subseteq \bigcup_{r=1}^{s} M^r_1 \subseteq M_1$.

We continue this construction for all $l < n$, i. e., we inductively choose transitions $((N_l, j_l), w(l), (N_{l+1}, j_{l+1}))$ which are at least as good as $((N_l, j_l), w(l), (\bigcup_{r=1}^{s} M^r_{l+1}, j'_{l+1}))$. Obviously, this ensures that $N_l \subseteq M_l$ for all $l < \omega$.

It remains to show that there is an $m \leq n$ such that $j_m = 0$. This follows directly for $n = 0$. For the case $n > 0$, assume that $j_l \neq 0$ for all $l < n$ (or we are done). Then, it follows by simple induction that $j_l \leq i_l$ for all $l < n$; especially, $j_{n_1} \leq i_{n-1}$. Assume that $z(q_r) = i_{n-1}$. Since $i_n = 0$, we have $q_r \notin M^r_n$ for the successor set M^r_n of $\{q_r\}$ (as a subset of M_n), and $z(q) \geq i_{n-1}$ for all $q \in M_n \cap P$. Consequently, $z(q) \geq i_{n-1} \geq j_{n-1}$ for all $q \in N_n \cap P \subseteq M_n \cap P$. That is, if $j_{n-1} < i_{n-1}$, then $j_n = 0$. Else if $j_{n-1} = i_{n-1}$ then (by choice of j_n according to the construction) $j_n \neq j_{n-1}$, which also implies $j_n = 0$. □

Lemma 2.2 *Let* $\mathbf{Q}$ *be a very weak alternating Büchi automaton. Let* (M_0, i_0), $(N_0, j_0) \in 2^Q \times (k+1)$ *be two states of* $\mathbf{Q}_{nd}$ *and* $\mathbf{Q}^{lo}_{nd}$, *respectively, such that* $N_0 \subseteq M_0$, *but* $i_0 < j_0$. *Let* $(M_l, i_l)_{l<\omega}$ *be a run of* $\mathbf{Q}_{nd}$ *on a word* $w \in \Sigma^\omega$ *starting from* (M_0, i_0) *such that for* $n < n' < \omega$, $i_n = i_{n'} = 0$ *and* $i_l \neq 0$ *for all* $n \neq l < n'$.

Then there is a run $(N_l, j_l)_{l<\omega}$ *of* $\mathbf{Q}^{lo}_{nd}$ *on* w *starting from* (N_0, j_0) *such that there is an* $m \leq n'$ *with* $j_m = 0$, *and* $N_l \subseteq M_l$ *for all* $l < \omega$.

Proof. We construct the run $(N_l, j_l)_{l<\omega}$ as in the proof of Lemma 2.1. That is, we have $N_l \subseteq M_l$ for all $l < \omega$.

It remains to show that there is an $m \leq n'$ with $j_m = 0$.

If $i_0 = 0$ (i. e., $n = 0$), then $i_1 = \text{next}(M_1, k+1)$. Now if $j_1 \neq 0$, then $i_1 = \text{next}(M_1, k+1) = \max\{z(p) \mid p \in M_1 \cap P\} \geq j_1$ and $i_1 \neq 0$, because $N_1 \subseteq M_1$ and thus $N_1 \cap P \subseteq M_1 \cap P$; also, $M_1 \cap P$ is not empty—it contains the state $z^{-1}(j_1)$. The claim now follows by Lemma 2.1.

Else if $i_0 \neq 0$ and $j_l \neq 0$ for all $l \leq n$, then the situation after n steps, where $i_n = 0$, is equal to the case $i_0 = 0$, i. e., by the same arguments, either $j_{n+1} = 0$ or $j_m = 0$ for an $m \leq n'$ by Lemma 2.1, because $0 \neq i_{n+1} \geq j_{n+1}$. □

From Lemmas 2.1 and 2.2, it follows immediately that local optimization is correct.

Theorem 2.3 *Let* $\mathbf{Q}$ *be a very weak alternating Büchi automaton. Then* $L(\mathbf{Q}_{nd}) = L(\mathbf{Q}^{lo}_{nd})$.

Proof. It suffices to show $L(\mathbf{Q}_{nd}) \subseteq L(\mathbf{Q}^{lo}_{nd})$. For an accepting run $(M_l, i_l)_{l<\omega}$ of $\mathbf{Q}_{nd}$ on a word $w \in L(\mathbf{Q}_{nd})$, we construct a run $(N_l, j_l)_{l<\omega}$ of $\mathbf{Q}^{lo}_{nd}$ on w according to what is done in the proof of Lemma 2.1. By combining Lemmas 2.1 and 2.2, it follows that this run is accepting. □

2.4.3 Application to the construction of Büchi automata from LTL

The above construction can be used in a straightforward manner to construct nondeterministic Büchi automata from LTL formulas via legal ε-ABA which are constructed according to Section 2.2. As stated in Section 2.2, these ε-ABA are not very weak in a proper sense (their transition graphs do have SCCs containing more than one state), but they have the following property: If (q_0, ε, q_1) $(q_1, \varepsilon, q_2) \ldots (q_{n-2}, \varepsilon, q_{n-1})$ (q_{n-1}, t, q_n) is a sequence of transitions in such an automaton (where (q_{n-1}, t, q_n) is the only transition labeled by a term $t \in \mathrm{term}_\Sigma$), then either $q_0 = q_n$ or q_0 is not reachable from q_n. That is, since the states of legal ε-ABA constructed from LTL are formulas, q_n is a subformula of q_0. In this sense, these ε-ABA can be regarded as very weak for the purpose of successor sets for terms (see below for a detailed treatment).

Let P_φ be the set of U- and F-subformulas in an LTL formula φ in negation normal form over a set of propositions Σ, and let $k_\varphi = \left|P_\varphi\right|$. Let z be a bijection $P_\varphi \to \{1, \ldots, k_\varphi\}$.

We then define a nondeterministic Büchi automaton $\mathbf{Q}^{td}(\varphi)$ with $L(\mathbf{Q}^{td}(\varphi)) = L(\varphi)$, also called the *top-down automaton* of φ, by

$$\mathbf{Q}^{td}(\varphi) = (Q_\varphi, \Sigma, q_I^\varphi, \Delta^{td}, F_\varphi) \tag{2.58}$$

where

$$Q_\varphi = 2^{\mathrm{sub}(\varphi)} \times (k_\varphi + 1) \ , \tag{2.59}$$

$$\Delta^{td} \subseteq Q_\varphi \times \mathrm{term}_\Sigma \times Q_\varphi \ , \text{ and} \tag{2.60}$$

$$F_\varphi = 2^{\mathrm{sub}(\varphi)} \times \{0\} \ . \tag{2.61}$$

In the state set, formulas of the form $\varphi_0 \wedge \varphi_1$ are identified with the union of the conjunctive subformulas, i. e., a state (M, i) with $\varphi_0 \wedge \varphi_1 \in M$ is identified with $(M \cup \{\varphi_0, \varphi_1\} \setminus \{\varphi_0 \wedge \varphi_1\}, i)$. Especially, the formula $\mathtt{tt}$ is identified with the empty conjunction, i. e., $(\{\mathtt{tt}\}, 0)$ is identified with $(\emptyset, 0)$. In this sense, the initial state of $\mathbf{Q}^{td}(\varphi)$ is $q_I^\varphi = (\{\varphi\}, \max\{z(\psi) \mid \psi \in \{\varphi\} \cap P_\varphi\})$.

We now define the notion of a successor set of an LTL formula (and thus of a state of the legal ε-ABA constructed from LTL) for a term $t \in \mathrm{term}_\Sigma$. The transitions in Δ^{td} are then defined as in Section 2.4, Equations (2.50) to (2.52), where the function next is defined as in Equation (2.49) with $P = P_\varphi$.

The relation scs is the smallest relation that satisfies the following.

- $\mathrm{scs}(\emptyset, \mathtt{tt}) = \{\emptyset\}$
- For singleton sets, we distinguish the following cases.

  - The set $\{\mathsf{ff}\}$ does not have successor sets. If $\varphi = a$ or $\varphi = \neg a$ for some $a \in \Sigma$, then $\mathrm{scs}(\{\varphi\}, \varphi) = \{\emptyset\}$. (The set $\{\mathsf{tt}\}$ is identified with $\emptyset$.)
  - For all $t \in \mathrm{term}_\Sigma$, $\mathrm{scs}(\{\psi \vee \rho\}, t) = \mathrm{scs}(\{\psi\}, t) \cup \mathrm{scs}(\{\rho\}, t)$. (The set $\{\psi \wedge \rho\}$ is identified with $\{\psi, \rho\}$.)
  - $\mathrm{scs}(\{\mathsf{X}\psi\}, \mathsf{tt}) = \{\psi\}$
  - $\mathrm{scs}(\{\psi \,\mathsf{U}\, \rho\}) = \{(t, N \cup \{\psi \,\mathsf{U}\, \rho\}) \mid (t, N) \in \mathrm{scs}(\{\psi\})\} \cup \mathrm{scs}(\{\rho\})$
    (This also covers the case $\varphi' = \mathsf{F}\rho \equiv \mathsf{tt} \,\mathsf{U}\, \rho$.)
  - $\mathrm{scs}(\{\psi \,\mathsf{R}\, \rho\}) = \{(t', N' \cup \{\psi \,\mathsf{R}\, \rho\}) \mid (t', N') \in \mathrm{scs}(\{\rho\})\} \cup \{(t \wedge t', N \cup N') \mid (t, N) \in \mathrm{scs}(\{\psi\}), (t', N') \in \mathrm{scs}(\{\rho\})\}$
    (This also covers the case $\varphi' = \mathsf{G}\rho \equiv \mathsf{ff} \,\mathsf{R}\, \rho$.)

- If $M \in 2^{\mathrm{sub}(\varphi)}$ such that $|M| > 1$, suppose that $M = \{\varphi_0, \ldots, \varphi_{r-1}\}$. Then $\mathrm{scs}(M) = \{(\bigwedge_{i<r} t_i, \bigcup_{i<r} N_i) \mid \forall i < r\colon (t_i, N_i) \in \mathrm{scs}(\{\varphi_i\})\}$.

Theorem 2.4 *Let φ be an LTL formula in negation normal form, and let $\mathbf{Q}(\varphi)$ be the alternating Büchi automaton for φ according to Section 2.2. For a fixed bijection z, $\mathbf{Q}(\varphi)_{nd} = \mathbf{Q}^{td}(\varphi)$, and $L(\varphi) = L(\mathbf{Q}^{td}(\varphi))$.*

Proof. The automaton $\mathbf{Q}^{td}(\varphi)$ results from the de-universalization construction of Subsection 2.4.1 applied to the alternating Büchi automaton for φ as defined in Section 2.2. The claim then directly follows by Theorem 2.2. □

We can adapt the local optimization of Definition 2.1 to this setting by adding a condition regarding the term labels of the transitions.

Definition 2.2 (local optimization and LTL) *Let φ be an LTL formula. For two transitions $((M,i),t,(N,j))$ and $((M,i),t',(N',j'))$ of $\mathbf{Q}^{td}(\varphi)$, we say that $((M,i),t',(N',j'))$ is a* better transition *than $((M,i),t,(N,j))$ if $t \to t'$, $N' \subseteq N$, and $j' \leq j$.*

The locally optimized automaton *$\mathbf{Q}^{lo}(\varphi)$ is defined like $\mathbf{Q}^{td}(\varphi)$, only the set of transitions is different. The set of transitions of $\mathbf{Q}^{lo}(\varphi)$ is*

$$\Delta^{lo} = \Delta^{td} \setminus \{((M,i),t,(N,j)) \mid \exists((M,i),t',(N',j')) \in \Delta^{td}\colon ((M,i),t',(N',j')) \text{ is better than } ((M,i),t,(N,j))\} \quad . \tag{2.62}$$

2.5 Usable Properties of Simulation Relations

We will use the construction of Section 2.4 for translating an alternating to a nondeterministic Büchi automaton. During this construction, we apply a set of simplification rules on-the-fly, i. e., after the construction of every single state of the

NBA. These simplifications are based on our notion of simulation as introduced in Sections 2.3 and 1.2. In this section, we therefore show useful properties of fair and delayed simulation, with a focus on the translation from ε-ABA to NBA and the special structure of ε-ABA from LTL as "almost" very weak automata (as detailed in Sections 2.2 and 2.4).

2.5.1 Deleting transitions using fair simulation

In order to make use of fair simulation for simplification purposes, we need more insights into the properties of this simulation relation. While fair simulation equivalence cannot be used for quotienting [ESW01], we know that fair simulation implies language containment (Theorem 1.1). So, as a starting point, we use the following Theorem taken from Somenzi and Bloem [SB00] (and adapted to our notation). Theorem 2.5 states the following, for nondeterministic automata: If two states p, q are successors of a state r such that the language accepted by every run starting from r and then going via p is included in the language of some run from r via q, then the transitions from r to p can be deleted, provided that r is not reachable from q.

Theorem 2.5 ([SB00]) *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ be a nondeterministic Büchi automaton. For $p, q \in Q$, $p \neq q$, assume that $L(\mathbf{Q}(p)) \subseteq L(\mathbf{Q}(q))$. Let Δ' be the result of removing all transitions (r,t,p) from Δ for which there is a transition $(r,t',q) \in \Delta$ such that $t \to t'$ and r is not reachable from q. Let $\mathbf{Q}' = (Q, \Sigma, q_I, \Delta', F)$.*

Then $L(\mathbf{Q}) = L(\mathbf{Q}')$.

If we replace the requirement $L(\mathbf{Q}(p)) \subseteq L(\mathbf{Q}(q))$ in Theorem 2.5 by $p \leq_f q$, then it follows that $\mathbf{Q} \equiv_f \mathbf{Q}'$, as Lemma 2.3 shows.

Lemma 2.3 *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ be a nondeterministic Büchi automaton. For $p, q \in Q$, $p \neq q$, assume that $p \leq_f q$. Let Δ' be the result of removing all transitions (r,t,p) from Δ for which there is a transition $(r,t',q) \in \Delta$ such that $t \to t'$ and r is not reachable from q. Let $\mathbf{Q}' = (Q, \Sigma, q_I, \Delta', F)$.*

Then $\mathbf{Q} \equiv_f \mathbf{Q}'$.

Proof. Obviously, $\mathbf{Q}' \leq_f \mathbf{Q}$.

To show $\mathbf{Q} \leq_f \mathbf{Q}'$, we have to show that Duplicator wins $G^f(\mathbf{Q}, \mathbf{Q}')$. Let σ_0 be a memoryless Duplicator winning strategy for $G^f(\mathbf{Q}, \mathbf{Q}')$. In order to win $G^f(\mathbf{Q}, \mathbf{Q}')$, Duplicator uses σ_0 until the play reaches a position $(q', r, t) \in Q \times Q \times \text{term}_\Sigma$ such that $\sigma_0(q', r, t) = (q', p)$, but there is no transition $(r, t', p) \in \Delta'$ such that $t \to t'$ (i. e., such a transition is in Δ, but not in Δ').

Then, by construction of Δ', there is a transition $(r, t'', q) \in \Delta'$ such that $p \leq_f q$, $t' \to t''$ and r is not reachable from q. We let Duplicator choose such a transition

(we have $t \to t''$) and continue the play at (q', q). Note that since the play was σ_0-conform so far, we have $q' \leq_f p \leq_f q$. Duplicator can therefore continue the play using a winning strategy σ_1 for $G^f(\mathbf{Q}(q'), \mathbf{Q}(q))$. That is, if Duplicator currently uses the strategy σ_i and encounters the above situation that σ_i requires to choose a transition which is not in Δ', he instead chooses a successor with the properties of q and switches to a new strategy σ_{i+1}.

Since such a switch of the strategy is necessary at most at the predecessors $r_0, \ldots, r_{n-1}$ of p and every such state r_i cannot be visited again if a switch of the strategy was necessary at it, Duplicator will finally stick to a winning strategy σ_m where $m < n$. Consequently, Duplicator wins $G^f(\mathbf{Q}, \mathbf{Q}')$. □

A variant of Lemma 2.3 can be applied to ε-ABA constructed from LTL according to Section 2.2. In these automata, we have to compare ε-labeled transitions starting from the same (existential or universal) state.

Corollary 2.1 (deleting ε-transitions) *Let φ be an LTL formula in negation normal form and $\mathbf{Q}(\varphi) = (Q, \Sigma, q_I, \Delta, E, U, F)$ be the legal ε-ABA according to Section 2.2. Let Δ' be the result of removing all transitions (r, ε, p) from Δ for which there is a transition $(r, \varepsilon, q) \in \Delta$ such that r is not reachable from q and*

(a) either $r \in E$ and $p \leq_f q$,

(b) or $r \in U$ and $q \leq_f p$.

Let $\mathbf{Q}'(\varphi) = (Q, \Sigma, q_I, \Delta', E, U, F)$. Then $\mathbf{Q}(\varphi) \equiv_f \mathbf{Q}'(\varphi)$.

Also from Lemma 2.3, we can deduce that it is possible to delete a transition in favor of a newly introduced transition to an equivalent state, provided that a reachability restraint similar to the above holds. This is also true for alternating Büchi automata.

Corollary 2.2 *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be an alternating Büchi automaton. For $p, q \in Q$, $p \neq q$, assume that $p \equiv_f q$ and p is not reachable from q. Let $Q' = Q \setminus \{p\}$ (analogously E', U', F') and $q'_I = q$ if $q_I = p$, else $q'_I = q_I$. Let Δ' be the result of replacing all transitions $(r, t, p) \in \Delta$ where $r \neq p$ by transitions (r, t, q). Let $\mathbf{Q}' = (Q', \Sigma, q'_I, \Delta', E', U', F')$.*

Then $\mathbf{Q} \equiv_f \mathbf{Q}'$.

Proof. (Sketch) Both in $G^f(\mathbf{Q}, \mathbf{Q}')$ and in $G^f(\mathbf{Q}, \mathbf{Q}')$, Duplicator switches his strategy whenever a pebble uses a new transition; this can happen only finitely often. □

That is, while quotienting w. r. t. fair simulation equivalence is not correct, Corollary 2.2 allows to do a kind of partial quotienting in which a fair equivalence

class is represented by only a subset of its elements. For very weak automata, of course, every *f*-equivalence class contains at least one state from which all the other states are not reachable. In this sense, very weak automata allow a full quotienting w. r. t. fair simulation. We are interested in very weak alternating automata from LTL, however, and since these automata are a direct translation of LTL formulas, rewriting the LTL formula is as good as changing its alternating automaton. That is, Corollaries 2.1 and 2.2 directly translate into simulation-based modifications of an LTL formula.

Corollary 2.3 (fair simulation rewriting) *Let* φ *be an LTL formula in negation normal form. Let* φ' *be the result of syntactically replacing*

1. *every subformula* ψ *of* φ *by a subformula* ρ *if* $\psi \equiv_f \rho$ *and* ψ *is not a subformula of* ρ*,*
2. *every subformula* $\psi \vee \rho$ *by* ρ *if* $\psi \leq_f \rho$*,*
3. *every subformula* $\psi \wedge \rho$ *by* ρ *if* $\rho \leq_f \psi$*,*
4. *every subformula* $\psi \mathsf{U} \rho$ *by* ρ *if* $\psi \wedge \mathsf{X}(\psi \mathsf{U} \rho) \leq_f \rho$ *or, equivalently, if* $\psi \mathsf{U} \rho \leq_f \rho$*, and*
5. *every subformula* $\psi \mathsf{R} \rho$ *by* ρ *if* $\rho \leq_f \psi \vee \mathsf{X}(\psi \mathsf{R} \rho)$ *or, equivalently, if* $\rho \leq_f \psi \mathsf{R} \rho$.

Then $\varphi \equiv_f \varphi'$.

Proof. Item 1 is the application of Corollary 2.2. Items 2 to 5 follow with Corollary 2.1.

For items 4 and 5, note that if $\psi \leq_f \rho$ then, clearly, also $\psi \wedge \mathsf{X}(\psi \mathsf{U} \rho)) \leq_f \rho$, and if $\rho \leq_f \psi$ then $\rho \leq_f \psi \vee \mathsf{X}(\psi \mathsf{R} \rho))$. □

Another useful observation is that it is possible to delete transitions to non-*f*-maximal successors of an NBA-state if this state is accepting or if there also is a transition to an accepting state which fairly simulates the disconnected successor. This is similar (but weaker) to what was shown in Remark 1.4.

Lemma 2.4 *Let* $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ *be a nondeterministic Büchi automaton. Let* Δ' *be the result of removing all transitions* (q,t,q') *from* Δ *for which there is a transition* $(q,t',q'') \in \Delta$ *such that* $q' \neq q''$*,* $q' \leq_f q''$*,* $t \to t'$*, and* $q \in F$ *or* $q'' \in F$*. Let* $\mathbf{Q}' = (Q, \Sigma, q_I, \Delta', F)$*.*

Then $\mathbf{Q} \equiv_f \mathbf{Q}'$.

Proof. Obviously, $\mathbf{Q}' \leq_f \mathbf{Q}$. To show $\mathbf{Q} \leq_f \mathbf{Q}'$, we construct a Duplicator winning strategy σ for $G^f(\mathbf{Q},\mathbf{Q}')$ as follows. Let σ_0 be a memoryless Duplicator winning strategy for $G^f(\mathbf{Q},\mathbf{Q})$, i. e., σ_0 is a partial function $Q \times Q \times \mathrm{term}_\Sigma \to Q$. Let D_0 be the domain of σ_0, and let $D_0' \subseteq D_0$ be those positions (p',q,t) for which $\sigma_0(p',q,t) = q'$, but $(q,t',q') \notin \Delta'$ for all terms t' such that $t \to t'$.

By construction of Δ', for every such position (p',q,t), we find a transition $f_0(p',q,t) = (q,t',q'') \in \Delta'$ such that $p' \leq_f q' \leq_f q''$, $t \to t'$ and $q'' \in F$ if $q \notin F$. We define $\sigma(p',q,t) = \sigma_0(p',q,t)$ if $(p',q,t) \in D_0 \setminus D_0'$, and $\sigma(p',q,t) = q''$ such that $f_0(p',q,t) = (q,t',q'')$ if $(p',q,t) \in D_0'$.

Now σ may be undefined at positions $(p',q'') = \sigma(p',q,t)$ where $(p',q,t) \in D_0'$ (if $\sigma(p',q,t) \notin D_0$). In the next step of the construction of σ, we choose such a state (p',q''). Since $p' \leq_f q''$, we find a Duplicator winning strategy σ_1 for $G^f(\mathbf{Q}(p'),\mathbf{Q}(q''))$. Let D_1 be the domain of σ_1 without D_0, and let $D_1' \subseteq D_1$ be those positions in D_1 where σ_1 requires Duplicator to choose a transition which is not in Δ'.

As above, we find new transitions in Δ' with the above properties for these positions. Note that for every position $(p',q,t) \in D_1'$ and the new transition $f_1(p',q,t) = (q,t',q'')$ for such a position, at least one of q and q'' is in F. Similar to the first step, we define $\sigma(p',q,t) = \sigma_1(p',q,t)$ for $(p',q,t) \in D_1 \setminus D_1'$ and $\sigma(p',q,t) = (p',q'')$ such that $f_1(p',q,t) = (q,t',q'')$, for $(p',q,t) \in D_1'$.

We continue this construction until the domain of σ is closed, i. e., until σ is defined on every position reachable in a σ-conform play. Assume that $n < \omega$ strategies $\sigma_0,\ldots,\sigma_{n-1}$ were used to construct σ.

Then σ is a (memoryless) Duplicator winning strategy for $G^f(\mathbf{Q},\mathbf{Q}')$. This is because every reachable position of the form (p,q) in a σ-conform play satisfies $p \leq_f q$, and because every Duplicator move in a σ-conform play is a move

- from a position in $D_i \setminus D_i'$ to a position in $D_j \setminus D_j'$ such that $j \leq i$, or
- from $D_i \setminus D_i'$ to D_j' such that $j \leq i$, or
- from D_i' to D_j,

for $i,j < n$.

So if moves of the second and third sort occur only finitely often in a σ-conform play, then the play will eventually stay in some set of positions $D_i \setminus D_i'$, and hence Duplicator will eventually only make σ_i-conform moves, for some $i < n$. Then, since σ_i is a winning strategy, Duplicator wins.

Or moves of the second and third sort occur infinitely often. Then, by construction of σ and the sets D_i', the Duplicator component is accepting infinitely often. Hence Duplicator also wins in this case. □

The following observation gives us a sufficient (but not necessary) condition to check fair simulation on-the-fly during the de-universalization construction of Section 2.4. Together with Lemma 2.4, this will allow us to identify unnecessary transitions of a state immediately after the state and its direct successor states are constructed.

Lemma 2.5 (deduced simulation in the NBA) *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ be a very weak alternating Büchi automaton, and let $\mathbf{Q}_{nd}$ be the NBA constructed from $\mathbf{Q}$ according to Section 2.4 based on the bijection $z\colon P \to \{1,\dots,k\}$. Let $(M_0, i_0), (N_0, j_0) \in 2^Q \times (k+1)$ be two states of $\mathbf{Q}_{nd}$ such that for every $q \in N_0$ there is a $q' \in M_0$ such that $q' \leq_f q$.*

Then $(M_0, i_0) \leq_f (N_0, j_0)$.

Proof. To show that Duplicator wins $G^f((M_0, i_0), (N_0, j_0))$, we use a construction similar to the logbook in the proof of Proposition 1.2: During a play of $G^f((M_0, i_0), (N_0, j_0))$, we update an assignment of the states in the N-component to states in the M-component, to a winning strategy and to a protoplay.

Initially, let $g_0\colon N_0 \to M_0$ be a function such that $g_0(q) \leq_f q$ for every $q \in N_0$. For every $q \in N_0$, let σ_q be a memoryless Duplicator winning strategy for $G^f(g_0(q), q)$. For technical convenience, we assume that there is a total ordering on the strategies $\{\sigma_q \mid q \in N_0\}$. Let h be a partial mapping of states in Q to strategies; initially, $h_0(q) = \sigma_q$ for every $q \in N_0$, that is, $h_0(q)$ is a Duplicator winning strategy for $G^f(g_0(q), q)$. And the function π maps the elements of the N-component to protoplays; initially, $\pi_0(q) = ((g_0(q), q), \varepsilon)$. The function π will be updated in such a way that $h_l(q)$ is a Duplicator winning strategy w. r. t. fair simulation for the protoplay $\pi_l(q)$, if $((M_l, i_l), (N_l, j_l))$ is the starting position of the $(l+1)$th round in a play of $G^f((M_0, i_0), (N_0, j_0))$ and $q \in N_l$.

We now suppose that the $(l+1)$th round of a play of $G^f((M_0, i_0), (N_0, j_0))$ starts with the position $((M_l, i_l), (N_l, j_l))$, g_l maps every state q in N_l to a state $g_l(q) \in M_l$ such that $g_l(q) \leq_f q$, and h_l maps every $q \in N_l$ to a strategy $h_l(q) \in \{\sigma_q \mid q \in N_0\}$ such that $h_l(q)$ is a Duplicator winning strategy for $\pi_l(q)$. Assume that Spoiler chooses the position $((M_{l+1}, i_{l+1}), (N_l, j_l), t_l)$.

Then, by the construction in Section 2.4, for every state $q \in M_l \cap E$, there is a state $q' \in M_{l+1}$ such that $(q, t, q') \in \Delta$, $t_l \to t$ and $q' \neq q$ if $0 \neq i_l \neq i_{l+1}$ and $z(q) = i_l$.

For every $q \in N_l$, we now update $\pi_l(q)$ according to $h(q)$. The basic concept is similar to the construction in the proof of Proposition 1.2; we there for only detail the case $(g_l(q), q) \in U \times E$. Different from the proof of Proposition 1.2, we may also have to update the strategies.

Let $q_l \in N_l$ and $(g_l(q_l), q_l) \in U \times E$. Assume that $(q'_{l+1}, q_l, t_l) = h_l(q_l)(g_l(q_l), q_l, t_l)$ and $(q'_{l+1}, q_{l+1}) = h_l(q_l)(q'_{l+1}, q_l, t_l)$. (Note that $q'_{l+1} \in M_{l+1}$.) If $h_{l+1}(q_{l+1})$

is undefined or if $h_{l+1}(q_{l+1})$ is strictly larger w. r. t. the ordering on the strategies than $h_l(q_l)$, then we let $h_{l+1}(q_{l+1}) = h_l(q_l)$ and $g_{l+1}(q_{l+1}) = q'_{l+1}$. We choose $\{q_{l+1}\}$ as the successor set of $\{q_l\}$ such that $q_{l+1} \in N_{l+1}$. The protoplay $\pi_{l+1}(q_{l+1})$ is $\pi_l(q_l)$ with (q'_{l+1}, q_{l+1}) and t_l appended. The condition that the strategy is changed only to a "smaller" strategy ensures that $h_{l+1}(q_{l+1}) \neq h_l(q_l)$ for only finitely many l.

That is, for $n < \omega$ and $q_n \in N_n$, $\pi_n(q_n) = ((g_i(q_i), q_i)_{i \leq n}, (t_i)_{i<n})$ is a protoplay such that every position $(g_i(q_i), q_i)$ satisfies $g_i(q_i) \leq_f q_i$, and $h_n(q_n)$ is a Duplicator winning strategy for $G^f(g_n(q_n), q_n)$. Moreover, it is ensured that there is an $m < \omega$ such that $h_m(q_m) = h_{m+1}(q_{m+1}) = \cdots$, i. e., for all $m' \geq m$, the protoplay $((g_i(q_i), q_i)_{m \leq i \leq m'}, (t_i)_{m-1 \leq i < m'})$ starting from round m is $h_m(q_m)$-conform.

Now assume that there are infinitely many positions $((M_l, i_l), (N_l, j_l))$ such that $i_l = 0$ in a play of $G^f((M_0, i_0), (N_0, j_0))$. Then it is ensured that the sequence

$$(\min\{|\{i \leq n \mid \pi_n(q_n) = ((g_i(q_i), q_i)_{i \leq n}, (t_i)_{i<n} \text{ and } g_i(q_i) \in F\}| : q_n \in N_n\})_{n<\omega} \tag{2.63}$$

is unbounded. In other words, in the limit, accepting states are visited infinitely often in the first component of every protoplay position that we keep book of.

Assume that there were only finitely many positions $((M_m, i_m), (N_m, j_m))$ such that $j_m = 0$. Then there is an m' such that $0 \neq j_{m'} = j_{m'+1} = \cdots$. Let $\bar{q} = z^{-1}(j_{m'})$, i. e., $\bar{q} \notin F$, and $\bar{q}$ is in the successor set chosen for $\bar{q}$ from round m' on. Then the last position of the protoplay $\pi_n(\bar{q})$ is $(g_n(\bar{q}), \bar{q})$ for all $n \geq m'$, but $g_n(\bar{q}) \in F$ for infinitely many $n \geq m'$. But, in contradiction, the protoplays $\pi_n(\bar{q})$ are conform to a Duplicator winning strategy from some point on.

Hence the assumption is wrong, i. e., there are also infinitely many positions $((M_m, i_m), (N_m, j_m))$ such that $j_m = 0$.

That is, Duplicator wins $G^f((M_0, i_0), (N_0, j_0))$ and $(M_0, i_0) \leq_f (N_0, j_0)$. □

2.5.2 Simulation-based NBA-state pruning

Pruning based on fair simulation

From Lemma 2.5 together with Corollary 2.2, we can deduce that from a set M in an NBA state (M, i), a state can be deleted if it is not $\leq_f$-minimal in M and if a reachability criterion is met.

Corollary 2.4 (fair simulation NBA-state pruning) *Let* $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, F)$ *be a very weak alternating Büchi automaton, and let* $\mathbf{Q}_{nd}$ *be the NBA constructed from* $\mathbf{Q}$ *according to Section 2.4 based on the bijection* $z\colon P \to \{1, \ldots, k\}$.

Let $(M, i) \in 2^Q \times (k+1)$ *be a state of* $\mathbf{Q}_{nd}$ *and let* $q, q' \in M$ *be two states such that* $q \leq_f q'$. *Let* $N = M \setminus \{q'\}$. *Let* $i' = \min\{z(q) > i \mid q \in N\}$ *if* $z(q') = i$, *else*

$i' = i$. Assume that (M,i) is not reachable from (N,i'). Let Δ'_{nd} be the result of replacing all transitions $(r,t,(M,i)) \in \Delta_{nd}$ by transitions $(r,t,(N,i'))$. Let $\mathbf{Q}'_{nd} = (Q_{nd},\Sigma,q_I^{nd},\Delta'_{nd},F_{nd})$.

Then $\mathbf{Q}_{nd} \equiv_f \mathbf{Q}'_{nd}$.

Proof. By Lemma 2.5, the states (M,i) and (N,i') are f-equivalent. Since (M,i) is not reachable from (N,i'), $\mathbf{Q}_{nd} \equiv_f \mathbf{Q}'_{nd}$ follows by Corollary 2.2. □

A sufficient local condition ensuring that (M,i) is not reachable from (N,i') is that q' is not reachable in $\mathbf{Q}$ from any state in N. A stronger criterion, specifically for our ε-ABA from LTL, is the following.

Proposition 2.2 (reachability) *For an LTL formula φ, let (M,i) be a state of $\mathbf{Q}^{td}(\varphi)$, and let q' and (N,i') be defined as in Corollary 2.4.*

If it is not *the case that both (1) q' (as an LTL formula) is a proper subformula of a* G*-,* U*- or* R*-formula $\psi \in N$, and (2a) every state/formula in N is a formula of the form $\mathsf{F}\psi$, $\mathsf{G}\psi$, $\psi \,\mathsf{U}\, \rho$ or $\psi \,\mathsf{R}\, \rho$ or (2b) a subformula of such a formula which also is in N, then (M,i) is not reachable from (N,i').*

The reachability criterion of Proposition 2.2 can also be used together with Lemma 2.3 in an on-the-fly fashion. Proposition 2.2 is quite awkward, however. In the following, we show that the reachability requirement of the simulation-based NBA-state pruning of Corollary 2.4 can be lifted for delayed and direct simulation under certain circumstances.

Pruning based on delayed and direct simulation

Since the delayed and direct simulation relations are subsets of the fair simulation relation, we can hope that the reachability criterion necessary in Corollary 2.4 can be weakened or lifted for these simulation relations, perhaps similar to Corollary 1.9 in Section 1.6. But, to see the basic problem, consider the formula $\varphi = \mathsf{G}((\mathsf{X}^2\mathsf{F}a) \wedge (\mathsf{X}^3\mathsf{F}a))$. For the ε-ABA $\mathbf{Q}(\varphi)$, we have $\varphi <_f \mathsf{X}^2\mathsf{F}a <_{di} \mathsf{XF}a <_{di} \mathsf{F}a$, cf. Section 2.2. Applying the NBA construction of Section 2.4, the initial state of the resulting NBA is $(\{\varphi\},0)$. From this state, the only transition is a $\mathtt{tt}$-transition to the state $(\{\varphi,\mathsf{X}^2\mathsf{F}a,\mathsf{XF}a\},0)$. The formula $\mathsf{XF}a$ is not minimal w. r. t. even direct simulation in this state. However, if we delete it, the resulting state is $(\{\varphi,\mathsf{X}^2\mathsf{F}a\},0)$, and from this state, there is a $\mathtt{tt}$-transition to $(\{\varphi,\mathsf{X}^2\mathsf{F}a,\mathsf{XF}a\},0)$ which again is simplified to $(\{\varphi,\mathsf{X}^2\mathsf{F}a\},0)$, that is, the resulting transition is $((\{\varphi,\mathsf{X}^2\mathsf{F}a\},0),\mathtt{tt},(\{\varphi,\mathsf{X}^2\mathsf{F}a\},0))$. That is, the resulting automaton would accept every sequence of sets of propositions, not just $L(\varphi)$, so an unconditional NBA-state pruning is wrong for direct and delayed simulation.

But we can show that pruning can be applied with accepting ABA-states in the set component of an NBA-state. We show this for ABA over alphabets as in Subsection 2.4.1; the application to NBA from LTL is straightforward.

Definition 2.3 (pruning with accepting states) *Let (M,i) be a state of a nondeterministic Büchi automaton $\mathbf{Q}_{nd} = (2^Q \times (k+1), \Sigma, (q_I, z(q_I)), \Delta_{nd}, 2^Q \times \{0\})$ constructed from a very weak alternating Büchi automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ over an alphabet Σ, cf. Subsection 2.4.1.*

For $x \in \{di, de\}$, the x-pruned state *for state (M,i) is the state*

$$(\{q \in M \mid \forall q' \in M\colon (q \neq q' \leq_x q) \rightarrow ((q' \notin F \ \wedge\ q \notin F) \ \vee\ z(q) = i)\}, i) \ . \tag{2.64}$$

That is, we remove an ABA state q from M if there is another state $q' \in M$ such that $q' \leq_x q$ and at least one of q and q' is an accepting state and $z(q) \neq i$. We then say that *q is pruned in favor of q'*.

Definition 2.4 (di/de-pruned NBA) *Let $\mathbf{Q}_{nd}$ and x be as in Definition 2.3.*

The x-pruned version *$\mathbf{Q}_{nd}^{pr-x}$ of $\mathbf{Q}_{nd}$ is inductively defined as follows.*

— The initial state of $\mathbf{Q}_{nd}^{pr-x}$ is the x-pruned state *for $(q_I, z(q_I))$, and*

— if (M,i) is a state of $\mathbf{Q}_{nd}^{pr-x}$ such that $((M,i),t,(N,j))$ is a transition from (M,i) according to Subsection 2.4.1 (that is, N is an unpruned successor set of M) and (N',j) is the x-pruned state *for state (N,j), then $((M,i),t,(N',j))$ is a transition of $\mathbf{Q}_{nd}^{pr-x}$ and (N',j) is a state of $\mathbf{Q}_{nd}^{pr-x}$.*

We claim that the $\{di, de\}$-pruned versions of a nondeterministic automaton $\mathbf{Q}_{nd}$ are equivalent w. r. t. *fair* simulation to $\mathbf{Q}_{nd}$.

Lemma 2.6 *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, F)$ be a very weak alternating Büchi automaton over an alphabet Σ, and let $\mathbf{Q}_{nd} = (2^Q \times (k+1), \Sigma, (q_I, z(q_I)), \Delta_{nd}, 2^Q \times \{0\})$ be the nondeterministic Büchi automaton constructed from $\mathbf{Q}$ according to Subsection 2.4.1. For $x \in \{di, de\}$, let $\mathbf{Q}_{nd}^{pr-x}$ be the* x-pruned version *of $\mathbf{Q}_{nd}$ according to Definition 2.4.*

Then $\mathbf{Q}_{nd} \equiv_f \mathbf{Q}_{nd}^{pr-x}$, and $L(\mathbf{Q}) = L(\mathbf{Q}_{nd}) = L(\mathbf{Q}_{nd}^{pr-x})$.

Proof. We will show the claim for $x = de$ only; from this, the claim for $x = di$ follows immediately.

It is easy to see that $\mathbf{Q}_{nd} \leq_f \mathbf{Q}_{nd}^{pr-de}$: In a play $(((M_j, i_j), (M'_j, i'_j))_{j<\omega}, w)$ of $G^f(\mathbf{Q}_{nd}, \mathbf{Q}_{nd}^{pr-de})$, $M'_j \subseteq M_j$ holds for every $j < \omega$, so if $(M_j, i_j)_{j<\omega}$ is an accepting run of $\mathbf{Q}_{nd}$, then $(M'_j, i'_j)_{j<\omega}$ is an accepting run of $\mathbf{Q}_{nd}^{pr-de}$.

To show $\mathbf{Q}_{nd}^{pr-de} \leq_f \mathbf{Q}_{nd}$, we use a variant of the logbook technique of the proof of Proposition 1.2 in Section 1.6. The situation will be more complicated, however: In the proof of Proposition 1.2, the logbook contains, for every state of the simulating alternating automaton which currently is in the set component of the simulating NBA, an ABA protoplay such that all these protoplays have the same length and are conform to the same Duplicator strategy. Here, we will also associate ABA states with ABA protoplays, but these protoplays need not be conform to the same strategy, nor will they necessarily all have the same length.

For all states $q, q' \in Q$ such that $q \leq_{de} q'$, let $\sigma(q,q')$ be a positional Duplicator winning strategy for $G^{de}(q,q')$.

We now inductively and synchronously construct a play $(((M'_j, i'_j), (M_j, i_j))_{j<\omega}, w)$ of $G^f(\mathbf{Q}_{nd}^{pr-de}, \mathbf{Q}_{nd})$ and a logbook as a partial function L with domain $Q \times \omega$. In every round $j < \omega$ of the play, L maps the elements of M_j to a pair consisting of an element of M'_j and a Duplicator winning strategy. That is, for the sake of simplicity, we do not store a proper protoplay; rather, if $L(q,j) = (q', \sigma)$, then (q', q) is the position in the jth round of a protoplay with Duplicator strategy σ. Our definition is as follows.

The first round of the play starts at $(((M'_0, i'_0), (M_0, i_0)), \varepsilon)$, where $M'_0 = M_0 = \{q_I\}$ and $i'_0 = i_0 = z(q_I)$. We associate with q_I in the first round the strategy $\sigma(q_I, q_I)$ and the state q_I, i. e., $L(q_I, 0) = (q_I, \sigma(q_I, q_I))$.

Now assume that the play and the mapping L is constructed up to some round j and that Spoiler in this play moves from (M'_j, i'_j) to (M'_{j+1}, i'_{j+1}) with letter $w(j) \in \Sigma$. Then, M'_{j+1} and i'_{j+1} result from M'_j and i'_j by a combination of successors of the states in M'_j and a pruning step. That is, there is a set $M''_{j+1} \in 2^Q$ such that

$$M''_{j+1} = \bigcup_{q \in M'_j \cap U} \Delta(q, w(j)) \cup \bigcup_{q \in M'_j \cap E} \{q'\} \ , \tag{2.65}$$

such that (1) for a state $q \in M'_j \cap E$ with $z(q) = i'_j$, $q \neq q'$ if and only if i'_j is different from i'_{j+1}, and such that (2) (M'_{j+1}, i'_{j+1}) is the *de*-pruned state of (M''_{j+1}, i'_{j+1}). Assume that f is a function $M'_j \to 2^{M''_{j+1}}$ that maps every state of M'_j to a successor set such that their combination is M''_{j+1}, i. e., $f(q) = \{q'\}$ if $q \in E$ and $f(q) = \Delta(q, w(j))$ if $q \in U$.

