Methods for the Transformation of ω-Automata: Complexity and Connection to Second Order Logic

Diploma Thesis (Revised Version)

Christof Löding

Supervisor: Prof. Dr. Wolfgang Thomas

Institute of Computer Science and Applied Mathematics
Christian-Albrechts-University of Kiel

Contents

Introduction

Finite automata on infinite objects were first introduced in the 60's. Büchi [Büc62], Trakhtenbrot [Tra62], and Rabin [Rab69] used them to solve decision problems in mathematics and logic by showing that every formula of the considered system is in some sense equivalent to a finite automaton. Following these ideas several other decision problems were solved (see the survey [Tho90]). Muller [Mul63] used automata on infinite words, called ω-automata, to describe problems in asynchronous switching theory. McNaughton [McN66] related the different models of automata considered by Büchi and Muller. Today automata on infinite objects are used for verification of nonterminating programs. A computation of a nonterminating program is an infinite sequence of program states. Such a state can be described by a set of propositions that hold in the state. Thus, a program defines a language of infinite words over sets of propositions. In many situations this language can also be defined by an automaton on infinite words. Therefore it is possible to reduce questions about programs to automata theoretic questions.

There has been a lot of research in the theory of ω-automata. Many different models of automata were introduced by several authors. An ω-automaton consists of a state set Q, an input alphabet Σ, an initial state from Q, a transition function, and an acceptance condition. The models differ in the way acceptance is defined and in the definitions of the transition functions. A main objective of this thesis is to give an integrated presentation of the different models. In this introduction we discuss some key ideas on the intuitive level, first regarding the concepts for transition functions, and secondly the acceptance conditions.

Transition Functions

The transition function of an automaton describes for every pair (q, a) of state and input letter the behavior of the automaton if it is in state q and reads the input letter a.

In deterministic automata there is exactly one successor state for every pair of state and input letter. Therefore an input word induces exactly one state sequence, called the run of the automaton on the input word. The input is accepted iff this run satisfies the acceptance condition.

In nondeterministic automata there is a set of successors for every pair of state and input letter. The automaton nondeterministically chooses one of these successors and then proceeds. Thus, for an infinite input word, there are many different possible runs. A word is accepted iff one of these state sequences satisfies the acceptance condition.

Dual to the concept of nondeterminism is the concept of universality. As in nondeterministic automata there is a set of successors for every pair of state and input letter.

A universal automaton does not choose one of these successors, but moves to each of these successors and then proceeds. Hence a run of a universal automaton is no longer a state sequence, but an infinite tree with the vertices labeled with states. The input is accepted iff every path through the tree satisfies the acceptance condition.

Instead of run trees we use a different notion, which can be obtained from the notion of a run tree for certain types of acceptance conditions as follows. On each level of a run tree the vertices may have only $|Q|$ different labels. If we merge vertices on the same level with the same label, then the tree transforms into an infinite directed acyclic graph. Every level of this graph has at most $|Q|$ vertices and therefore the graph is, in some sense, of bounded width. This view of runs is of interest in the investigation of another transition mode, namely alternation.

Alternation is a combination of nondeterminism and universality. The transition function gives a system of subsets of Q for each pair of state and input letter. The automaton nondeterministically chooses one of these sets and then moves to each state from this set, just as a universal automaton. So a run of an alternating automaton is built up from nondeterministic choices and has the same structure as a run of a universal automaton. In a universal automaton for every input there is exactly one run, while in an alternating automaton there may be more than one run for each input. An input is accepted by an alternating automaton iff there is a run such that every path through this run satisfies the acceptance condition.

For finite automata on finite words ($*$-automata) all these different modes of the transition functions have the same expressive power. As will be shown in the subsequent chapters for ω-automata this depends on the type of acceptance condition.

Acceptance Conditions

Finite automata on finite words accept an input iff the automaton reaches a final state after having read the whole input. For ω-automata acceptance has to be defined in a different way since the automaton never reaches the end of the (infinite) input. Büchi extended the idea of the final states for ω-automata. He defined an input to be accepted by an automaton iff the run of the automaton infinitely often visits a "final state". The languages that can be defined by these nondeterministic Büchi automata are called ω-regular, by an equivalence to ω-regular expressions. But, in contrast to $*$-automata, deterministic Büchi automata are strictly weaker than nondeterministic ones.

Dual to the definition of Büchi automata one can define an input to be accepted iff the run on this input only finitely often visits the set of final states. Such an acceptance condition we call a co-Büchi condition.

Muller [Mul63] worked with deterministic automata. The acceptance condition specifies a certain system $\mathcal{F}$ of subsets of states ($\mathcal{F} \subseteq 2^Q$). An input is accepted iff the infinity set of the run, i.e., the states occurring infinitely often in the run, belongs to $\mathcal{F}$. The fundamental theorem of McNaughton states that deterministic Muller automata and nondeterministic Büchi automata have the same expressive power [McN66]. In his translation from nondeterministic Büchi automata into deterministic Muller automata McNaughton used a special form of the Muller condition that was formalized by Rabin [Rab69]. The Rabin condition consists of pairs (E_i, F_i) of subsets of Q. An input is accepted iff there exists an i such that the run on this input infinitely often visits F_i and only finitely often visits E_i.