We now choose M_{j+1} according to the assignments in $L(.,j)$. That is, for a state $q \in M_j$, assume that $L(q,j) = (q', \sigma)$. By induction hypothesis, $q' \leq_{de} q$ and σ is a Duplicator winning strategy for the protoplay in position (q', q). It follows that there is a set $M_q \in \mathrm{scs}(q, w(j))$ such that for every $\bar{q} \in M_q$, there is a $\bar{q}' \in f(q')$ such that $\bar{q}' \leq_{de} \bar{q}$ and σ is a Duplicator winning strategy for the

protoplay in position $(\bar{q}', \bar{q})$. We set

$$M_{j+1} = \bigcup_{q \in M_j} M_q \ , \tag{2.66}$$

and we store $L(\bar{q}, j+1) = (\bar{q}', \sigma)$ in the logbook if $\bar{q}'$ also is in M'_{j+1}. If not, that is, if $\bar{q}'$ is pruned in favor of a state $\hat{q}' \leq_{de} \bar{q}'$, we distinguish two cases: (1) If $\bar{q}$ is accepting, we start a new protoplay starting in position $(\hat{q}', \bar{q})$ with Duplicator strategy $\sigma(\hat{q}', \bar{q})$, i. e., we store $L(\bar{q}, j+1) = (\hat{q}', \sigma(\hat{q}', \bar{q}))$. (2) If $\bar{q}$ is not accepting, we store $L(\bar{q}, j+1) = (\hat{q}', \sigma(\hat{q}', \bar{q}') \bowtie \sigma)$ where the intermediate sequence of $\sigma(\hat{q}', \bar{q}') \bowtie \sigma$ starts at $\bar{q}'$. Since at least one of the states $\hat{q}'$ and $\bar{q}'$ is accepting, this forces Duplicator to eventually reach an accepting state from $\bar{q}$.

Consequently, if the sequence of states $(M'_j, i'_j)_{j<\omega}$ is an accepting run, the sequence $(M_j, i_j)_{j<\omega}$ also is an accepting run since the successor sets of the ABA states in the sets M_j are chosen according to strategies such that accepting ABA states are reached infinitely often from all these ABA states. □

Compared to the pruning of MH-automaton states of Corollary 1.9 in Section 1.6, we see that the optimized construction for very weak alternating automata comes at a price. Intuitively, in the MH-construction, *all* relevant non-accepting states are "kept under special surveillance" in the second components of the states; in the top-down construction of Section 2.4, this surveillance is on *only one* non-accepting state at a time, and only on states in which a run really can get stuck without accepting. While this is sufficient for very weak ABA to control acceptance, it is not sufficient to control simulation-based pruning if the states involved in a possible pruning step are both non-accepting.

2.6 Computing the Nondeterministic Büchi Automaton with On-The-Fly Simplifications

We can now combine the considerations of the previous subsections into an algorithm for the construction of an equivalent nondeterministic automaton from an LTL formula with an on-the-fly use of simulation-based simplifications. The basic construction of the NBA follows the definitions of Section 2.4, and the delayed and fair simulation relations are computed according to Sections 2.2 and 2.3. These simulations relations are used for simplifications during the construction, based on the insights of Section 2.5.

Such an algorithm is given in Subsection 2.6.1. In Subsection 2.6.2, we give examples of the working of this algorithm on two LTL formulas. Subsection 2.6.3 reports comparative experiments of a prototypical implementation of an earlier version of this algorithm on random LTL formulas.

2.6.1 An algorithm

The considerations of the previous subsection suggest the following algorithm for translating an LTL formula φ over a set of propositions Σ to an equivalent nondeterministic Büchi automaton $\mathbf{Q}_{nd}^{sim}(\varphi)$.

1. Convert φ to negation normal form.
2. Compute the fair simulation relation relation $\leq_f$ for the automaton $\mathbf{Q}(\varphi)$ of Section 2.2, using the simulation game of Section 2.3.
3. **(fair simulation rewriting)** Apply Corollary 2.3 for formula rewriting to φ. In the following, we assume that φ is the result of these rewritings.
4. Compute the delayed simulation relation $\leq_{de}$ for $\mathbf{Q}(\varphi)$, i. e., for the alternating automaton resulting from the simplified formula after formula rewriting, again using the game of Section 2.3.
5. Choose a bijection $z\colon P_\varphi \to \{1,\ldots,k\}$.
6. The initial state of $\mathbf{Q}_{nd}^{sim}(\varphi)$ is $q_I^\varphi = (\{\varphi\}, \min\{z(\psi) \mid \psi \in \{\varphi\} \cap P_\varphi\})$, where conjunctive formulas are identified with the collection of their conjunctive subformulas, cf. Subsection 2.4.3.

   Apply NBA state pruning according to step (11) of this algorithm to q_I^φ and add the resulting state as q_I^φ to the auxiliary set of new states, i. e., *newStates* $= \{q_I^\varphi\}$.
7. Choose a state (M,i) from *newStates*, remove it from *newStates* and add it to the set Q_{nd}^{sim} of states of $\mathbf{Q}_{nd}^{sim}(\varphi)$.
8. Compute the transitions starting at (M,i) as described in Subsection 2.4.3, based on the bijection z. Let $T_{(M,i)} \subseteq \mathrm{term}_\Sigma \times (2^{Q(\varphi)} \times (k+1))$ be the set of these transitions.
9. **(local optimization)** Remove all transitions from $T_{(M,i)}$ which are superfluous according to the local optimization criterion of Definition 2.2.
10. **(fair transition elimination)** For every two transitions $(t,(N,j))$, $(t',(N',j')) \in T_{(M,i)}$, if $i = 0$ or $j' = 0$, and $t \to t'$, and $(N,j) \leq_f (N',j')$ according to Lemma 2.5, then remove $(t,(N,j))$ from $T_{(M,i)}$; this is the application of Lemma 2.4.

    Also, if $t \to t'$ and $(N,j) \leq_f (N',j')$, and (M,i) is not reachable from (N',j') (Proposition 2.2), then remove $(t,(N,j))$ from $T_{(M,i)}$; this is the application of Lemma 2.3.

11. **(NBA state pruning)** For every remaining transition $(t,(N,j)) \in T_{(M,i)}$, if $(N,j) \notin Q_{nd}^{sim}$, we prune (N,j) w. r. t. fair simulation (and adjust j, if necessary) by applying Corollary 2.4, taking care of the reachability criterion of Proposition 2.2, and w. r. t. delayed simulation according to Definition 2.3. If this step changes a transition, check if any (further) transition can be deleted by local optimization now.

12. For all transitions $(t,(N,j))$ which are now in $T_{(M,i)}$, we add $((M,i),t,(N,j))$ to the set Δ_{nd}^{sim} of transitions of $\mathbf{Q}_{nd}^{sim}(\varphi)$, and we add (N,j) to *newStates* if $(N,j) \notin Q_{nd}^{sim}$.

13. If $newStates \neq \emptyset$, continue with step 7.

14. The set of accepting states of $\mathbf{Q}_{nd}^{sim}(\varphi)$ is $F_{nd}^{sim} = \{(M,i) \in Q_{nd}^{sim} \mid i = 0\}$.

Steps 9 to 11 of this algorithm are optional and independent of each other—it is possible to skip any of these steps. Alternatively, it is also possible to compute the transitions at a given state one after the other and apply (some of) steps 9 to 11 after the computation of every single transition to the set of transitions computed so far. Especially, step 11 can be of use in this approach since it only deals with a single transition, not with comparing transitions.

Also, the algorithm can be combined with other simplification techniques, namely with formula rewriting based on syntactical rewrite rules in step 3 (see, e. g., [EH00]). After the computation of the nondeterministic automaton, further optimizations, both simulation-based and based on criteria like the SCC structure of the NBA, can be applied, see, e. g., [EH00, SB00, ESW01, GO01]. Experiments with the program TMP [Ete] indicate that computing delayed or fair simulation for the NBA can easily become too time consuming. Computing direct simulation and applying minimax quotienting as described in Section 1.4 can be a good alternative.

Remember that, by Lemma 2.5, the fair simulation relation for the NBA is already partly computed. With the complete NBA computed, we can apply Corollary 2.4 to its full extent and also apply Lemma 2.3 to the NBA. The partially computed fair simulation relation may also serve as a starting point for the fair simulation minimizations of [GBS02].

We suggest to apply the following post-processing steps.

1. Remove all unproductive states from $\mathbf{Q}_{nd}^{sim}(\varphi)$, i. e., all states from which an accepting state is not reachable.

2. Compute direct simulation and the minimax quotient w. r. t. direct simulation of $\mathbf{Q}_{nd}^{sim}(\varphi)$, cf. Chapter 1. Continue with this quotient automaton $\mathbf{Q}_{nd}^{di}(\varphi)$.

3. Since $\mathbf{Q}_{nd}^{di}(\varphi)$ is direct simulation equivalent to $\mathbf{Q}_{nd}^{sim}(\varphi)$ (Theorem 1.2) and the direct simulation relation is a subset of the fair simulation relation (Lemma 1.1), we have $[q] \leq_f [q']$ at least for those states of $\mathbf{Q}_{nd}^{di}(\varphi)$ for which there are representatives $\bar{q} \in [q]$, $\bar{q}' \in [q']$ such that $\bar{q} \leq_f \bar{q}'$ holds according to Lemma 2.5, or $\bar{q} \leq_{di} \bar{q}'$ holds, as computed in step 2 of this post-processing.

   With all information about reachability at hand, we apply Lemma 2.3 for deleting transitions in $\mathbf{Q}_{nd}^{di}(\varphi)$ with this partially computed relation $\leq_f$.

We thus use all our three modes of simulation—direct, delayed, and fair—for simulation-based simplification: Computing the direct simulation relation is comparably fast, so we can compute the direct simulation quotient of the nondeterministic automaton in the post-processing. Delayed simulation is useful for the pruning of NBA states, and fair simulation can be used for pruning, for identifying superfluous transitions of the NBA and for formula rewriting. Local optimization, while not based on simulation, is easy to use and worthwhile.

2.6.2 Example: From LTL to NBA

We exemplify the working of the algorithm of Subsection 2.6.1 for two input formulas.

Example: $\mathsf{F}((\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b)))$

Our first example is the input formula $\varphi = \mathsf{F}((\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b)))$ over the set of propositions $\Sigma = \{a,b\}$. Figure 2.1 shows the ε-ABA $\mathbf{Q}(\varphi)$. As in Chapter 1, existential states are shown as diamonds and universal states as boxes; accepting states have double lines. Labels "ε" are omitted on the transitions, as is the state ff.

We first compute the fair simulation relation for $\mathbf{Q}(\varphi)$ and find that $\varphi \equiv_f (\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b)) \equiv_f a\ \mathsf{R}\ (\mathsf{F}b) <_f \mathsf{F}b <_f \mathsf{tt}$, and $\mathsf{F}b$ is incomparable via fair simulation to $a \vee \mathsf{X}(a\ \mathsf{R}\ (\mathsf{F}b))$.

To further illustrate the game rules of Section 2.3, we clarify here why the states $a\ \mathsf{R}\ (\mathsf{F}b)$ and $(\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b))$ are fair simulation equivalent: In a simulation game with the red pebble on $(\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b))$ and the green pebble on $a\ \mathsf{R}\ (\mathsf{F}b)$ at the beginning of the first round, Spoiler chooses some term from $\mathrm{posTerms}((\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b))) = \{\mathsf{tt}, a, b, a \wedge b\}$ and then moves the green pebble to either $a \vee \mathsf{X}(a\ \mathsf{R}\ (\mathsf{F}b))$ or to $\mathsf{F}b$. Then, Duplicator can move the red pebble, and he can now move it via $a\ \mathsf{R}\ (\mathsf{F}b)$ to the same state on which the green pebble now is. In fact, since $a\ \mathsf{R}\ (\mathsf{F}b)$ and $(\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b))$ are both accepting states of $\mathbf{Q}(\varphi)$, this shows that even $(\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b)) \leq_{di} a\ \mathsf{R}\ (\mathsf{F}b)$.

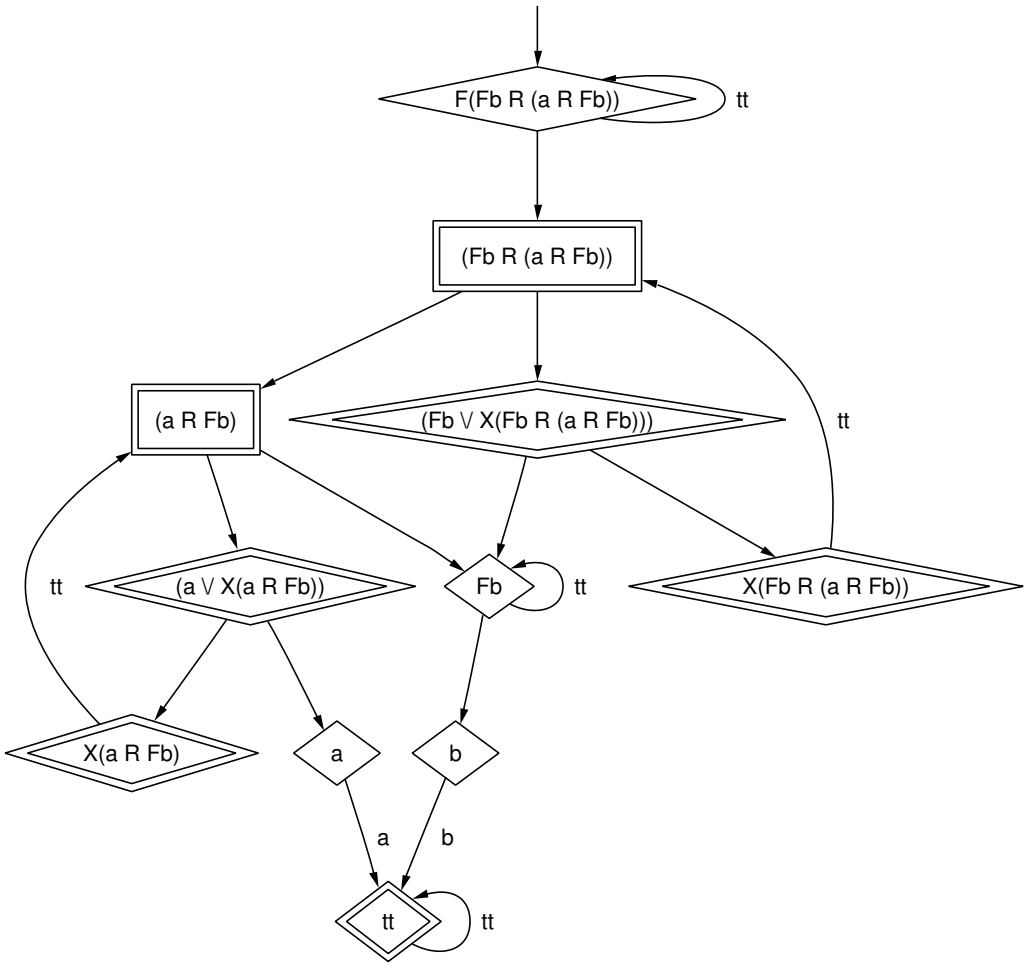

Figure 2.1: ε-ABA for $\mathsf{F}((\mathsf{F}b)\ \mathsf{R}\ (a\ \mathsf{R}\ (\mathsf{F}b)))$

Conversely, if the red pebble starts on $a \mathsf{R} (\mathsf{F}b)$ and the green pebble starts on $(\mathsf{F}b) \mathsf{R} (a \mathsf{R} (\mathsf{F}b))$, Spoiler can also choose a term from $\{\mathtt{tt}, a, b, a \wedge b\}$. He then has to move the green pebble, and he will lose if he moves it to $a \mathsf{R} (\mathsf{F}b)$. If he moves the green pebble to $(\mathsf{F}b) \vee \mathsf{X}((\mathsf{F}b) \mathsf{R} (a \mathsf{R} (\mathsf{F}b)))$, Duplicator can then move the red pebble to $\mathsf{F}b$. Depending on whether he has chosen one of the terms b or $a \wedge b$, Spoiler can now end the moves of the red pebble in this round on state $\mathsf{F}b$ or on state $\mathtt{tt}$, but Duplicator can then move the green pebble to the same state, such that the round ends with both pebbles on the same state. This shows that we even have $(\mathsf{F}b) \mathsf{R} (a \mathsf{R} (\mathsf{F}b)) \equiv_{di} a \mathsf{R} (\mathsf{F}b)$.

The following steps are numbered according to the algorithm of Subsection 2.6.1.

(3) Since φ and $a \mathsf{R} (\mathsf{F}b)$ are equivalent w. r. t. fair simulation, we continue the computation with $\varphi = a \mathsf{R} (\mathsf{F}b)$.

(4) Computing $\leq_{de}$, we see that $\varphi = a \mathsf{R} (\mathsf{F}b) \not\leq_{de} \mathsf{F}b$, basically because $a \mathsf{R} (\mathsf{F}b)$ is an accepting state and $\mathsf{F}b$ is not.

(5) The bijection z is $\mathsf{F}b \mapsto 1$, so ...

(6) ... the initial state of $\mathbf{Q}_{nd}^{sim}(\varphi)$ is $(\{\varphi\}, 0)$, which is added to *newStates* and ...

(7) ... then moved from *newStates* to Q_{nd}^{sim}.

(8) The transitions of $(M,i) = q_I = (\{\varphi\}, 0) = (\{a \mathsf{R} (\mathsf{F}b)\}, 0)$, i. e., the elements of $T_{(M,i)}$, are $(\mathtt{tt}, (\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1))$, $(a, (\{\mathsf{F}b\}, 1))$, $(b, (\{a \mathsf{R} (\mathsf{F}b)\}, 0))$ and $(a \wedge b, (\emptyset, 0))$.

(9) No transitions can be deleted from $T_{(M,i)}$ by local optimization.

(10) By Lemma 2.5, we have $(\{a \mathsf{R} (\mathsf{F}b)\}, 0) \equiv_f (\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1) \leq_f (\{\mathsf{F}b\}, 1) \leq_f (\emptyset, 0)$. Also, $b \to \mathtt{tt}$ and $i = 0$ in $(M,i) = (\{a \mathsf{R} (\mathsf{F}b)\}, 0)$. So by Lemma 2.4, we may remove the transition $(b, (\{a \mathsf{R} (\mathsf{F}b)\}, 0))$.

(11) While $\mathsf{F}b \geq_f a \mathsf{R} (\mathsf{F}b)$ in the set $\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}$ of state $(\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1)$, we must not delete $\mathsf{F}b$ from this set, because the reachability criterion of Proposition 2.2 is not met: $\mathsf{F}b$ is a proper subformula of the remaining formula $a \mathsf{R} (\mathsf{F}b)$, and this formula is an R-formula.

(12) We thus add the transitions $(q_I, t, (N, j))$ to Δ_{nd}^{sim} with $(t, (N, j)) \in \{(\mathtt{tt}, (\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1)), (a, (\{\mathsf{F}b\}, 1)), (a \wedge b, (\emptyset, 0))\}$. The states $(\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1)$, $(\{\mathsf{F}b\}, 1)$ and $(\emptyset, 0)$ are added to *newStates*. Since *newStates* now is not empty, the algorithm continues with step 7.

In the following loops through steps 7 to 13, no new states are added to *newStates*, i. e., the resulting set of states Q_{nd}^{sim} contains the above states in *newStates* and the initial state. The transitions of $(\{\mathsf{F}b\}, 1)$ are $\{(\mathtt{tt}, (\{\mathsf{F}b\}, 1)), (b, (\emptyset, 0))\}$ and the transition of $(\emptyset, 0)$ is $((\emptyset, 0), \mathtt{tt}, (\emptyset, 0))$.

The state $(\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1)$ has transitions to the initial state via b, to itself via $\mathtt{tt}$, to $(\{\mathsf{F}b\}, 1)$ via a and to $(\emptyset, 0)$ via $a \wedge b$. Note that the transition (b, q_I)

must not be deleted in favor of the transition $(\mathtt{tt}, (\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1))$ in step 10, since $(\{a \mathsf{R} (\mathsf{F}b), \mathsf{F}b\}, 1)$ is not an accepting state.

The post-processing does not yield any further simplifications.

Figure 2.2 shows the resulting nondeterministic Büchi automaton for $\mathsf{F}((\mathsf{F}b) \mathsf{R} (a \mathsf{R} (\mathsf{F}b))) \equiv_f a \mathsf{R} (\mathsf{F}b)$.

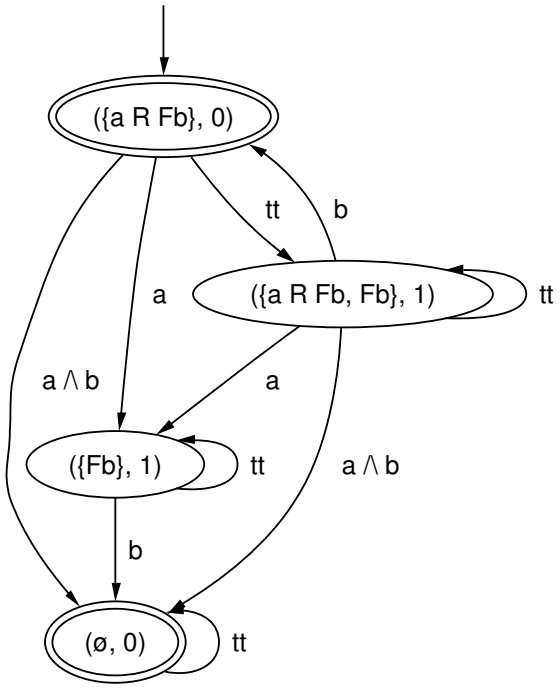

Figure 2.2: NBA for $\mathsf{F}((\mathsf{F}b) \mathsf{R} (a \mathsf{R} (\mathsf{F}b))) \equiv_f a \mathsf{R} (\mathsf{F}b)$

Example: $\mathsf{G}(a \mathsf{R} ((\mathsf{F}b) \mathsf{U} c))$

The next example is the formula $\varphi = \mathsf{G}(a \mathsf{R} ((\mathsf{F}b) \mathsf{U} c))$. Again, the steps are numbered according to the algorithm of Subsection 2.6.1.

(2) We define $\psi = a \mathsf{R} ((\mathsf{F}b) \mathsf{U} c)$, i. e., $\varphi = \mathsf{G}\psi$. We have $\varphi <_f \psi <_f (\mathsf{F}b) \mathsf{U} c$. Further, $b <_f \mathsf{F}b$, and the subformulas a and c are incomparable to the other subformulas. No two subformulas are f-equivalent, and also $(\mathsf{F}b) \wedge \mathsf{X}((\mathsf{F}b) \mathsf{U} c) \not\leq_f c$ and $(\mathsf{F}b) \mathsf{U} c \not\leq_f a \vee \mathsf{X}\psi$.

(3) That is, no formula rewriting is possible.

(4) We have $\varphi <_{de} \psi$ and $b <_{de} \mathsf{F}b$, but $\psi \not\leq_{de} (\mathsf{F}b) \mathsf{U} c$.

(5) We choose the bijection $z\colon \mathsf{F}b \mapsto 1, (\mathsf{F}b) \mathsf{U} c \mapsto 2$.

(6) The initial state of $\mathbf{Q}_{nd}^{sim}(\varphi)$ is $q_I^\varphi = (\{\varphi\}, 0)$.

(8) There are six transitions to six different states starting at $q_I^\varphi = (\{\varphi\}, 0)$: (i) $(\mathtt{tt}, (\{\varphi, \psi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2))$, (ii) $(b, (\{\varphi, \psi, (\mathsf{F}b)\ \mathsf{U}\ c\}, 2))$, (iii) $(c, (\{\varphi, \psi\}, 0))$, (iv) $(a, (\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2))$, (v) $(a \wedge b, (\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c\}, 2))$, (vi) $(a \wedge c, (\{\varphi\}, 0))$.

(9) None of these six transition can be deleted by local optimization.

(10) We have $(\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2) \equiv_f (\{\varphi, \psi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2)$ by Lemma 2.5, and $a \rightarrow \mathtt{tt}$, and q_I^φ is an accepting state. That is, the transition (iv) can be deleted according to Lemma 2.4. Similarly, transition (v) can be deleted in favor of transition (ii) and transition (vi) can be deleted in favor of transition (iii) according to Lemma 2.4.

(11) Pruning w. r. t. fair simulation (Corollary 2.4) is not possible for the target states of the remaining three transitions: While $\varphi \leq_f \psi$, the reachability criterion fails for deleting ψ. But we also have $\varphi \leq_{de} \psi$, and both φ and ψ are accepting states. That is, we can delete ψ from the set component of the three target states by pruning w. r. t. delayed simulation. Our transitions starting at q_I^φ are thus (i) $(\mathtt{tt}, (\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2))$, (ii) $(b, (\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c\}, 2))$, (iii) $(c, (\{\varphi\}, 0))$.

Generally speaking, a formula $\varphi_0\ \mathsf{R}\ \varphi_1$ can always be pruned in favor of the formula $\mathsf{G}(\varphi_0\ \mathsf{R}\ \varphi_1)$. It follows that pruning w. r. t. delayed simulation effectively implements the rewrite rule $\mathsf{G}(\varphi_0\ \mathsf{R}\ \varphi_1) \mapsto \mathsf{G}\varphi_1$, i. e., adding this rule to the rules of Corollary 2.3 does not change the output of the algorithm.

(12) Consequently, the *newStates* are $(\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2)$ and $(\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c\}, 2)$. The state $(\{\varphi\}, 0)$ is the initial state and is already in Q_{nd}^{sim}.

The further computation yields one more state: There is a transition from $(\{\varphi, (\mathsf{F}b)\ \mathsf{U}\ c, \mathsf{F}b\}, 2)$ to $(\{\varphi, \mathsf{F}b\}, 1)$ labeled with c.

The final automaton $\mathbf{Q}_{nd}^{sim}(\varphi)$ is shown in Figure 2.3.

2.6.3 Experiments

The main ideas of the above automata construction from LTL are prototypically implemented as the program LTL→NBA [FTb]. Different from Subsection 2.6.1, in LTL→NBA only the delayed simulation relation is computed for the ε-ABA of an input LTL formula, so that on-the-fly simplifications of the resulting NBA are based on delayed simulation only, and the translation from ε-ABA is via the construction of [MH84] (cf. Section 1.6). The following tables, taken from [Fri03], compare this implementation to the LTL-to-NBA translators TMP [Ete] of Etessami and LTL2BA [Odd] of Gastin and Oddoux on random formulas.

Test 1. 1000 formulas and their negations, of length 8 to 10, with at most 3 different propositions and an equal frequency of the operators $\vee$, $\wedge$, F, G, U, R.

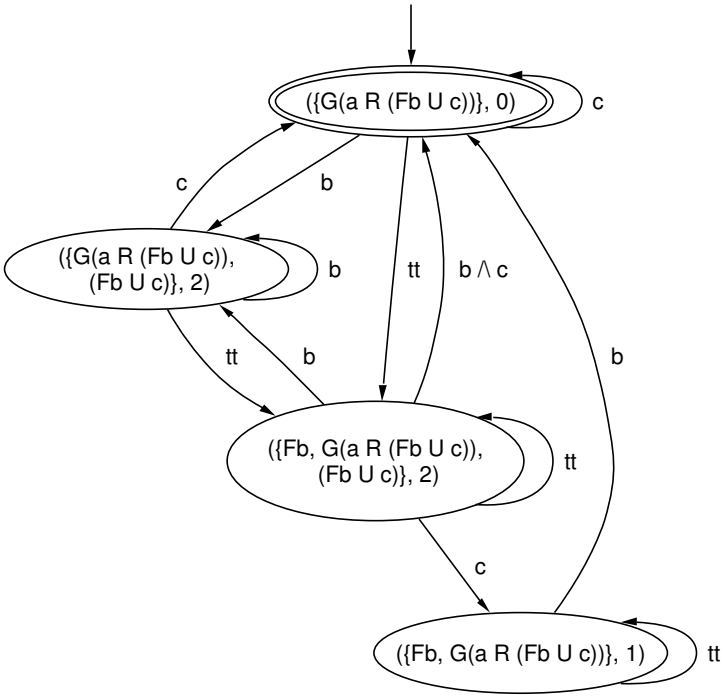

Figure 2.3: NBA for $\mathsf{G}(a\,\mathsf{R}\,((\mathsf{F}b)\,\mathsf{U}\,c))$

Test 1	LTL→NBA	TMP	LTL2BA
Avg. no. of states	3.54	3.59	3.76
Avg. no. of transitions	6.96	6.65	7.75
Total time (sec)	209.1	70.2	12.7

Test 2. 1000 formulas and their negations, of length 10 to 14.

Test 2	LTL→NBA	TMP	LTL2BA
Avg. no. of states	4.37	4.71	4.93
Avg. no. of transitions	10.07	9.71	12.28
Total time (sec)	425.0	197.1	12.9

Test 3. 1000 formulas and their negations, of length 11 to 15 and with an increased frequency of the operators U and R.

Test 3	LTL→NBA	TMP	LTL2BA
Avg. no. of states	5.06	5.71	5.74
Avg. no. of transitions	12.59	13.83	15.98
Total time (sec)	575.0	14737.1[2]	13.1

Test 4. 1000 formulas and their negations, of length 15, with the same frequency of the operators U and R as in Test 3.

Test 4	LTL→NBA	TMP	LTL2BA
Avg. no. of states	5.80	6.55	6.68
Avg. no. of transitions	16.05	16.86	20.63
Total time (sec)	986.6	3001.9	13.4

That is, with the complexity of the formulas increasing, our implementation LTL→NBA becomes faster than TMP, while the resulting automata are even somewhat smaller on the average. LTL2BA is extremely fast, but the resulting automata are considerably larger.

To interpret these results, first note that LTL→NBA is implemented in Python [Pyt], which is a rather slow interpreted language; we suppose that a C implementation would be at least 10 times faster. The program TMP is written in SML [MTHM97] while LTL2BA is implemented in C [KR88].

Basically, TMP works as follows. TMP computes a nondeterministic generalized Büchi automaton from the input LTL formula using the algorithm of [DGV99]. As detailed in Section 2.7, this automaton is then adapted to the format of a nondeterministic Büchi automaton with transition labels as terms, as in our setting, and then the delayed simulation quotient for this NBA is computed, cf. [ESW01]. Since the size of these NBA can be exponential in the length of the input formula, it is not surprising that the average computation time of TMP increases rapidly for longer formulas and with more (nested) U-operators.

It is surprising, however, that the automata output by TMP are not, on average, smaller than the output of LTL→NBA, since LTL→NBA does not compute a simulation quotient for the NBA and hence does not detect all equivalences with respect to delayed simulation that exist between the states of the output NBA. The discussion of the [DGV99] algorithm in Section 2.7 shows that this is not due to using this particular algorithm for the translation of LTL to automata. We conjecture that the main reason for this is a disadvantageous translation of the generalized Büchi condition into a simple Büchi condition, based on the following observation: For the equivalent formulas $(\mathsf{GFX}a) \vee (c\ \mathsf{U}\ (c \vee b))$, $(c\ \mathsf{U}\ (c \vee b)) \vee \mathsf{GFX}a$ and $(\mathsf{GFX}a) \vee c \vee b$, TMP produced three NBA which are not pairwise isomor-

[2]TMP spent 13997.3 sec (nearly 4 hrs.) on the formula $\neg(((\mathsf{X}b)\ \mathsf{U}\ (((\mathsf{G}a)\ \mathsf{R}\ c)\ \mathsf{U}\ (\mathsf{G}\neg a)))\ \mathsf{R}\ a)$, resulting in an automaton of 115 states (LTL→NBA: 7.5 sec, 50 states; LTL2BA: 0.04 sec, 83 states). So without this particular formula, the total time drops to 739.8 sec.

phic, while our algorithm does produce isomorphic automata for all three formulas, both with simplifications based on delayed simulation and with fair simulation simplifications. (For the latter, note that the subformula $c \mathsf{U} (c \vee b)$ is replaced by $c \vee b$ according to item 4 of Corollary 2.3.)

Our automaton for these formulas, shown in Figure 2.4, is isomorphic to the TMP automaton for $(\mathsf{GFX}a) \vee c \vee b$, which only contains one U- or F-operator.

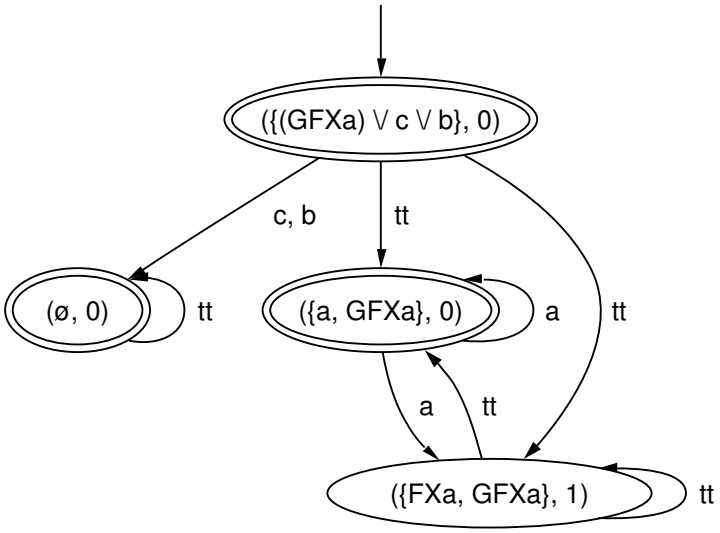

Figure 2.4: $Q_{nd}^{sim}((\mathsf{GFX}a) \vee c \vee b)$, isomorphic to the TMP automaton for this formula.

Compare this automaton to the TMP automaton for $(\mathsf{GFX}a) \vee (c \mathsf{U} (c \vee b))$ shown in Figure 2.5.

It is quite obvious that this automaton is in fact equivalent to the automaton for $(\mathsf{GFX}a) \vee c \vee b$; only the state T2 is superfluous in the automaton for $(\mathsf{GFX}a) \vee (c \mathsf{U} (c \vee b))$. The TMP automaton for $(c \mathsf{U} (c \vee b)) \vee \mathsf{GFX}a$ also has a strongly connected component with three states for the acceptance of the subformula $\mathsf{GFX}a$. Our explanation for this is that, in the TMP implementation, the equivalent of the counter component, which is necessary when a generalized Büchi condition is translated to a simple Büchi condition, is decreased by at most 1 along every transition. That is, we assume that the TMP automaton for $(\mathsf{GFX}a) \vee (c \mathsf{U} (c \vee b))$ can be considered to be based on the bijection $\mathsf{FX}a \mapsto 2$, $c \mathsf{U} (c \vee b) \mapsto 1$ such that the state T1 corresponds to $(\{\mathsf{FX}a, \mathsf{GFX}a\}, 0)$, T4 corresponds to $(\{\mathsf{FX}a, \mathsf{GFX}a\}, 2)$ and T2 corresponds to $(\{a, \mathsf{GFX}a\}, 1)$, i. e., T2 corresponds to an inconsistent state in the sense of Subsection 2.4.1.

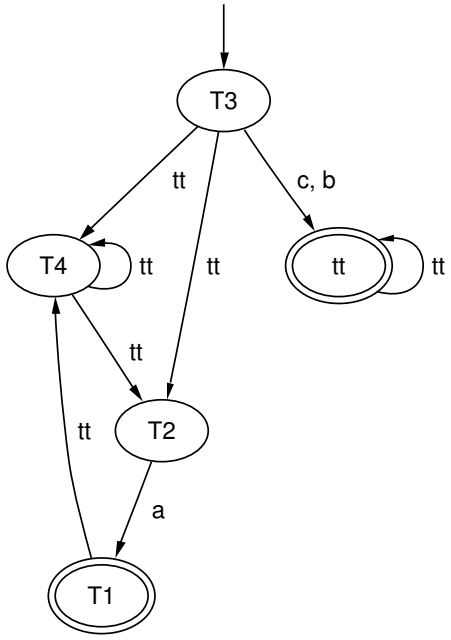

Figure 2.5: TMP automaton for $(\mathsf{GFX}a) \vee (c \,\mathsf{U}\, (c \vee b))$.

In contrast, the experimental results for LTL2BA are as expected: LTL2BA is extremely fast, but the resulting automata are considerably larger than the automata resulting from the other programs. It can be assumed that this is because LTL2BA does not use simulation-based simplifications. The program LTL2BA relies on a small set of simplification rules; these are mainly comparing transitions to identify removable transitions and merging states with the same transitions and the same acceptance status. These rules are applied in an on-the-fly fashion to the very weak alternating Büchi automaton computed for the input LTL formula, then to a generalized Büchi automaton computed from this ABA and, finally, to an NBA resulting from the intermediate generalized automaton. See [GO01] for a detailed description of LTL2BA. While these rules can be applied very fast, they miss "deeper" connections between automata states which are revealed by a simulation-based analysis.

In summary, computing a simulation relation at the intermediate level of alternating automata seems to be a viable compromise between time-consuming approaches like computing a simulation quotient of the resulting NBA (as in TMP) and fast simplifications based on local criteria only (as in LTL2BA).

2.7 A Comparison of LTL-to-NBA Constructions

The classical approach to ABA-to-NBA translation is tableau-based, as in [VW86] and refined in [GPVW95, DGV99]. It is easy to see that the worst-case size of the resulting NBA is the same for our approach and for these refined versions of the tableau-based translation. The exact relation between the automata resulting from these two approaches is not that obvious, however. This is especially because the algorithms of [GPVW95, DGV99] translate LTL to generalized Büchi automata with term labels on the transitions rather than on the states. That is, a comparison requires a common automaton format and a standard translation to that format in the first place.

In this section, we first present the automaton resulting from the [GPVW95] algorithm, henceforth called the *GPVW-automaton*, in Subsection 2.7.1. We then give a straightforward adaptation of the GPVW-automaton to the format of our top-down automaton in Subsection 2.7.2; this adaptation is similar to what is done in [GL02] and implemented in TMP [Ete]. In Subsection 2.7.3, we show that there is a strong connection between the local optimization of Definition 2.2 and the concept of syntactical implication introduced in [DGV99] and used in an optimized translation from LTL to a generalized Büchi automaton; this automaton constructed according to [DGV99] will henceforth be called the *DGV-automaton*. Using this careful analysis of local optimization, we show in Subsection 2.7.4 that the GPVW-automaton is the same as our automaton via ABA for all input formulas in next normal form, provided that both automata are also locally optimized. Since the use of syntactical implication is the main difference between the GPVW-automaton and the DGV-automaton, we can also show that the same is true for the DGV-automaton (Subsection 2.7.5). In Subsection 2.7.6, we discuss why next normal form is crucial for these results.

2.7.1 The GPVW-automaton

The GPVW-automaton differs from our top-down automaton in some basic aspects. Let

$$\mathcal{A}(\varphi) = (Q_\varphi, \Sigma, I, \rightarrow, \mathcal{F}, \mathcal{L}) \tag{2.67}$$

be the GPVW-automaton for an LTL formula φ. Then $\mathcal{A}(\varphi)$ is a generalized Büchi automaton, i. e., the acceptance condition $\mathcal{F}$ is given as

$$\mathcal{F} = \{F_0, \ldots, F_{n-1}\} \subseteq 2^{Q_\varphi} \tag{2.68}$$

such that a run $\pi = (q_i)_{i<\omega} \in Q_\varphi^\omega$ of $\mathcal{A}(\varphi)$ is accepting if and only if $\mathrm{Inf}(\pi) \cap F_i \neq \emptyset$ for all $i < n$. Moreover, $I \subseteq Q_\varphi$ is a set of initial states rather than a single initial

state, and the terms which are used as labels of the transitions in our construction are labels of the states in the GPVW-automaton while there are no labels on the transitions. That is, $\mathcal{L}$ is a labeling function $Q_\varphi \to \mathrm{term}_\Sigma$, and a sequence $(q_i)_{i<\omega} \in Q_\varphi^\omega$ is a run of the GPVW-automaton on a word $w \in (2^\Sigma)^\omega$ if and only if $q_0 \in I$, $(q_i, q_{i+1}) \in \to$ and $w(i) \models \mathcal{L}(q_i)$, for all $i < \omega$.

We therefore apply some simple modifications to the GPVW-automaton: (1) We introduce a new state as the single initial state, (2) we transfer the labels of a state to its incoming transitions, and (3) we change the generalized Büchi acceptance condition to a simple Büchi condition by introducing a counter. Then we show that this modified GPVW-automaton is equal to our automaton w. r. t. a local simplification criterion for input formulas in next normal form.

In this section, we give the details of the GPVW-automaton. In Subsection 2.7.2, we discuss the effects of the above modifications.

In the algorithmic definition of the GPVW-automaton in [GPVW95], a state of the automaton is described as an object with the fields *Name*, *Incoming*, *New*, *Old*, *Next* and *Father*. Of these, only *Old* and *Next* are necessary to describe a node while the other fields contain auxiliary data. Both *Old* and *Next* are sets of subformulas of the input formula. Consequently, we will describe a state of the GPVW-automaton as an element of $2^{\mathrm{sub}(\varphi)} \times 2^{\mathrm{sub}(\varphi)}$, with the first component representing the *Old*-field and the second component representing the *Next*-field.

To define the transition relation of the GPVW-automaton, we first define the relation $\rightsquigarrow \subseteq 2^{\mathrm{sub}(\varphi)} \times (2^{\mathrm{sub}(\varphi)} \times 2^{\mathrm{sub}(\varphi)})$: We have $N \rightsquigarrow (M', N')$ if there is a $t \in \mathrm{term}_\Sigma$ such that $\mathrm{scs}(N, t, N')$ and $M = \mathrm{lit}(t) \cup N$. As in Subsection 2.4.3, we identify a subset $\{\psi \wedge \rho\}$ with $\{\psi, \rho\}$.

The GPVW-automaton $\mathcal{A}(\varphi)$ is now defined by

$$\to \;=\; \{((M,N),(M',N')) \in (2^{\mathrm{sub}(\varphi)})^4 \mid N \rightsquigarrow (M',N')\} \;, \quad (2.69)$$

$$Q_\varphi \;=\; \{(M,N) \in 2^{\mathrm{sub}(\varphi)} \times 2^{\mathrm{sub}(\varphi)} \mid (\emptyset, \{\varphi\}) \to^+ (M,N)\} \;, \quad (2.70)$$

$$I \;=\; \{(M,N) \in Q_\varphi \mid \{\varphi\} \rightsquigarrow (M,N)\} \;, \quad (2.71)$$

$$\mathcal{L}\colon (M,N) \;\mapsto\; \bigwedge_{\alpha \in M \text{ a literal}} \alpha \;. \quad (2.72)$$

To define the set $\mathcal{F}$ of accepting sets, let z be a bijection of the set P_φ of F- and U-formulas in $\mathrm{sub}(\varphi)$ to $\{1, \ldots, k_\varphi\}$. We have

$$\mathcal{F} = \{F_1, \ldots, F_{k_\varphi}\} \;, \quad (2.73)$$

where

$$F_i = \{(M,N) \in Q_\varphi \mid (\psi \,\mathsf{U}\, \rho \in M \wedge z(\psi \,\mathsf{U}\, \rho) = i) \implies \rho \in M\} \;; \quad (2.74)$$

here, a formula $\mathsf{F}\rho$ is regarded as $\mathsf{tt}\,\mathsf{U}\,\rho$.

We note:

Lemma 2.7 *For all $(M,N),(M',N') \in Q_\varphi$, if $(M,N) \to (M',N')$, then $N' \in \mathrm{scs}(N, \mathcal{L}(M',N'))$ in the sense of Subsection 2.4.3, that is, if there is an $i \leq k_\varphi$ such that (N,i) is a state of $\mathbf{Q}^{td}(\varphi)$, then there is a $j \leq k_\varphi$ such that $((N,i), \mathcal{L}(M',N'),(N',j))$ is a transition in $\mathbf{Q}^{td}(\varphi)$.*

Conversely, if $((N,i),t,(N',j))$ is a transition in $\mathbf{Q}^{td}(\varphi)$ and there is an $M \in 2^{\mathrm{sub}(\varphi)}$ such that $(M,N) \in Q_\varphi$, then there is an $M' \in 2^{\mathrm{sub}(\varphi)}$ such that $(M,N) \to (M',N')$ and $\mathcal{L}(M',N') \equiv t$.

Proof. Straightforward induction over the the size of N and the structure of LTL formulas. □

2.7.2 The adjusted GPVW-automaton

In the first step of the modification, we introduce a single initial state and put the term labels on the transitions. That is, we define

$$\mathcal{A}^{(1)}(\varphi) = (Q_\varphi^{(1)}, \Sigma, q_I, \Delta, \mathcal{F}) \ , \tag{2.75}$$

where $Q_\varphi^{(1)} = Q_\varphi \cup \{q_I\}$ and $q_I = (\emptyset, \{\varphi\})$, if φ is not a F- or U-formula, else $q_I = (\{\varphi\},\{\varphi\})$. If φ is a conjunction $\bigwedge_{i<n} \psi_i$ of subformulas of which, say, $\psi_0,\ldots,\psi_{r-1}$ are F- or U-formulas, then $q_I = (\{\psi_i \mid i<r\},\{\psi_i \mid i<n\})$.

The transitions in $\to$ starting from q_I and the assignment of q_I to accepting sets follow from the above definitions. As the new set of transitions, we have

$$\Delta = \{(q,t,q') \mid q \to q' \text{ and } t = \mathcal{L}(q')\} \ . \tag{2.76}$$

With the terms on the transitions rather than on the states, the only relevant information encoded in the first component of a state (the *Old*-field) now is the state's assignment to the acceptance sets. That is, we can now change the first component to a set of integers in the range $1 \ldots k_\varphi$ such that, for all $1 \leq i \leq k_\varphi$ and for all changed states (M,N), $i \in M$ if and only if $(M,N) \in F_i$. We therefore define the auxiliary function

$$f\colon 2^{\mathrm{sub}(\varphi)} \ \to \ 2^{k_\varphi+1} \ , \tag{2.77}$$

$$M \ \mapsto \ \{0 < i \leq k_\varphi \mid \forall \psi \,\mathsf{U}\, \rho \in M\colon z(\psi \,\mathsf{U}\, \rho) = i \to \rho \in M\}, \tag{2.78}$$

where again a formula $\mathsf{F}\rho$ is regarded as $\mathsf{tt}\,\mathsf{U}\,\rho$.

Our new automaton is

$$\mathcal{A}^{(2)}(\varphi) = (Q_\varphi^{(2)}, \Sigma, q_I^{(2)}, \Delta^{(2)}, \mathcal{F}^{(2)}) \ , \tag{2.79}$$

where

$$Q_\varphi^{(2)} = \{(f(M),N) \mid (M,N) \in Q_\varphi^{(1)}\} \ , \tag{2.80}$$
$$q_I^{(2)} = (f(\mathrm{pr}_1(q_I)),\mathrm{pr}_2(q_I)) \ , \tag{2.81}$$

and, consequently,

$$\Delta^{(2)} = \{(f(M),N),t,(f(M'),N') \mid ((M,N),t,(M',N')) \in \Delta \ , \tag{2.82}$$
$$F_i^{(2)} = \{(M,N) \in Q_\varphi^{(2)} \mid i \in M\}, \text{ for all } 1 \leq i \leq k_\varphi. \tag{2.83}$$

The third step is to change the generalized Büchi condition into a simple Büchi condition. The standard approach is to introduce a counter in the range $0 \ldots k_\varphi = |\mathcal{F}^{(2)}|$; this counter is decreased from i to $i-1$ if the current state in the run belongs to $F_i^{(2)}$. A more sophisticated approach is to decrease, in this case, the counter from i to the largest $j < i$ such that the current state does not belong to $F_j^{(2)}$, and to change the counter to 0 if there is no such j. In both cases, a run is accepting if the counter becomes 0 infinitely often.

This sort of counter behavior (with increasing counters, though) and the translation of term labels to the incoming transitions is also discussed in [GL02].

Another approach is not to use an integer counter but a bit string of length k_φ. Bit no. i is switched from 0 to 1 if the current state belongs to $F_i^{(2)}$, and a run is accepting if the bit string becomes 1^{k_φ} infinitely often. The obvious drawback of this approach is a possible blow-up of the state space by a factor of 2^{k_φ}, while the other approaches have a blow-up factor of $k_\varphi + 1$ in the worst case.

It is easy to see that the second approach corresponds to our construction in Section 2.4 while the third approach corresponds to the result of a Miyano–Hayashi construction applied to the alternating automaton of an LTL formula. As discussed in Subsection 2.6.3, the first approach seems to be used in the LTL-to-NBA implementation TMP [Ete].

To compare the GPVW-automaton to our construction of Section 2.4, we will use the second approach here to turn the generalized Büchi condition of $\mathcal{A}^{(2)}(\varphi)$ into a simple Büchi condition.

We will first define, as an intermediate format, the automaton $\mathcal{A}^{(3)}(\varphi)$ where

$$Q_\varphi^{(3)} = Q_\varphi^{(2)} \times (k_\varphi + 1) \ , \tag{2.84}$$
$$q_I^{(3)} = (q_I^{(2)}, \max\{0 < l \leq k_\varphi \mid l \notin \mathrm{pr}_1(q_I^{(2)})\}) \ , \tag{2.85}$$

and the transition relation is

$$\begin{aligned}\Delta^{(3)} \;=\; & \{((M,N,i),t,(M',N',j)) \mid ((M,N),t,(M',N')) \in \Delta^{(2)},\\ & \text{if } i=0 \text{ then } j=\max\{l \le k_\varphi \mid l \notin M'\}\\ & \text{else if } i \in M' \text{ then } j=\max\{l < i \mid l \notin M'\}\\ & \text{else } j=i\}\ . \end{aligned} \tag{2.86}$$

The Büchi acceptance set is

$$F^{(3)} = \{(q,0) \mid q \in Q^{(2)}\}\ . \tag{2.87}$$

In the next step, we merge any two states if they contain the same counter and the same set of subformulas. That is, the adjusted automaton $\mathcal{A}^{ad}(\varphi)$ has the state set

$$Q^{ad}_\varphi = \{(N,i) \mid \exists M \in 2^{k_\varphi+1} \colon (M,N,i) \in Q^{(3)}_\varphi\}\ , \tag{2.88}$$

and the transitions in Δ^{ad} and the accepting states in F^{ad} are adjusted accordingly.

We note:

Property 2.1 (consistency) *For every state (N,i) of $\mathcal{A}^{ad}(\varphi)$, if $i > 0$, then $z^{-1}(i) \in N$, i. e., the states of $\mathcal{A}^{ad}(\varphi)$ are consistent.*

Proof. Let (N,i) be a state of $\mathcal{A}^{ad}(\varphi)$ such that $i > 0$. By (2.88), there is an $M \in 2^{k_\varphi+1}$ such that (M,N,i) is a reachable state of $\mathcal{A}^{(3)}(\varphi)$. By (2.86), $i \notin M$, so there is $\psi \mathsf{U} \rho \in M$ such that $z(\psi \mathsf{U} \rho) = i$ and $\rho \notin M$ by (2.78) and (2.80). By the definition of the relation $\rightsquigarrow$ and the basic GPVW-automaton, it follows that $\psi \mathsf{U} \rho \in N$. □

2.7.3 Local optimization and syntactical implication

At this stage, $\mathcal{A}^{ad}(\varphi)$ and $\mathbf{Q}^{td}(\varphi)$ are not the same automaton. Consider the automata of Figure 2.6 for the input formula $\varphi = (a \mathsf{U} b) \wedge \mathsf{G}b$.

In the above example, one can observe that, in both automata, the $(a \wedge b)$-labeled transitions are superfluous in the sense of Definition 2.2, because there is a better transition $((\{\varphi\},1),b,(\{\mathsf{G}b\},0))$.

In the following, let $\mathcal{A}^{lo}(\varphi)$ be the locally optimized version of $\mathcal{A}^{ad}(\varphi)$. In the example, we then have $\mathcal{A}^{lo}(\varphi) = \mathbf{Q}^{lo}(\varphi)$, because the $(a \wedge b)$-labeled transitions and, consequently, the state q^{ad}_1 of $\mathcal{A}^{ad}(\varphi)$ are deleted. (Note that the resulting automata can be further optimized by merging the two remaining states—they are equivalent w. r. t. delayed simulation equivalence.)

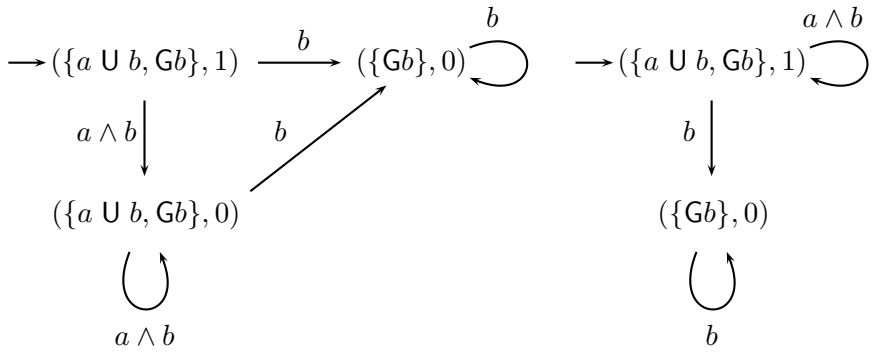

Figure 2.6: $\mathcal{A}^{ad}(\varphi)$ (left) and $\mathbf{Q}^{td}(\varphi)$ (right) for $\varphi = (a \mathsf{U} b) \wedge \mathsf{G}b$

We claim that our observation for the example is not a coincidence: The modified and locally optimized GPVW-automaton is the same as the locally optimized top-down automaton for all LTL formulas in next normal form, provided that the same bijection z is used for the set of U- and F-subformulas.

In the following, we write, e. g., $\mathbf{Q}_z^{td}(\varphi)$ for the top-down automaton of φ based on a fixed bijection z.

We first turn our attention to the redundancy checks via syntactical implication of [DGV99]. There is a close connection between local optimization and these redundancy checks, and the careful analysis of this connection will also lead us to an important observation about locally optimized automata. We introduce the notion of syntactical implication following [GL02].

Definition 2.5 (syntactical implication, cf. [DGV99, GL02]) *For sets A, B of LTL formulas over Σ, $\mathtt{SI}(A,B)$ is the set of LTL formulas over Σ defined inductively as follows.*

1. $\mathsf{tt} \in \mathtt{SI}(A,B)$,
2. $\varphi \in \mathtt{SI}(A,B)$, *if* $\varphi \in A$,
3. $\varphi \in \mathtt{SI}(A,B)$, *if one of the following holds:*
   - $\varphi = \mathsf{X}\psi$ *and* $\psi \in B$, [3] *or*
   - $\varphi = \psi \vee \rho$ *and* ($\psi \in \mathtt{SI}(A,B)$ *or* $\rho \in \mathtt{SI}(A,B)$), *or*
   - $\varphi = \psi \wedge \rho$ *and* $\{\psi, \rho\} \subseteq \mathtt{SI}(A,B)$, *or*
   - $\varphi = \psi \mathsf{U} \rho$ *and* ($\psi \in \mathtt{SI}(A,B)$ *and* $\varphi \in B$, *or* $\rho \in \mathtt{SI}(A,B)$), *or*

[3] We add this rule for technical convenience. It is not included in the definition of [GL02].

- $\varphi = \psi \mathsf{R} \rho$ *and* ($\rho \in \mathtt{SI}(A,B)$ *and* $\varphi \in B$, *or* $\{\psi,\rho\} \subseteq \mathtt{SI}(A,B)$).

If $\varphi \in \mathtt{SI}(A,B)$, *we say that* φ *is* syntactically implied *by A and B, or that* φ *is* syntactically redundant *w. r. t. A and B.*

Obviously, if $A' \subseteq A$ and $B' \subseteq B$, then $\mathtt{SI}(A',B') \subseteq \mathtt{SI}(A,B)$.

Syntactical implication is used in [GL02] in the following way[4]: When computing a transition starting from a set of formulas M, the formulas in M are processed one after the other such that a processed formula is removed from M. (If a subformula φ' of a formula $\varphi \in M$ has to be processed in order to process φ, then φ' is added to M.)

Now before any formula φ taken from M is processed, it is checked whether φ is syntactically redundant w. r. t. the formulas already known to be elements of the target state of the currently computed transition and w. r. t. the literals which are known to be part of the term label of this transition. If this is the case, φ needs not be further processed.

This on-the-fly procedure speeds up the automaton construction: Some formulas are not processed, which can result in less automaton states, and these states also may contain less formulas. However, the performance of this redundancy check obviously depends on the order in which the formulas are taken from M.

To show the connection between syntactical implication and local optimization, we first need a connection between syntactical implication and successor sets.

Lemma 2.8 *Let A be a set of literals, let B be a set of LTL formulas, and let* $\varphi \not\equiv \mathsf{ff}$ *be an LTL formula.*

(1) If $\varphi \in \mathtt{SI}(A,B)$, *then there are sets* $A' \subseteq A$, $B' \subseteq B$ *such that* $B' \in \mathrm{scs}(\{\varphi\}, \mathrm{term}(A'))$.

(2) Conversely, if $B \in \mathrm{scs}(\{\varphi\},\mathrm{term}(A))$, *then* $\varphi \in \mathtt{SI}(A,B)$.

Proof. By induction over the structure of LTL formulas.

(1) *Basic cases:* If $\varphi = \mathsf{tt}$, then $A' = B' = \emptyset$ satisfies the claim. If φ is a literal, then $\varphi \in \mathtt{SI}(A,B)$ because $\varphi \in A$, so the claim holds with $A' = \{\varphi\}$, $B = \emptyset$.

Composed formulas: If $\varphi = \mathsf{X}\psi$, we have $\psi \in B$, so with $A' = \emptyset$ and $B' = \{\psi\}$, we have $B' \in \mathrm{scs}(\{\varphi\},\mathrm{term}(A'))$.

If $\varphi = \psi \mathsf{U} \rho$, we may have $\varphi \in \mathtt{SI}(A,B)$ because $\psi \in \mathtt{SI}(A,B)$ and $\varphi \in B$. By induction hypothesis, there are sets $A_\psi \subseteq A$, $B_\psi \subseteq B$ such that $B_\psi \in \mathrm{scs}(\{\psi\},\mathrm{term}(A_\psi))$. The claim follows with $A' = A_\psi$ and $B' = B_\psi \cup \{\varphi\}$. If,

[4]The use is somewhat different in the details in [DGV99] because there are no transition labels as such in that setting.

on the other hand, $\rho \in \mathtt{SI}(A,B)$, there are $A_\rho \subseteq A$, $B_\rho \subseteq B$ such that $B_\rho \in \mathrm{scs}(\{\rho\}, \mathrm{term}(A_\rho))$, and the claim follows with $A' = A_\rho$, $B' = B_\rho$.

Analogously, if $\varphi = \psi \mathsf{R} \rho$, we either set $A' = A_\rho$ and $B' = B_\rho \cup \{\varphi\}$, or $A' = A_\psi \cup A_\rho$ and $B' = B_\psi \cup B_\rho$.

For $\varphi = \psi \vee \rho$ and $\varphi = \psi \wedge \rho$, the claim follows immediately by the induction hypothesis.

(2) The basic cases are obvious. For $\varphi = \psi \mathsf{U} \rho$, first assume that B also is a successor set of $\{\rho\}$ for $\mathrm{term}(A)$. Then, by induction hypothesis, $\rho \in \mathtt{SI}(A,B)$, hence also $\varphi \in \mathtt{SI}(A,B)$. If, on the other hand, $B = B' \cup \{\varphi\}$ such that $B' \in \mathrm{scs}(\{\psi\}, \mathrm{term}(A))$, then $\psi \in \mathtt{SI}(A,B') \subseteq \mathtt{SI}(A,B)$ and, since also $\varphi \in B$, we also have $\varphi \in \mathtt{SI}(A,B)$. The proofs are also straightforward if φ is a conjunction, a disjunction or an X- or R-formula. □

For a deeper analysis of syntactical implication versus local optimization, we need a terminology that allows us to precisely track the generation of successor sets from formulas and subformulas. We therefore introduce in the following Definitions 2.6 to 2.8 a notion of an LTL formula as a tree and a labeling of these trees by successor sets for the subformulas.

Definition 2.6 (syntax tree) *The* syntax tree $\mathrm{syn}(\varphi)$ *of an LTL formula* φ *is the labeled directed graph* $(V_\varphi, E_\varphi, L_\varphi)$ *representing the syntactical structure of* φ*. That is, if* φ *is a literal or* $\varphi \in \{\mathtt{tt}, \mathtt{ff}\}$ *then* $\mathrm{syn}(\varphi) = (\{v\}, \emptyset, \{(v,\varphi)\})$.

For a composed formula φ*, assume that the syntax trees* $\mathrm{syn}(\psi) = (V_\psi, E_\psi, L_\psi)$ *and* $\mathrm{syn}(\rho) = (V_\rho, E_\rho, L_\rho)$ *for the subformulas* ψ *and* ρ *are already defined such that* V_ψ *and* V_ρ *are disjoint and* $v_0 \in V_\psi$*,* $v_1 \in V_\rho$ *such that* $L_\psi(v_0) = \psi$*,* $L_\rho(v_1) = \rho$*. Let* $v \notin V_\psi \cup V_\rho$ *be a new node.*

For $\varphi \in \{\psi \vee \rho, \psi \wedge \rho, \psi \mathsf{U} \rho, \psi \mathsf{R} \rho\}$*, we define* $\mathrm{syn}(\varphi) = (V_\psi \cup V_\rho \cup \{v\}, E_\psi \cup E_\rho \cup \{(v,v_0),(v,v_1)\}, L_\psi \cup L_\rho \cup \{(v,\varphi)\})$.

For $\varphi = \mathsf{X}\psi$*, we define* $\mathrm{syn}(\varphi) = (V_\psi \cup \{v\}, E_\psi \cup \{(v,v_0)\}, L_\psi \cup \{(v,\varphi)\})$.

Obviously, $\mathrm{syn}(\varphi)$ is a tree. A node v of $\mathrm{syn}(\varphi)$ is a *leaf* if and only if $L_\varphi(v)$ is a literal, tt or ff. A node v is the *root* of $\mathrm{syn}(\varphi)$ if and only if $L_\varphi(v) = \varphi$.

Definition 2.7 (successor labeling) *A* successor labeling sl *of a syntax tree* $\mathrm{syn}(\varphi) = (V,E,L)$ *is a mapping* $V \to \mathrm{term}_\Sigma \times 2^{\mathrm{sub}(\varphi)}$ *which is inductively defined in analogy to the successor sets.*

That is, $\mathrm{sl}(v) = (L(v), \emptyset)$ *if* $L(v)$ *is* tt *or a literal (*$\mathrm{sl}(v)$ *is undefined for* $L(v) =$ ff*).*

If $\{(v,v_0),(v,v_1)\} \subseteq E$*,* $L(v_0) = \psi$ *and* $L(v_1) = \rho$*, and* $\mathrm{sl}(v_0) = (t,N)$ *and* $\mathrm{sl}(v_1) = (t',N')$ *are already defined, then*

– if $L(v) = \psi \vee \rho$*, then either* $\mathrm{sl}(v) = \mathrm{sl}(v_0)$ *or* $\mathrm{sl}(v) = \mathrm{sl}(v_1)$*,*

– *if* $L(v) = \psi \wedge \rho$, *then* $\mathrm{sl}(v) = (t \wedge t', N \cup N')$.
– *if* $L(v) = \psi \mathsf{U} \rho$, *then either* $\mathrm{sl}(v) = (t, N \cup \{L(v)\})$ *or* $\mathrm{sl}(v) = \mathrm{sl}(v_1) = (t', N')$,
– *if* $L(v) = \psi \mathsf{R} \rho$, *then either* $\mathrm{sl}(v) = (t', N' \cup \{L(v)\})$ *or* $\mathrm{sl}(v) = (t \wedge t', N \cup N')$.
If $(v, v_0) \in E$, $L(v_0) = \psi$ *and* $L(v) = \mathsf{X}\psi$, *then* $\mathrm{sl}(v) = (\mathsf{tt}, \{\psi\})$.

A successor labeling sl_M *of a set* M *of (disjoint) syntax trees is a mapping defined for all nodes of the trees in* M *such that* sl_M *restricted to the nodes of a single tree in* M *is a successor labeling.*

That is, $(t,N) \in \mathrm{scs}(\{\varphi\})$ if and only if there is a successor labeling sl of $\mathrm{syn}(\varphi)$ such that for the root v_r of $\mathrm{syn}(\varphi)$, we have $\mathrm{sl}(v_r) = (t,N)$. Analogously, $(t,N) \in \mathrm{scs}(M)$ if and only if there is a successor labeling sl of M such that for the roots $v_0, \ldots, v_{n-1}$ of the trees in M, we have $\mathrm{sl}(v_i) = (t_i, N_i)$ for every $i < n$, $t \equiv \bigwedge_{i<n} t_i$ and $N = \bigcup_{i<n} N_i$. Note that several different successor labelings may all result in the same term and successor set.

Definition 2.8 (characteristic function) *A* characteristic function cf *of a successor labeling* sl *of a syntax tree* $\mathrm{syn}(\varphi) = (V,E,L)$ *is a mapping* $V \to \{0,1\}$ *such that* $\mathrm{cf}(v) = 0$ *if* $L(v)$ *is a literal,* tt, ff *or a* X- *or* $\wedge$*-formula; if* $L(v)$ *is* $\psi \vee \rho$, $\psi \mathsf{U} \rho$ *or* $\psi \mathsf{R} \rho$ *such that* $\{(v,v_0),(v,v_1)\} \subseteq E$ *and* $L(v_0) = \psi$, $L(v_1) = \rho$, *and* $\mathrm{sl}(v_0) = (t,N)$ *and* $\mathrm{sl}(v_1) = (t',N')$, *then*

- *if* $L(v) = \psi \vee \rho$, *then if* $\mathrm{sl}(v) = \mathrm{sl}(v_0)$, *then* $\mathrm{cf}(v) = 0$, *else* $\mathrm{cf}(v) = 1$;
- *if* $L(v) = \psi \mathsf{U} \rho$, *then if* $\mathrm{sl}(v) = \mathrm{sl}(v_1)$, *then* $\mathrm{cf}(v) = 0$, *else* $\mathrm{cf}(v) = 1$;
- *if* $L(v) = \psi \mathsf{R} \rho$, *then if* $\mathrm{sl}(v) = (t \wedge t', N \cup N')$, *then* $\mathrm{cf}(v) = 0$, *else* $\mathrm{cf}(v) = 1$.

A characteristic function of a successor labeling of a set of syntax trees is defined analogously.

That is, the characteristic function is a kind of roadmap on how to choose successor sets of subformulas: For a fixed syntax tree $\mathrm{syn}(\varphi)$ and a characteristic function cf for $\mathrm{syn}(\varphi)$, there is one and only one successor labeling sl of $\mathrm{syn}(\varphi)$ such that cf is a characteristic function for sl. Consequently, for a set of syntax trees M, the choice of a characteristic function defined for every node in one of the trees in M determines a successor labeling for M and thus a term t and a set N such that N is a successor set of M for t.

Part (1) of the following lemma can be read as follows: We are given a set of formulas M such that a syntactical redundancy check *might* detect (given the right order of processing of the formulas in M) that $\varphi \in M$ does not need any processing. If the result of this is that a transition from M to N is not constructed, then, for every possible i such that (M,i) is a state in our automaton, a transition to (N,j) with any possible j is not locally optimal and will be deleted.

In this sense, redundancy checks by syntactical implication are covered by local optimization.

Lemma 2.9 *Let $((M,i),t,(N,j))$ be a transition of the NBA $\mathbf{Q}_z^{td}(\varphi_0)$, for an LTL formula φ_0. Let* cf *be a characteristic function for M such that t and N are the resulting term and successor set such that also, if* sl *is the resulting successor labeling, $i \neq 0$ and v_i is the root of* $\mathrm{syn}(z^{-1}(i))$*, then $z^{-1}(i) \in \mathrm{sl}(v_i)$ if and only if $i = j$.*

Let $\varphi \in M$, and let $(t',N') \in \mathrm{scs}(M \setminus \{\varphi\})$ be such that t' and N' result from cf *restricted to $M \setminus \{\varphi\}$; especially, $t \to t'$ and $N' \subseteq N$.*

1. *If $\varphi \in \mathtt{SI}(\mathrm{lit}(t'),N')$ such that also $\varphi \notin N'$ if φ is an* U*-formula, then $(t',N') \in \mathrm{scs}(M)$, and there is a transition $((M,i),t',(N',j'))$ which is as least as good as $((M,i),t,(N,j))$, i. e., $j' \leq j$.*

2. *If there is a $j' \leq j$ such that $((M,i),t',(N',j'))$ is a transition of $\mathbf{Q}_z^{td}(\varphi_0)$ (i. e., $((M,i),t',(N',j'))$ is at least as good as $((M,i),t,(N,j))$), then $\varphi \in \mathtt{SI}(\mathrm{lit}(t'),N')$.*

Proof. (1) If $\varphi \in \mathtt{SI}(\mathrm{lit}(t'),N')$, then there is a subset N'' of N' and a subformula t'' of t' such that $N'' \in \mathrm{scs}(\{\varphi\},t'')$, by Lemma 2.8(1). That is, $(t' \wedge t'', N' \cup N'') = (t',N'') \in \mathrm{scs}((M \setminus \{\varphi\}) \cup \{\varphi\}) = \mathrm{scs}(M)$. It follows that $((M,i),t',(N',j'))$ is a transition in $\mathbf{Q}_z^{td}(\varphi_0)$ for some j'.

We thus find a characteristic function cf′ (and a resulting successor labeling sl′) for M which is identical to cf (to sl) on $M \setminus \{\varphi\}$ and which results in the root label (t'',N'') for $\mathrm{syn}(\varphi)$. That is, cf′ determines the transition $((M,i),t',(N',j'))$ such that it is guaranteed that $j' \leq j$. This is because if $z^{-1}(i)$ is in the sl′-root label of $\mathrm{syn}(z^{-1}(i))$, then it also is in the sl-root label of the same tree, and if another U-formula is in some sl′-root label, then it also is in some sl-root label.

(2) This is an obvious application of Lemma 2.8(2). □

The following is now easy to see.

Proposition 2.3 *Lemma 2.9 is also true for transitions of $A_z^{ad}(\varphi_0)$, but for the better transition $((M,i),t',(N',j'))$ in part (1), the value of j' may be different from the value of j' for $\mathbf{Q}_z^{td}(\varphi_0)$.*

Lemma 2.9 is about the connection between the redundancy of a formula in a state and the *outgoing* transitions of that state. There is also an important connection to the *incoming* transitions, which is especially interesting for redundant U- and R-formulas.

Lemma 2.10 *Let $((M,i),t,(N,j))$ be a transition of the NBA $\mathbf{Q}_z^{td}(\varphi_0)$ or $\mathcal{A}_z^{ad}(\varphi_0)$, for an LTL formula φ_0 in next normal form. Let $\varphi \in N$ be an* U- *or* R*-formula and $\varphi \in \mathtt{SI}(\mathrm{lit}(t), N \setminus \{\varphi\})$.*

Then there is a transition $((M,i),t',(N',j'))$ of $\mathbf{Q}_z^{td}(\varphi_0)$ or $\mathcal{A}_z^{ad}(\varphi_0)$, respectively, which is strictly better than $((M,i),t,(N,j))$.

Proof. Let $M = \{\varphi_1,\ldots,\varphi_n\}$. For $1 \leq l \leq n$, we find characteristic functions cf_l of successor labelings sl_l of $\mathrm{syn}(\varphi_l)$ such that the transition $((M,i),t,(N,j))$ results from the combination of the successor labels of the roots of the syntax trees. That is, if $v_1,\ldots,v_n$ are the root nodes of $\mathrm{syn}(\varphi_1),\ldots,\mathrm{syn}(\varphi_n)$ and $(t_l,N_l) = \mathrm{sl}_l(v_l)$ for $1 \leq l \leq n$, then $N = \bigcup_{l<n} N_l$ and $t \equiv \bigwedge_{l<n} t_l$. Also, if $i \neq 0$ and $z(\varphi_l) = i$ then $\varphi_l \in N_l$ if and only if $i = j$.

Assume that $\varphi_1,\ldots,\varphi_r$ are those formulas that contain the formula φ as a subformula such that also $\varphi \in N_l$, for $1 \leq l \leq r \leq n$. We now construct new successor labelings for $\mathrm{syn}(\varphi_1),\ldots,\mathrm{syn}(\varphi_r)$, as follows. Assume that v_φ is a node in one of these syntax trees, say, in $\mathrm{syn}(\varphi_l)$, such that $L_{\varphi_l}(v_\varphi) = \varphi$ and $\mathrm{cf}_l(v_\varphi) = 1$. By Lemma 2.8(1), there is a term t_φ and a set $N_\varphi \subseteq N \setminus \{\varphi\}$ such that $t \to t_\varphi$ and $N_\varphi \in \mathrm{scs}(\{\varphi\},t_\varphi)$. Let cf_φ be a characteristic function of a successor labeling sl_φ of $\mathrm{syn}(\varphi)$ such that $\mathrm{sl}_\varphi(v) = (t_\varphi,N_\varphi)$ for the root v of $\mathrm{syn}(\varphi)$. Note that $\mathrm{cf}_\varphi(v) = 0$. The new characteristic function cf'_l for $\mathrm{syn}(\varphi_l)$ is now defined like cf_φ for the subtree rooted at v_φ; for the other nodes, it is defined like cf_l. Let sl'_l be the successor labeling for cf'_l and $\mathrm{syn}(\varphi_l)$, and let $(t'_l,N'_l) = \mathrm{sl}_l(v_l)$ be the new root successor label of $\mathrm{syn}(\varphi_l)$.

If we apply these changes to all such nodes v_φ, then φ is not an element of any set N'_l, for $1 \leq l \leq r$. This is because we assume negation normal form: There is no subformula $\mathsf{X}\varphi$, so the successor labeling φ can only be propagated to a root node if φ is in the successor labeling of a node v_φ, and this is excluded by our choice of cf'.

Instead $N_\varphi \subseteq N'_l$, and every formula in N'_l belongs to N_l if it does not belong to N_φ. Since $N_\varphi \subseteq N \setminus \{\varphi\}$,

$$N' = \bigcup_{1\leq l\leq r} N'_l \cup \bigcup_{r<m\leq n} N_m \tag{2.89}$$

also is a subset of $N \setminus \{\varphi\}$. Similarly, every literal in $\mathrm{lit}(t'_l)$ is in $\mathrm{lit}(t_l)$ or in $\mathrm{lit}(t_\varphi)$, and these two sets are subsets of $\mathrm{lit}(t)$, i. e., $t \to t'_l$ and

$$t' = \bigwedge_{1\leq l\leq r} t'_l \wedge \bigwedge_{r<m\leq n} t_k \tag{2.90}$$

also is implied by t.

By construction, N' is a successor set of M for t'. That is, by Lemma 2.7, there is a transition $((M,i),t',(N',j'))$ in both $\mathbf{Q}_z^{td}(\varphi)$ and $\mathcal{A}_z^{ad}(\varphi)$. For $\mathbf{Q}_z^{td}(\varphi)$, we have $j' \leq j$ because N' contains less U-formulas.

We also have $j' \leq j$ for $\mathcal{A}_z^{ad}(\varphi)$. This is because for sets of subformulas N_0 and N_1 such that $M \rightsquigarrow (N_0,N)$ and $M \rightsquigarrow (N_1,N')$, the U-formulas in N_1 are also in N_0, and if a formula $\psi \mathsf{U} \rho$ is in $N_0 \cap N_1$ such that also $\rho \in N_0$, then also $\rho \in N_1$.

Consequently, $((M,i),t',(N',j'))$ is a strictly better transition than $((M,i),t,(N,j))$. □

We can extract the following corollary from the proof of Lemma 2.10.

Corollary 2.5 *Let φ be an LTL formula in next normal form. Let $((M,i),t,(N,j))$ be a transition of $\mathbf{Q}_z^{td}(\varphi)$ or $\mathcal{A}_z^{ad}(\varphi)$.*

If there is a formula $\psi \mathsf{U} \rho \in N$ (where $\mathsf{F}\rho$ is regarded as $\mathtt{tt} \mathsf{U} \rho$) such that $N_\rho \subseteq N$ for a successor set N_ρ of $\{\rho\}$ by a term t' such that $t \to t'$, then this transition is not a transition in $\mathbf{Q}_z^{lo}(\varphi)$ or $\mathcal{A}_z^{lo}(\varphi)$, because it is removed during the local optimization.

2.7.4 Equality of locally optimized top-down automaton and GPVW-automaton

From Corollary 2.5, it follows that for a transition $((M,i),t,(N,j))$ of an optimized automaton with formulas in next normal form, the value j is determined by i and N only.

Corollary 2.6 *Let φ be an LTL formula in next normal form. Let $((M,i),t,(N,j))$ be a transition of $\mathbf{Q}_z^{lo}(\varphi)$ or $\mathcal{A}_z^{lo}(\varphi)$.*

If there is a formula $\psi \mathsf{U} \rho \in N$ such that $z(\psi \mathsf{U} \rho) \leq i$, then $i \geq j \geq z(\psi \mathsf{U} \rho)$, else $j = 0$.

More precisely, $j = \max\{z(\psi \mathsf{U} \rho) \leq i \mid \psi \mathsf{U} \rho \in N\}$.

Proof. Let first $((M,i),t,(N,j))$ be a transition of $\mathbf{Q}_z^{lo}(\varphi)$, and let $\psi \mathsf{U} \rho \in N$ such that $z(\psi \mathsf{U} \rho) \leq i$. Assume that $j < z(\psi \mathsf{U} \rho)$. It follows that there is a term t' and a set N_ρ such that N_ρ is a successor set of $\{\psi \mathsf{U} \rho\}$ for t', $N_\rho \subseteq N$ and t' is a subformula of t, i. e., $t \to t'$. This contradicts Corollary 2.5, hence $j \geq z(\psi \mathsf{U} \rho)$. If, on the other hand, there is no formula $\psi \mathsf{U} \rho \in N$ such that $z(\psi \mathsf{U} \rho) \leq i$, then clearly $j = 0$ by construction of $\mathbf{Q}_z^{lo}(\varphi)$.

Now let $((M,i),t,(N,j))$ be a transition of $\mathcal{A}_z^{lo}(\varphi)$ such that again $\psi \mathsf{U} \rho \in N$ and $z(\psi \mathsf{U} \rho) \leq i$. Assume that $j < z(\psi \mathsf{U} \rho)$. We then find a set $M' \in 2^{\mathrm{sub}(\varphi)}$ such that $M \rightsquigarrow (M',N)$ and $\rho \in M'$, which implies that a successor set of $\{\rho\}$ is a subset of N. So this is again the situation of Corollary 2.5 where the transition

$((M,i),t,(N,j))$ is deleted during the local optimization. Hence $j \geq z(\psi \mathsf{U} \rho)$. If there is no formula $\psi \mathsf{U} \rho \in N$ such that $z(\psi \mathsf{U} \rho) \leq i$, then there is also no such U-formula in M' for every $M' \in 2^{\mathrm{sub}(\varphi)}$ such that $M \rightsquigarrow (M',N)$; consequently, $j = 0$.

The precise value of j then follows by the consistency of automata states. □

With these considerations, we can now prove the following crucial lemma.

Lemma 2.11 *Let φ be an LTL formula in next normal form, and let (M,i) be a state of both $\mathbf{Q}_z^{lo}(\varphi)$ and $\mathcal{A}_z^{lo}(\varphi)$.*

Then $((M,i),t,(N,j))$ is a transition of $\mathcal{A}_z^{lo}(\varphi)$ if and only if it is a transition of $\mathbf{Q}_z^{lo}(\varphi)$.

Proof. First let $((M,i),t,(N,j))$ be a transition of $\mathcal{A}_z^{lo}(\varphi)$. That is, there is no (strictly) better transition in $\mathcal{A}_z^{ad}(\varphi)$ than $((M,i),t,(N,j))$. By Lemma 2.7, there is a transition $((M,i),t,(N,\hat{j}))$ in $\mathbf{Q}_z^{td}(\varphi)$ for some $\hat{j}$.

Assume that there is a transition in $\mathbf{Q}_z^{td}(\varphi)$ which is strictly better than $((M,i),t,(N,j))$, and let $((M,i),t',(N',j'))$ be a best transition with this property, i. e., $((M,i),t',(N',j'))$ also is a transition in $\mathbf{Q}_z^{lo}(\varphi)$. Then again by Lemma 2.7, there is a transition $((M,i),t',(N',j''))$ in $\mathcal{A}_z^{ad}(\varphi)$ such that $t \to t'$ and $N' \subseteq N$.

The value of j'' is either $\max\{z(\psi \mathsf{U} \rho) \mid \psi \mathsf{U} \rho \in N', z(\psi \mathsf{U} \rho) \leq i \text{ if } i \neq 0\}$, which implies that $j'' \leq j$. Consequently, $((M,i),t',(N',j''))$ is at least as good as $((M,i),t,(N,j))$. Since there is no strictly better transition in $\mathcal{A}_z^{ad}(\varphi)$ than $((M,i),t,(N,j))$, we have $t' \equiv t$, $N' = N$ and $j'' = j$. Since we assume that $((M,i),t',(N',j'))$ is strictly better than $((M,i),t,(N,j))$, we have $j \neq 0$ and either $j' < j$. But since $N = N'$ and $((M,i),t',(N',j'))$ is a transition in $\mathbf{Q}_z^{lo}(\varphi)$, it must be the case that $j = j'$ by Corollary 2.6. Contradiction.

Or j'' is not a function of i and N'. In this case, the transition $((M,i),t',(N',j''))$ is not a transition of $\mathcal{A}_z^{lo}(\varphi)$ (Corollary 2.6), so there is an even better transition $((M,i),t'',(N'',\bar{j}))$ in $\mathcal{A}_z^{ad}(\varphi)$ such that $t \to t' \to t''$ and $N'' \subseteq N' \subseteq N$, where $\bar{j} = \max\{z(\psi \mathsf{U} \rho) \mid \psi \mathsf{U} \rho \in N'', z(\psi \mathsf{U} \rho) \leq i \text{ if } i \neq 0\}$. This again contradicts the choice of $((M,i),t,(N,j))$.

Hence there is no better transition in $\mathbf{Q}_z^{td}(\varphi)$ than $((M,i),t,(N,j))$. Since $((M,i),t,(N,\hat{j}))$ is a transition of $\mathbf{Q}_z^{td}(\varphi)$, we have $\hat{j} \geq j$. If $((M,i),t,(N,\hat{j}))$ is a transition of $\mathbf{Q}_z^{lo}(\varphi)$, then $\hat{j} = j$ by Corollary 2.6, and we are done. Else there is a transition $((M,i),\bar{t},(\bar{N},\bar{j}))$ in $\mathbf{Q}_z^{td}(\varphi)$ and in $\mathbf{Q}_z^{lo}(\varphi)$ which is better than $((M,i),t,(N,\hat{j}))$ but at most as good as $((M,i),t,(N,j))$. Following Corollary 2.6, we then have $\bar{t} = t$, $\bar{N} = N$, and $\bar{j} = j$.

The other direction is completely similar (exchange $\mathcal{A}_z^{ad}(\varphi)$ and $\mathbf{Q}_z^{td}(\varphi)$ as well as $\mathcal{A}_z^{lo}(\varphi)$ and $\mathbf{Q}_z^{lo}(\varphi)$). □

It now follows directly that $\mathcal{A}_z^{lo}(\varphi)$ and $\mathbf{Q}_z^{lo}(\varphi)$ are the same automaton for LTL formulas φ in next normal form.

Theorem 2.6 (equality to GPVW-automaton) *Let φ be an LTL formula in next normal form over a set of propositions Σ. Let P_φ be the set of* U- *and* F*-subformulas of φ, and let z be a bijection $P_\varphi \to \{1,\ldots,\left|P_\varphi\right|\}$.*

Then $\mathcal{A}_z^{lo}(\varphi) = \mathbf{Q}_z^{lo}(\varphi)$.

Proof. By definition, $\mathcal{A}_z^{lo}(\varphi)$ and $\mathbf{Q}_z^{lo}(\varphi)$ have the same initial state.

By Lemma 2.11, it then follows inductively that we have exactly the same states and transitions in both $\mathcal{A}_z^{lo}(\varphi)$ and $\mathbf{Q}_z^{lo}(\varphi)$. In consequence, also the set of accepting states is the same. □

2.7.5 Equality to the DGV-automaton

Ignoring redundancy and contradiction checks, the difference between the GPVW-automaton and the DGV-automaton can be described as follows.

- For the DGV-automaton, we have $M \rightsquigarrow (M',N')$ if there is a $t \in \mathrm{term}_\Sigma$ such that $\mathrm{scs}(M,t,N')$ and $M' = \mathrm{lit}(t)$.
- The acceptance sets in the generalized Büchi condition are $F_i = \{(M,N) \in Q_\varphi \mid (z(\psi \mathsf{U} \rho) = i \wedge \psi \mathsf{U} \rho \in \mathtt{SI}(M,N)) \implies \rho \in \mathtt{SI}(M,N)\}$, for $1 \leq i \leq k_\varphi$.

All other definitions and the adjustments are as for the GPVW-automaton. We write $\mathcal{B}^{ad}(\varphi)$ for the adjusted DGV-automaton for an input formula φ, or $\mathcal{B}_z^{ad}(\varphi)$ to stress that the adjustment is based on the bijection z.

We assume that, in the construction of the automaton $\mathcal{B}^{ad}(\varphi)$, redundancy and contradiction checks were not performed. By Lemma 2.9, the locally optimized version $\mathcal{B}^{lo}(\varphi)$ corresponds to the DGV-automaton for which the redundancy checks progressed in an optimal way, i. e., all possible redundancies were detected. The effect of the contradiction check of [DGV99], also using syntactical implication, can be covered by the assumption that all automata are reduced to their productive states, where a state q is productive if it is reachable from the initial state and there is an accepting run (on an arbitrary word) starting from q, i. e., a strongly connected component of the transition graph containing an accepting state is reachable from q.

For an arbitrary input formula φ, $\mathcal{B}^{ad}(\varphi)$ can be different from both $\mathcal{A}^{ad}(\varphi)$ and $\mathbf{Q}^{td}(\varphi)$ (see Subsection 2.7.6). Nevertheless, the locally optimized DGV-automaton $\mathcal{B}^{lo}(\varphi)$ for an input formula φ in next normal form is equal to the locally optimized top-down and GPVW-automata.

Theorem 2.7 (equality to DGV-automaton) *Let φ be an LTL formula in next normal form over a set of propositions Σ. Let P_φ be the set of* U- *and* F*-subformulas of φ, and let z be a bijection $P_\varphi \to \{1,\ldots,|P_\varphi|\}$.*

Then $\mathcal{A}_z^{lo}(\varphi) = \mathcal{B}_z^{lo}(\varphi) = \mathbf{Q}_z^{lo}(\varphi)$.

Proof. By definition, the transition structure of $\mathcal{B}_z^{ad}(\varphi)$ is based on the same successor set structure as for the top-down automaton. Therefore, it is easy to see that Lemmas 2.9 and 2.10 are also true for $\mathcal{B}_z^{ad}(\varphi)$. The condition "$\psi \,\mathsf{U}\, \rho \in \mathtt{SI}(M,N)$" from the above definition of the acceptance sets translates to "$\psi \,\mathsf{U}\, \rho \in N$ and $\psi \in \mathtt{SI}(M,N)$" by the definition of syntactical implication.

Now if the input formula φ is in next normal form, then $\psi \,\mathsf{U}\, \rho \in N$ implies $\psi \in \mathtt{SI}(M,N)$. This is because, in this case, $\psi \,\mathsf{U}\, \rho \in N$ implies $N_\psi \subseteq N$ for a set N_ψ and a term t' such that $N_\psi \in \mathrm{scs}(\{\psi\},t')$ and $\mathrm{lit}(t') \subseteq M$. By Lemma 2.8, $\psi \in \mathtt{SI}(\mathrm{lit}(t'),N_\psi) \subseteq \mathtt{SI}(M,N)$. Also by Lemma 2.8, $\rho \in \mathtt{SI}(M,N)$ implies that there are sets $M' \subseteq M$, $N_\rho \subseteq N$ such that $N_\rho \in \mathrm{scs}(\{\rho\},\mathrm{term}(M'))$.

That is, for φ in next normal form, the acceptance sets are $F_i = \{(M,N) \in Q_\varphi \mid (z(\psi \,\mathsf{U}\, \rho) = i \wedge \psi \,\mathsf{U}\, \rho \in N) \implies \exists M' \subseteq M, N_\rho \subseteq N\colon N_\rho \in \mathrm{scs}(\{\rho\},\mathrm{term}(M'))\}$, for $1 \leq i \leq k_\varphi$. But having both $\psi \,\mathsf{U}\, \rho \in N$ and $N_\rho \subseteq N$ is exactly the situation of Corollary 2.5. Consequently, for the locally optimized automaton, the requirement for the ith acceptance set reduces to "$z(\psi \,\mathsf{U}\, \rho) = i \to \psi \,\mathsf{U}\, \rho \notin N$".

That is, we have exactly the situation of Corollary 2.6: In a transition $((M,i),t,(N,j))$ of $\mathcal{B}_z^{lo}(\varphi)$, the value j depends on (the U-formulas in) N and i only.

The rest of the argumentation now is as in Subsection 2.7.4. □

2.7.6 Why do we need next normal form?

To see the problem for formulas that are not in next normal form, consider the automata $\mathcal{A}^{lo}(\varphi)$ and $\mathbf{Q}^{lo}(\varphi)$ for the formula $\varphi = \mathsf{GX}(a \,\mathsf{U}\, b)$ in Figure 2.7.

The problem here is that $\mathcal{A}(\varphi)$ has the states $(\{\varphi,\mathsf{X}(a \,\mathsf{U}\, b),a \,\mathsf{U}\, b,b\},\{\varphi,a \,\mathsf{U}\, b\})$ and $(\{\varphi,\mathsf{X}(a \,\mathsf{U}\, b),a \,\mathsf{U}\, b,a\},\{\varphi,a \,\mathsf{U}\, b\})$, which have self-transitions and transitions to each other. The former state is in F_1, since while $a \,\mathsf{U}\, b$ is in its first component, b is also in its first component; the latter state is not in F_1. Consequently, $\mathcal{A}^{lo}(\varphi)$ has a similar structure in which a b-transition always leads to the accepting state $(\{\varphi,a \,\mathsf{U}\, b\},0)$.

But in $\mathbf{Q}^{lo}(\varphi)$, we have the transition $((\{\varphi,a \,\mathsf{U}\, b\},0),b,(\{\varphi,a \,\mathsf{U}\, b\},\mathbf{1}))$, because the counter is set from 0 to the highest possible value 1.

For $\varphi = \mathsf{GX}(a \,\mathsf{U}\, b)$, $\mathcal{A}^{lo}(\varphi)$ and $\mathcal{B}^{lo}(\varphi)$ are the same automaton. This is *not* the case for the formula $\varphi = a \wedge \mathsf{X}(a \,\mathsf{U}\, b)$. For this formula, $\mathcal{A}(\varphi)$ has a transition $((\{\mathsf{X}(a \,\mathsf{U}\, b),a\},\{a \,\mathsf{U}\, b\}),(\{a \,\mathsf{U}\, b,a\},\{a \,\mathsf{U}\, b\}))$ where the state $(\{\mathsf{X}(a \,\mathsf{U}$

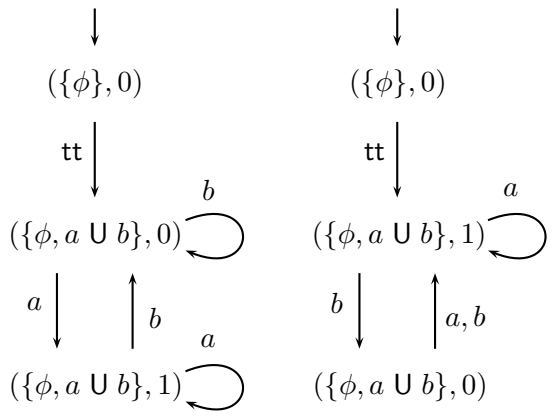

Figure 2.7: $\mathcal{A}^{lo}(\varphi)$ (left) and $\mathbf{Q}^{lo}(\varphi)$ (right) for $\varphi = \mathsf{GX}(a \mathsf{U} b)$

$b), a\}, \{a \mathsf{U} b\})$ is the initial state and is in F_1, but the state $(\{a \mathsf{U} b, a\}, \{a \mathsf{U} b\})$ is not in F_1. This translates in $\mathcal{A}^{lo}(\varphi)$ to a transition $((\{a \mathsf{U} b\}, 0), a, (\{a \mathsf{U} b\}, 1))$.

But in $\mathcal{B}(\varphi)$, there is only one state $(\{a\}, \{a \mathsf{U} b\})$ instead, which is initial and not in F_1 since $a \mathsf{U} b \in \mathtt{SI}(\{a\}, \{a \mathsf{U} b\})$, but $b \notin \mathtt{SI}(\{a\}, \{a \mathsf{U} b\})$. As a consequence, $(\{a \mathsf{U} b\}, 0)$ is not a state in $\mathcal{B}^{lo}(\varphi)$; only $(\{a \mathsf{U} b\}, 1)$ is. Note that, in this example, $\mathbf{Q}^{lo}(\varphi) = \mathcal{B}^{lo}(\varphi)$.

In summary, locally suboptimal transitions and U-formulas "popping up" out of the scope of a X-operator interoperate with the acceptance / the counter behavior of the three automata in three different ways, but for optimal transitions and formulas in next normal form, the counter behavior turns out to be the same in all three cases.

2.8 Inductive Bottom-Up NBA Construction from LTL

It is also possible to construct an NBA from an LTL formula in a bottom-up way, by defining nondeterministic automata inductively over the structure of the formula. For a given LTL formula, the inductive construction described below and the top-down approach of Section 2.4 yield the same nondeterministic automaton (up to the order induced on the states in P_φ by the bijection z). The main advantage of the bottom-up construction is the possibility to make use of simplifications of automata for subformulas. For example, the size of $\mathbf{Q}^{td}(\psi \mathsf{U} \rho)$ may be quadratic in the size of both $\mathbf{Q}^{td}(\psi)$ and $\mathbf{Q}^{td}(\rho)$. It may be too time-consuming to substan-

tially simplify $\mathbf{Q}^{td}(\psi \mathsf{U} \rho)$, but it may be possible to simplify $\mathbf{Q}^{td}(\psi)$ and $\mathbf{Q}^{td}(\rho)$; we describe how to make use of these simplifications in the construction of an automaton for $\psi \mathsf{U} \rho$.

We will first give an inductive construction and then discuss the advantages and disadvantages of the two approaches.

2.8.1 The bottom-up construction

We inductively define, for every LTL formula in negation normal form φ over a set of propositions Σ, an equivalent nondeterministic Büchi automaton (the *bottom-up automaton* of φ)

$$\mathbf{Q}^{bu}(\varphi) = (\mathbf{Q}_\varphi, \Sigma, q_I^\varphi, \Delta^{bu}, F_\varphi) \tag{2.91}$$

where the components $\mathbf{Q}_\varphi$, q_I^φ and F_φ are defined as in Subsection 2.4.3.

Again, let P_φ be the set of U- and F-subformulas in φ and $k_\varphi = \left|P_\varphi\right|$.

The transition structure of the defined automata will similar to the structure of the automata defined in Subsection 2.4.3, that is, for a fixed bijection $z\colon P_\varphi \to \{1,\dots,k_\varphi\}$, every transition of the automaton $\mathbf{Q}^{bu}(\varphi)$ defined here is also a transition of the automaton $\mathbf{Q}^{td}(\varphi)$ constructed via the top-down approach and vice versa. This underlying bijection z_φ will be defined inductively together with $\mathbf{Q}^{bu}(\varphi)$.

The construction is as follows.

Atomic formulas: If φ is an atomic formula tt, ff, a or $\neg a$ with $a \in \Sigma$, we have $\mathbf{Q}^{bu}(\varphi) = \mathbf{Q}^{td}(\varphi)$. The mapping z_φ is empty.

In the following, we assume that

$$\mathbf{Q}^{bu}(\psi) = (2^{\mathrm{sub}(\psi)} \times (k_\psi + 1), \Sigma, q_I^\psi, \Delta_\psi, 2^{\mathrm{sub}(\varphi)} \times \{0\}) \tag{2.92}$$

and

$$\mathbf{Q}^{bu}(\rho) = (2^{\mathrm{sub}(\rho)} \times (k_\rho + 1), \Sigma, q_I^\rho, \Delta_\rho, 2^{\mathrm{sub}(\rho)} \times \{0\}) \tag{2.93}$$

and the bijections $z_\psi\colon P_\psi \to \{1,\dots,k_\psi\}$ and $z_\rho\colon P_\rho \to \{1,\dots,k_\rho\}$ are already defined.

We then define the nondeterministic automaton

$$\mathbf{Q}^{bu}(\varphi) = (2^{\mathrm{sub}(\varphi)} \times (k_\varphi + 1), \Sigma, q_I^\varphi, \Delta^{bu}, 2^{\mathrm{sub}(\varphi)} \times \{0\}) \tag{2.94}$$

and the mapping $z_\varphi\colon P_\varphi \to \{1,\dots,k_\varphi\}$.

Initial states, initial transitions and z_φ:

- For $\varphi = \mathsf{X}\psi$, we have $q_I^\varphi = (\{\varphi\}, 0)$ and $(q_I^\varphi, \mathsf{tt}, q_I^\psi) \in \Delta^{bu}$.

  We set $z_\varphi = z_\psi$.

- For $\varphi = \psi \vee \rho$, we have $q_I^\varphi = (\{\varphi\}, 0)$ and
  - $(q_I^\varphi, t, (N_\psi, j_\psi)) \in \Delta^{bu}$ for every transition $(q_I^\psi, t, (N_\psi, j_\psi)) \in \Delta_\psi$,
  - $(q_I^\psi, t, (N_\rho, 0)) \in \Delta^{bu}$ for every transition $(q_I^\rho, t, (N_\rho, 0)) \in \Delta_\rho$,
  - $(q_I^\psi, t, (N_\rho, j_\rho + k_\psi)) \in \Delta^{bu}$ for every transition $(q_I^\rho, t, (N_\rho, j_\rho)) \in \Delta_\rho$ such that $j_\rho > 0$.

  We define $z_\varphi \colon P_\varphi \to \{1, \ldots, k_\varphi = k_\psi + k_\rho\}$ by
  - $z_\varphi(p) = z_\psi(p)$ for $p \in P_\psi$,
  - $z_\varphi(p) = z_\rho(p) + k_\psi$ for $p \in P_\rho$.
- For $\varphi = \psi \wedge \rho$, assume that the initial states of $\mathbf{Q}^{bu}(\psi)$ and $\mathbf{Q}^{bu}(\rho)$ are $q_I^\psi = (I_\psi, i_\psi)$ and $q_I^\rho = (I_\rho, i_\rho)$, respectively. The initial state of $\mathbf{Q}^{bu}(\varphi)$ is $q_I^\varphi = (I_\psi \cup I_\rho, i_\varphi)$, where
  - $i_\varphi = 0$ if $i_\psi = i_\rho = 0$,
  - $i_\varphi = i_\psi$ if $i_\psi \neq 0$,
  - $i_\varphi = i_\rho + k_\psi$ if $i_\psi = 0$ and $i_\rho \neq 0$.

  The initial transitions are covered by the set of all transitions below.

  The mapping z_φ is the same as for the case $\psi \vee \rho$.
- For $\varphi = \psi \mathsf{U} \rho$, the initial state of $\mathbf{Q}^{bu}(\varphi)$ is $q_I^\varphi = (\{\varphi\}, k_\varphi)$, with $k_\varphi = k_\psi + k_\rho + 1$.

  For all transitions $(q_I^\psi, t_\psi, (N_\psi, j_\psi)) \in \Delta_\psi$, $(q_I^\rho, t_\rho, (N_\rho, j_\rho)) \in \Delta_\rho$, we have as initial transitions
  - $(q_I^\varphi, t, (\{\varphi\} \cup N_\psi, k_\varphi)) \in \Delta^{bu}$,
  - $(q_I^\varphi, t, (N_\rho, 0)) \in \Delta^{bu}$ if $j_\rho = 0$,
  - $(q_I^\varphi, t, (N_\rho, j_\rho + k_\psi)) \in \Delta^{bu}$ if $j_\rho \neq 0$.

  The bijection $z_\varphi \colon P_\varphi \to \{1, \ldots, k_\varphi\}$ is defined by
  - $z_\varphi(\varphi) = k_\varphi$,
  - $z_\varphi(p) = z_\psi(p)$ for $p \in P_\psi$,
  - $z_\varphi(p) = z_\rho(p) + k_\psi$ for $p \in P_\rho$.
- For $\varphi = \psi \mathsf{R} \rho$, the initial state of $\mathbf{Q}^{bu}(\varphi)$ is $q_I^\varphi = (\{\varphi\}, 0)$.

  For all transitions $(q_I^\psi, t_\psi, (N_\psi, j_\psi)) \in \Delta_\psi$, $(q_I^\rho, t_\rho, (N_\rho, j_\rho)) \in \Delta_\rho$, we have as initial transitions

- $(q_I^{\varphi}, t_\rho, (\{\varphi\} \cup N_\rho, j_\rho)) \in \Delta^{bu}$,
- $(q_I^{\varphi}, t_\psi \wedge t_\rho, (N_\psi \cup N_\rho, j_\psi)) \in \Delta^{bu}$ if $j_\psi \neq 0$,
- $(q_I^{\varphi}, t_\psi \wedge t_\rho, (N_\psi \cup N_\rho, j_\rho + k_\psi)) \in \Delta^{bu}$ if $j_\psi = 0$ and $j_\rho \neq 0$,
- $(q_I^{\varphi}, t_\psi \wedge t_\rho, (N_\psi \cup N_\rho, 0)) \in \Delta^{bu}$ if $j_\psi = j_\rho = 0$.

The mapping z_φ is the same as for the case $\psi \vee \rho$.

Other transitions: Let (M,i) be some state of $\mathbf{Q}^{bu}(\varphi)$ other than the initial state.

Let $M_\psi = M \cap 2^{\mathrm{sub}(\psi)}$, $M_\rho = M \cap 2^{\mathrm{sub}(\rho)}$ and $M_\varphi = M \setminus (M_\psi \cup M_\rho)$, i. e., either $M_\varphi = \emptyset$ or $M_\varphi = \{\varphi\}$. (For simplicity in notation, we assume that $((\emptyset,0), \mathtt{tt}, (\emptyset,0)) \in \Delta^{bu} \cap \Delta_\psi \cap \Delta_\rho$). Then there is a transition $((M,i), t_\varphi \wedge t_\psi \wedge t_\rho, (N_\varphi \cup N_\psi \cup N_\rho, j))$ if the terms t_φ, t_ψ and t_ρ, the sets N_φ, N_ψ and N_ρ and the integer j satisfy the following.

- $((M_\varphi, \max\{z(\psi) \mid \psi \in M_\varphi \cap P_\varphi\}), t_\varphi, (N_\varphi, j_\varphi)) \in \Delta^{bu}$
- If $i = k_\psi + k_\rho + 1$ then $((M_\psi, 0), t_\psi, (N_\psi, j_\psi)) \in \Delta_\psi$ and $((M_\rho, 0), t_\rho, (N_\rho, j_\rho)) \in \Delta_\rho$.
- If $0 \leq i \leq k_\psi$ then $((M_\psi, i), t_\psi, (N_\psi, j_\psi)) \in \Delta_\psi$ and $((M_\rho, 0), t_\rho, (N_\rho, j_\rho)) \in \Delta_\rho$.
- If $k_\psi < i \leq k_\psi + k_\rho$ then $((M_\psi, 0), t_\psi, (N_\psi, j_\psi)) \in \Delta_\psi$ and $((M_\rho, i - k_\psi), t_\rho, (N_\rho, j_\rho)) \in \Delta_\rho$.
- $j'_\rho = j_\rho + k_\psi$ if $j_\rho \neq 0$, else $j'_\rho = 0$.
- $j = \max\{0 < l \leq i \mid l \in \{j_\varphi, j_\psi, j'_\rho\}\}$

Theorem 2.8 *Let φ be an LTL formula in negation normal form. Then*

$$L(\varphi) = L(\mathbf{Q}^{bu}(\varphi)) \ . \tag{2.95}$$

Proof. (Sketch) By induction over the structure of LTL formulas, it can be easily shown that $\mathbf{Q}^{bu}(\varphi)$ is the same as $\mathbf{Q}^{td}(\varphi)$. □

2.8.2 The bottom-up construction and simulation-based simplifications

While the inductive bottom-up construction seems to be technically more difficult than the top-down approach, it offers an interesting possibility for NBA construction with simplifications: Given a formula in negation normal form, e. g., $\varphi = \psi \,\mathsf{U}\, \rho$, it is possible to first construct the nondeterministic automata $\mathbf{Q}(\psi)$ and $\mathbf{Q}(\rho)$, simplify these sub-automata, and merge the two simplified automata into an automaton for φ using the above construction. This is especially interesting if a global simplification of $\mathbf{Q}(\varphi)$ is too expensive.

This outlined construction is simple (or even trivial) for formulas φ of the form $\psi \vee \rho$, $\psi \wedge \rho$, $\mathsf{X}\psi$ and $\mathsf{F}\psi$. This is because, in these cases, in constructing the automaton for φ from the sub-automata, it is not necessary to merge two states of the same sub-automaton into a new state.

The situation is different if φ is of the form $\mathsf{G}\rho$, $\psi \,\mathsf{U}\, \rho$ or $\psi \,\mathsf{R}\, \rho$: In these cases, it is necessary to merge two states of the same sub-automaton into a new state. Now if the sub-automaton is a simplified automaton $\mathbf{S}(\psi)$ such that its states do not relate to elements of $2^{\mathrm{sub}(\psi)} \times (k_\psi + 1)$, then we have to apply a Miyano–Hayashi construction and create new states which are subsets of the state set of the simplified sub-automaton (i. e., the size of the resulting automaton may be doubly exponential in the length of the formula).

To avoid this, it is necessary to identify the states of simplified automata for sub-formulas with states (or classes of states) of the original automata. In order to make best use of the simplifications in $\mathbf{S}(\psi)$, it must be possible to switch between the states of $\mathbf{S}(\psi)$ and elements of $2^{\mathrm{sub}(\psi)} \times (k_\psi + 1)$ during the construction.

The formal treatment is as follows. For an LTL formula ψ, let

$$\mathbf{Q}(\psi) = (\mathbf{Q}_\psi, \Sigma, q_I^\psi, \Delta_\psi, F_\psi) \tag{2.96}$$

be a nondeterministic Büchi automaton for $L(\psi)$ constructed according to the bottom-up or top-down approach detailed above. We may assume that $\mathbf{Q}_\psi$ only contains states that are reachable from q_I^ψ via Δ_ψ. That is,

$$\mathbf{Q}_\psi \subseteq 2^{\mathrm{sub}(\psi)} \times (k_\psi + 1) \ . \tag{2.97}$$

Note that, for every state $(M,i) \in \mathbf{Q}_\psi$, the language of the automaton with initial state (M,i), in symbols $L(\mathbf{Q}(M,i))$, is determined by the elements of M, that is,

$$\forall (M,i) \in \mathbf{Q}_\psi \colon L(\mathbf{Q}(M,i)) = \bigcap_{\alpha \in M} L(\alpha) \ , \tag{2.98}$$

where $\bigcap_{\alpha \in \emptyset} L(\alpha) = (2^\Sigma)^\omega$.

The accepting states are

$$F_\psi = \mathbf{Q}_\psi \cap (2^{\mathrm{sub}(\psi)} \times \{0\}) \ . \tag{2.99}$$

We may assume that simplifications preserving equations (2.97) to (2.99), were already applied to $\mathbf{Q}(\psi)$. This is true for the simplifications of the algorithm of Subsection 2.6.1, excluding quotienting w. r. t. direct simulation.

We further assume that the simplified automaton

$$\mathbf{S}(\psi) = (\mathbf{S}_\psi, \Sigma, [q_I^\psi], \Delta_\psi^S, F_\psi^S) \tag{2.100}$$

is a quotient automaton of $\mathbf{Q}(\psi)$ w. r. t. an appropriate simulation relation $\equiv$. For example, $\equiv$ may be direct or delayed simulation equivalence.

That is, we have $\mathbf{S}_\psi \subseteq 2^{\mathbf{Q}_\psi}$, and we have

$$\text{for all } s, s' \in \mathbf{S}_\psi, \ s \cap s' = \emptyset \ , \tag{2.101}$$

$$\text{for all } s \in \mathbf{S}_\psi \text{ and for all } q \in s, \ \mathbf{S}(s) \equiv \mathbf{Q}(q) \ , \tag{2.102}$$

and also

$$\Delta_\psi^S \subseteq \{([q], t, [q']) \mid (q, t, q') \in \Delta_\psi\} \text{ and} \tag{2.103}$$

$$F_\psi^S = \{s \in \mathbf{S}_\psi \mid s \cap F_\psi \neq \emptyset\} \ . \tag{2.104}$$

We now choose, for every such state $s \subseteq 2^{\mathrm{sub}(\psi)} \times (k_\psi + 1)$, a representative $\mathrm{rep}(s) = (M_s, i_s) \in s$ such that

- $i_s = \min\{i \mid \exists M\colon (M, i) \in s\}$ (note that s is an accepting state if and only if this is the case), and
- M_s is an inclusion-minimal element of $\{M \mid (M, i_s) \in s\}$.

We further define a partial mapping $\mathrm{cls}\colon \mathbf{Q}_\psi \to \mathbf{S}_\psi$ such that $\mathrm{cls}(q) = s$ if and only if $q \in s$, and such that $\mathrm{cls}(q)$ is undefined if there is no such s (remember that the elements of $\mathbf{S}$ are pairwise disjoint). We thus have $\mathrm{cls}(\mathrm{rep}(s)) = s$ for every $s \in \mathbf{S}$, and $\mathrm{rep}(\mathrm{cls}(q)) \equiv q$ for every $q \in \mathbf{Q}$ such that $\mathrm{cls}(q)$ is defined.

We can now identify the states in $\mathbf{S}_\psi$ with their representatives, i. e., we continue the construction with the *representative automaton*

$$\mathbf{S}^{\mathrm{rep}}(\psi) = (\mathbf{S}_\psi^{\mathrm{rep}}, \Sigma, \mathrm{rep}([q_I^\psi]), \Delta_\psi^{\mathrm{rep}}, F_\psi^{\mathrm{rep}}) \ , \tag{2.105}$$

where

$$\mathbf{S}_\psi^{\mathrm{rep}} = \{\mathrm{rep}(s) \mid s \in \mathbf{S}_\psi\} \ , \tag{2.106}$$

$$\Delta_\psi^{\mathrm{rep}} = \{(\mathrm{rep}(s), t, \mathrm{rep}(s')) \mid (s, t, s') \in \Delta_\psi^S\} \ , \tag{2.107}$$

$$F_\psi^{\mathrm{rep}} = \{\mathrm{rep}(s) \in \mathbf{S}_\psi^{\mathrm{rep}} \mid s \in F_\psi^S\} \ . \tag{2.108}$$

Obviously, $\mathbf{S}^{\mathrm{rep}}(\psi) \equiv \mathbf{S}(\psi) \equiv \mathbf{Q}(\psi)$. A crucial property of the representative automaton and the reason for the constraints in the choice of the representatives is that sequences of states of the representative automaton are similar to runs of the bottom-up automaton w. r. t. the counter behavior:

Proposition 2.4 *Let $(M_j, i_j)_{j<\omega}$ be a run of a representative automaton $\mathbf{S}^{\mathrm{rep}}(\psi)$ as defined above. Then, for every $j < \omega$, $i_j \geq i_{j+1}$ or $i_j = 0$.*

The basic idea of this automaton construction is to use the states in $\mathbf{S}^{\mathrm{rep}}_{\psi}$ and the transitions in $\Delta^{\mathrm{rep}}_{\psi}$ as much as possible. We exemplify this for the Until operator; the generalization to other operators is straightforward.

Let $\varphi = \psi \mathbin{\mathsf{U}} \rho$, and assume that the automata $\mathbf{Q}(\psi)$ and $\mathbf{Q}(\rho)$ (together with the mappings z_ψ and z_ρ) and their simplified versions $\mathbf{S}(\psi)$ and $\mathbf{S}(\rho)$ as well as the representative automata $\mathbf{S}^{\mathrm{rep}}(\psi)$ and $\mathbf{S}^{\mathrm{rep}}(\rho)$ are already computed.

We now want to construct the nondeterministic Büchi automaton $\mathbf{Q}^S(\varphi)$ which makes use of the simplifications in $\mathbf{S}(\psi)$ and $\mathbf{S}(\rho)$. As with $\mathbf{Q}^{bu}(\varphi)$, the states $\mathbf{Q}^S_\varphi$ of $\mathbf{Q}^S(\varphi)$ are elements of $2^{\mathrm{sub}(\varphi)} \times (k_\varphi + 1)$, the initial state is $(\{\varphi\}, k_\varphi)$, and the set of accepting states is $\mathbf{Q}^S_\varphi \cap (2^{\mathrm{sub}(\varphi)} \times \{0\})$.

The set of transitions Δ^S_φ of $\mathbf{Q}^S(\varphi)$ results from Δ^{bu} by applying the following modifications. For simplicity in notation, we define the function

$$\mathrm{offset}_\psi \colon 2^{\mathrm{sub}(\rho)} \times (k_\rho + 1) \to 2^{\mathrm{sub}(\varphi)} \times (k_\varphi + 1) \tag{2.109}$$

by setting

$$\mathrm{offset}_\psi(M, i) = \begin{cases} (M, i + k_\psi), & \text{if } i \neq 0 \ , \\ (M, 0), & \text{if } i = 0 \ . \end{cases} \tag{2.110}$$

1. For the initial state $(\{\varphi\}, k_\varphi)$, delete all transitions starting from $(\{\varphi\}, k_\varphi)$ in Δ^{bu}. Instead, let

   $$((\{\varphi\}, k_\varphi), t, (\{\varphi\} \cup N_\psi, k_\varphi)) \in \Delta^S_\varphi \tag{2.111}$$

   for every transition $(\mathrm{rep}([q^\psi_I]), t, (N_\psi, j)) \in \Delta^{\mathrm{rep}}_\psi$, and

   $$((\{\varphi\}, k_\varphi), t, \mathrm{offset}_\psi(N_\rho, j)) \in \Delta^S_\varphi \tag{2.112}$$

   for every transition $(\mathrm{rep}([q^\rho_I]), t, (N_\rho, j)) \in \Delta^{\mathrm{rep}}_\rho$.

2. Now let (M, i) be a state such that $M \in 2^{\mathrm{sub}(\psi)}$.

   (a) If $\mathrm{cls}(M, i)$ is defined, delete all transitions starting from (M, i). Instead, let

   $$((M, i), t, (N, j)) \in \Delta^S_\varphi \tag{2.113}$$

   for every transition $(\mathrm{rep}(\mathrm{cls}(M, i)), t, (N, j)) \in \Delta^{\mathrm{rep}}_\psi$.

(b) If $\mathrm{cls}(M,i)$ is undefined, replace every transition $((M,i),t,(N,j)) \in \Delta^{bu}$ such that $\mathrm{cls}(N,j)$ is defined by a transition

$$((M,i),t,\mathrm{rep}(\mathrm{cls}(N,j))) \in \Delta^S_\varphi \ . \tag{2.114}$$

3. Analogously replace transitions $((M,i),t,(N,j))$ such that $M \in 2^{\mathrm{sub}(\rho)}$, and $\mathrm{cls}(\mathrm{offset}_\psi^{-1}(M,i))$ is defined (as in (2a)) or $\mathrm{cls}(\mathrm{offset}_\psi^{-1}(N,j))$ is defined (as in (2b)).

4. If (M,i) is a state such that $M \in 2^{\mathrm{sub}(\psi)\cup\mathrm{sub}(\rho)}$, delete all transitions starting from (M,i). Let $M = M_\psi \cup M_\rho$ with $M_\psi \in 2^{\mathrm{sub}(\psi)}$ and $M_\rho \in 2^{\mathrm{sub}(\rho)}$.

   (a) First assume that $i \le k_\psi$. If $((M_\psi,i),t,(N_\psi,j))$ is a transition in Δ^S_φ (case (2)) and $((M_\rho,0),t',(N_\rho,j')) \in \Delta^S_\varphi$ (case (3)), then

   $$((M,i),t\wedge t',(N_\psi\cup N_\rho,j)) \in \Delta^S_\varphi \ . \tag{2.115}$$

   (b) Now assume that $k_\psi < i \le k_\psi + k_\rho$. If $((M_\psi,0),t,(N_\psi,j))$ is a transition in Δ^S_φ (case (2)) and $((M_\rho,i),t',(N_\rho,j')) \in \Delta^S_\varphi$ where $j' > 0$ (case (3)), then

   $$((M,i),t\wedge t',(N_\psi\cup N_\rho,j')) \in \Delta^S_\varphi \ ; \tag{2.116}$$

   if $j' = 0$, then

   $$((M,i),t\wedge t',(N_\psi\cup N_\rho,j)) \in \Delta^S_\varphi \ . \tag{2.117}$$

5. Now assume that $M = M' \cup \{\varphi\}$ for some $M' \in 2^{\mathrm{sub}(\psi)\cup\mathrm{sub}(\rho)}$. Let $\hat{\imath} = 0$ if $i = k_\varphi$, else $\hat{\imath} = i$.

   For all $(t,(N,j)) \in \Delta^S_\varphi((\{\varphi\},k_\varphi))$ (by case (1)) and $(t',(N',j')) \in \Delta^S_\varphi((M',\hat{\imath}))$ (by case (4)), let

   $$\bar{j} = \begin{cases} \max\{\min\{i,j\},j'\} & \text{if } j>0 \text{ and } j'>0, \\ j & \text{if } j>0 \text{ and } j'=0, \\ j' & \text{if } j=0. \end{cases} \tag{2.118}$$

   Now let $L_\psi = (N\cup N')\cap\mathrm{sub}(\psi)$ and $L_\rho = (N\cup N')\cap\mathrm{sub}(\rho)$. Let $l_\psi = \bar{j}$ if $0 < \bar{j} \le k_\psi$, else $l_\psi = 0$, and let $l_\rho = \bar{j} - k_\psi$ if $k_\psi < \bar{j} \le k_\psi + k_\rho$, else $l_\rho = 0$.

   That is, we separate the state $(N\cup N',\bar{j})$ into the two states (L_ψ,l_ψ) and (L_ρ,l_ρ), which are states of $\mathbf{Q}(\psi)$ and $\mathbf{Q}(\rho)$, respectively. We now look for representatives for these two states, i. e., if $\mathrm{cls}(L_\psi,l_\psi)$ is defined, we

set $(N_\psi, j_\psi) = \mathrm{rep}(\mathrm{cls}(L_\psi, l_\psi))$, else $(N_\psi, j_\psi) = (L_\psi, l_\psi)$. We analogously define (N_ρ, j_ρ).

We then define

$$\hat{j} = \begin{cases} \bar{j} & \text{if } \bar{j} = 0 \text{ or } \bar{j} = k_\varphi, \\ l_\psi & \text{if } 0 < \bar{j} \leq k_\psi \text{ and } l_\psi > 0, \\ \mathrm{offset}_\psi(l_\rho) & \text{else,} \end{cases} \tag{2.119}$$

and we have

$$((M,i), t \wedge t', (N_\psi \cup N_\rho \cup \{\varphi\}, \hat{j})) \in \Delta_\varphi^S \ , \tag{2.120}$$

if $\varphi \in N$, or

$$((M,i), t \wedge t', (N_\psi \cup N_\rho, \hat{j})) \in \Delta_\varphi^S \ , \tag{2.121}$$

if $\varphi \notin N$.

It is not difficult (but tedious) to derive the construction for $\mathsf{G}\rho$ and $\psi \,\mathsf{R}\, \rho$ from this description. The correctness of this construction, i. e., that $L(\mathbf{Q}^S(\varphi)) = L(\varphi)$, follows from the correctness of the bottom-up construction and of the simplified automata.

2.9 Conclusion of Chapter 2

We have extended simulation relations for alternating Büchi automata to relations for comparing LTL formulas. We have developed an algorithm for the translation of LTL formulas to equivalent nondeterministic Büchi automata; this algorithm uses simulation relations for the simplification of the input LTL formula as well as for an on-the-fly simplification of the resulting automaton. Experimental data indicates that our approach is a good compromise between consumption of computing resources and quality of result. We show by a detailed analysis that our basic concept of translating LTL via alternating automata to nondeterministic automata (simulation-based simplifications left aside) is not inferior to the tableau-based translation of LTL to automata.

Chapter 3

Simulation Relations and the Parity Acceptance Condition

We have seen in Chapter 1 how to define (in a game-based approach) simulation relations and simulation-based quotienting for alternating Büchi automata. In this chapter, we expand these considerations to the *parity* acceptance mode of ω-automata. In a parity automaton [Mos84, EJ91], the acceptance condition is given as a function mapping every state to a *priority*, i. e., a natural number. A run of a parity automaton is accepting if the minimal priority visited infinitely often is even. (Some authors favor the definition that the maximal priority must be even.) Obviously, the parity acceptance condition can be seen as a generalization of the Büchi acceptance condition: A Büchi acceptance condition is a parity condition with priorities 0 and 1 only. (It is well-known that the class of languages recognizable by parity automata is the same as the class recognizable by Büchi automata.) From this point of view, many considerations in this chapter can be seen as enhancements of the ideas of Chapter 1.

A lot of scientific attention has been spent on *parity games*. A parity game is a two-player-one-pebble game on a finite board for which the winning condition is given as a parity condition, i. e., one of the players wins a play of the parity game if the smallest priority of the positions visited infinitely often during the play is even, and the other player wins if it is odd. The problem of deciding the winner of a parity game is intriguing: It is known to be in $\mathbf{UP} \cap \mathbf{co\text{-}UP}$ [Jur98], that is, it can be solved by a so-called unambigious Turing machine in polynomial time, cf. [Pap94]; the best known algorithms for solving parity games work in time exponential in the number of priorities occurring in the input parity game [Jur00]. The problem is also important, because the model checking problem and the satisfiability problem for the modal μ-calculus [Koz83] can be reduced to the problem of deciding the winner of a parity game, see, e. g., the survey [Wil01] and the book [GTW02]. Also note that Etessami et al. [ESW01] reduce the problem of

computing the fair simulation relation for a Büchi automaton to the problem of deciding the winner of a parity game with priorities 0, 1, and 2 only; we discuss the application of the resulting algorithm in Section 1.7.

In this chapter, we discuss delayed simulation for parity games and, more general, alternating parity automata. We introduce a delayed simulation relation for alternating parity automata in Section 3.2. We show in Section 3.3 that this relation can be computed in polynomial time and can be used to compare parity games w. r. t. the win of one of the players and parity automata w. r. t. language inclusion. In Section 3.4, we introduce two dual restricted versions of this relation and show that these restricted relations can be used for state space reduction by merging of equivalent states (*quotienting*) and deletion of transitions. From this, we develop in Section 3.5 a simulation-based simplification algorithm for alternating parity automata. In Section 3.6, we a give a sketch of how to apply these insights to the simplification of formulas of a fragment of the modal μ-calculus, via a translation to alternating parity tree automata. We do not discuss direct or fair simulation for parity automata. Following the basic concepts developed in this chapter, defining direct simulation is relatively straightforward; Janin [Jan01] introduces synchronous and asynchronous direct simulation for simplifying parity games. A notion of fair simulation for parity automata seems to be impractical.

3.1 Basic Definitions: Alternating Parity Automata and Parity Games

An alternating parity automaton (APA, for short) $\mathbf{Q}$ is a tuple

$$(Q, \Sigma, q_I, \Delta, E, U, \Omega) \tag{3.1}$$

where Q is a finite, non-empty set of states, Σ, as usual, is an alphabet with typical elements a, b, c ..., $q_I \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $\{U, E\}$ is a partition of Q, where $U = \emptyset$ or $E = \emptyset$ is allowed, and $\Omega\colon Q \to \omega$ is the *priority function*. That is, an alternating parity automaton has the same structure as the alternating Büchi automaton defined in Subsection 1.1.2, only the acceptance condition is different. Without loss of generality, we assume in this chapter that Δ is *complete*, i. e., for every $q \in Q$, $a \in \Sigma$, there is a $q' \in Q$ such that $(q, a, q') \in \Delta$.

As in the case of alternating Büchi automata, the language $L(\mathbf{Q})$ of an APA $\mathbf{Q}$ can be defined via a word game, as in Subsection 1.1.2. For technical convenience, we will introduce a slight modification: For an ω-word w, let $\mathrm{suf}(w) = \{w[n..] \mid n < \omega\}$ be the set of suffixes of w. Note that $\mathrm{suf}(w)$ is finite if and only if w is *eventually periodic*, i. e., if $w = uv^\omega$ for some words $u, v \in \Sigma^+$. The word game of

an APA $\mathbf{Q}$ and an ω-word w is the game

$$G(\mathbf{Q},w) = (P,P_0,P_1,p_I,Z,W) \ , \tag{3.2}$$

where $P = Q \times \mathrm{suf}(w)$ is the set of positions, $P_0 = U \times \mathrm{suf}(w)$ and $P_1 = E \times \mathrm{suf}(w)$ are the sets of positions of Player 0 (Pathfinder) and Player 1 (Automaton), respectively, $p_I = (q_I,w)$ is the initial position and $Z = \{((q,w[i..]),(q',w[i+1..])) \mid (q,w(i),q') \in \Delta\}$ is the set of moves. The winning set is

$$W = \{(q_n,w[n..])_{n<\omega} \in P^\omega \mid \min\{\Omega(q) \mid q \in \mathrm{Inf}((q_n)_{n<\omega})\} \equiv 0 \bmod 2\} \ , \tag{3.3}$$

that is, Automaton wins a play of $G(\mathbf{Q},w)$ if the smallest priority of the states visited infinitely often during the play is even. We call this acceptance mode *parity acceptance*.

As usual, a word w is accepted by $\mathbf{Q}$ if Automaton wins the game $G(\mathbf{Q},w)$, i. e., if he has a winning strategy for $G(\mathbf{Q},w)$, and the language of $\mathbf{Q}$ is

$$L(\mathbf{Q}) = \{w \in \Sigma^\omega \mid \text{Automaton wins the game } G(\mathbf{Q},w)\} \ . \tag{3.4}$$

Note that an alternating Büchi automaton $(Q,\Sigma,q_I,\Delta,U,E,F)$ can thus be regarded as an alternating parity automaton $(Q,\Sigma,q_I,\Delta,E,U,\Omega)$ with $\Omega(q) = 0$ if $q \in F$, else $\Omega(q) = 1$.

We will use the term *parity game* for an APA $\mathbf{Q}$ such that $|\Sigma| = 1$. There is a natural isomorphism between a parity game and its word game: If $|\Sigma| = 1$, then $|\Sigma^\omega| = 1$ and $|\mathrm{suf}(w)| = 1$ for $w \in \Sigma^\omega$. We will identify the two and say *Player 1 wins the parity game* $\mathbf{Q}$ if $L(\mathbf{Q}) \neq \emptyset$.

We will use the following ordering of natural numbers (also used in, e. g., [JV00]): The *reward order* $\leq_\Omega \subseteq \omega \times \omega$ is defined by $m \leq_\Omega n$ if and only if

- m is even and n is odd, or
- m and n are even and $m \leq n$, or
- m and n are odd and $n \leq m$

for all $m,n \in \omega$. That is, $0 <_\Omega 2 <_\Omega 4 <_\Omega \ldots <_\Omega 5 <_\Omega 3 <_\Omega 1$. We will also phrase $n <_\Omega m$ as *n is better than m*, while terms like *minimum* and *smaller than* will always be used w. r. t. the standard order $\leq$.

We say that an APA is *normalized* if, for every SCC of its transition graph, the set of priorities occurring in that SCC is of the form $\{0,\ldots,n\}$ or $\{1,\ldots,n\}$ for some $n < \omega$. It is easy to see that there is a normalized APA for every APA such that the two accept the same language and have the same transition structure. A normalized APA can be computed from a given APA as follows: An APA is not normalized if and only if there is an SCC in its transition graph such that $\Omega(q) = m \geq 2$ for a state q in the SCC, but there is no q' in the SCC such that $\Omega(q') = m-1$. Then change the priority of q to $m-2$. Repeat this until the APA is normalized.

3.2 A Delayed Simulation Relation for the Parity Condition

In this section, we introduce a delayed simulation relation for alternating parity automata, based on the basic game of Section 1.2. The winning condition will be more complicated as in Chapter 1, yet still allow to compute the winner in polynomial time.

Recall the winning condition of the delayed simulation game of Subsection 1.2.1: A play $(q_n, s_n)_{n<\omega}$ of a delayed simulation game $G^{de}(\mathbf{Q}, \mathbf{S})$ is a win for Duplicator if for every n with $q_n \in F^q$ there is an $m \geq n$ such that $s_m \in F^s$. An alternating Büchi automaton can be regarded as an alternating parity automaton with two priorities, priority 0 for accepting states and priority 1 for non-accepting states. In this spirit, we may assume that, whenever p_n is accepting (priority 0) but q_n is not (priority 1), the smaller of the two values is stored as a flag to indicate that Spoiler will win the play unless there is an $m > n$ such that q_m has priority 0, i. e., a priority that is less than or equal to the flag value and even, in which case the flag (or *priority memory*) is erased.

In other words, whenever at the end of a round the priority of the pebble in the simulated automaton is better than the simulating pebble's priority, we store the smaller of the two priorities in a priority memory. This value serves as a threshold for the priorities encountered by the two pebbles: It is lowered if the simulated pebble's priority is even and below the threshold, or if the simulating pebble's priority is odd and below. It is erased if the simulated pebble's priority is odd and below or equal to the threshold, or if the simulating pebble's priority is even and below or equal to the threshold.

Assume that the priorities of q_n and s_n at the end of round n are i and j, respectively, and that the priority memory value of the former round is k_{n-1}. We assume that the memory takes on the special value $\surd$ when erased.

The updated memory value k_n at the end of round n is then computed as follows:

– If $i <_\Omega j$ and $k_{n-1} = \surd$, then k_n is the minimum of i and j. If $i <_\Omega j$, but $k_{n-1} \neq \surd$, the priority memory takes on this new value only if it is smaller than k_{n-1}.

– If $j \leq_\Omega i$ and $k_{n-1} = \surd$, the new flag is again $\surd$.

– The flag value is erased (switches from $k_{n-1} \neq \surd$ to $k_n = \surd$) if $j \leq_\Omega i$, j is even and $j \leq k_{n-1}$.

– It is also erased if $j \leq_\Omega i$, i is odd and $i \leq k_{n-1}$.

– In all other cases, we have $k_n = k_{n-1}$.

Note that if the alternating parity automaton only has two priorities 0 and 1, i. e., if it is an alternating Büchi automaton, then these are just the rules for the

delayed simulation game for ABA. In this case, the only possible memory values are 0 and $\surd$, in analogy to the notion of a winning bit in Chapter 1.

3.2.1 A formal definition of the simulation game

In this subsection, we formally define the delayed simulation relation for the parity condition outlined above, again using the game-based approach of Sections 1.2 and 2.3.

Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta^q, E^q, U^q, \Omega^q)$ and $\mathbf{S} = (S, \Sigma, s_I, \Delta^s, E^s, U^s, \Omega^s)$ be alternating parity automata.

The *delayed simulation game* $G^{de}(q_I, s_I) = (P, P_0, P_1, p_I, Z)$ is defined as follows:

- The basic positions are $P_B = Q \times S \times (\Omega^q(Q) \cup \Omega^s(S) \cup \{\surd\})$. The positions P of $G^{de}(q_I, s_I)$ are $P_B \cup P_B \times \Sigma \times \{sp, du\} \times \{0,1\} \cup P_B \times \Sigma \times (\{sp, du\} \times \{0,1\})^2$. The P_B-part represents the basic situation of a play (i. e., the positions of the simulated and simulating pebble) and the priority memory, and other control components are used to determine who of the players must move which pebble along which transitions. To this end, we define the function $c\colon Q \times S \to (\{sp, du\} \times \{0,1\})^2$ mapping every pair of positions to the appropriate control components. That is,
  - $c(q,s) = (sp, 0, du, 1)$ if $(q,s) \in E^q \times E^s$,
  - $c(q,s) = (sp, 0, sp, 1)$ if $(q,s) \in E^q \times U^s$,
  - $c(q,s) = (du, 0, du, 1)$ if $(q,s) \in U^q \times E^s$,
  - $c(q,s) = (sp, 1, du, 0)$ if $(q,s) \in U^q \times U^s$.

  For convenience, we extend the domain of c to P_B by defining $c((q,s,k)) = c(q,s)$.
- If $\Omega^q(q_I) <_\Omega \Omega^s(s_I)$, then $p_I = (q_I, s_I, \min\{\Omega^q(q_I), \Omega^s(s_I)\})$, else $p_I = (q_I, s_I, \surd)$.
- The set $P_0 \subseteq P$ are those positions where Spoiler must move. It contains all positions with sp as fifth component and the positions in P_B. As usual, Duplicator's positions P_1 are $P \setminus P_0$.
- To define the set of moves, we first define the function

$$\mathrm{pm}\colon \omega^2 \times (\omega \cup \{\surd\}) \to \omega \cup \{\surd\} \tag{3.5}$$

  as follows:

1. $\mathrm{pm}(i,j,\surd) = \min\{i,j\}$ if $i <_\Omega j$,
2. $\mathrm{pm}(i,j,\surd) = \surd$ if $j \leq_\Omega i$,
3. $\mathrm{pm}(i,j,k) = \min\{i,j,k\}$ if $i <_\Omega j$,
4. $\mathrm{pm}(i,j,k) = \surd$ if $j \leq_\Omega i$, i is odd and $i \leq k$, and j is odd or $k < j$,
5. $\mathrm{pm}(i,j,k) = \surd$ if $j \leq_\Omega i$, j is even and $j \leq k$, and i is even or $k < i$; we want the cases 4 and 5 to be disjoint for the considerations of Section 3.4,
6. $\mathrm{pm}(i,j,k) = \surd$ if i is odd, j is even, and both $i \leq k$ and $j \leq k$,
7. else $\mathrm{pm}(i,j,k) = k$. Note that in this case, both i and j must be strictly larger than k.[1]

We extend this function to P_B by defining $\mathrm{pm}(q,s,k) = \mathrm{pm}(\Omega^q(q), \Omega^s(s), k)$

The set of moves $Z \subseteq P \times P$ then contains the following pairs:

- $((q,s,k),(q,s,k,a,c(q,s)))$, for all $a \in \Sigma$,
- $((q,s,k,a,c(q,s)),(q',s,k,a,\mathrm{pr}_{3,4}(c(q,s)),$ for $\mathrm{pr}_2(c(q,s)) = 0$ and $(q,a,q') \in \Delta^q$,
- $((q,s,k,a,c(q,s)),(q,s',k,a,\mathrm{pr}_{3,4}(c(q,s)),$ for $\mathrm{pr}_2(c(q,s)) = 1$ and $(s,a,s') \in \Delta^s$,
- $((q,s,k,a,x,0),(q',s,\mathrm{pm}(q',s,k)))$, for $x \in \{sp,du\}$ and $(q,a,q') \in \Delta^q$,
- $((q,s,k,a,x,1),(q,s',\mathrm{pm}(q,s',k)))$, for $x \in \{sp,du\}$ and $(s,a,s') \in \Delta^s$,

A play $\pi = (p_i)_{i<\omega}$ of the delayed simulation game is a win for Duplicator iff there are infinitely many $i < \omega$ such that $\mathrm{pr}_3(p_i) = \surd$. That is, the delayed simulation game has a Büchi winning condition and can thus be solved using the approach of [ESW01], see also Section 1.7, and see Subsection 3.3.2 for the details here.

As usual, if Duplicator wins $G^{de}(\mathbf{Q},\mathbf{S})$, we write $\mathbf{Q} \leq_{de} \mathbf{S}$ and say "$\mathbf{S}$ *de*-simulates $\mathbf{Q}$".

3.2.2 The relation $\leq_{de}$ is a preorder

Obviously, $\leq_{de}$ is reflexive. The proof of the transitivity is quite involved; it is given in this subsection.

The following proposition follows directly from the definition of the simulation game.

[1] This implies that, equivalently, we can delete case (3) and rewrite case (7) as *else* $\mathrm{pm}(i,j,k) = \min\{i,j,k\}$.

Proposition 3.1 *Let $\pi = (q_i, s_i, k_i)_{i<\omega}$ be a play of a delayed simulation game for alternating parity automata $G^{de}(q_0, s_0)$. Let $0 \leq n < m \leq \omega$ be such that $k_{n-1} = \surd$ if $n > 0$, and $k_m = \surd$ if $m < \omega$. For all i such that $n \leq i < m$, let $k_i \neq \surd$.*

Then, $(k_i)_{n \leq i < m}$ is a monotonic decreasing sequence.

The following lemma states an important property of plays of simulation games for APA.

Lemma 3.1 *Let π, n and m be defined like in Proposition 3.1.*

Then,

1. *for every i such that $n \leq i < m$,*

$$k_i = \min\{\Omega(p) \mid p \in \{q_l, s_l \mid n \leq l \leq i\}\} \ . \tag{3.6}$$

If $m < \omega$, then

2. *$\Omega(q_m)$ is odd and*

$$\Omega(q_m) = \min\{\Omega(p) \mid p \in \{q_i, s_i \mid n \leq i \leq m\}\} \ , \tag{3.7}$$

or

3. *$\Omega(s_m)$ is even and*

$$\Omega(s_m) = \min\{\Omega(p) \mid p \in \{q_i, s_i \mid n \leq i \leq m\}\} \ . \tag{3.8}$$

Proof. Note that $k_i = \mathrm{pm}(q_i, s_i, k_{i-1})$, for $n < i \leq m$, and $k_n = \min\{\Omega(q_n), \Omega(s_n)\}$.

If we assume that $m < \omega$, it follows directly by the definition of the simulation game that $\Omega(q_m)$ is odd or $\Omega(s_m)$ is even: By construction, $k_m = \surd = \mathrm{pm}(q_m, s_m, k_{m-1})$ with $k_{m-1} \neq \surd$, so one of the cases 4 or 5 in the definition of pm applies if $n > 0$.

We will first show inductively that $k_i = \min\{\Omega(p) \mid p \in \{q_l, s_l \mid n \leq l \leq i\}\}$, for every $n \leq i < m$. This is true by definition for $i = n$. Now let $n < i < m$, and assume that $k_{i-1} = \min\{\Omega(p) \mid p \in \{q_l, s_l \mid n \leq l < i\}\}$. Since $k_i \neq \surd$, either $k_i = \min\{\Omega(q_i), \Omega(s_i), k_{i-1}\}$ (if $\Omega(q_i) <_\Omega \Omega(s_i)$) and we are done.

Or $k_i = k_{i-1}$ (case 7 in the definition of pm). In this case, $k_{i-1} < \Omega(q_i)$ and $k_{i-1} < \Omega(s_i)$. Hence again $k_i = \min\{\Omega(q_i), \Omega(s_i), k_{i-1}\}$.

This shows claim (1). The other two claims of the lemma now follow from claim (1) because $k_m = \surd = \mathrm{pm}(q_m, s_m, k_{m-1})$ by application of one of the cases 4 or 5 in the definition of pm. □

The following is a direct consequence of Lemma 3.1.

Corollary 3.1 *Let π, n and m be defined like in Proposition 3.1. Let $m < \omega$.*

1. *If $\Omega(q_m)$ is odd and $\Omega(q_m) = \min\{\Omega(p) \mid p \in \{q_i, s_i \mid n \leq i \leq m\}\}$ then $\Omega(q_m) < \Omega(q_i)$ for all $n \leq i < m$.*
2. *If $\Omega(s_m)$ is even and $\Omega(s_m) = \min\{\Omega(p) \mid p \in \{q_i, s_i \mid n \leq i \leq m\}\}$ then $\Omega(s_m) < \Omega(s_i)$ for all $n \leq i < m$.*

Proof. To proof the first claim, suppose that $\Omega(q_m) = \Omega(q_i)$ for an i such that $n \leq i < m$ (from Lemma 3.1, we know that $\Omega(q_m) \leq \Omega(q_i)$). Since $\Omega(q_m) \leq k_{i-1}$ by the above lemma and $\Omega(q_m)$ is odd, we have $\Omega(q_i) <_\Omega \Omega(s_i)$ (else $k_i = \surd$ by the definition of the game; the same conclusion is valid for $i = n = 0$). That is, $\Omega(s_i)$ is odd and $\Omega(s_i) < \Omega(q_i) = \Omega(q_m)$, in contradiction to the minimality of $\Omega(q_m)$ according to Lemma 3.1 (2).

The proof of the second claim is symmetrical. □

Given two delayed simulation games for APA $G^{de}(\mathbf{Q},\mathbf{R})$ and $G^{de}(\mathbf{R},\mathbf{S})$, the join of two Duplicator strategies σ^0, σ^1 for $G^{de}(\mathbf{Q},\mathbf{R})$ and $G^{de}(\mathbf{R},\mathbf{S})$, respectively, can be defined in a straightforward manner, similar to what is done in Section 1.3.

Using this notion, we will use Proposition 3.1, Corollary 3.1 and Lemma 3.1 to proof the following main theorem of this section, Theorem 3.1.

Theorem 3.1 *Let $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$ be alternating parity automata such that $\mathbf{Q} \leq_{de} \mathbf{R}$ and $\mathbf{R} \leq_{de} \mathbf{S}$. Let σ^0 and σ^1 be Duplicator winning strategies for $G^{de}(\mathbf{Q},\mathbf{R})$ and $G^{de}(\mathbf{R},\mathbf{S})$, respectively.*

Then, $\sigma^0 \bowtie \sigma^1$ is a Duplicator winning strategy for $G^{de}(\mathbf{Q},\mathbf{S})$.

Proof. Let $T = ((q_i, s_i, k_i)_{i<\omega}, w)$ be a $(\sigma^0 \bowtie \sigma^1)$-conform protoplay of $G^{de}(\mathbf{Q},\mathbf{S})$ with intermediate sequence $(r_i)_{i<\omega}$ (cf. Section 1.3). Note that the protoplays $T' = ((q_i, r_i, k'_i)_{i<\omega}, w)$ and $T'' = ((r_i, s_i, k''_i)_{i<\omega}, w)$ are σ^0- and σ^1-conform, respectively.

We have to show that, whenever there is an n such that $k_n \neq \surd$, there is an $m > n$ such that $k_m = \surd$.

We therefore introduce a *progress measure* $f_T : \omega \to \omega$ such that $f_T(i) = -1$ if and only if $k_i = \surd$, and else $f_T(i) \geq 0$. We call a round i of T *initial* if and only if $k_i \neq \surd$ and either $i = 0$ or $k_{i-1} = \surd$. We show that, for every initial round n, there is a finite sequence $n = n_0, n_1, \ldots, n_l$ of rounds such that $f_T(n_i) > f_T(n_{i+1})$ (for $i < l$) and $f_T(n_l) = -1$. As a necessary invariant, we will also show that at least one of k'_i and k''_i is less than or equal to k_i, for $i < l$. We will call a turn *active* if and only if it satisfies this invariant.

Our progress measure is

$$f_T(i) = \textbf{if } k_i \neq \surd \textbf{ then } k_i \cdot (\max(\mathbf{R}) + 1) + \Omega^r(r_i) \textbf{ else } -1 \ , \qquad (3.9)$$

where $\max(\mathbf{R}) = \max\{\Omega^r(r) \mid r \in R\}$ is the maximal priority occurring in $\mathbf{R}$.

For ease of notation, in the following we identify game positions with their priorities. Especially, we will extend the reward order $\leq_\Omega$ and the normal ordering of integers $\leq$ to game positions and simply write, e. g., $q_0 \leq_\Omega s_0$ instead of $\Omega^q(q_0) \leq_\Omega \Omega^s(s_0)$.

To start our argumentation, we will at first show that initial rounds are active. Assume that round n of T is an initial round. Hence $q_n <_\Omega s_n$, i. e., either (1) $r_n \leq_\Omega q_n <_\Omega s_n$, or (2) $q_n <_\Omega r_n <_\Omega s_n$, or (3) $q_n <_\Omega s_n \leq_\Omega r_n$. Note that in a $<_\Omega$-ordered chain $n_0 <_\Omega n_1 <_\Omega \ldots <_\Omega n_i$, either n_0 or n_1 are the minimal elements w. r. t. $<$. Hence in case (1), either r_n is minimal (and $k''_n \leq k_n$ by Lemma 3.1(1)) or s_n is minimal (and again $k''_n \leq k_n$). Similar considerations show that the initial round is also active in cases (2) and (3).

We now assume that $n < \omega$ is an active round such that $f_T(n) \neq -1$, i. e., $k_n \neq \surd$. We will show that there is an active round $m > n$ such that $f_T(m) = -1$, or m is active and $f_T(m) < f_T(n)$. It follows that there is an active round $m' > n$ such that $f_T(m') = -1$, i. e., $k_{m'} = \surd$, and this proves the theorem.

We distinguish several cases.

1. $k_n = 0$, i. e., $f_T(n) \leq z$. Since n is an active round, at least one of k'_n, k''_n is less than or equal to k_n, i. e., equal to 0 in this case. Assume that $k'_n = 0$. Since T' is σ^0-conform, there is an $m \geq n$ such that $\Omega^r(r_m) = 0$. Thus $k_m = \surd$, or $k''_m = 0$ (Start your argumentation here if not k'_n but k''_n is equal to 0; substitute n for m.) and there is an $l > m$ such that $\Omega^s(s_l) = 0$ since T'' is σ^1-conform. Consequently, $k_l = \surd$.

For the other cases, we assume that n is an active round and $k_n > 0$. We distinct two main cases: (2) $k''_n \leq k_n$ and (3) $k'_n \leq k_n$. If both k''_n and k'_n are not equal to $\surd$, we choose case (2) if $k''_n \leq k'_n$, else case (3). Each main case is divided into several subcases based on the comparison of q_n, r_n and s_n w. r. t. $\leq_\Omega$.

2. $k_n \neq \surd \neq k''_n$ and $k''_n \leq k_n$. In this case, there is a minimal $m > n$ such that $k''_m = \surd$ and $s_m \leq_\Omega r_m$. We assume that there is no $n < i \leq m$ such that $k_i = \surd$ (in that case, we are done, of course).

   (a) First, let $q_m <_\Omega s_m \leq_\Omega r_m$, and assume that case 3 of Lemma 3.1 and case 2 of Corollary 3.1 apply to s_m in T''. (In the following, we will simply write, e. g., *s_m has triggered $k''_m = \surd$* to describe this case.) Let $l = \max\{i \mid q_i = \min\{q_j \mid n \leq j \leq m\}\}$. We have $s_m < s_i$ for all $n \leq i < m$, and $q_m < s_m$ since $q_m <_\Omega s_m$ and s_m is even. Hence $k_m = q_l < s_m \leq k''_n \leq k_n$, i. e., $f_T(m) < f_T(n)$. And $k'_m \leq q_l = k_m$ since $q_l < s_m \leq r_i$ for all $n \leq i \leq m$, i. e., round m is active. (We have tacitly used Lemma 3.1(1) at several points of our argumentation. We will continue doing so.)

(b) Again, let $q_m <_\Omega s_m \leq_\Omega r_m$, but assume that r_m has triggered $k'' = \surd$. That is, $r_m < r_n$ (hence $f_T(m) < f_T(n)$). By Lemma 3.1, $k'_m \leq r_m \leq s_i$ for all $n \leq i \leq m$ and also $k'_m \leq r_m \leq k''_n \leq k_n$. That is, if $k_m < k'_m \leq k_n$ then there is an l such that $n < l \leq m$ and $q_l = k_m < k'_m \leq p_m \leq p_i$ for all $n \leq i \leq m$. But $\Omega^q(q_l)$ must be even (else $k_l = \surd$), hence $k'_i \neq \surd$ for $l \leq i \leq m$ and consequently $k'_m \leq q_l$ (contradiction). We conclude that $k'_m \leq k_m$, i. e., round m is active.

(c) Now let $s_m \leq_\Omega q_m \leq_\Omega r_m$, and assume that s_m has triggered $k'' = \surd$. Since we assume that $k_i \neq \surd$ for $n < i \leq m$, we know that $k_m < s_m$ and $k_m < q_m$. Especially, $k_m < k_n$ since $s_m < k''_n \leq k_n$, and hence $f_T(m) < f_T(n)$. By Lemma 3.1, we know that $k_m < s_m \leq s_i$ for $n \leq i \leq m$. Hence $k_m = q_l$ for an l such that $n < l \leq m$. As in case 2a, we conclude that $k'_m \leq q_l = k_m$.

(d) Let $s_m \leq_\Omega q_m \leq_\Omega r_m$, and assume that r_m has triggered $k'' = \surd$. Since $k_m \leq k_n$ and $r_m < r_n$, we have $f_T(m) < f_T(n)$. As above, we can conclude that $k_m < s_m$ and $k_m < q_m$. We know that $r_m \leq k''_n \leq k_n$ and $r_m \leq s_i$ for all $n \leq i \leq m$, i. e., if $k_m < r_m$ then there is an l such that $n < l \leq m$ and $q_l = k_m < r_m$ is even. Hence $k'_m \leq k_m$ (cf. case 2a). Or $k_m \geq r_m$, and then $k'_m \leq r_m \leq k_m$ follows, because $k'_m = \surd$ implies $\Omega^q(q_m) = \Omega^r(r_m) \leq k_m$, and $\Omega^r(r_m)$ is odd and $r_m \leq s_m$. And this implies $k_m = \surd$.

(e) The case that $s_m \leq_\Omega r_m <_\Omega q_m$ and that s_m has triggered $k'' = \surd$ is completely similar to case 2c.

(f) Finally, we turn to the case $s_m \leq_\Omega r_m <_\Omega q_m$ where r_m has triggered $k'' = \surd$. Since $s_m <_\Omega q_m$ but $k_m \neq \surd$, we have $k_m < s_m$ and $k_m < q_m$. Now $\Omega^r(r_m)$ is odd and $r_m <_\Omega q_m$, hence $\Omega^q(q_m)$ is odd and $q_m < r_m$, and also $r_m \leq s_i$ for $n \leq i \leq m$ and $r_m < k'_n \leq k_n$. That is, using Lemma 3.1, it is not possible that $k_m = s_l$ for $n \leq l \leq m$. Hence $k_m = q_l$ for an l such that $n \leq l \leq m$, and q_l is even and $q_l < q_m$. Consequently, $k'_m \leq q_l = k_m < k_n$, i. e., round m is active and $f_T(m) < f_T(n)$.
This concludes our first subcase distinction.

3. The second main case is $k_n \neq \surd \neq k'_n$ and $k'_n \leq k_n$. In this case, there is a minimal $m > n$ such that $k'_m = \surd$ and $r_m \leq_\Omega q_m$. Again assume that there is no $n < i \leq m$ such that $k_i = \surd$.

(a) The first subcase is $s_m <_\Omega r_m \leq_\Omega q_m$, where r_m has triggered $k'_m = \surd$. Note that, in this main case, *r_m has triggered $k'_m = \surd$* implies that $\Omega^r(r_m)$ is even. This case is similar to case 2f if we swap k' with k'', q with s and *odd* with *even*. That is, we have $k_m < q_m$, $k_m < s_m$, we

conclude that $s_m < r_m$ and $r_m \leq q_i$ for $n \leq i \leq m$, we then have $k_m = s_l$ for an l such that $n \leq l \leq m$ and arrive at $k''_m \leq s_l = k_m < r_m < k'_n \leq k_n$.

(b) Now let $s_m <_\Omega r_m \leq_\Omega q_m$, but assume that q_m has triggered $k'_m = \surd$. This case is similar to case 2e if you swap k' with k'' and q with s.

(c) Let $r_m \leq_\Omega s_m \leq_\Omega q_m$ and assume that r_m has triggered $k'_m = \surd$. This case is similar to case 2d if you swap k' with k'', q with s and *odd* with *even*.

(d) Again, let $r_m \leq_\Omega s_m \leq_\Omega q_m$, but assume that q_m has triggered $k'_m = \surd$. This case is similar to case 2c if you swap k' with k'' and q with s.

(e) We now assume that $r_m \leq_\Omega q_m <_\Omega s_m$ and that r_m has triggered $k'_m = \surd$. This case is similar to case 2b if you swap k' with k'', q with s and *odd* with *even*.

(f) Finally, we have $r_m \leq_\Omega q_m <_\Omega s_m$ where q_m has triggered $k'_m = \surd$. This case is similar to case 2a if you swap k' with k'', q with s and *odd* with *even*.

As stated above, it follows that $\leq_{de}$ is transitive. □

Since the relation $\leq_{de}$ also is reflexive, it is a preorder.

Corollary 3.2 *The relation $\leq_{de}$ is a preorder.*

3.3 Properties of Delayed Simulation for the Parity Condition

In this section, we show that the delayed simulation preorder defined in Section 3.2 implies language containment in the sense that if an APA **Q** is simulated by an APA **S**, i. e., $\mathbf{Q} \leq_{de} \mathbf{S}$, then the language accepted by **Q** is a subset of the language accepted by **S**.

We further show that the delayed simulation relation for APA can be efficiently computed (Subsection 3.3.2), and that the relation is compatible with the dualization of parity automata (Subsection 3.3.3). But, in Subsection 3.3.4, we see that our version of delayed simulation seems to be ill-suited for automata minimization via quotienting, as can be done for alternating Büchi automata (cf. Section 1.5).

In Section 3.4, we discuss a way to alleviate this problem.

3.3.1 Delayed simulation implies language containment

As in the case of delayed simulation for alternating Büchi automata, delayed simulation for alternating parity automata implies language containment between the simulated and the simulating automaton. The basic idea of the proof is similar to the proof of Theorem 1.1, so we concentrate on the special aspects of parity acceptance.

Theorem 3.2 *Let* $\mathbf{Q}$, $\mathbf{S}$ *be alternating parity automata over an alphabet* Σ *such that* $\mathbf{Q} \leq_{de} \mathbf{S}$. *Then,* $L(\mathbf{Q}) \subseteq L(\mathbf{S})$.

Proof. The basic idea is the same as in the proof of Theorem 1.1. We fix a word $w \in L(\mathbf{Q})$. Given an arbitrary strategy τ^1 of Pathfinder for $G(\mathbf{S}, w)$, we choose an Automaton winning strategy σ^0 for $G(\mathbf{Q}, w)$ and a Duplicator winning strategy σ for the simulation game $G^{de}(\mathbf{Q}, \mathbf{S})$.

We then look at the simulation game where Spoiler moves the $\mathbf{Q}$-pebble according to σ^0 and the $\mathbf{S}$-pebble according to τ^1.

Duplicator uses σ, and we have to show that the projection of the simulation game to the $\mathbf{S}$-pebble is a win for Automaton in the word game $G(\mathbf{S}, w)$; see Subsection 1.2.2 for the technical details.

Now let π be a protoplay of this simulation game, that is, $\pi = ((q_i, s_i)_{i<\omega}, w) \in (Q \times S)^\omega \times \Sigma^\omega$.

Since the projection of π to the word game $\pi_0 = ((q_i)_{i<\omega}, w)$ of $\mathbf{Q}$ is a win for Automaton (π_0 is σ^0-conform), finally every odd priority that the $\mathbf{Q}$-pebble encounters is succeeded by a smaller even priority. That is, there is a $j < \omega$ such that for all $k \geq j$, if $\Omega(q_k)$ is odd, there is an $l > k$ such that $\Omega(q_l) < \Omega(q_k)$ and $\Omega(q_l)$ is even.

Now suppose that $\min(\mathrm{Inf}((s_i)_{i<\omega}))$ is odd. That is, there are infinitely many $k' > j$ such that $\Omega(s_{k'})$ is odd and there is no $l' > k'$ such that both $\Omega(s_{l'})$ is even and $\Omega(s_{l'}) < \Omega(s_{k'})$.

Given such a k', either $\Omega(q_{k'}) <_\Omega \Omega(s_{k'})$. Then, for the priority memory to be erased, there must be an $l > k'$ such that $\Omega(q_l) \leq \Omega(s_{k'})$, $\Omega(q_l)$ is odd and there is no $k' < l' < l$ such that $\Omega(q_{l'}) < \Omega(q_l)$ is even. But then there is an $m > l$ such that $\Omega(q_m) < \Omega(q_l) \leq \Omega(s_{k'})$ is even (especially, $\Omega(q_m) <_\Omega \Omega(s_{k'})$), and there is no $m' > m$ such that $\Omega(q_{m'})$ is both odd and less than $\Omega(q_m)$. Hence the play is a loss for Duplicator (contradiction).

Or $\Omega(s_{k'}) \leq_\Omega \Omega(q_{k'})$, that is, $\Omega(q_{k'})$ is odd and $\Omega(q_{k'}) \leq \Omega(s_{k'})$. But then there is an $l > k'$ such that $\Omega(q_l) < \Omega(q_{k'}) \leq \Omega(s_{k'})$ is even, i. e., by choice of k', $\Omega(q_l) <_\Omega \Omega(s_l) \leq_\Omega \Omega(s_{k'})$. We can then argue as in the first case. □

3.3.2 Computing the delayed simulation relation

As pointed out in Subsection 3.2.1, the delayed simulation game for alternating parity automata is a finite game with a Büchi winning condition. That is, the simulation game can be solved as described in Section 1.7, cf. [ESW01]. The running time of this algorithm is $O(m'n_1)$, where m' is the number of edges of the game graph and n_1 is the number of rejecting positions, i. e., positions to which priority 1 is assigned according to [ESW01].

For an alternating parity automaton with n states, m transitions and l priorities, the simulation game graph has $O(mnl)$ nodes and edges, and we may assume that the rejecting nodes are the nodes of the form $Q \times Q \times \omega$, of which there are $O(ln^2)$. We thus have the following result

Theorem 3.3 (computing delayed simulation for APA) *The relation $\leq_{de}$ for an alternating parity automaton with n states, m transitions and l priorities can be computed in time $O(n^3l^2m)$ and space $O(mnl)$.*

That is, compared to the computation of $\leq_{de}$ for alternating Büchi automata (Theorem 1.6), we have an additional factor of l^2 in the asymptotic time consumption and an additional factor of l in the asymptotic space consumption for computing $\leq_{de}$ for alternating parity automata.

3.3.3 Dualities

For an alternating parity automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$, we define the *dual automaton* $\bar{\mathbf{Q}} = (Q, \Sigma, q_I, \Delta, \bar{E}, \bar{U}, \bar{\Omega})$, where $\bar{\Omega}\colon Q \to \omega$, $q \mapsto \Omega(q)+1$, $\bar{E} = Q \setminus E$ and $\bar{U} = Q \setminus U$. That is, $\bar{E} = U$ and $\bar{U} = E$, the two sets are swapped. The language accepted by $\bar{\mathbf{Q}}$ is the complement of the language accepted by $\mathbf{Q}$, see [MS87, Löd98]. Dualization is compatible with our simulation relation in the following way.

Lemma 3.2 *For all alternating parity automata $\mathbf{Q}$ and $\mathbf{S}$, $\mathbf{Q} \leq_{de} \mathbf{S}$ is equivalent to $\bar{\mathbf{S}} \leq_{de} \bar{\mathbf{Q}}$.*

Proof. Let $\mathbf{Q} \leq_{de} \mathbf{S}$ and let σ be a memoryless Duplicator winning strategy for $G^{de}(\mathbf{Q}, \mathbf{S})$. We define the Duplicator strategy $\bar{\sigma}$ for $G^{de}(\bar{\mathbf{S}}, \bar{\mathbf{Q}})$ as follows, where $\surd - 1 = \surd = \surd + 1$.

- For a position $(s, q', k, a, du, 0)$ with $s \in \bar{E}^s$, assume that $\sigma(q', s, k-1, a, du, 1) = (q', s', k'-1)$. We define $\bar{\sigma}(s, q', k, a, du, 0) = (s', q', k')$.
- For a position $(s', q, k, a, du, 1)$ with $q \in \bar{U}^q$, assume that $\sigma(q, s', k-1, a, du, 0) = (q', s', k'-1)$. We define $\bar{\sigma}(s', q, k, a, du, 1) = (s', q', k')$.

- For a position $(s,q,k,a,du,0,du,1)$ with $(s,q) \in \bar{E}^s \times \bar{U}^q$, assume that $\sigma(q,s,k-1,a,du,0,du,1) = (q',s,k-1,a,du,1)$ and $\sigma(q',s,k-1,a,du,1) = (q',s',k'-1)$. We define $\bar{\sigma}(s,q,k,a,du,0,du,1) = (s',q,k,a,du,1)$ and $\bar{\sigma}(s',q,k,a,du,1) = (s',q',k')$.

We claim that $\bar{\sigma}$ is a winning strategy. To see this, note that $n <_\Omega m$ if and only if $m+1 <_\Omega n+1$. Using induction and a case distinction over the modes of (q_i,s_i), it is easy to see that $((q_i,s_i,k_i)_{i<\omega},w)$ is a σ-conform protoplay if and only if $((s_i,q_i,k_i+1)_{i<\omega},w)$ is $\bar{\sigma}$-conform. It follows that $\bar{\sigma}$ is a Duplicator winning strategy for $G^{de}(\bar{\mathbf{S}},\bar{\mathbf{Q}})$. □

3.3.4 Quotienting is a problem

Unfortunately, the full delayed simulation relation for parity automata poses difficulties for quotienting. It seems hard to find a working definition of the full quotient of alternating parity automata w. r. t. our delayed simulation, because a merging of equivalent states may result in an automaton which is not delayed simulation equivalent to the original automaton, and it may even change the language of an automaton.

Consider the automaton in Figure 3.1 on the left-hand side. As usual, existential states are shown as diamonds and universal states are shown as boxes. The labels of the states are of the form "state name: priority". We have $q_I \equiv_{de} q_1 <_{de} q_2 <_{de} q_3$, so the resulting quotient automaton is shown on the right-hand side.

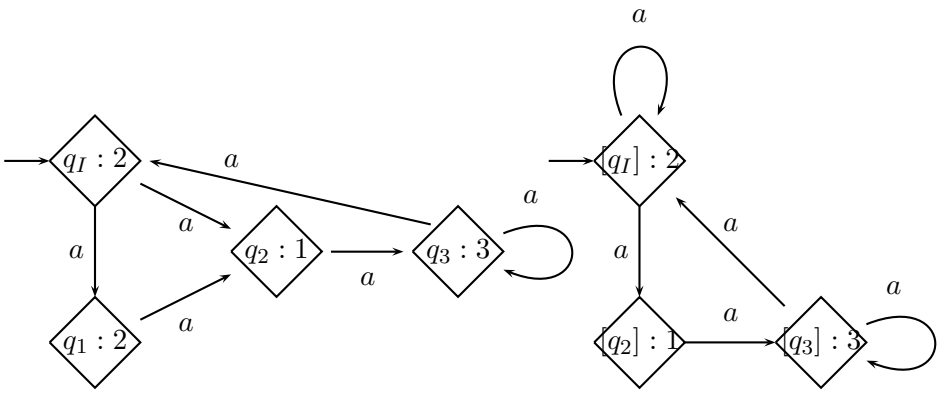

Figure 3.1: A parity game and its (non-equivalent) naive delayed simulation quotient

The language of the automaton is empty, while a naive quotient automaton accepts the word a^ω. Note that the automaton is normalized, $|\Sigma| = 1$ (the automaton is a parity game as defined in Section 3.1), all states have the same modality, only states in the same SCC are merged and there are only three different priorities.

In the in the example of Figure 3.1, one gets a quotient automaton simulation-equivalent to the original automaton if the transition $([q_I], a, [q_I])$ is deleted. This might suggest to use a minimax or semi-elective approach as in Chapter 1. This would in fact help in the example of Figure 3.1, but remember that, in the semi-elective quotient, transitions originating from existential (diamond) states must not be removed. Removing non-maximal transitions from diamond states may change the language of an automaton even if only transitions in a single SCC are removed. Consider Figure 3.2. Here we have $q_I >_{de} q_1 >_{de} q_2$, and the language changes from a^ω to $\emptyset$ if the transition from q_I to q_1 is removed.

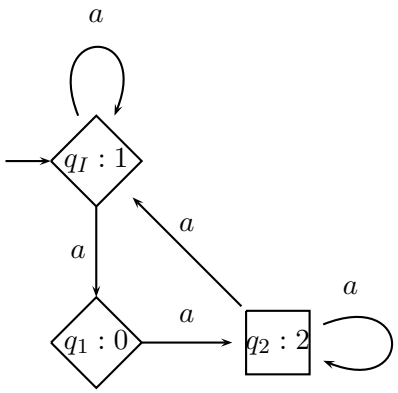

Figure 3.2: Removing non-maximal transitions changes the language

The considerations of Subsection 1.5.4 about pseudo-accepting states in the context of Büchi automata might suggest an approach in which existential states only have maximal transitions provided that their priority is even. This would be correct for the automata of Figures 3.1 and 3.2. But consider the automaton of Figure 3.3.

There, we have $q_2 \leq_{de} q_I$ and $q_2 \leq_{de} q_1$ while q_I and q_1 are incomparable, that is, $\max_a(q_1) = \max_b(q_1) = q_I$ and the priority of q_1 is even. But removing the transitions $(q_1, a/b, q_2)$ obviously changes the language of the automaton.

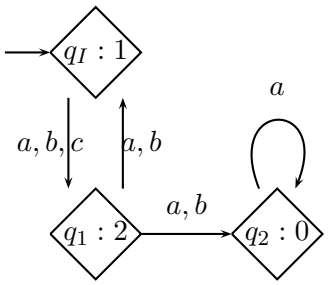

Figure 3.3: Non-maximal transitions of existential states with even priority must not be removed

3.4 Quotienting with Smaller Relations

In Subsection 3.3.4, we have seen that our delayed simulation relation makes it difficult to merge equivalent automata states. As a remedy, we here present two simulation relations that do allow this kind of quotienting in such a way that the quotient automaton is simulation equivalent to the original automaton. These relations are strictly smaller than the full delayed simulation relation and hence less states are identified as equivalent. This will be partly compensated by the possibility to combine the quotienting with respect to both new relations.

3.4.1 Smaller delayed simulation relations for alternating parity automata

The relations $\leq^l_{de}, \leq^r_{de} \subseteq \leq_{de}$ are defined by the the same simulation game as $\leq_{de}$, with the following changes: The memory update function pm in the definition of $\leq_{de}$ of Subsection 3.2.1 is replaced by the functions pm^l and pm^r, respectively. The difference between pm and pm^r is that case 4 in the definition of pm yields an pm^r-result of k, that is,

$$\mathrm{pm}^r(i,j,k) = k, \text{ if } j \leq_\Omega i,\ i \text{ is odd and } i \leq k, \text{ and } j \text{ is odd or } k < j \ , \quad (3.10)$$

while $\mathrm{pm}^r(i,j,k)$ is equal to $\mathrm{pm}(i,j,k)$ in all the other cases of the definition of pm. Symmetrically, case 5 yields a pm^l-result of k while pm^l is equal to pm in the other cases, that is,

$$\mathrm{pm}^l(i,j,k) = k, \text{ if } j \leq_\Omega i,\ j \text{ is even and } j \leq k, \text{ and } i \text{ is even or } k < i \ . \quad (3.11)$$

The respective simulation games are denoted $G^{r-de}(\mathbf{Q},\mathbf{S})$ and $G^{l-de}(\mathbf{Q},\mathbf{S})$, for automata $\mathbf{Q}$ and $\mathbf{S}$.

That is, in the case of $\leq_{de}^{r}$, once the value of the priority memory is not $\surd$, it will change back to the value $\surd$ again only if this is triggered by a small even priority in the simulating automaton, while small odd priorities in the simulated automaton are ignored. In the case of $\leq_{de}^{l}$, only a small odd priority in the simulated automaton can trigger a switch to the memory value $\surd$ while small even priorities are ignored.

We call $\leq_{de}^{r}$ the *right-hand delayed simulation relation*, using the intuition that the simulating automaton (the automaton on the right-hand side of $\leq_{de}^{r}$) has to trigger a switch to the memory value $\surd$. Analogously, $\leq_{de}^{l}$ is the *left-hand delayed simulation relation*.

It follows immediately that left-hand and right-hand delayed simulation are subsets of the full delayed simulation, and it is easy to see that they are proper subsets, see also Subsection 3.4.3. Computing these relations is as asymptotically time- and space-consuming as computing the full delayed simulation relation.

Again, note the analogy to delayed simulation for Büchi automata: If a parity automaton $\mathbf{Q}$ is a Büchi automaton (priorities 0 and 1 only), then the right-hand delayed simulation relation for $\mathbf{Q}$ is the same as the delayed simulation relation for $\mathbf{Q}$ as a Büchi automaton and as the full delayed simulation relation for $\mathbf{Q}$ as a parity automaton. Dually, the left-hand delayed simulation relation is a natural notion for delayed simulation for alternating co-Büchi automata, i. e., parity automata with priorities 1 and 2 only.

In analogy to Lemma 3.2, $\leq_{de}^{r}$ and $\leq_{de}^{l}$ are connected via dualization.

Remark 3.1 *For alternating parity automata* $\mathbf{Q}$ *and* $\mathbf{S}$, $\mathbf{Q} \leq_{de}^{l} \mathbf{S}$ *is equivalent to* $\bar{\mathbf{S}} \leq_{de}^{r} \bar{\mathbf{Q}}$.

That is, in order to compute $\leq_{de}^{l}$ for an automaton $\mathbf{Q}$, we may dualize $\mathbf{Q}$ and compute $\leq_{de}^{r}$ for $\bar{\mathbf{Q}}$. The relation $\leq_{de}^{l}$ for $\mathbf{Q}$ is the inverse relation of the relation $\leq_{de}^{r}$ for $\bar{\mathbf{Q}}$ (and vice versa).

It is not entirely obvious that these relations are preorders. Especially, Lemma 3.1 does not hold if $\leq_{de}^{r}$ or $\leq_{de}^{l}$ are inserted for $\leq_{de}$, because certain priorities encountered during a play do not influence the priority memory. But Lemma 3.1 is a cornerstone of the proof of Theorem 3.1, showing the transitivity of the delayed simulation relation for alternating parity automata. Nevertheless, we can modify Lemma 3.1 and then show that the right-hand and left-hand simulation relations are preorders.

Corollary 3.3 *The relations* $\leq_{de}^{r}$ *and* $\leq_{de}^{l}$ *are preorders.*

Proof. Obviously, the two relations are reflexive. Note that we can equivalently restate the cases 1 and 3 in the definition of the function pm in Subsection 3.2.1 as

1. If $i <_\Omega j$ then

   (a) if i is odd then $\mathrm{pm}(i,j,\surd) = j$,

   (b) if i is even then $\mathrm{pm}(i,j,\surd) = i$.

3. If $i <_\Omega j$ then

   (a) if i is odd then $\mathrm{pm}(i,j,k) = \min\{j,k\}$,

   (b) if i is even then $\mathrm{pm}(i,j,k) = \min\{i,k\}$.

This is because if $i <_\Omega j$ and i is odd then $i > j$, hence $\min\{i,j\} = j$, and symmetrically for an even i.

Based on this description, we can replace equation (3.6) in Lemma 3.1 by

$$k_i = \min\{\Omega(p) \mid p \in \{q_l \mid n \leq l \leq i, \Omega(q_l) \text{ even}\} \cup \{s_l \mid n \leq l \leq i\}\} \quad , \qquad (3.12)$$

and equation (3.8) in Lemma 3.1 by

$$\Omega(s_m) = \min\{\Omega(p) \mid p \in \{q_i \mid n \leq i \leq m, \Omega(q_i) \text{ even}\} \cup \{s_i \mid n \leq i \leq m\}\} \quad . \qquad (3.13)$$

In the case of $\leq^r_{de}$, it is easy to check that only a restricted version of Lemma 3.1 holds true, where equation (3.12) holds for every $n \leq i < m$ and equation (3.13) holds for $m < \omega$: For a play $\pi = (q_i, s_i, k_i)_{i<\omega}$ of a right-hand delayed simulation game $G^{r-de}(q_0, s_0)$ and $0 \leq n < m \leq \omega$ such that $k_{n-1} = \surd$ if $n > 0$, $k_m = \surd$ if $m < \omega$, and $k_i \neq \surd$ for all i in $\{n, n+1, \ldots, m-1\}$, (3.12) and (3.13) hold, but an analog of part 2 of Lemma 3.1 does not hold for right-hand simulation games.

But these restricted versions are then sufficient to show the transitivity of $\leq^r_{de}$ with the same proof technique as for Theorem 3.1—note that only the subcases 2a, 2c, 2e, 3a, 3c and 3e apply to $\leq^r_{de}$.

The proof of Theorem 3.1 for $\leq^l_{de}$ is completely symmetrical, that is, a restriction of Lemma 3.1 with modified versions of Lemma 3.1(1) and 3.1(2) is true for $\leq^l_{de}$ and can then be used to show transitivity in the style of the proof of Theorem 3.1. □

3.4.2 Quotienting

The quotient automaton of an APA $\mathbf{Q}$ w. r. t. $\equiv^r_{de}$, denoted $\mathbf{Q}^{se}_{r-de}$ is an enhancement of the semi-elective quotient automaton of Section 1.5. We have to remove transitions starting from universal states that might enable Spoiler to win the game $G^{r-de}(\mathbf{Q}, \mathbf{Q}^{\equiv}_{r-de})$ by not allowing the pebble of the simulating automaton to reach

a state with a small even priority although Duplicator can enforce to reach the respective state in $G^{r-de}(\mathbf{Q},\mathbf{Q})$. Dually, we have to remove transitions starting from existential states in $\mathbf{Q}^{\equiv}_{l-de}$ to prevent Spoiler from winning the game $G^{l-de}(\mathbf{Q}^{\equiv}_{l-de},\mathbf{Q})$ by avoiding a small odd priority which is unavoidable in $\mathbf{Q}$.

That is, while the definition of the semi-elective quotient of an alternating Büchi automaton w. r. t. delayed simulation in Chapter 1 is asymmetric in the context of Büchi acceptance, we can now perceive a symmetry in that definition if we regard Büchi acceptance as a special case of parity acceptance.

For an alternating parity automaton $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,\Omega)$, we define the *right-semi-elective quotient automaton*

$$\mathbf{Q}^{se}_{r-de} = (Q^r,\Sigma,[q_I]^r,\Delta^r,E^r,U^r,\Omega) \ , \tag{3.14}$$

where

$$\begin{aligned}
Q^r &= Q/\equiv^r_{de} = \{[q]^r \mid q \in Q\} \ , && (3.15)\\
\Delta^r &= \{([q]^r,a,[q']^r) \mid (q,a,q') \in \Delta, q \in E\} \\
&\quad \cup \{([q]^r,a,[q']^r) \mid [q]^r \subseteq U, q' \in \min^{r-de}_a(q)\} \ , && (3.16)\\
U^r &= \{[q]^r \in Q^r \mid [q]^r \subseteq U\} \ , && (3.17)\\
E^r &= \{[q]^r \in Q^r \mid [q]^r \cap E \neq \emptyset\} \ . && (3.18)
\end{aligned}$$

The notations $\min^x_a(q)$ and $\max^x_a(q)$ are introduced in Subsection 1.4.1.

For the priority function of the quotient automaton, we extend the domain of Ω to subsets of Q: $\Omega(M) = \min\{\Omega(m) \mid m \in M\}$ for $M \subseteq Q$.
Symmetrically, there is also the *left-semi-elective quotient automaton*

$$\mathbf{Q}^{se}_{l-de} = (Q^l,\Sigma,[q_I]^l,\Delta^l,E^l,U^l,\Omega) \ , \tag{3.19}$$

where

$$\begin{aligned}
Q^l &= Q/\equiv^l_{de} = \{[q]^l \mid q \in Q\} \ , && (3.20)\\
\Delta^l &= \{([q]^l,a,[q']^l) \mid (q,a,q') \in \Delta, q \in U\} \\
&\quad \cup \{([q]^l,a,[q']^l) \mid [q]^l \subseteq E, q' \in \max^{l-de}_a(q)\} \ , && (3.21)\\
U^l &= \{[q]^l \in Q^l \mid [q]^l \cap U \neq \emptyset\} \ , && (3.22)\\
E^l &= \{[q]^l \in Q^l \mid [q]^l \subseteq E\} \ . && (3.23)
\end{aligned}$$

Again, the duality principle holds: The dual automaton of $\mathbf{Q}^{se}_{l-de}$ is the same as $\bar{\mathbf{Q}}^{se}_{r-de}$ and, conversely, the dual automaton of $\mathbf{Q}^{se}_{r-de}$ is the same as $\bar{\mathbf{Q}}^{se}_{l-de}$.

Q simulates $\mathbf{Q}^{se}_{r-de}$

To show that $\mathbf{Q}$ *r-de*-simulates $\mathbf{Q}^{se}_{r-de}$, we can use the proof technique of Section 1.5 *mutatis mutandis*, namely, with another rule of updating the joint strategy, cf. the proof of Theorem 1.3.

In the proof, we use the analogs of Corollaries 1.6 and 1.7 for $\leq^r_{de}$; the proofs of these corollaries for $\leq^r_{de}$ are the same as the proofs in Section 1.5.2.

Theorem 3.4 *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$ be an alternating parity automaton, and let p_0, q_0 be states such that $p_0 \leq^r_{de} q_0$. Then $\mathbf{Q}(q_0)$* r-de-*simulates $\mathbf{Q}^{se}_{r-de}(p_0)$, i. e., there is a Duplicator winning strategy for $G^{r-de}([p_0]^r, q_0)$.*

Proof. We fix,

1. for every $K \in Q^r_{de}$, a representative $\mathrm{rep}(K) \in K$ such that $\Omega(\mathrm{rep}(K)) = \Omega(K)$,
2. for every $(K,p) \in Q^r_{de} \times Q$ such that $K \sqsubseteq^r_{de} p$, a $\sqsubseteq^r_{de}$-respecting minimax strategy $\sigma^o_{K,p}$ of Duplicator for $G(K,p)$, and
3. for every $(p,q) \in Q \times Q$ such that $p \leq^r_{de} q$, a Duplicator winning strategy $\sigma^{de}_{p,q}$ for $G^{r-de}(p,q)$.

Let $T_n = (t_i)_{i \leq n} = (K_i, q_i, l_i)_{i \leq n}$ be the prefix of a $G^{r-de}([p_0]^r, q_0)$-play T (with $(K_i, q_i, l_i) \in Q^r \times Q \times (\omega \cup \{\surd\})$). Let

$$j' = \min\{i \leq n \mid l_i \neq \surd \wedge \forall i'(i \leq i' \leq n \rightarrow l_{i'} \neq \surd)\} \ , \tag{3.24}$$

or $j' = 0$ if this set is empty, and let

$$j = \max\{i \mid j' \leq i \leq n \wedge \Omega(K_i) = l_i < l_{i-1} \wedge l_i <_\Omega \Omega(q_i)\} \ , \tag{3.25}$$

or $j = j'$ if this set is empty.
Let $T_{[j,n]}$ be the suffix of T_n starting with $([K_j, q_j, l_j)$, and define

$$\sigma(T_n) = \sigma^o_{K_j, \mathrm{rep}(K_j)} \bowtie \sigma^{de}_{\mathrm{rep}(K_j), q_j}(T_{[j,n]}) \ . \tag{3.26}$$

Note that σ is $\leq^r_{de}$-respecting.
Now if (K_i, q_i, l_i) is
(1) the first $(Q^r \times Q \times \omega)$-position after the last $(Q^r \times Q \times \{\surd\})$-position, or
(2) the first $(Q^r \times Q \times \omega)$-position at all, or
(3) the last position where the priority memory value l_i has decreased because $\Omega(K_i)$ is even and $\Omega(K_i) < l_{i-1}$, and $\Omega(K_i) <_\Omega \Omega(q_i)$,

we have $\mathrm{rep}(K_i) \leq^r_{de} q_i$. The strategy σ is updated to $\sigma^o_{\mathrm{rep}(K_i),\mathrm{rep}(K_i)} \bowtie \sigma^{de}_{\mathrm{rep}(K_i),q_i}$ where only the suffix starting with (K_i, q_i, l_i) is taken into account for the following moves of Duplicator.

Then there is either an $m > i$ such that $l_m < l_i$ and $l_{m'} = l_i$ for all $i \leq m' < m$, in which case σ is updated again (and this can happen consecutively only finitely often—we use here that the priority memory is monotonically decreasing between two memory values of $\surd$, see Proposition 3.1). Or, since $\sigma^{de}_{\mathrm{rep}(K_i),q_i}$ is winning, there is a $k > i$ such that $\Omega(q_k) \leq l_{k-1} = l_i$ and $\Omega(q_k)$ is even (hence $\Omega(q_k) \leq_\Omega l_i = \Omega(K_k)$, and still $K_k \sqsubseteq^r_{de} q_k$ by construction of σ). Hence $l_k = \surd$.

That is, every position in $Q^r \times Q \times \omega$ is followed by a position in $Q^r \times Q \times \{\surd\}$ in a σ-conform play. Thus σ is a Duplicator winning strategy in $G^{r-de}([p_0]^r, q_0)$. □

$\mathbf{Q}^{se}_{r-de}$ simulates $\mathbf{Q}$

To show that $\mathbf{Q}^{se}_{r-de}$ *r-de*-simulates $\mathbf{Q}$, first note that, by construction of $\mathbf{Q}^{se}_{r-de}$, the analog of Corollary 1.8 holds for $\leq^r_{de}$.

Corollary 3.4 (cf. Corollary 1.8) *Let $q'_I \in [q_I]^r$. There is a Duplicator strategy $\sigma^{\equiv}_r$ for $G^{de}(q'_I, [q_I]^r)$ such that, for every $Q \times Q^r$-position $(q'_i, [q_i]^r)$ of a $\sigma^{\equiv}_r$-conform play, $q'_i \in [q_i]^r$ holds, that is, $\sigma^{\equiv}_r$ is $\equiv^r_{de}$-respecting.*

We can then proof that $\mathbf{Q}^{se}_{r-de}$ *r-de*-simulates $\mathbf{Q}$. The style of this proof is mainly similar to the proofs of Theorem 3.4 above and of Theorem 1.3; the proof of Theorem 1.4 is simpler because of the restrictions of Büchi acceptance as compared to parity acceptance.

Theorem 3.5 *Let $\mathbf{Q}$ be a parity automaton with positions p_0, q_0 such that $p_0 \leq_{de} q_0$. The automaton $\mathbf{Q}^{se}_{r-de}([q_0]^r)$* r-de-*simulates $\mathbf{Q}(p_0)$, that is, there is a winning strategy for Duplicator in $G^{r-de}(p_0, [q_0]^r)$.*

Proof. For every $(p,q) \in Q \times Q$ such that $p \leq^r_{de} q$, let $\sigma_{p,q}$ be a Duplicator winning strategy for $G^{r-de}(p,q)$.

We fix, for every $K \in Q^r$, a representative $\mathrm{rep}(K) \in K$ such that $\Omega(\mathrm{rep}(K)) = \Omega(K)$. For every $K \in Q^r$, let $\sigma^{\equiv}_K$ be a $\equiv^r_{de}$-respecting Duplicator strategy for $G^{r-de}(\mathrm{rep}(K), K)$, and let $K_0 = [q_0]^r$.

Here, we define for the prefix of a play $T_n = (p_i, K_i, l_i)_{i \leq n}$ of length $n+1$, the values j' and j as follows. Let

$$j' = \min\{i \leq n \mid l_i \neq \surd \wedge \forall i'(i \leq i' \leq n \to l_{i'} \neq \surd)\} \ , \tag{3.27}$$

or $j' = 0$ if this set is empty, and let

$$j = \max\{i \mid j' \leq i \leq n \ \wedge \ \Omega(K_i) = l_i < l_{i-1}\} \ , \tag{3.28}$$

or $j = j'$ if this set is empty. Let $T_{[j,n]}$ be the suffix of T_n starting with (p_j, K_j, l_j), and define

$$\sigma(T_n) = \sigma_{p_j,\mathrm{rep}(K_j)} \bowtie \sigma^{\equiv}_{K_j}(T_{[j,n]}) \ . \tag{3.29}$$

The argumentation then continues very similar to the proof of Theorem 3.4: The strategy σ is $\leq^r_{de}$-respecting. If (p_i, K_i, l_i) is

(1) the first $(Q \times Q^r \times \omega)$-position after the last $(Q \times Q^r \times \{\surd\})$-position, or

(2) the first $(Q \times Q^r \times \omega)$-position at all, or

(3) the last position where the priority memory value l_i has decreased because $\Omega(K_i)$ is odd and $\Omega(K_i) < l_{i-1}$, and $\Omega(p_i) <_\Omega \Omega(K_i)$,

we have $p_i \leq^r_{de} \mathrm{rep}(K_i)$. The strategy σ is updated to $\sigma_{p_i,\mathrm{rep}(K_i)} \bowtie \sigma^{\equiv}_{K_i}$ where only the suffix starting with (p_i, K_i, l_i) is taken into account for the following moves of Duplicator. Assume that $(k_m)_{m \geq i}$ is the intermediate sequence for the play with strategy $\sigma_{p_i,\mathrm{rep}(K_i)} \bowtie \sigma^{\equiv}_{K_i}$, that is, $k_i = \mathrm{rep}(K_i)$, but k_{i+1} may be different from $\mathrm{rep}(K_{i+1})$, but $k_m \in K_m$ for all $m \geq i$, of course, since $\sigma^{\equiv}_{K_i}$ is $\equiv^r_{de}$-respecting, and, consequently, $\Omega(k_m) \geq \Omega(K_m)$.

Then there is either an $m > i$ such that $l_m < l_i$ and $l_{m'} = l_i$ for all $i \leq m' < m$, in which case σ is updated again (again, we use Proposition 3.1). Or, since $\sigma_{p_i,\mathrm{rep}(K_i)}$ is winning, there is an $m > i$ such that $\Omega(k_m) \leq l_{m-1} = l_i$ and $\Omega(k_m)$ is even. Since $\Omega(K_m) \leq \Omega(k_m)$, either $\Omega(K_m)$ is even and thus $l_m = \surd$. Or $\Omega(K_m)$ is odd and $l_m < l_i$, and we update σ as above.

It is thus ensured that every position in $Q \times Q^r \times \omega$ is followed by a position in $Q \times Q^r \times \{\surd\}$ in a σ-conform play. Thus σ is a Duplicator winning strategy in $G^{r-de}(p_0, [q_0]^r)$. □

Equivalence of Q and $\mathbf{Q}^{se}_{l-de}$

The proofs for the respective claims for left-hand delayed simulation are "mirror-symmetric" to the above proofs in the following sense. To show that $\mathbf{Q}^{se}_{l-de} \leq^l_{de} \mathbf{Q}$, we derive the following corollary from the definition of $\mathbf{Q}^{se}_{l-de}$, a "mirrored" version of Corollary 3.4.

Corollary 3.5 *Let $q'_I \in [q_I]^l$. There is a Duplicator strategy $\sigma^{\equiv}_l$ for $G^{de}([q_I]^l, q'_I)$ such that, for every $Q^l \times Q$-position $([q_i]^l, q'_i)$ of a $\sigma^{\equiv}_l$-conform play, $q'_i \in [q_i]^l$ holds, that is, $\sigma^{\equiv}_l$ is $\equiv^l_{de}$-respecting.*

We then can use the updatable join of a $\equiv^l_{de}$-respecting strategy and a winning strategy to show that $\mathbf{Q}$ *l-de*-simulates $\mathbf{Q}^{se}_{l-de}$ (as opposed to above, where the join of a winning strategy and a $\equiv^r_{de}$-respecting strategy is used to show $\mathbf{Q} \leq^r_{de} \mathbf{Q}^{se}_{r-de}$).

To show, conversely, that $\mathbf{Q} \leq^l_{de} \mathbf{Q}^{se}_{l-de}$, we can use the updatable join of a winning strategy and a $\sqsubseteq^l_{de}$-respecting minimax strategy (while above and, in a sense, also in Section 1.5, we use the join of a $\sqsubseteq^r_{de}$-respecting minimax strategy and a winning strategy to show $\mathbf{Q}^{se}_{r-de} \leq^r_{de} \mathbf{Q}$).

In summary, we have the following theorem.

Theorem 3.6 (left- and right-semi-elective quotients) *For every alternating parity automaton* $\mathbf{Q}$*, the automata* $\mathbf{Q}$ *and* $\mathbf{Q}^{se}_{l-de}$ l-de-*simulate each other, and the automata* $\mathbf{Q}$ *and* $\mathbf{Q}^{se}_{r-de}$ r-de-*simulate each other. In particular,* $L(\mathbf{Q}) = L(\mathbf{Q}^{se}_{l-de}) = L(\mathbf{Q}^{se}_{r-de})$.

3.4.3 Example: Quotienting

In this subsection, we give an example for quotienting w. r. t. left-hand and right-hand delayed simulation. Our example is the parity game $H_{2,2}$ shown in Figure 3.4. (In the figures of parity games in this subsection, we omit transition labels and do not mark an initial position.) It is taken from Jurdziński [Jur00]. The game $H_{2,2}$ belongs to a family of parity games $H_{i,j}$ for which Jurdziński's lifting algorithm for deciding the winner of a parity game shows a worst-case behavior. Note that Pathfinder wins $H_{2,2}$ on the positions q_{0i}, $0 \leq i \leq 4$, while Automaton wins on all the other positions.

There are eight equivalence classes w. r. t. $\equiv^r_{de}$ of $H_{2,2}$, of which three contain more than one element: We have $q_{00} \equiv^r_{de} q_{02} \equiv^r_{de} q_{04}$ and $q_{01} \equiv^r_{de} q_{03}$, and further $q_{13} \equiv^r_{de} q_{14}$. In detail, q_{13} and q_{14} *r-de*-simulate every state, and we have $q_{10} \leq^r_{de} q_{12}$, $q_{20} \leq^r_{de} q_{21}$ and $[q_{00}]^r \leq^r_{de} q_{11}$.

Since q_{11} and q_{13} simulate the elements of $[q_{00}]^r$ but not vice versa, the transitions (q_{01}, q_{11}) and (q_{03}, q_{13}) are deleted in the quotient: q_{01} and q_{03} are universal states, and q_{11} is not in $\min^{r-de}(q_{01})$ and q_{13} is not in $\min^{r-de}(q_{03})$. The resulting quotient game is shown in Figure 3.5.

A subsequent quotienting w. r. t. $\equiv^l_{de}$ results in the game of Figure 3.6. The states $[q_{00}]$ and $[q_{01}]$ of Figure 3.5 are equivalent w. r. t. $\equiv^l_{de}$. And $\max^{l-de}([q_{13}]) = \{[q_{13}]\}$, which is why the transitions $([q_{13}], [q_{01}])$ and $([q_{13}], [q_{12}])$ do not appear in Figure 3.6.

In the above example, it does not really matter in which order the quotienting w. r. t. the two equivalence relations is applied. We have first done a quotienting w. r. t. $\equiv^r_{de}$ and then w. r. t. $\equiv^l_{de}$. The other way around, we get virtually the same quotient, only state $[q_{13}]$ of Figure 3.6 is a box state in that order of quotienting.

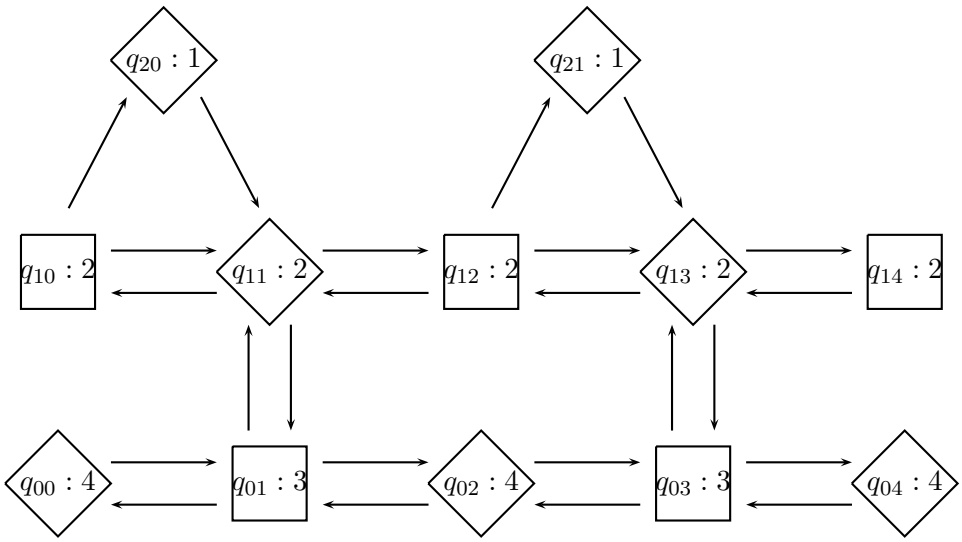

Figure 3.4: Quotienting example: $H_{2,2}$ of [Jur00]

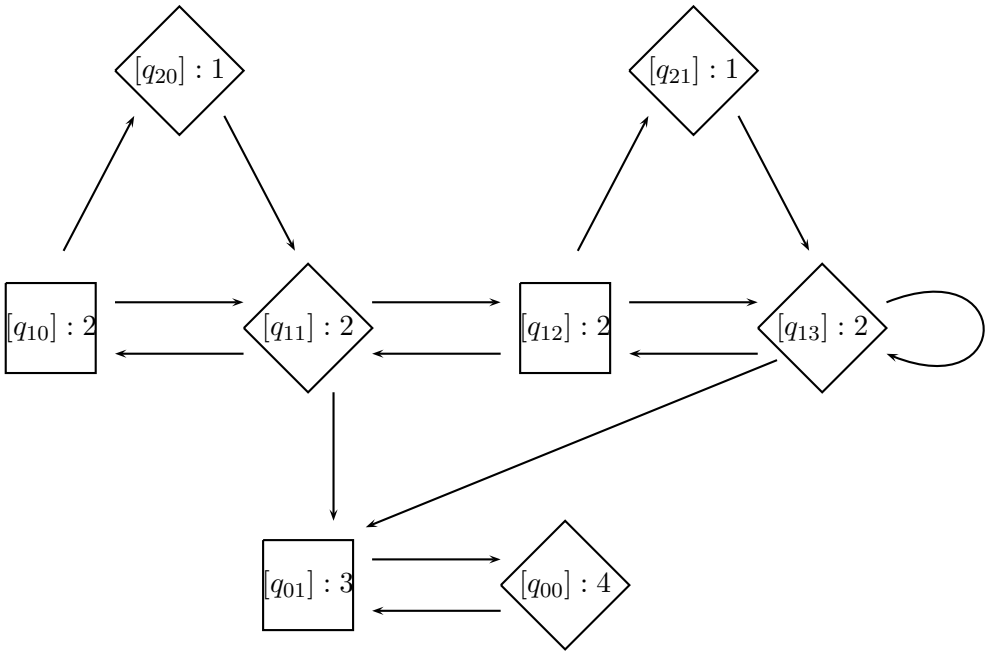

Figure 3.5: The right-semi-elective quotient of $H_{2,2}$

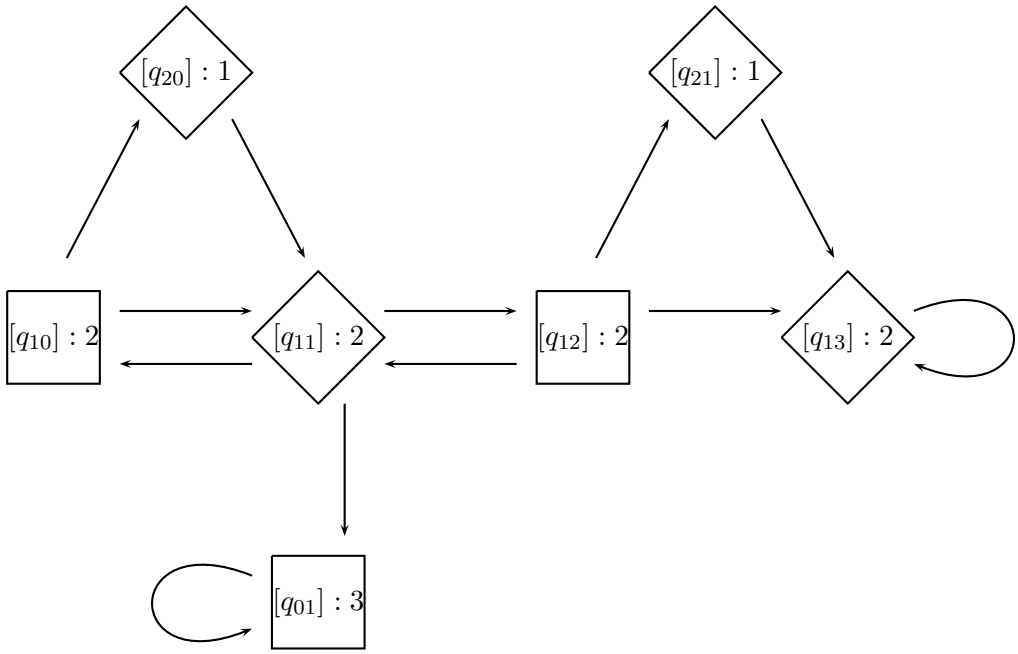

Figure 3.6: The left-semi-elective quotient of the right-semi-elective quotient of $H_{2,2}$

Moreover, all positions which are equivalent w. r. t. $\equiv_{de}$ are merged in the quotient game of Figure 3.6.

But these nice properties do *not* hold for all parity automata or, at least, for all parity games. It is possible that, for states q, s, we have $q \leq_{de} s$, but neither $q \leq^l_{de} s$ nor $q \leq^r_{de} s$. And it is possible that both $q \leq^r_{de} s$ and $s \leq^l_{de} q$ hold, but neither $q \leq^l_{de} s$ nor $s \leq^r_{de} q$. Consider the parity game in Figure 3.7.

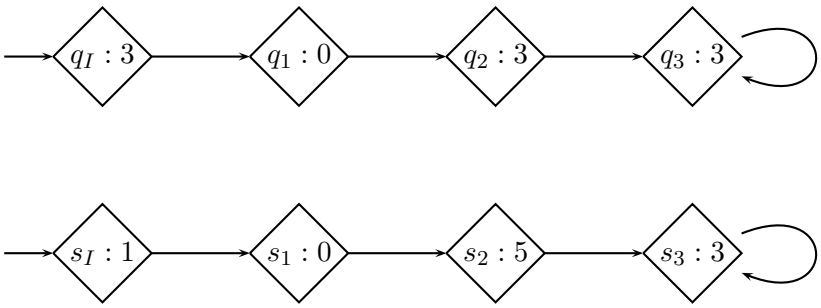

Figure 3.7: Mutually strictly larger

There, we have $q_I <^r_{de} s_I$ but not $s_I <^r_{de} q_I$, and $s_I <^l_{de} q_I$ but not $q_I <^l_{de} s_I$. We now integrate q_I and s_I into a single parity game as shown in Figure 3.8. (The dots in Figure 3.8 replace the two diamond positions p_2 and p_3 with priority 5.)

In this game, we have $p_1 <^r_{de} p_0$ and $p_2 <^l_{de} s_1$, but $p_2 \not\leq^r_{de} q_1$. The positions p_1 and p_2 are incomparable w. r. t. $\leq^l_{de}$.

That is, in the quotient w. r. t. $\equiv^r_{de}$, the transition from p_I to p_0 is deleted. In the quotient w. r. t. $\equiv^l_{de}$, however, the transition from p_0 to s_I is deleted. Consequently, $p_1 <^r_{de} p_0$ no longer holds in this quotient, and the transition from p_I to p_0 is *not* deleted in a consecutive quotienting w. r. t. $\equiv^r_{de}$. This shows that the quotient game w. r. t. both $\equiv^r_{de}$ and $\equiv^l_{de}$ depends on the order in which the two quotientings are applied.

With a little modification, this example also demonstrates that it is not possible to mend the situation by normalizing the game: Merge the states q_3, s_3 and p_5 to a single box state and connect it via two states with priorities 2 and 4 to p_I. The resulting game is normalized, but the relations between p_0, p_1, p_2, q_1 and s_1 still hold.

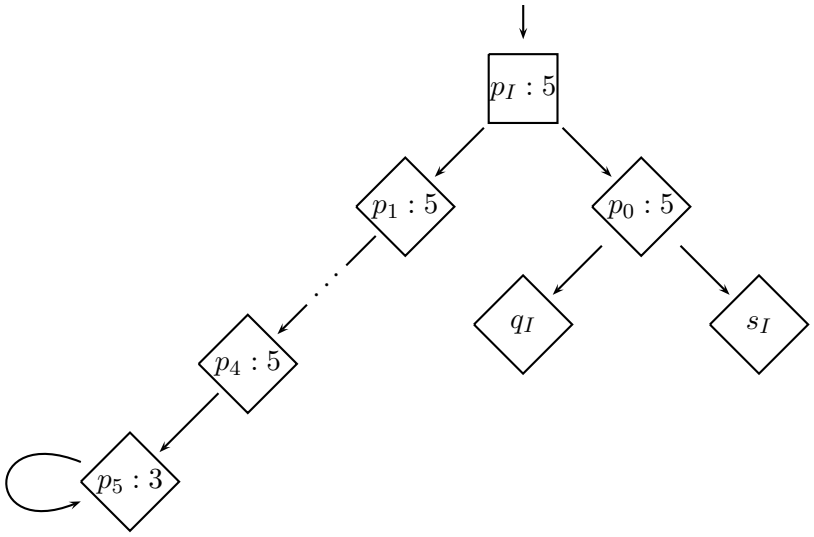

Figure 3.8: Quotienting modulo $\equiv^{l}_{de}$ destroys a $\leq^{r}_{de}$-relation

3.5 A Simulation-Based Simplification Algorithm for Alternating Parity Automata

In this section, we present some observations that will allow us to use both the full delayed simulation relation and the left-hand and right-hand versions for the efficient simplification of alternating parity automata.

That is, we give an algorithm for the simplification of alternating parity automata and show that it is correct in the sense that the input automaton and the resulting simplified automaton accept the same language, and the simplified automaton is not larger than the input automaton.

3.5.1 The simplification algorithm

INPUT An alternating parity automaton **Q**.

Simplification Algorithm

1. Normalize **Q**.
2. Compute the relation $\leq_{de}$ for **Q**.
   (a) Assign the same priority to *de*-equivalent states, namely, the minimum of the priorities of the states in that equivalence class. [Lemma 3.3]

   (b) Change transitions to topologically maximal representatives for every class. [Lemma 3.4]

   (c) Apply 0-1-minimaxing. [Lemma 3.5]

   (d) Apply reachability minimaxing. [Lemma 3.6]

3. Let $\mathbf{Q}'$ be the modified automaton computed so far. Compute $\leq^r_{de}$ and $\leq^l_{de}$ for $\mathbf{Q}'$.

   (a) Delete transitions according to $\leq^r_{de}$ and $\leq^l_{de}$ as described in Lemma 3.7.

   (b) Merge states according to $\leq^r_{de}$ and $\leq^l_{de}$ as described in Lemma 3.8.

4. Let $\mathbf{Q}''$ be the automaton computed from $\mathbf{Q}$ in steps 1 to 3b. If $\mathbf{Q}''$ has less states or transitions than the input automaton $\mathbf{Q}$, continue with step 1 with $\mathbf{Q}''$ as $\mathbf{Q}$, else return $\mathbf{Q}''$ as the simplified automaton.

That is, there is a working cycle of alternating normalizations (step 1) and simulation-based simplifications (steps 2 and 3). This is because, on the one hand, normalization may change the simulation relations between automata states and, on the other hand, the simulation-based simplifications may change the priorities of states and result in deleting transitions, such that a new normalization may again change the automaton.

It is easy to see that normalization changes the simulation relation. As a very simple example, consider two states q, q' with outgoing transitions (q,a,q) and (q',a,q') only, such that $\Omega(q) = 0$ and $\Omega(q') = 2$. Then, q' does not *de*-simulate q, but, in a normalized automaton, the priority of q' is 0, so q and q' are equivalent after normalization.

For the correctness of the algorithm, we will show that the normalized automaton after step (1) of the algorithm is equivalent w. r. t. $\equiv_{de}$ to the simplified automaton after step (3b). This is sufficient since we know that *de*-equivalence implies language equivalence and normalization does not change the accepted language.

3.5.2 Usable properties of the delayed simulation relations

The homogenized automaton

We first show that we can assign the same priority to two *de*-equivalent states, namely, the minimum of all priorities in the *de*-equivalence class. This is can be seen as an extension of the concept of pseudo-accepting states (Lemma 1.8 in Subsection 1.5.4).

Lemma 3.3 (homogenized automaton) *Let* $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$ *be a normalized APA. We define*

$$\mathbf{Q}^{2a} = (Q, \Sigma, q_I, \Delta, E, U, \Omega') \tag{3.30}$$

by

$$\Omega'(q) = \min\{\Omega(q') \mid q' \equiv_{de} q\} \ , \tag{3.31}$$

for all $q \in Q$.

Then, $\mathbf{Q}^{2a} \equiv_{de} \mathbf{Q}$. *We call* $\mathbf{Q}^{2a}$ *the* homogenized automaton *of* $\mathbf{Q}$.

Proof. We can basically follow the proof scheme of Theorems 1.3, 3.4, and 3.5. That is, we get a winning strategy as the join of two strategies (here, of two winning strategies) which is updated at certain points of the simulation game.

Therefore, let $\sigma_{q,q'}$ be a Duplicator winning strategy for $G^{de}(q,q')$, for every pair of states $q, q' \in Q$ such that $q \leq_{de} q'$. For every state $q \in Q$, let $\mathrm{rep}(q)$ be a state in Q such that $\mathrm{rep}(q) \equiv_{de} q$ and $\Omega(\mathrm{rep}(q)) = \Omega'(\mathrm{rep}(q))$. We assume that $\mathrm{rep}(q) = q$ if $\Omega(q) = \Omega'(q)$.

To show that $\mathbf{Q} \leq_{de} \mathbf{Q}^{2a}$, we start a play of the game $G^{de}(\mathbf{Q}, \mathbf{Q}^{2a})$ with the Duplicator strategy $\sigma_{q_I,q_I} \bowtie \sigma_{q_I,q_I}$, that is, Duplicator chooses his moves in $\mathbf{Q}^{2a}$ as if $\mathbf{Q}^{2a}$ was the same as $\mathbf{Q}$.

We update this strategy if, during a play, the priority memory decreases or changes from $\surd$ to a natural number at a position (q_j, q'_j, k_j), i. e., in round n, the last update was in round

$$\begin{aligned} j \ = \ \max\{i \leq n \mid & ((k_{i-1} = \surd \wedge k_i \neq \surd) \vee (k_{i-1} > k_i)) \\ & \wedge \forall i' (i \leq i' \leq n \rightarrow k_{i'} \neq \surd)\} \ , \end{aligned} \tag{3.32}$$

or $j = 0$ if this set is empty.

A strategy update at position (q_j, q'_j, k_j) means that Duplicator continues the play with the strategy $\sigma_{q,\mathrm{rep}(q')} \bowtie \sigma_{\mathrm{rep}(q'),q'}$ and only takes the positions starting from round j into account for his further moves.

As in previous proofs, it follows that, whenever the priority memory is different from $\surd$, the strategy will eventually switch again (and this can happen consecutively only finitely often, because the priority memory can decrease only finitely often without taking on the value $\surd$) or become $\surd$, that is, Duplicator wins $G^{de}(\mathbf{Q}, \mathbf{Q}^{2a})$.

The proof for $\mathbf{Q}^{2a} \leq_{de} \mathbf{Q}$ is completely symmetrical. □

The shortcut automaton

As in Section 2.5, we can choose, for every equivalence class C, a topologically maximal representative r_C of C and replace every transition (p,a,q) such that $q \in C$ but $r_C \not\leq_R q$ by a transition (p,a,r_C). That is, the following lemma is the translation of Corollary 2.2 to this setting; also, the proof is the same as for Corollary 2.2.

Lemma 3.4 (shortcut automaton) *Let $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,\Omega)$ be a normalized and homogenized APA, see above. Let $C_0 \leq C_1 \leq \ldots \leq_T C$ be a topological sorting of the SCCs $C_0,\ldots,C_n$ of the directed graph $A_G = (Q,\{(q,q') \mid \exists a \in \Sigma : (q,a,q') \in \Delta\})$.*

For every de*-equivalence class D, let* $\mathrm{rep}(D) \in D$ *be a representative such that if* $\mathrm{rep}(D) \in C_i$ *then $D \cap C_j = \emptyset$, for all $j > i$.*

Let $\leq_R$ be the reachability preorder on A_G, i. e., for all $q,q' \in Q$, $q \leq_R q'$ if and only if there is a path in A_G from q to q'.

We define

$$\mathbf{Q}^{2b} = (Q,\Sigma,\mathrm{rep}([q_I]_{de}),\Delta',E,U,\Omega) \tag{3.33}$$

by

$$\begin{aligned} \Delta' \;=\; & \Delta \setminus \{(q,a,q') \in \Delta \mid \mathrm{rep}([q']_{de}) \not\leq_R q'\} \\ & \cup \{(q,a,\mathrm{rep}(D)) \mid D \in Q/\equiv_{de}, \\ & \qquad \exists q' \in D\colon (q,a,q') \in \Delta, \mathrm{rep}(D) \not\leq_R q'\} \;. \end{aligned} \tag{3.34}$$

Then, $\mathbf{Q}^{2b} \equiv_{de} \mathbf{Q}$. We call $\mathbf{Q}^{2b}$ the shortcut automaton *of* $\mathbf{Q}$.

The 0-1-minimax automaton

It is easy to see that, in analogy to Remark 1.4, existential states of minimal even priority only need maximal successors w. r. t. $\leq_{de}$. Dually, universal states with minimal odd priority only need minimal successors. With the same reasoning, it follows that if there is are two transitions $(q,a,q'),(q,a,q'')$ such that $q \in E$, $\Omega(q') = 0$ and $q'' \leq_{de} q'$, then the transition (q,a,q'') can be deleted, in analogy to Remark 1.5. Dually, a transition (q,a,q'') from a universal state q can be deleted in favor of a transition (q,a,q') such that $q' \leq_{de} q''$ and $\Omega(q') = 1$.

Lemma 3.5 (0-1-minimaxing) *Let $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,\Omega)$ be a normalized, homogenized and shortcut APA as defined above. Let $\mathbf{Q}^{2c}$ be defined like $\mathbf{Q}$, but*

with Δ' instead of Δ, where

$$\begin{aligned}\Delta' = \Delta \;&\setminus\; \{(q,a,q') \mid q \in E, \Omega(q) = 0, q' \notin \max_a^{de}(q)\} \\ &\setminus\; \{(q,a,q') \mid q \in U, \Omega(q) = 1, q' \notin \min_a^{de}(q)\} \\ &\setminus\; \{(q,a,q'') \mid q \in E, \exists (q,a,q') \in \Delta\colon q' \neq q'' \wedge q'' \leq_{de} q' \wedge\ \Omega(q') = 0\} \\ &\setminus\; \{(q,a,q'') \mid q \in U, \exists (q,a,q') \in \Delta\colon q' \neq q'' \wedge q' \leq_{de} q'' \wedge \Omega(q') = 1\}\ .\end{aligned} \tag{3.35}$$

Then, $\mathbf{Q} \equiv_{de} \mathbf{Q}^{2c}$. We call this step 0-1-minimaxing.

The reachability minimax automaton

The next lemma is an adaptation of Lemma 2.3, with the same proof.

Lemma 3.6 (reachability minimaxing) *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$ be an alternating parity automaton simplified like $\mathbf{Q}^{2c}$ in Lemma 3.5. Let $\leq_R$ be the reachability preorder of Lemma 3.4.*

Let $\mathbf{Q}^{2d} = (Q, \Sigma, q_I, \Delta', E, U, \Omega)$ defined by

$$\begin{aligned}\Delta' = \Delta \;&\setminus\; \{(q,a,q') \mid q \in E, \exists p \in \Delta(q,a)\colon q' \leq_{de} p \text{ and } p \not\leq_R q\} \\ &\setminus\; \{(q,a,q') \mid q \in U, \exists p \in \Delta(q,a)\colon p \leq_{de} q' \text{ and } p \not\leq_R q\}\ .\end{aligned} \tag{3.36}$$

Then, $\mathbf{Q} \equiv_{de} \mathbf{Q}^{2d}$. We call this step reachability minimaxing.

We call the automaton to which the simplifications of steps (1) to (2d) have been applied the de-*simplified* automaton (of $\mathbf{Q}$).

Note that the *de*-simplified automaton has the following property: Every two *de*-equivalent states belong to the same strongly connected component of the automaton's transition graph and have the same priority.

The rl-edge-reduced automaton

We can delete transitions in an alternating parity automaton based on right-hand and left-hand delayed simulation.

Lemma 3.7 (rl-edge-reduced automaton) *Let $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$ be an alternating parity automaton, and let $\mathbf{R}$ be the* de-*simplified automaton of $\mathbf{Q}$. Let $\mathbf{Q}^{rl}$ be defined like $\mathbf{R}$, but with Δ^{rl} instead of Δ, where*

$$\begin{aligned}\Delta^{rl} = \Delta \;&\setminus\; \{(q,a,q') \mid q \in E, \exists p \in \Delta(q,a)\colon q' <^l_{de} p\} \\ &\setminus\; \{(q,a,q') \mid q \in U, \exists p \in \Delta(q,a)\colon p <^r_{de} q'\}\ .\end{aligned} \tag{3.37}$$

Then, $\mathbf{Q} \equiv_{de} \mathbf{R} \equiv_{de} \mathbf{Q}^{rl}$.

We call $\mathbf{Q}^{rl}$ the rl-edge-reduced automaton *of $\mathbf{Q}$.*

Proof. Using the proofs in Subsection 3.4.2, it is easy to show that an automaton $\mathbf{Q}^r$ with transition relation

$$\Delta^r = \Delta \setminus \{(q,a,q') \mid q \in U, \exists p \in \Delta(q,a) \colon p <^r_{de} q'\} \tag{3.38}$$

is equivalent w. r. t. $\equiv^r_{de}$ (and hence w. r. t. $\equiv_{de}$) to $\mathbf{Q}$. Similarly, an automaton $\mathbf{Q}^l$ with transitions

$$\Delta^{rl} = \Delta \setminus \{(q,a,q') \mid q \in E, \exists p \in \Delta(q,a) \colon q' <^l_{de} p\} \tag{3.39}$$

is equivalent w. r. t. $\equiv^l_{de}$ and $\equiv_{de}$ to $\mathbf{Q}$.

Now $\mathbf{Q}^{rl}$ evolves from $\mathbf{Q}^r$ by removing only transitions starting at existential states. Consequently, $\mathbf{Q}^{rl} \leq_{de} \mathbf{Q}^r \equiv_{de} \mathbf{Q}$. But $\mathbf{Q}^{rl}$ also evolves from $\mathbf{Q}^l$ by removing only transitions from universal states. Hence $\mathbf{Q}^{rl} \geq_{de} \mathbf{Q}^l \equiv_{de} \mathbf{Q} \equiv_{de} \mathbf{Q}^r \geq_{de} \mathbf{Q}^{rl}$.

More precisely, this argumentation shows that $\mathbf{Q}^{rl} \leq^r_{de} \mathbf{Q} \leq^l_{de} \mathbf{Q}^{rl}$. □

Simultaneous quotienting w. r. t. left-hand and right-hand delayed simulation

Although quotienting w. r. t. $\leq^l_{de}$ may change the relation $\leq^r_{de}$ and vice versa, it is possible to apply a *simultaneous* quotienting w. r. t. both relations. The resulting automaton may not be equivalent to the original automaton w. r. t. both $\equiv^l_{de}$ and $\equiv^r_{de}$, but it will be equivalent w. r. t. $\equiv_{de}$.

We start with the following corollary which is basically similar to Corollary 1.3, but we also exploit the deletion of transitions according to Lemma 3.7.

Corollary 3.6 *Let* $\mathbf{Q} = (Q,\Sigma,q_I,\Delta,E,U,\Omega)$ *be a* de*-simplified alternating parity automaton, and let* $\mathbf{Q}^{rl} = (Q,\Sigma,q_I,\Delta^{rl},E,U,\Omega)$ *be the rl-edge-reduced automaton of* $\mathbf{Q}$*. Let* $p,q \in Q$ *such that* $p \equiv^r_{de} q$ *or* $p \equiv^l_{de} q$ *in* $\mathbf{Q}$*, i. e., w. r. t. the transition relation* Δ*. Let* $a \in \Sigma$*.*

If $\{p,q\} \subseteq E$ *or* $\{p,q\} \subseteq U$ *then for every* $p' \in \Delta^{rl}(p,a)$ *there is a* $q' \in \Delta^{rl}(q,a)$ *such that* $p' \equiv_{de} q'$*.*

Else for every $p' \in \Delta^{rl}(p,a)$ *and for every* $q' \in \Delta^{rl}(q,a)$*, we have* $p' \equiv_{de} q'$*.*

We now define a partition of the state set of an alternating parity automaton into equivalence classes; these equivalence classes will serve as states of the simultaneous quotient.

Let $\mathbf{Q}$ be an alternating parity automaton as above, and let $\mathbf{Q}'$ be the *de*-simplified automaton of $\mathbf{Q}$. Let $\leq^r_{de}$, $\equiv^r_{de}$, $\leq^l_{de}$ and $\equiv^l_{de}$ denote the simulation preorders and equivalences of Section 3.4 with respect to $\mathbf{Q}'$. Let $\mathbf{Q}^{rl}$ be the rl-edge-reduced automaton of $\mathbf{Q}$ (and $\mathbf{Q}'$) as defined in Lemma 3.7.

Let $R = \{R_0, \ldots, R_n\}$ be the set of *r-de*-equivalence classes of Q, i. e., R is the partition of Q according to $\equiv^r_{de}$. Let $L = \{L_0, \ldots, L_m\}$ be the partition according to $\equiv^l_{de}$.

Starting from these partitions, we define a partition RL of Q as follows.

1. If there are an $R_i \in R$ and an $L_j \in L$ such that $R_i \subseteq L_j$, delete R_i from R. Conversely, if $L_j \subset R_i$, delete L_j from L.

2. If step 1 cannot be applied any further: If there are $R_i \in R$, $L_j \in L$ such that $R_i \cap L_j \neq \emptyset$, let $R_i = R_i \setminus (L_j \cap E)$ and $L_j = L_j \setminus (R_i \cap U)$.

If step 2 cannot be applied any further, the partition RL is $R \cup L$.

Let $\equiv_{rl}$ be the equivalence relation on $Q \times Q$ such that RL is its set of equivalence classes. Obviously, $q \equiv_{rl} q'$ implies $q \equiv_{de} q'$.

Lemma 3.8 (simultaneous quotienting) *Let* $\mathbf{Q}$, $\mathbf{Q}^{rl}$, $\leq^r_{de}$, $\equiv^r_{de}$, $\leq^l_{de}$ *and* $\equiv^l_{de}$ *be defined like above. Especially,* $\mathbf{Q}^{rl}$ *is* de*-equivalent to* $\mathbf{Q}$ *but need not be* r-de*-equivalent or* l-de*-equivalent to* $\mathbf{Q}$.

Let RL be the partitioning of Q as defined above, and let $\equiv_{rl}$ *be the induced equivalence relation. Let R and L be the two sets of sets of states from above such that* $RL = R \cup L$.

Define $\mathbf{Q}^{de} = (Q^{rl}, \Sigma, [q_I]^{rl}, \Delta^{rl}_{de}, E^{rl}, U^{rl}, \Omega_{de})$, *where*

$$
\begin{aligned}
Q^{rl} &= \{[q]^{rl} \mid q \in Q\} = \{\{q' \in Q \mid q' \equiv_{rl} q\} \mid q \in Q\} \ , && (3.40)\\
\Delta^{rl}_{de} &= \{([p]^{rl}, a, [q]^{rl}) \mid a \in \Sigma, \\
&\qquad \exists p' \in [p]^{rl}, q' \in [q]^{rl} : (p', a, q') \in \Delta\} \ , && (3.41)\\
E^{rl} &= \{[q]^{rl} \in L \mid [q]^{rl} \subseteq E\} \\
&\quad \cup \{[q]^{rl} \in R \mid [q]^{rl} \cap E \neq \emptyset\} \ , && (3.42)\\
U^{rl} &= \{[q]^{rl} \in L \mid [q]^{rl} \cap U \neq \emptyset\} \\
&\quad \cup \{[q]^{rl} \in R \mid [q]^{rl} \subseteq U\} \ , && (3.43)\\
\Omega_{de}([q]^{rl}) &= \Omega(q) \ .^2 && (3.44)
\end{aligned}
$$

Then, $\mathbf{Q}^{de} \equiv_{de} \mathbf{Q}$.

Proof. We first introduce the following notation: We say that an automaton $\mathbf{Q}' = (Q', \Sigma, q'_I, \Delta', E', U', \Omega')$ results from an automaton $\mathbf{Q} = (Q, \Sigma, q_I, \Delta, E, U, \Omega)$ by *E-merging states* $M \subseteq Q$ if

[2] In the definition of Ω_{de}, we use the facts that all states in an *rl*-equivalence class are *de*-equivalent, and that we work on *de*-simplified automata. Hence all states in an *rl*-class have the same priority.

$$
\begin{aligned}
Q' &= Q \setminus M \cup \{M\} \ , && (3.45)\\
U' &= U \setminus M \ , && (3.46)\\
E' &= E \setminus M \cup \{M\} \ , && (3.47)\\
\textbf{if} \quad q_I \notin M \quad & \textbf{then } q'_I = q_I \textbf{ else } q'_I = M \ , && (3.48)\\
\Delta' &= \Delta \cap Q' \times \Sigma \times Q' \\
&\quad \cup \{(M,a,M) \mid \exists q,q' \in M : (q,a,q') \in \Delta\} \\
&\quad \cup \{(q,a,M) \mid q \in Q \setminus M, \exists q' \in M : (q,a,q') \in \Delta\} \\
&\quad \cup \{(M,a,q) \mid q \in Q \setminus M, \exists q' \in M : (q,a,q') \in \Delta\} \ , && (3.49)\\
\Omega'(q) &= \Omega(q) \text{ for } q \in Q \setminus M \ , && (3.50)\\
\Omega'(M) &= \min\{\Omega(q) \mid q \in M\} \ . && (3.51)
\end{aligned}
$$

If the states are *U-merged* instead, the definition is the same, only the new state M belongs to U' in this case, i. e., equations (3.46) and (3.47) are replaced by the following for U-merging.

$$
\begin{aligned}
U' &= U \setminus M \cup \{M\} \ , && (3.52)\\
E' &= E \setminus M \ . && (3.53)
\end{aligned}
$$

That is, $\mathbf{Q}^{de}$ results from $\mathbf{Q}^{rl}$ by

1. U-merging every set in L that contains a universal state, and
2. U-merging every set in R that contains only universal states, and
3. E-merging every set in R that contains an existential state, and
4. E-merging every set in L that contains only existential states.

We write, e. g., $\mathbf{Q}^{rl}[1]$ for the automaton that results from $\mathbf{Q}^{rl}$ by applying the merging 1, and, e. g., $\mathbf{Q}^{rl}[234]$ for the automaton resulting from applying the mergings 2, 3, and 4 to $\mathbf{Q}^{rl}$. Especially, we have $\mathbf{Q}^{de} = \mathbf{Q}^{rl}[1234] = \mathbf{Q}^{rl}[3241]$, i. e., the order inside the brackets is not important; we will use the order inside the brackets in which we have applied the mergings.

Now after mergings of sort 3 and 4, the resulting automaton $\mathbf{Q}^{rl}[34]$ still *de*-simulates $\mathbf{Q}^{rl}$ because we only introduce new paths at existential states and merge some universal states into existential states. Conversely, it is clear that $\mathbf{Q}^{rl}$ *de*-simulates $\mathbf{Q}^{rl}[12]$.

Consequently, to show that $\mathbf{Q} \equiv_{de} \mathbf{Q}^{de}$, we proceed as follows. We show that $\mathbf{Q}^{rl}[43] \leq^{r}_{de} \mathbf{Q}^{rl}$; then it follows that $\mathbf{Q}^{rl}[43]$ is *de*-equivalent to $\mathbf{Q}^{rl}$. We then show

that $\mathbf{Q}^{rl} \leq^{l}_{de} \mathbf{Q}^{rl}[4321]$; it follows that $\mathbf{Q}^{rl}$ and $\mathbf{Q}^{rl}[4321]$ are *de*-equivalent. Since $\mathbf{Q}^{rl} \equiv_{de} \mathbf{Q}$ by Lemma 3.7 and $\mathbf{Q}^{rl}[4321] = \mathbf{Q}^{de}$, we then have $\mathbf{Q} \equiv_{de} \mathbf{Q}^{de}$.

Now to show that $\mathbf{Q}^{rl}[43] \leq^{r}_{de} \mathbf{Q}^{rl}$, we proceed in two steps: We first show that $\mathbf{Q}^{rl}[4] \leq^{r}_{de} \mathbf{Q}^{rl}$ and then use this result to show that $\mathbf{Q}^{rl}[43] \leq^{r}_{de} \mathbf{Q}^{rl}$.

To first show that $\mathbf{Q}^{rl}[4] \leq^{r}_{de} \mathbf{Q}^{rl}$, we can use as the Duplicator strategy in $G^{r-de}(\mathbf{Q}^{rl}[4], \mathbf{Q}^{rl})$ the join of a $\equiv_{de}$-respecting Duplicator strategy $\sigma^{\equiv}$ for $G^{de}(\mathbf{Q}^{rl}[4], \mathbf{Q}^{rl})$ and a Duplicator winning strategy σ^{r} for $G^{r-de}(\mathbf{Q}^{rl}, \mathbf{Q}^{rl})$. From Corollary 3.6, it follows directly that there is a $\equiv_{de}$-respecting strategy for $G^{de}(\mathbf{Q}^{rl}[4], \mathbf{Q}^{rl})$. Note that if Duplicator uses a $\equiv_{de}$-respecting strategy in $G^{de}(\mathbf{Q}^{rl}[4], \mathbf{Q}^{rl})$, the two pebbles in that game always end their turns on states with the same priority, so there is no need for an updatable strategy as in the proofs of Subsection 3.4.2 and Section 1.5.

In the next step, we add mergings of sort 3. We can now use the well-known proof scheme with a joint and updatable strategy to show that still $\mathbf{Q}^{rl}[43] \leq^{r}_{de} \mathbf{Q}^{rl}$ holds: We join a $\sqsubseteq^{r}_{de}$-respecting Duplicator strategy for $G^{r-de}(\mathbf{Q}^{rl}[43], \mathbf{Q}^{rl}[4])$ and a Duplicator winning strategy for $G^{r-de}(\mathbf{Q}^{rl}[4], \mathbf{Q}^{rl})$, and we update this strategy whenever the priority of the state in the simulated automaton triggers a decrease of the priority memory of $G^{r-de}(\mathbf{Q}^{rl}[43], \mathbf{Q}^{rl})$. That is, this proof is similar to the proof of Theorem 3.4 that $\mathbf{Q}$ *r-de*-simulates $\mathbf{Q}^{se}_{r-de}$. We thus have $\mathbf{Q}^{rl}[43] \equiv_{de} \mathbf{Q}^{rl}$.

We can now first show that $\mathbf{Q}^{rl} \leq^{l}_{de} \mathbf{Q}^{rl}[432]$ and then show that $\mathbf{Q}^{rl} \leq^{l}_{de} \mathbf{Q}^{rl}[4321]$. The proofs are completely symmetric, i. e., to show $\mathbf{Q}^{rl} \leq^{l}_{de} \mathbf{Q}^{rl}[432]$, we join a Duplicator winning strategy for $G^{l-de}(\mathbf{Q}^{rl}, \mathbf{Q}^{rl}[43])$ with a $\equiv_{de}$-respecting strategy for $G^{l-de}(\mathbf{Q}^{rl}[43], \mathbf{Q}^{rl}[432])$. To show $\mathbf{Q}^{rl} \leq^{l}_{de} \mathbf{Q}^{rl}[4321]$, we use the updatable join of a Duplicator winning strategy for $G^{l-de}(\mathbf{Q}^{rl}, \mathbf{Q}^{rl}[432])$ and a $\sqsubseteq^{l}_{de}$-respecting strategy for $G^{l-de}(\mathbf{Q}^{rl}[432], \mathbf{Q}^{rl}[4321])$, i. e., the proof for this last step is similar to the proof that $\mathbf{Q} \leq^{l}_{de} \mathbf{Q}^{se}_{l-de}$.

We then have $\mathbf{Q} \equiv_{de} \mathbf{Q}^{rl} \equiv_{de} \mathbf{Q}^{rl}[4321] = \mathbf{Q}^{de}$. □

3.5.3 Example: Simplification algorithm

As an example for the application of the simplification algorithm, we reconsider the parity game $H_{2,2}$ of Figure 3.4 in Subsection 3.4.3. The following numeration refers to the steps of the simplification algorithm in Subsection 3.5.1.

(1) $H_{2,2}$ is normalized.

(2) The relation $\leq_{de}$ for $H_{2,2}$ is $q_{0i} \equiv_{de} q_{0j} \leq_{de} q_{11} \leq_{de} q_{13} \equiv_{de} q_{14}$ for $i, j < 5$, $q_{10} \leq_{de} q_{12} \leq_{de} q_{21} \leq_{de} q_{13}$ and $q_{20} \leq_{de} q_{21}$.

(2a) In the homogenized automaton $H^{2a}_{2,2}$, the priority of the states q_{0i} is 3 ($i < 5$); q_{13} and q_{14} already have the same priority.

(2b) The transition graph of $H^{2a}_{2,2}$ is strongly connected, so the shortcut automaton $H^{2b}_{2,2}$ is the same as $H^{2a}_{2,2}$.

(2c) We have q_{21} and q_{13} as successors of q_{12}, which is a universal state. The priority of q_{21} is 1, and $q_{21} \leq_{de} q_{13}$. That is, we can delete the transition (q_{12}, q_{13}) in $H^{2c}_{2,2}$.

(2d) The transition graph of $H^{2c}_{2,2}$ still is strongly connected, so no transitions can be deleted by reachability minimaxing in $H^{2d}_{2,2}$.

(3) Let $H'_{2,2} = H^{2d}_{2,2}$. The relation $\leq^l_{de}$ for $H'_{2,2}$ is the same as $\leq_{de}$ for $H_{2,2}$, and $\leq^r_{de}$ differs from $\leq^l_{de}$ in that $q_{12} \not\leq^r_{de} q_{21}$.

(3a) According to $\leq^l_{de}$, we delete the transitions (q_{11}, q_{10}), (q_{13}, q_{12}) and (q_{13}, q_{03}). According to $\leq^r_{de}$, we delete the transitions (q_{01}, q_{11}) and (q_{03}, q_{13}).

(3b) The set L of *l-de*-equivalence classes is

$$\{\{q_{10}\}, \{q_{20}\}, \{q_{11}\}, \{q_{12}\}, \{q_{21}\}, \{q_{13}, q_{14}\}, \{q_{0i} \mid i < 5\}\} \ ; \tag{3.54}$$

this is the same as the set R of *r-de*-equivalence classes. According to the partition rules, we set $R = \emptyset$, and L also is our partition RL. According to Lemma 3.8, $\{q_{13}, q_{14}\}$ is merged into a universal state $[q_{13}]$, and $\{q_{0i} \mid i < 5\}$ is merged into a universal state $[q_{00}]$.

The intermediate result is shown in Figure 3.9.

(4) We continue with this result as the new input of step (1) and call it $\mathbf{Q}$.

(1) This parity game $\mathbf{Q}$ is not strongly connected; there are six SCCs in the transition graph. In the normalized version, priorities 2 are changed to 0, and priority 3 to 1.

(2) The relation $\leq_{de}$ for the normalized $\mathbf{Q}$ now is very simple: The states $[q_{10}]$, $[q_{20}]$, $[q_{11}]$, $[q_{12}]$, $[q_{21}]$ and $[q_{13}]$ are all *de*-equivalent and strictly simulate $[q_{00}]$.

(2a) These *de*-equivalent states are assigned priority 0.

(2b) In the shortcut automaton, the transitions from $[q_{10}]$, $[q_{20}]$ and $[q_{12}]$ as well as the transition $([q_{11}], [q_{12}])$ are replaced by transitions to $[q_{13}]$.

(2c) 0-1-minimaxing then deletes the transition $([q_{11}], [q_{00}])$.

(3) The *de*-equivalent states are also all *r-de*-equivalent and are thus merged into a single existential state with priority 0 and only a self transition (the relation $\leq^l_{de}$ is strictly smaller). The state $[q_{00}]$ remains a single state with priority 1 and only a self transition.

That is, after two cycles of the simplification algorithm, $H_{2,2}$ is simplified to two states, and it is obvious that Player 1 wins at states $\{q_{ij} \mid (i = 1 \wedge j < 5) \vee (i = 2 \wedge j < 2)\}$ while Player 0 wins at states $\{q_{0i} \mid i < 5\}$.

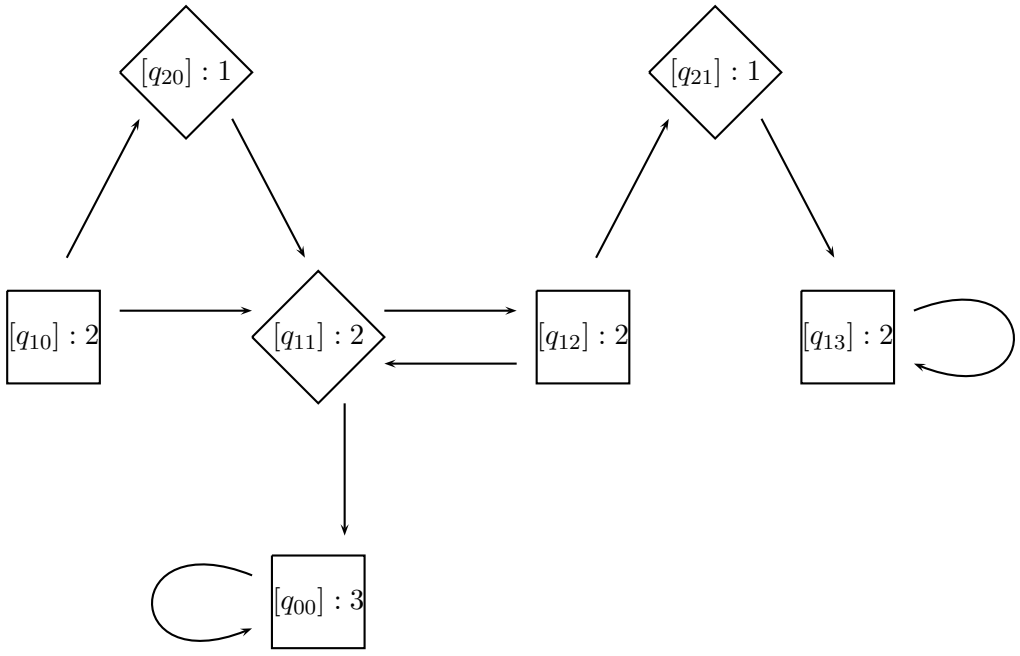

Figure 3.9: The intermediate result of the simplification algorithm applied to $H_{2,2}$

3.6 An Application to the μ-Calculus

It is possible to use our simulation-based simplifications for alternating parity automata for the simplification of parity games, in the hope that first simplifying and then solving a parity game is faster than solving the unsimplified parity game. But experiments with an implementation indicate that solving a given parity game using Jurdziński's lifting algorithm [Jur00] directly is faster than first simplifying the parity game using our approach and then solving it using Jurdziński's algorithm.

In this section, we outline the application of simulation-based simplifications to the modal μ-calculus [Koz83]. The modal μ-calculus is closely connected to alternating parity *tree* automata [Niw97], that is, for a modal μ-calculus formula, there is an equivalent alternating parity tree automaton (APTA, for short) such that the size of this automaton is linear in the length of the μ-calculus formula, and some simulation-based simplifications of the automaton can be seen as simplifications of the equivalent formula. So the connection between modal μ-calculus and alternating parity tree automata resembles the connection between LTL and alternating Büchi (word) automata analyzed in Chapter 2.

There are several new technical difficulties in the application to APTA from modal μ-calculus. First, runs of APTA are defined on Kripke structures, not on words. A Kripke structure is a (not necessarily finite) directed graph together with an interpretation of propositions in each vertex of the graph. For our purposes of defining a simulation relation via a simulation game for APTA, however, we can abstract from the structure of the automaton input as a graph: We assume that player Spoiler in the simulation game on APTA chooses a graph vertex given as an interpretation of the propositions at the beginning of every round of a play of the simulation game, such that a play only follows a single path through a graph where this input graph is not fixed a priori, just as the input word of play of a simulation game for word automata is not fixed from the start.

Another difficulty lies in the combination of the acceptance condition and the occurrence of unlabeled or ε-transitions in APTA from μ-calculus. In Chapter 2, alternating Büchi automata from LTL are constructed in such a way that only the acceptance status of the state reached at the end of each simulation game round is relevant. For APTA from μ-calculus, at least the priorities of states corresponding to fixed point formulas are relevant, but it cannot be ensured that such a relevant state is visited by a pebble only at the end of a game round. To ensure this property, we restrict ourselves to a subclass of μ-calculus formulas, which we call the *strictly guarded* formulas. This is class is introduced in Subsection 3.6.1. As discussed below, we can embed formulas of the logic CTL (Computation Tree Logic) [EH85] in the class of strictly guarded μ-calculus formulas.

In the definition of modal μ-calculus, APTA and the translation of μ-calculus formulas to APTA, we basically follow the definitions in the survey of

Wilke [Wil01]. We will point out differences of our definitions to [Wil01], and we keep our usual convention that, e. g., Q is the set of states and Σ is the set of propositions. We do not provide theorems and proofs in this section, but outline the definition of a delayed simulation relation and simulation-based simplification steps for an example formula.

We first define the syntax of modal μ-calculus, APTA, and the translation from μ-calculus formulas to APTA. We only give an informal definition of APTA acceptance here, and we define the semantics of μ-calculus formulas only by way of defining an equivalent APTA for a μ-calculus formula; see [Wil01] for detailed definitions.

We then define a delayed simulation game and a delayed simulation relation for APTA constructed from μ-calculus and illustrate possible simplifications based on this relation for an example automaton and formula.

3.6.1 The modal μ-calculus, APTA, and a translation from μ-calculus to APTA

Syntax of modal μ-calculus

According to [Wil01], *modal μ-calculus is modal logic augmented by least and greatest fixed points operators.* The set L_μ of μ-calculus formulas is defined inductively as follows, for a fixed set of propositions Σ. The propositions play a double role both as propositional variables and as propositional constants (in LTL formulas, all propositions are propositional constants in this sense).

- The formulas $\mathtt{tt}$ and $\mathtt{ff}$ are in L_μ.
- The literals a and $\neg a$ are in L_μ, for every $a \in \Sigma$.
- For all $\psi, \rho \in L_\mu$, $\psi \vee \rho$ and $\psi \wedge \rho$ are formulas in L_μ.
- If $\psi \in L_\mu$, then $\Box\psi$ and $\Diamond\psi$ are formulas in L_μ. The operators $\Box$ and $\Diamond$ are called *next modalities.*
- If $Z \in \Sigma$ and $\psi \in L_\mu$ such that Z occurs in ψ at most as a positive literal, then $\mu Z\psi$ and $\nu Z\psi$ are formulas in L_μ. The operators μ and ν are the *fixed point operators* or *fixed point quantifiers.* Formulas of the form $\mu Z\psi$ and $\nu Z\psi$ are called *fixed point formulas.*

A formula in L_μ is in *normal form* if every proposition Z is only quantified at most once (i. e., it is in the scope of at most one fixed point operator μZ or νZ) and, if it is quantified, then all occurrences of Z are in the scope of this quantification. If a proposition is quantified in a formula, we say that it is *bound,* else it is *free.*

If a proposition Z is bound in a formula φ in normal form, then φ_Z is the unique fixed point subformula of φ such that $\varphi_Z = \mu Z\psi$ or $\varphi_Z = \nu Z\psi$ and Z occurs in ψ. A formula is *guarded* if every quantified proposition Z also is in the scope of a next modality, and this next modality itself is in the scope of the fixed point quantifier to which Z is bound. For a given μ-calculus formula, an equivalent guarded formula in normal form can be computed in linear time [KVW00, Wil01].

As stated in the introduction, we impose here an even stronger requirement, namely, that every bound proposition occurs in a subformula $\Box Z$ or $\Diamond Z$ only, i. e., a next modality has to directly precede every bound proposition. If this is the case for a μ-calculus formula, we say that it is *strictly guarded.* Note that this is a real restriction on the structure of the formulas—an equivalent strictly guarded formula cannot be computed in linear time for an arbitrary formula.

But we can express CTL formulas as strictly guarded μ-calculus formulas in normal form, by using the following standard translation of the temporal CTL operators:

$$\mathsf{AX}\psi \equiv \Box[\psi]_{L_\mu} \qquad \mathsf{EX}\psi \equiv \Diamond[\psi]_{L_\mu} \ , \tag{3.55}$$

$$\mathsf{AF}\varphi \equiv \mu Z([\varphi]_{L_\mu} \vee \Box Z) \qquad \mathsf{EF}\psi \equiv \mu Z([\varphi]_{L_\mu} \vee \Diamond Z) \ , \tag{3.56}$$

$$\mathsf{AG}\varphi \equiv \nu Z([\varphi]_{L_\mu} \wedge \Box Z) \qquad \mathsf{EG}\psi \equiv \nu Z([\varphi]_{L_\mu} \wedge \Diamond Z) \ , \tag{3.57}$$

$$\mathsf{A}[\psi\mathsf{U}\rho] \equiv \mu Z([\rho]_{L_\mu} \vee ([\psi]_{L_\mu} \wedge \Box Z)) \qquad \mathsf{E}[\psi\mathsf{U}\rho] \equiv \mu Z([\rho]_{L_\mu} \vee ([\psi]_{L_\mu} \wedge \Diamond Z)) \ , \tag{3.58}$$

$$\mathsf{A}[\psi\mathsf{R}\rho] \equiv \nu Z([\rho]_{L_\mu} \wedge ([\psi]_{L_\mu} \vee \Box Z)) \qquad \mathsf{E}[\psi\mathsf{R}\rho] \equiv \nu Z([\rho]_{L_\mu} \wedge ([\psi]_{L_\mu} \vee \Diamond Z)) \ , \tag{3.59}$$

where $[\psi]_{L_\mu}$ denotes the CTL formula ψ as a formula in L_μ and Z does not occur in $[\psi]_{L_\mu}$ and $[\rho]_{L_\mu}$. See, e. g., [Eme90] for more details on CTL.

In the following, we only consider strictly guarded formulas in normal form.

Alternating parity tree automata (APTA)

An alternating parity tree automaton is a tuple

$$\mathbf{Q} = (Q, q_I, \delta, \Omega) \tag{3.60}$$

where Q is a finite set of states, q_I is the initial state of $\mathbf{Q}$, δ is a transition function, and Ω is a priority function $Q \to \omega$.

The transition function δ maps every state to a transition condition over Q, where a transition condition is defined by:

- $\mathtt{tt}$ and $\mathtt{ff}$ are transition conditions over Q, called *final transitions,*

- for all $a \in \Sigma$, a and $\neg a$ are transitions conditions over Q, called *propositional transitions,*
- for all $q \in Q$, q is a transition condition over Q, called *simple transition*,
- for all $q \in Q$, $\Box q$ and $\Diamond q$ are transition conditions over Q, called *next transitions,*
- for all $q, q' \in Q$, $q \wedge q'$ and $q \vee q'$ are transition conditions over Q. A transition condition $q \wedge q'$ is called a *universal transition* and a transition $q \vee q'$ is called an *existential transition.*

A state $q \in Q$ is a *universal state* if $\delta(q) = q' \wedge q''$ for some $q', q'' \in Q$, else it is an *existential state.*

The input of an APTA is a pointed Kripke structure. A pointed Kripke structure over a set of propositions Σ is given as

$$\mathbf{K} = (W, A, \kappa, w) \tag{3.61}$$

where (W,A) is a directed graph, κ is a function $\Sigma \to 2^W$, and $w \in W$. The set W is called the *universe* of $\mathbf{K}$, and its elements are called *worlds*. Both the universe and the *accessibility relation* $A \subseteq W \times W$ may be infinite. The function κ is an *interpretation* assigning to every proposition the set of worlds where it holds true. The world w is the *distinguished world* of $\mathbf{K}$; the computation of an APTA on a pointed Kripke structure starts at the distinguished world.

The following paragraph, quoted from [Wil01] with remarks and changes in brackets, gives an informal description of a computation (or run) of an APTA on a pointed Kripke structure.

> Alternating [parity] tree automata are finite-state devices designed to accept or reject pointed Kripke structures. The computation of an alternating [parity] tree automaton on a pointed Kripke structure proceeds in rounds. At the beginning of every round there are several copies of the alternating [parity] tree automaton in different worlds of the Kripke structure, each of them in its own state; some worlds might be occupied by many copies, others might not accommodate a single one. During a round, each copy splits up in several new copies, which are sent to neighbored worlds and change their states, all this done according to the transition function. Initially, there is only one copy of the alternating tree automaton; it resides in the distinguished world of the pointed Kripke structure and starts in the initial state of the alternating [parity] tree automaton. To determine acceptance or rejectance of a computation of an alternating [parity] tree automaton

> on a pointed Kripke structure the entire computation tree is inspected; acceptance is then defined via path conditions for the infinite branches of the computation tree. Namely, every state [is] assigned a priority and an infinite branch of the computation will be accepting if the [minimum] priority occurring infinitely often is even; a computation tree will be accepting if each of its infinite branches is accepting.

That *each copy splits up in several new copies, which are sent to neighbored worlds and change their states, all this done according to the transition function* means

- if a copy (a computation branch) takes a final transition, it stops, and it accepts if the transition condition is $\mathtt{tt}$ and it rejects if it is $\mathtt{ff}$.
- It also stops taking a propositional transition. If it stops on a world w taking a propositional transition a, then it accepts if and only if $w \in \kappa(a)$; if the propositional transition is $\neg a$, it accepts if and only if $w \notin \kappa(a)$.
- If a copy takes a simple transition q with $q \in Q$, then it only changes its state to q, but it stays on the same world.
- If a copy takes a universal transition $q \wedge q'$, then it also stays on the same world, but it splits up in two new copies, one in state q and one in state q'.
- If a copy takes an existential transition $q \wedge q'$, there is a nondeterministic choice whether the computation continues in state q or in state q'; the current world stays the same.
- A copy of the APTA changes the current world of the input Kripke structure only via next transitions.
  - By taking a next transition $\Diamond q$ in a world w, the automaton nondeterministically chooses a successor world w' such that $(w, w') \in A$ and continues the computation branch on the world w' in state q. The computation branch ends and rejects if there is no such successor world.
  - By taking a next transition $\Box q$ in a world w, the computation branches into as many copies as there are worlds w' such that $(w, w') \in A$, and one such copy in state q is sent to every such world w'. If there are no successor worlds of w, then, by taking a transition $\Box q$, the respective copy stops and accepts.

Existential and universal transitions and next transitions can also be considered in terms of word games—or "Kripke structure games", more precisely. That

is, we may assume that there are two players Pathfinder and Automaton such that Pathfinder chooses one successor state for universal transitions and one successor world for transitions $\Box q$ while Automaton chooses the successor states for existential transitions and for transitions $\Diamond q$.

The translation from L_μ to APTA

For a guarded formula $\varphi \in L_\mu$ in normal form, we define an equivalent APTA

$$\mathbf{Q}(\varphi) = (Q, q_I, \delta, \Omega) \tag{3.62}$$

as follows.

- The set of states Q contains, for every subformula ψ of φ, a state denoted $\langle\psi\rangle$.
- The initial state is $q_I = \langle\varphi\rangle$.
- The transition function δ is defined by
  - $\delta(\langle\mathsf{tt}\rangle) = \mathsf{tt}$ and $\delta(\langle\mathsf{ff}\rangle) = \mathsf{ff}$,
  - for $a \in \Sigma$, $\delta(\langle a\rangle) = a$ if a is free in φ, and $\delta(\langle a\rangle) = \langle\varphi_a\rangle$ if a is bound in φ,
  - for $a \in \Sigma$, $\delta(\langle\neg a\rangle) = \neg a$,
  - $\delta(\langle\psi \wedge \rho\rangle) = \langle\psi\rangle \wedge \langle\rho\rangle$ and $\delta(\langle\psi \vee \rho\rangle) = \langle\psi\rangle \vee \langle\rho\rangle$,
  - $\delta(\langle\Box\psi\rangle) = \Box\langle\psi\rangle$ and $\delta(\langle\Diamond\psi\rangle) = \Diamond\langle\psi\rangle$,
  - $\delta(\langle\mu Z\psi\rangle) = \langle\psi\rangle$ and $\delta(\langle\nu Z\psi\rangle) = \langle\psi\rangle$.

That is, in the APTA $\mathbf{Q}(\varphi)$ for a strictly guarded formula φ in normal form, a state $\langle z\rangle$ with z a bound proposition is reachable only via a next transition $\delta(\langle\Box z\rangle) = \Box\langle z\rangle$ or $\delta(\langle\Diamond z\rangle) = \Diamond\langle z\rangle$.

Different from Wilke [Wil01], we will, as in the previous sections, use the convention that the Automaton player wins a game if the *smallest* priority visited infinitely often is even. We define the priority function Ω via the transition graph of $\mathbf{Q}(\varphi)$ and the alternation depth of φ as follows. The *transition graph* of $\mathbf{Q}(\varphi)$ is the directed graph $\mathbf{G}(\varphi) = (Q, E_\delta)$ with $(q, q') \in E_\delta$ if and only if q' occurs in $\delta(q)$; especially, states $\langle\mathsf{tt}\rangle$, $\langle\mathsf{ff}\rangle$, $\langle\neg a\rangle$, and $\langle a\rangle$ for free a do not have outgoing edges. Following [Wil01], we define the *alternation depth* $\alpha(\varphi)$ of a formula φ as follows.

- If no proposition is bound in φ, then $\alpha(\varphi) = 0$.

- If φ is a fixed point formula $\eta Z\psi$ with $\eta \in \{\mu,\nu\}$ such that Z is free in ψ and ψ is without fixed point operators, then $\alpha(\varphi) = 1$.
- Else suppose that $\alpha(\psi)$ has already been computed for every proper subformula ψ of φ, and let $m = \max\{\alpha(\psi) \mid \psi \text{ a proper subformula of } \varphi\}$. If $\varphi = \eta Z\psi$ is a fixed point formula and there is a fixed point formula $\eta' Y\rho$ such that $\alpha(\eta' Y\rho) = m$ and $\langle \eta' Y\rho\rangle$ is in the SCC of $\langle\varphi\rangle$ in $\mathbf{G}(\varphi)$ and $\eta \neq \eta'$, then $\alpha(\varphi) = m+1$, else $\alpha(\varphi) = m$.

We then define $\Omega\colon Q \to \omega$ via an auxiliary function Ω'. For a state $\langle \mu Z\psi\rangle$ such that $\alpha(\mu Z\psi) > 0$, $\Omega'(\langle \mu Z\psi\rangle) = 2\lceil \alpha(\mu Z\psi)/2\rceil - 1$, and for a state $\langle \nu Z\psi\rangle$ such that $\alpha(\nu Z\psi) > 0$, $\Omega'(\langle \nu Z\psi\rangle) = 2\lfloor \alpha(\nu Z\psi)/2\rfloor$.

Let m be the maximal value of Ω' for a state of $\mathbf{Q}(\varphi)$. We define Ω as follows.

1. If ψ is a fixed point subformula of φ and m is even, then $\Omega(\langle\psi\rangle) = m - \Omega'(\langle\psi\rangle)$.
2. If ψ is a fixed point subformula of φ and m is odd, then $\Omega(\langle\psi\rangle) = m - \Omega'(\langle\psi\rangle) + 1$.
3. If Z is a bound proposition, then $\Omega(\langle Z\rangle) = \Omega(\varphi_Z)$.
4. If $\psi = \circ_1 \ldots \circ_n Z$ with $\circ_i \in \{\Box, \Diamond\}$ for all $1 \leq i \leq n$ and Z a bound proposition, then $\Omega(\langle\psi\rangle) = \Omega(\langle Z\rangle)$.
5. If $\Omega(\langle\psi\rangle)$ is not defined after steps 1 to 4, then if Ω is defined for some state of the SCC of $\langle\psi\rangle$ in $\mathbf{G}(\varphi)$, then $\Omega(\langle\psi\rangle)$ is the maximal priority already assigned to a state in this SCC.
6. If $\Omega(\langle\psi\rangle)$ is not defined after steps 1 to 5, then $\langle\psi\rangle$ is a trivial SCC in $\mathbf{G}(\varphi)$, and $\Omega(\langle\psi\rangle)$ is the maximal priority assigned to a state so far, or $\Omega(\langle\psi\rangle) = 0$, if no priority has been assigned in steps 1 to 5.

Note that Ω' is the priority function as defined in [Wil01]. Steps 5 and 6 are in analogy to the remarks in [Wil01, Subsection 2.2.5], while we add steps 3 and 4 for technical reasons.

As an example of computing the alternation depth and the priorities, consider the formula

$$\varphi = \mu Z(\varphi_1 \vee \Diamond Z) \ , \tag{3.63}$$

where

$$\varphi_1 = \nu Y(\varphi_2 \wedge \Diamond Y) \ , \tag{3.64}$$

$$\varphi_2 = \mu X(a \vee (\varphi_3 \wedge \Diamond X)) \ , \text{ and} \tag{3.65}$$

$$\varphi_3 = \mu W((a \vee b) \vee (\Diamond W \vee \Diamond Z)) \ . \tag{3.66}$$

The alternation depth of φ_3 is 1, so the alternation depth of φ_2 is also 1, since $\langle\varphi_3\rangle$ is not in the same SCC of $\mathbf{G}(\varphi_2)$ as $\langle\varphi_2\rangle$. Similarly, the alternation depth of φ_1 is 1. The alternation depth of φ is 2, because φ_1 is in the same SCC of $\mathbf{G}(\varphi)$ as φ_1 (notice the proposition Z in φ_3), and φ is a least fixed point formula while φ_1 is a greatest fixed point formula.

Consequently, we have $\Omega'(\langle\varphi_3\rangle) = \Omega'(\langle\varphi_2\rangle) = 1$, $\Omega'(\langle\varphi_1\rangle) = 0$, and $\Omega'(\langle\varphi\rangle) = 1$. So the maximal value w. r. t. Ω' is 1, and thus $\Omega(\langle\varphi_3\rangle) = \Omega(\langle\varphi_2\rangle) = \Omega(\langle\varphi\rangle) = 1$ and $\Omega(\langle\varphi_1\rangle) = 2$. The priority of, e. g., $\langle\varphi_2 \wedge \Diamond Y\rangle$ is 2 because this state is in the SCC of $\langle\varphi_1\rangle$, and the priority of $\langle\Diamond X\rangle$ is 1 because the priority of $\langle X\rangle$ is 1, which in turn follows from the priority of φ_2. It follows that every priority in $\mathbf{Q}(\varphi)$ is either 1 or 2.

3.6.2 Simulation relations for APTA from strictly guarded L_μ-formulas in normal form

We now define a basic delayed simulation game and, using this game, a delayed simulation relation for alternating parity tree automata constructed from μ-calculus formulas as defined above.

The basic ideas are:

- As for the other simulation games in this work, there are two players, Spoiler and Duplicator, who move two pebbles, a red one and a green one, on the transition graphs of two automata (often, the two automata are one and the same, but the pebbles start at different states).

- A play can be interpreted as a run of the two automata along the same path of a Kripke structure. This path is not fixed but is the result of a sequence of choices of Spoiler, just as Spoiler constructs a sequence of letters or terms in the other simulation games. That is, at the beginning of each round, Spoiler chooses a term over the propositional constants, similar to what is done in Section 2.3.

- Universal and existential transitions as well as simple transitions are treated like ε-labeled transitions in Section 2.3.

- A play ends early if, at the end of a round, at least one of the players has chosen a final transition or a propositional transition, that is, if a pebble has to be moved on a state without an outgoing edge in the transition graph.

- Since we assume that the two automata follow the same path, a play also ends early if both pebbles are moved along a next transition at the end of a round, but with different modalities. In this case, Spoiler wins immediately.

We justify this as follows: If the red pebble (the pebble in the automaton to be simulated) moves along a transition $\Box q$ while the green pebble moves along a transition $\Diamond q'$, we can assume that the path of the two automata has reached a world without a successor world, and in this case, the simulated automaton accepts immediately while the simulating automaton rejects immediately. If, on the other hand, the red pebble moves along a transition $\Diamond q$ while the green pebble moves along a transition $\Box q'$, then the paths of the two automata might deviate such that, consequently, Spoiler can present two different terms at the beginning of each round, one for the red pebble and one for the green pebble. We assume that Spoiler could use this possibility to win the game, so we stop at this point and declare Spoiler the winner. (This is a simplification, of course, since the simulated automaton might accept every Kripke structure starting from the current state of the green pebble, but since this simplification can only lead to a smaller relation, this is not a problem.)

For a detailed definition of the simulation game, we first define the function posTerms, as in Section 2.3. The function posTerms here is a mapping from the guarded μ-calculus formulas in normal form to term_Σ and is defined inductively as follows.

$$\text{posTerms}(\mathsf{ff}) = \emptyset \ , \tag{3.67}$$

$$\text{posTerms}(\varphi) = \{\varphi\}, \text{ for } \varphi \in \{\mathsf{tt}, a, \neg a \mid a \in \Sigma\} \ , \tag{3.68}$$

$$\text{posTerms}(\psi \vee \rho) = \text{posTerms}(\psi) \cup \text{posTerms}(\rho) \ , \tag{3.69}$$

$$\text{posTerms}(\psi \wedge \rho) = \{t \wedge t' \mid t \in \text{posTerms}(\psi),\ t' \in \text{posTerms}(\rho)\} \ , \tag{3.70}$$

$$\text{posTerms}(\circ\psi) = \{\mathsf{tt}\}, \text{ for } \circ \in \{\Box, \Diamond\} \ , \tag{3.71}$$

$$\text{posTerms}(\eta Z \varphi) = \text{posTerms}(\varphi)[\mathsf{tt}/Z], \text{ for } \eta \in \{\mu, \nu\}, \text{ i. e.,} \tag{3.72}$$

Z is substituted by tt in the terms in $\text{posTerms}(\varphi)$; this is equivalent to removing every occurrence of Z from these terms.

Now let $\mathbf{Q}(\varphi) = (Q, q_I, \delta, \Omega)$ be an alternating parity tree automaton constructed from a strictly guarded μ-calculus formula in normal form as described above. A round of the basic simulation game proceeds very similar to what is described in Section 2.3, that is, we assume that, at the beginning of a round, the red pebble (the pebble to be simulated) is placed on $\langle\psi\rangle \in Q$ and the green pebble (the simulating pebble) on $\langle\rho\rangle \in Q$. Then, the players Spoiler and Duplicator play as follows.

1. Spoiler chooses a term $t \in \text{posTerms}(\psi)$.

   Spoiler loses early if $\text{posTerms}(\psi) = \emptyset$ or $t \equiv \mathsf{ff}$ for all $t \in \text{posTerms}(\psi)$.

2. The progression of the round depends on whether the pebbles are placed on an existential or on a universal state, and on the statuses of the pebbles (free or locked). Initially, both pebbles are free, and the round ends when both pebbles are locked. A player moves a free pebble on a state q depending on the transition condition $\delta(q)$, i. e.,

   (a) if $\delta(q)$ is a final or propositional transition, then the pebble becomes locked,

   (b) if $\delta(q) = q'$ is a simple transition, then the round continues with the still free pebble on q',

   (c) if $\delta(q) = \Box q'$ or $\delta(q) = \Diamond q'$ is a next transition, the round continues with the now locked pebble on q',

   (d) if $\delta(q) = q' \vee q''$ or $\delta(q) = q' \wedge q''$, then the round continues with the still free pebble on q' or q'', depending on the choice of the player who has to move the pebble.

   The following rules determine who of the players has to move which pebble.

   - If $\langle\psi\rangle$ is existential and $\langle\rho\rangle$ is universal (that is, ρ is of the form $\rho' \wedge \rho''$) and both pebbles are free, then Spoiler moves one of the pebbles (he can choose which one).
   - If $\langle\psi\rangle$ is existential and the red pebble is free, but $\langle\rho\rangle$ is existential or the green pebble is locked, then Spoiler has to move the red pebble.
   - Conversely, if $\langle\psi\rangle$ is universal or the red pebble is locked, but $\langle\rho\rangle$ is universal and the green pebble is free, then Spoiler has to move the green pebble.

   If these cases do not apply, Duplicator has to move a free pebble in a symmetric fashion, as follows.

   - If $\langle\psi\rangle$ is universal and $\langle\rho\rangle$ is existential and both pebbles are free, then Duplicator chooses one of the pebbles and moves it.
   - If $\langle\psi\rangle$ is universal and the red pebble is free while the green pebble is locked, then Duplicator moves the red pebble.
   - And if $\langle\rho\rangle$ is existential, the green pebble is free, and the red pebble is locked, then Duplicator moves the green pebble.

3. At the end of a round, both pebbles are locked. Since the automata in question are constructed from guarded formulas, it is ensured that every round is finite, i. e., the resulting automata are legal in the sense of Section 2.2.

The play of the simulation game ends early at the end of a round if one of the following holds.

(a) At least one of the pebbles has taken a propositional or final transition in step 2a.

(b) One of the pebbles has ended its round by taking a transition $\Diamond q$ while the other pebble has taken a transition $\Box q$ in step 2c.

If the play does not end early, then the play continues with the next round in step 1.

If the play ends early, the winner is determined as follows.

- As stated above, Duplicator wins if Spoiler cannot choose a term different from ff in step 1.
- If the green pebble has taken a propositional or final transition in step 2a, then Duplicator wins if
  - the green pebble has taken a transition tt or a propositional transition $t' \in \{a, \neg a \mid a \in \Sigma\}$ such that $t \to t'$, or
  - if both the red and the green pebble have taken a transition ff or a transition $t' \in \{a, \neg a \mid a \in \Sigma\}$ such that $t \not\to t'$.
- If the red pebble has taken a propositional or final transition in step 2a but the green pebble has not, then Duplicator wins if the red pebble has taken a transition ff or a transition $t' \in \{a, \neg a \mid a \in \Sigma\}$ such that $t \not\to t'$.
- In all other cases, Spoiler wins; especially, Spoiler wins if the play ends early in the above case 3b.

In order to get a delayed simulation game, we add a priority memory, and we update this priority memory according to the priorities of the pebbles using the function pm of Subsection 3.2.1, or the functions pm^r and pm^l of Subsection 3.4.1 for right-hand and left-hand simulation, respectively. We have to consider that, basically, all visited priorities are significant for acceptance of an APTA. However, our APTA from μ-calculus formulas are constructed in such a way that only the priority of states $\langle Z \rangle$ with $\delta(\langle Z \rangle) = \langle \varphi_Z \rangle$ are significant. Since we only consider strictly guarded formulas in normal form, such a state can be visited at most once during a round of the simulation game by each of the pebbles, and if it is visited by a pebble, then the round ends with this pebble on the visited state $\langle Z \rangle$. It is therefore sufficient to only consider the priorities of the states reached by the pebbles at the end of each round for the priority memory.

That is, at the beginning of the first round with the red pebble on $\langle\psi\rangle$ and the green pebble on state $\langle\rho\rangle$, the initial value of the priority memory is $\mathrm{pm}(\langle\psi\rangle, \langle\rho\rangle, \surd)$. If a round ends with the red pebble on $\langle\psi'\rangle$, the green pebble on $\langle\rho'\rangle$ and the priority memory value $k \in \omega \cup \{\surd\}$, then the priority memory is set to $\mathrm{pm}(\langle\psi'\rangle, \langle\rho'\rangle, k)$. For right-hand and left-hand delayed simulation, we use pm^r or pm^l, respectively, instead of pm. As usual, Duplicator wins an infinite play of the delayed simulation game if the priority memory value is $\surd$ infinitely often. We write $\langle\psi\rangle \leq_{de} \langle\rho\rangle$ (or $\langle\psi\rangle \leq^l_{de} \langle\rho\rangle$ or $\langle\psi\rangle \leq^r_{de} \langle\rho\rangle$) if Duplicator has a winning strategy for the simulation game starting with the red pebble on $\langle\psi\rangle$ and the green pebble on $\langle\rho\rangle$.

In a formal definition of the game graph, a position needs the components known from the definition of the simulation game for ε-ABA in Section 2.3: Aside from encoding the current states of the two pebbles and the term chosen by Spoiler in the current round, we need two bits to store whether each pebble is free or locked. Here, we also need a priority memory, of course, and we need an additional bit to store whether the pebble which was locked first has taken a next transition $\Box q$ or a next transition $\Diamond q$.

3.6.3 Application and example

As an example, we reconsider the formula $\varphi = \mu Z(\varphi_1 \vee \Diamond Z)$ of equation (3.63). Most interesting, in a first approach, is the comparison between the two alternatives of existential and universal transitions. Computing $\leq_{de}$ as well as $\leq^r_{de}$ and $\leq^l_{de}$ for $\mathbf{Q}(\varphi)$, we see that

- both $\langle\Diamond Z\rangle \leq^r_{de} \langle\varphi_1\rangle$ and $\langle\Diamond Z\rangle \leq^l_{de} \langle\varphi_1\rangle$ hold, but $\langle\varphi_1\rangle \not\leq_{de} \langle\Diamond Z\rangle$,
- $\langle\Diamond Y\rangle \leq^l_{de} \langle\varphi_2\rangle$, but $\langle\Diamond Y\rangle \not\leq^r_{de} \langle\varphi_2\rangle$ and also $\langle\varphi_2\rangle \not\leq_{de} \langle\Diamond Y\rangle$,
- both $\langle\Diamond X\rangle \leq^r_{de} \langle\varphi_3\rangle$ and $\langle\Diamond X\rangle \leq^l_{de} \langle\varphi_3\rangle$ hold, but $\langle\varphi_3\rangle \not\leq_{de} \langle\Diamond X\rangle$,
- and $\langle\Diamond W\rangle$ is also strictly larger than $\langle\Diamond Z\rangle$ w. r. t. all our three modes of delayed simulation.

To see that, e. g., $\langle\Diamond Z\rangle \leq_{de} \langle\Diamond W\rangle$, consider the simulation game starting with the red pebble on $\langle\Diamond Z\rangle$ and the green pebble on $\langle\Diamond W\rangle$. Both states have priority 1, so the initial value of the priority memory is $\surd$. Both pebbles must take the only possible next transition in the first round (and both these transitions are $\Diamond$-transitions), so the next round starts with the red pebble on $\langle Z\rangle$ and the green pebble on $\langle W\rangle$, and both these states have priority 1. The possible terms for Spoiler to choose are $\mathtt{tt}$, a, and b, but if he chooses a or b, then Duplicator will move the green pebble to $\langle a\rangle$ or $\langle b\rangle$, respectively, take a propositional transition

in accordance with the chosen term and win early. Hence we assume that Spoiler chooses the term tt. Spoiler may now move the red pebble via $\langle \varphi_1 \vee \Diamond Z \rangle$ back to $\langle Z \rangle$, but then Duplicator will move the green pebble to the same state via the states $\langle (a \vee b) \vee (\Diamond W \vee \Diamond Z) \rangle$ and $\langle \Diamond W \vee \Diamond Z \rangle$, and Spoiler loses. But if Spoiler moves the red pebble to $\langle \varphi_2 \wedge \Diamond Y \rangle$, then Duplicator will move it to $\langle \Diamond Y \rangle$ and then, in order not to lose early by moving the red pebble to $\langle a \rangle$ via $\langle a \vee (\varphi_3 \wedge \Diamond X) \rangle$, the red pebble will reach the state $\langle \varphi_3 \wedge \Diamond X \rangle$, and Duplicator can move it to $\langle \varphi_3 \rangle$. Duplicator has not moved the green pebble in this round so far, it is still on $\langle W \rangle$, and it is now obvious that Duplicator will win regardless of what Spoiler does: Duplicator can now play in such a way that the two pebbles end the round on the same state, and since the priority memory is still $\surd$, this is sufficient for Duplicator to win.

The first rule that we may use now to simplify $\mathbf{Q}(\varphi)$ is similar to the *rl-edge-reduction* of Lemma 3.7: We may change an existential transition $\delta(q) = q' \vee q''$ into a simple transition $\delta(q) = q''$ if $q' <^{l}_{de} q''$, and, symmetrically, we may change a universal transition $\delta(q) = q' \wedge q''$ into a simple transition $\delta(q) = q''$ if $q'' <^{r}_{de} q'$. That is, we may change the transition condition $\delta(\langle \varphi_1 \vee \Diamond Z \rangle) = \langle \varphi_1 \rangle \vee \langle \Diamond Z \rangle$ to $\delta(\langle \varphi_1 \vee \Diamond Z \rangle) = \langle \varphi_1 \rangle$, and this is similar to replacing the subformula $\varphi_1 \vee \Diamond Z$ in φ by φ_1. Similarly, we can replace the subformula $\Diamond W \vee \Diamond Z$ in φ_3 by the formula $\Diamond W$, and the formula $\varphi_3 \wedge \Diamond X$ by $\Diamond X$. Note that we must not apply a similar replacement for the subformula $\varphi_2 \wedge \Diamond Y$, because we only have left-hand simulation for the successor states there.

The simplified formula now is

$$\varphi' = \mu Z \varphi'_1 \tag{3.73}$$

with

$$\varphi'_1 = \nu Y (\varphi'_2 \wedge \Diamond Y) \ , \text{ and} \tag{3.74}$$
$$\varphi'_2 = \mu X (a \vee \Diamond X) \ . \tag{3.75}$$

Obviously, the fixed point operator μZ can be removed because there is no proposition bound to it, so we have

$$\varphi \equiv \nu Y (\mu X (a \vee \Diamond X) \wedge \Diamond Y) \ , \tag{3.76}$$

that is, φ is equivalent to the CTL formula $\mathsf{EGEF}a$.

Aside from deleting states based on $\leq^{l}_{de}$ and $\leq^{r}_{de}$ as described above, APTA from strictly guarded μ-calculus formulas in normal form can be *homogenized* based on delayed simulation equivalence (similar to Lemma 3.3), we can introduce *shortcuts* (similar to Lemma 3.4) and we can do a *reachability minimaxing* of the APTA (similar to Lemma 3.6).

3.7 Conclusion of Chapter 3

We have adapted delayed simulation to alternating automata with a parity acceptance condition, in the form of a general delayed simulation relation and of two restricted dual versions of this relation. We have introduced quotient automata for these restricted relations, and we have integrated the three notions of delayed simulation for parity acceptance into an incremental simplification algorithm for alternating parity automata and parity games. We can state that delayed simulation is an efficient tool for the comparison and simplification of alternating automata with parity acceptance, but our experiments indicate that using it for simplifying parity games is of limited practical use only. We have outlined an application to formulas of the modal μ-calculus and to alternating parity tree automata.

Conclusion and Directions of Future Research

Simulation Relations for Alternating Büchi Automata

Alternating automata as well as simulation relations have a natural and very intuitive connection to infinite two-player games. The run of an alternating automaton on an ω-word can be described, both on an intuitive and on a formal level, as an infinite game between two players Automaton and Pathfinder: Automaton wants the word to be accepted, and Pathfinder wants the word to be rejected. Similarly, the notion of simulation of one automaton by another automaton can be defined via a game of a Duplicator player who wants to demonstrate this relationship by showing that one automaton can duplicate the runs of the other automaton, and a Spoiler player who wants to spoil this demonstration.

In Chapter 1, we use this game-based approach to extend the notions of direct, delayed, and fair simulation to alternating Büchi automata. In the case of nondeterministic Büchi automata, these relations are preorders and imply language containment. We show that this is also the case for alternating Büchi automata. Especially, to show transitivity is nontrivial; we therefore introduce the notion of a join of two Duplicator strategies which formally describes the idea that two strategies can be combined, with a synchronizing man-in-the-middle, to a single strategy. In many proofs, we make use of the fact that such a joint strategy inherits properties of the two strategies; especially, the join of two winning strategies is again a winning strategy.

For nondeterministic Büchi automata, it is known that direct and delayed simulation allow to merge simulation-equivalent states into a single state—the quotient automaton w. r. t. these simulation equivalences accepts the same language as the original automaton. We show that similar quotient automata (the minimax quotient and the semi-elective quotient) can be defined for alternating Büchi automata such that the quotient automaton is simulation-equivalent to the original automaton (and thus accepts the same language). The main problem in defining these quotient automata is the treatment of universal states and the case where

a universal state is simulation-equivalent to an existential state. We also discuss the extent to which delayed simulation allows to identify transitions that can be removed from the quotient automaton.

Alternating Büchi automata can be translated to equivalent nondeterministic Büchi automata of exponential size, using the method of Miyano and Hayashi. We show that our notions of simulation are compatible with this translation in the sense that if an alternating automaton simulates another automaton, then so do their nondeterministic versions. This opens up the possibility of simulation-based on-the-fly simplifications during the construction of a nondeterministic automaton; we analyze this in detail for the case of alternating automata from LTL in Chapter 2.

We show that computing the simulation relations is not more difficult for alternating automata than for nondeterministic automata—we can use the known algorithms with the same asymptotic space and time consumption. We show that for the case of weak alternating automata, which are of special interest for the application to LTL, delayed and fair simulation can be computed as fast as direct simulation.

Simulation Relations and Büchi Automata from LTL

New considerations and insights are necessary for the application of the results of Chapter 1 to the automata construction from LTL. In Chapter 2, we develop a framework for simulation relations as relations on the set of LTL formulas over a set of propositions. We use the well-known fact that an LTL formula can be viewed as an alternating automaton, but we have to adjust both these automata and our notion of simulation games in order to get a useful notion of simulation relations for LTL.

Alternating automata from LTL are very weak. We show that this property allows us to optimize the de-universalization construction for alternating automata such that the resulting nondeterministic automata have at most $(n+1)2^n$ states instead of the worst-case $\Omega(4^n)$ of nondeterministic automata resulting from the Miyano–Hayashi construction. This optimized de-universalization can be applied to LTL formulas directly, without using alternating automata explicitly, and allows local optimization as a simple criterion for identifying and deleting superfluous transitions.

In order to use simulation relations in an on-the-fly fashion during the de-universalization, we further analyze these relations in the special setting of a translation from LTL to nondeterministic automata. We show that both the fair and the delayed simulation relation introduced for alternating automata from LTL can be used for simplifications in the nondeterministic automaton. The main results here

are a set of rules for LTL formula rewriting based on fair simulation (in lieu of simulation-based quotienting of the alternating automaton), criteria for deleting transitions based on fair simulation, and rules for NBA state pruning based on both fair and delayed simulation.

We combine these rules and the de-universalization construction into a simple algorithm for the construction of a nondeterministic automaton from an LTL formula with on-the-fly simplifications. We discuss appropriate post-processing steps and the interaction of our algorithm with other approaches, give two detailed examples and report experimental data for an earlier version of the algorithm.

We compare our optimized de-universalization to tableau-based approaches. This comparison is complicated by the fact that the usual tableau-based algorithms result in state-labeled generalized Büchi automata. Using a straightforward translation of these automata into our format, it turns out that the results of the two approaches are the same for input formulas in next normal form, if the local optimization is applied to the automata resulting from both approaches. In order to get this result, we also show that local optimization covers the syntactical implication of Daniele et al.

We introduce a bottom-up variant of our automata construction from LTL. While the resulting automata are not different from the top-down de-universalization, we show that this approach offers the possibility to copy simulation-based simplifications (especially quotienting w. r. t. a simulation relation) of automata for subformulas into simplifications of the automaton for the overall formula. It is thus possible to decompose a (large) formula into smaller subformulas, compute quotient automata for these subformulas and then assemble the quotient automata into an automaton for the overall formula. This can be seen as a sort of partial quotienting: The simplifications of the subautomata become simplifications of the overall automaton, without the need of a costly computation of a quotient automaton for the whole input formula.

Simulation Relations and the Parity Acceptance Condition

The notion of delayed simulation relations can be extended to alternating automata with a parity acceptance condition. We first define, via infinite games and using the new notion of a priority memory, a relation on the state space of alternating parity automata. We show that this relation has the basic properties of other simulation relations in this work: It is a preorder, it implies language containment, and it can be computed in polynomial time. This relation models delayed simulation for parity acceptance in the following sense: Whenever in a play of the simu-

lation game the priority of the current state of the automaton to be simulated is better than the current priority of the simulating automaton, we store the smaller of the two priorities as an obligation for the Duplicator player. This obligation has to be fulfilled at some point in the future, by the simulating automaton or by the simulated automaton. Thus, the priority memory is similar to the winning bit introduced for delayed simulation of Büchi automata in Chapter 1.

This general delayed simulation relation, which allows to erase the priority memory based on the moves in both automata, poses problems for merging equivalent states and for deleting transitions. We therefore introduce two restricted versions of the general relation, called left-hand and right-hand delayed simulation, following the idea that the states encountered either in the simulating or in the simulated automaton only must fulfill the obligation of the parity memory. This allows us to define two dual notions of quotients for alternating parity automata, and this also rounds out the picture of Chapter 1: The semi-elective quotient for alternating Büchi automata, where existential and universal states must be handled in an asymmetric fashion, now is generalized as the quotient w. r. t. right-hand delayed simulation, and its dual counterpart is the quotient w. r. t. left-hand simulation.

By extending ideas of the previous chapters to the case of parity acceptance, we develop a simulation-based simplification algorithm for alternating parity automata. This algorithm makes use of the two restricted left-hand and right-hand simulations as well as the general delayed simulation relation. We show that the latter can be used for, e. g., homogenizing the priorities of a parity automaton, which is an adaptation of the concept of pseudo-accepting states of Chapter 1. We combine the use of simulation relations with normalizing the parity automaton to an incremental simplification algorithm.

Solving parity games (that is, deciding the winner) is important for model checking of fixed point logics. Experiments indicate that simplifying parity games using our approach before solving them is not faster than solving them outright in practice. But alternating parity automata on tree-like structures are a natural representation of μ-calculus formulas, and branching-time logics like CTL can be seen as fragments of the modal μ-calculus. We give a sketch of how to apply our concepts of simulation-based simplifications to alternating parity tree automata constructed from a fragment of the modal μ-calculus; this fragment contains CTL.

Directions of Ongoing and Further Research

An active direction of research is the adaptation of useful notions of simulation to different acceptance modes of ω-automata. Juvekar and Piterman [JP06] extend fair and delayed simulation to generalized (nondeterministic) Büchi automata.

Since these generalized Büchi automata occur as the result of a tableau-based translation of LTL to automata [GPVW95, DGV99], it may be interesting to combine our on-the-fly simplifications and analysis of LTL-to-automata translations in Chapter 2 with the approach of [JP06]. Active research also explores the combination of delayed simulation with other notions of simulation and simplification techniques, see, e. g., [CC04].

Kesten et al. [KPP05] use fair simulation to check trace containment between reactive systems modeled as Streett automata. Klein [Kle05] analyzes the simplification of deterministic ω-automat with Rabin and Streett acceptance [Rab72, Str82] using bisimulation among other techniques (for deterministic automata, bisimulation is the same as direct simulation equivalence). Deterministic Rabin automata result from determinizing nondeterministic Büchi automata using the method of Safra [Saf88] while Streett acceptance is dual to Rabin acceptance. Since Safra's construction is essentially a sophisticated version of the subset construction, Etessami's concept of k-pebble simulation [Ete02] may be useful for further research in this field. Future research may also analyze whether it is useful and practicable to extend the concepts of [Ete02, KPP05, JP06] to alternation.

For ω-automata with empty-word transitions, we have defined simulation relations via games such that the two players synchronize their moves at the end of each round. While this is appropriate for automata constructed from LTL, it poses problems for automata from μ-calculus formulas, such that we resorted to a fragment of the modal μ-calculus. Future research seems necessary to better handle empty transitions. In this context, an analysis of asynchronous simulation as in [Jan01] seems necessary.

Bibliography

[AHKV98] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR '98: Concurrency Theory, 9th Internat. Conf., Nice, France*, volume 1466 of *LNCS*, pages 163–178. Springer, Berlin, 1998.

[And94] Henrik R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, 1994.

[AS03] André Arnold and Luigi Santocanale. Ambigious classes in the games μ-calculus hierarchy. In A. D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th Int. Conf. (FoSSaCS 2003), Warsaw, Poland*, volume 2620 of *Lecture Notes in Computer Science*, pages 70–86. Springer-Verlag, 2003.

[BG02] Doron Bustan and Orna Grumberg. Applicability of fair simulation. In J.-P. Katoen and P. Stevens, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 8th Int. Conf. (TACAS 2002), Grenoble, France*, volume 2280 of *Lecture Notes in Computer Science*, pages 401–414. Springer, Berlin, 2002.

[Blo] Roderick Bloem. Wring: an LTL to Buechi translator. URL: `http://vlsi.colorado.edu/~rbloem/wring.html`.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[Büc62] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Logic, Methodology, and Philosophy of Science: Proc. of the 1960 International Congress*, pages 1–11, Stanford, Calif., 1962. Stanford University Press.

[CC04] Jean-Marc Champarnaud and Fabien Coulon. Büchi automata reduction by means of left and right trace inclusion preorders. In *Proc. Journées Montoise d'Informatique Théorique (JM 2004)*, 2004.

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[DGV99] Marco Daniele, Fausto Giunchiglia, and Moshe Y. Vardi. Improved automata generation for linear time temporal logic. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification, 11th Internat. Conf., CAV '99, Trento, Italy*, volume 1633 of *Lecture Notes in Computer Science*, pages 249–260. Springer, Berlin, 1999.

[DHWT91] David L. Dill, Alan J. Hu, and Howard Wong-Toi. Checking for language inclusion using simulation preorders. In K. Guldstrand Larsen and A. Skou, editors, *Computer Aided Verification, 3rd Internat. Workshop, CAV '91, Aalborg, Denmark*, volume 575 of *Lecture Notes in Computer Science*, pages 255–265. Springer, Berlin, 1991.

[EH85] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.

[EH00] Kousha Etessami and Gerard Holzmann. Optimizing Büchi automata. In C. Palamidessi, editor, *11th Int. Conf. on Concurrency Theory (CONCUR 2000), University Park, PA, USA*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167. Springer, Berlin, 2000.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. 32nd Ann. Symp. on Foundations of Computer Science (FoCS '91), San Juan, Puerto Rico*, pages 368–377. IEEE Computer Society Press, 1991.

[Eme90] E. Allen Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, pages 995–1072. Elsevier Publishing, Amsterdam, 1990.

[ESW01] Kousha Etessami, Rebecca Schuller, and Thomas Wilke. Fair simulation relations, parity games, and state space reduction for Büchi automata. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th Internat. Coll.,*

ICALP 2001, Crete, Greece, volume 2076 of *Lecture Notes in Computer Science*, pages 694–707. Springer, Berlin, 2001.

[Ete] Kousha Etessami. Temporal massage parlor. URL: `http://www1.bell-labs.com/project/TMP/`.

[Ete02] Kousha Etessami. A hierarchy of polynomial-time computable simulations for automata. In L. Brim, P. Janar, M. Ketínský, and A. Kuera, editors, *Concurrency Theory: 13th Internat. Conf., Brno, Czech Republic (CONCUR 2002)*, volume 2421 of *LNCS*, pages 131–144. Springer, Berlin, 2002.

[Fri03] Carsten Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In O. H. Ibarra and Z. Dang, editors, *Implementation and Application of Automata, 8th Internat. Conf., CIAA 2003, Santa Barbara, CA, USA*, volume 2759 of *Lecture Notes in Computer Science*, pages 35–48. Springer, Berlin, 2003.

[Fri05a] Carsten Fritz. Concepts of automata construction from LTL. In G. Sutcliffe and A. Voronkov, editors, *12th Int. Conf. on Logic for Programming, Artifical Intelligence, and Reasoning (LPAR 2005), Montego Bay, Jamaica*, volume 3835 of *Lecture Notes in Artificial Intelligence*, pages 728–742. Springer, Berlin, 2005.

[Fri05b] Carsten Fritz. Concepts of automata construction from LTL. Technical report, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 2005. URL: `http://www.ti.informatik.uni-kiel.de/~fritz/AutFromLTL-TR.pdf`.

[FTa] Carsten Fritz and Björn Teegen. LTL → NBA (improved version). URL: `http://www.ti.informatik.uni-kiel.de/~fritz/ABA-Simulation/ltl.cgi`.

[FTb] Carsten Fritz and Björn Teegen. LTL→NBA. URL: `http://www.ti.informatik.uni-kiel.de/~teegen/ABA-Simulation/ltl.cgi`.

[FW01] Carsten Fritz and Thomas Wilke. Simulation relations for alternating Büchi automata. Technical Report 2019, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, Nov. 2001. Available at `http://www.informatik.uni-kiel.de/reports/2001/2019.html`.

[FW02] Carsten Fritz and Thomas Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In M. Agrawal and A. Seth, editors, *22nd Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2002), Kanpur, India*, volume 2556 of *Lecture Notes in Computer Science*, pages 157–168. Springer, Berlin, 2002.

[FW05] Carsten Fritz and Thomas Wilke. Simulation relations for alternating Büchi automata. *Theoretical Computer Science*, 338(1–3):275–314, 2005.

[GBS02] Sankar Gurumurthy, Roderick Bloem, and Fabio Somenzi. Fair simulation minimization. In E. Brinksma and K. Guldstrand Larsen, editors, *Computer Aided Verification, 14th Internat. Conf., CAV 2002, Copenhagen, Denmark*, volume 2404 of *LNCS*, pages 610–623. Springer, Berlin, 2002.

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *14th ACM Symp. on the Theory of Computing, San Francisco, CA, USA*, pages 60–65. ACM Press, 1982.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1979.

[GL02] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In D. Peled and M. Y. Vardi, editors, *Proc. of 22nd IFIP Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE 2002), Houston, TX, USA*, volume 2529 of *LNCS*, pages 308–326. Springer, 2002.

[GO01] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In G. Berry, H. Comon, and A. Finkel, editors, *Computer Aided Verification, 13th Internat. Conf., CAV 2001, Paris, France*, volume 2102 of *LNCS*, pages 53–65. Springer, Berlin, 2001.

[GPVW95] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. 15th Workshop on Protocol Specification, Testing, and Verification, Warsaw, Poland*, pages 3–18. Chapman & Hall, London, 1995.

[GS05] Gregor Gramlich and Georg Schnitger. Minimizing NFA's and regular expressions. In V. Diekert and B. Durand, editors, *22nd Symp.*

on Theoretical Aspects of Computer Science, STACS 2005, Stuttgart, Germany, volume 3404 of *Lecture Notes in Computer Science*, pages 399–411. Springer, Berlin, 2005.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[HKR97] Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. In *CONCUR '97: Concurrency Theory, 8th Internat. Conf., Warsaw, Poland*, volume 1243 of *LNCS*, pages 273–287. Springer, Berlin, 1997.

[Hol] Gerard J. Holzmann. The SPIN homepage. URL: `http://netlib.bell-labs.com/netlib/spin/whatispin.html`.

[HR00] Thomas A. Henzinger and Sriram K. Rajamani. Fair bisimulation. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for Construction and Analysis of Systems, 6th Internat. Conf., TACAS 2000, Berlin, Germany*, volume 1785 of *Lecture Notes in Computer Science*, pages 299–314. Springer, Berlin, 2000.

[HRHK95] Monika Henzinger Rauch, Thomas A. Henzinger, and Peter W. Kopke. Computing simulations on finite and infinite graphs. In *36th Ann. Symp. on Foundations of Computer Science (FOCS '95), Milwaukee, WI, USA*, pages 453–462. IEEE Computer Society Press, 1995.

[Jan01] David Janin. Simplifying parity games with synchronous and asynchronous simulation relations. Unpublished note, 2001.

[JP06] Sudeep Juvekar and Nir Piterman. Minimizing generalized Büchi automata. Submitted for publication, 2006.

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998.

[Jur00] Marcin Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *STACS 2000, 17th Ann. Symp. on Theoretical Aspects of Computer Science, Lille, France*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, Berlin, 2000.

[JV00] Marcin Jurdziński and Jens Vöge. A discrete improvement algorithm for solving parity games. Technical Report 2, Aachener Informatik-Berichte, RWTH Aachen, 2000.

[Kam68] Johan A. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, Calif., 1968.

[Kle05] Joachim Klein. Linear time logic and deterministic omega-automata. Diploma thesis, Institut für Informatik, Rheinische Friedrich-Wilhelms-Universität Bonn, 2005.

[Knu68] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, 1968. Second edition 1973.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KPP05] Yonit Kesten, Nir Piterman, and Amir Pnueli. Bridging the gap between fair simulation and trace inclusion. *Information and Computation*, 200(1):35–61, 2005.

[KR88] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition edition, 1988.

[KV97] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. In *5th Israeli Symposium on the Theory of Computing Systems (ISTCS '97)*, pages 147–158, Ramat-Gan, Israel, 1997. IEEE.

[KVW00] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[LT00] Christof Löding and Wolfgang Thomas. Alternating automata and logics over infinite words. In *Proc. of the IFIP Int. Conf. on Theoretical Computer Science (IFIP TCS2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer, 2000.

[Löd98] Christof Löding. Methods for the transformation of omega-automata: Complexity and connection to second order logic. Diploma thesis, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 1998.

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[Mil71] Robin Milner. An algebraic definition of simulation between programs. In D. C. Cooper, editor, *Proc. 2nd Internat. Joint Conf. on Artificial Intelligence, London, UK*, pages 481–489. William Kaufmann, 1971.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, Englewood Cliffs, NJ, USA, 1989.

[Mos84] Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984.

[Mos91] Andrzej W. Mostowski. Hierarchies of weak automata and weak monadic formulas. *Theoretical Computer Science*, 83(2):323–335, June 1991.

[MP71] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, Cambridge, Mass., 1971.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, Berlin, New York, 1992.

[MS87] David E. Muller and Paul E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, October 1987.

[MSS86] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Coll. on Automata, Languages, and Programming (ICALP '86)*, volume 226 of *Lecture Notes in Computer Science*, 1986.

[MSS88] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *3rd IEEE Ann. Symp. on Logic in Computer Science, Edinburgh, Scotland, UK*, pages 422–427, 1988.

[MTHM97] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. The MIT Press, May 1997.

[Mul63] David E. Muller. Infinite sequences and finite machines. In *Proc. 4th Ann. IEEE Symp. on Switching Circuit Theory and Logical Design*, pages 3–16, 1963.

[Niw97] Damian Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189:1–69, 1997.

[Odd] Denis Oddoux. LTL2BA. URL: `http://www.liafa.jussieu.fr/~oddoux/ltl2ba/`.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, Providence, RI, 1977. IEEE Computer Society.

[Pyt] Python Software Foundation. Python programming language. `http://www.python.org`.

[Rab72] Michael O. Rabin. *Automata on infinite objects and Church's problem*, volume 13 of *Regional Conf. Series in Mathematics*. American Mathematical Society, 1972.

[Roh97] Scott Rohde. *Alternating Automata and the Temporal Logic of Ordinals*. PhD thesis, Department of Mathematics, University of Illinois at Urbana-Champaign, 1997.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *29th Annual Symposium on Foundations of Computer Science*, pages 319–327, White Plains, New York, 1988. IEEE Computer Society.

[SB00] Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In E. A. Emerson and A. P. Sistla, editors, *Computer Aided Verification, 12th Internat. Conf., CAV 2000, Chicago, IL, USA*, volume 1855 of *LNCS*, pages 248–263. Springer, Berlin, 2000.

[Str82] Robert S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54(1–2):121–141, 1982.

[Tar72] Robert E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[TH02] Heikki Tauriainen and Keijo Heljanko. Testing LTL formula translation into Büchi automata. *International Journal on Software Tools for Technology Transfer*, 4(1):57–70, 2002.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, pages 134–191. Elsevier, Amsterdam, 1990.

[Tho97] Wolfgang Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3: Beyond Words, pages 389–455. Springer-Verlag, Berlin, 1997.

[Var94] Moshe Y. Vardi. Nontraditional applications of automata theory. In M. Hagiya and J. C. Mitchell, editors, *Theoretical Aspects of Computer Software, Internat. Conf. TACS '94, Sendai, Japan*, volume 789 of *Lecture Notes in Computer Science*, pages 575–597. Springer, Berlin, 1994.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In D. Kozen, editor, *1st Ann. IEEE Symp. on Logic in Computer Science (LiCS'86)*, pages 332–344, Cambridge, Mass., USA, 16–18 June 1986. IEEE Computer Society.

[Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8:359–391, 2001.

List of Figures

Index