Other conditions for deterministic automata that have the same expressive power as the Muller condition are the Streett condition [Str82] and the parity condition [Mos84]. A Streett condition, as a Rabin condition, consists of pairs (E_i, F_i) of subsets of Q. But dual to the Rabin condition an input is accepted iff its run infinitely often visits E_i or only finitely often visits F_i for each i. Parity conditions are also called Rabin chain conditions, because they can be represented like Rabin conditions with the restriction that the E_i's and F_i's form a chain, i.e., $E_1 \subseteq F_1 \subseteq E_2 \subseteq F_2 \cdots \subseteq E_r \subseteq F_r$. Another representation of this condition is via a mapping c ("coloring") that assigns to every state a number from a set $\{1, \ldots, k\}$. The run is accepted iff the smallest number that is visited infinitely often is even.

In [MSS86] a different ("weak") form of acceptance was introduced. In this form of acceptance not infinitely many visits of states, but mere visits are considered. More precisely, in weak automata the state set is partitioned into several subsets. There is an order on these sets of the partition, and transitions may either stay in the same set of the partition or move to a lower one. Some of these sets in the partition are positive and some are negative. A run is accepting iff the lowest set in the partition that is visited by the run is positive.

In the deterministic case weak automata have the same expressive power as the automata considered by Staiger and Wagner [SW74]. Allowing alternation in the transition function of weak automata, one can obtain the same expressive power as for nondeterministic Büchi automata.

Aim and Structure of the Thesis

This thesis gives an overview over the basic ideas for the transformations of ω-automata. We present the main constructions, analyze their complexity, and give upper and lower bounds for the transformations.

The first chapter is an introduction to ω-automata, explaining the different modes of transition functions, the different acceptance types, and ω-regular expressions. Furthermore we introduce the notion of an infinite game and explain the relation between ω-automata and infinite games. In the last section of chapter 1 we use infinite games to prove that the dual of an alternating automaton accepts the complementary language [MS87].

The second chapter deals with the different modes of transition functions. The basic concepts for the transformation between models with different modes of transition functions are extensions of the powerset construction of Myhill and Rabin/Scott for $*$-automata. We present Safra's construction [Saf88], transforming nondeterministic Büchi automata into deterministic Rabin automata. The other construction we introduce in the second chapter, the breakpoint construction, serves to transform alternating Büchi automata into nondeterministic ones. It was suggested in [MH84]. Both constructions are of optimal complexity. Concatenating these two constructions yields a doubly exponential construction transforming alternating automata into deterministic ones. In the last section of chapter 2 we show that this can not be improved.

The third chapter describes constructions to transform the type of acceptance condition. One concept (introduced by Büchi in an unpublished manuscript) is based on a data structure called later appearance record (LAR). With this structure it is possible to show that Muller, Rabin, Streett, and parity conditions have the same expressive power. As a new result we show the optimality of the LAR constructions. The proof is based on an idea from [DJW97], where the state appearance record, one kind of LAR, was shown to be a memory of optimal size for winning strategies in infinite games.

Another important construction is a rank construction as it appears e.g. in [KV97]. It transforms alternating co-Büchi automata into alternating weak automata and is of quadratic complexity. The known lower bound for such a transformation is of order $n \cdot \log n$, where n is the number of states.

Based on [KV97] we use this construction to give a complementation procedure for nondeterministic Büchi automata. A Büchi automaton is transformed into an equivalent alternating weak automaton, using a slightly modified rank construction. The resulting automaton is complemented and then transformed back into a nondeterministic Büchi automaton. The complementation procedure presented here uses a similar idea as the one in [KV97], but, due to the fact that the complementation step is done on the level of the alternating weak automata, it is clearer and more symmetric as the one presented in [KV97].

As a further contribution we investigate the fine structure of the class of ω-regular languages, based on the different models of ω-automata. Using the results from chapter 2 and 3, we can classify all of the considered models in the Landweber hierarchy, as shown in figure 1.3 on page 17. Landweber introduced his hierarchy using deterministic Muller (DM), deterministic Büchi (DB), and deterministic co-Büchi (DCB) automata. This means the mode of transition function is fixed (as deterministic) and the acceptance condition is changed to characterize the different levels.

$$\begin{tikzcd} & \mathrm{DM} \arrow[dl, no head] \arrow[dr, no head] & \\ \mathrm{DB} \arrow[dr, no head] & & \mathrm{DCB} \arrow[dl, no head] \\ & \mathrm{DB} \cap \mathrm{DCB} & \end{tikzcd}$$

Now we can characterize these levels by fixing the acceptance condition (as the weak one) and changing the mode of transition function, using alternating weak (AW), nondeterministic weak (NW), universal weak (UW), and deterministic weak (DW) automata.

$$\begin{tikzcd} & \mathrm{AW} \arrow[dl, no head] \arrow[dr, no head] & \\ \mathrm{UW} \arrow[dr, no head] & & \mathrm{NW} \arrow[dl, no head] \\ & \mathrm{DW} & \end{tikzcd}$$

In the last chapter we analyze the connection of the different types of automata to second order logic. A new result is the characterization of alternating automata in second order logic. We give characterizations of the form

$$\exists Y_1 \cdots Y_m \forall Z_1 \cdots Z_n (\text{first order kernel})$$

and

$$\forall Y_1 \cdots Y_m \exists Z_1 \cdots Z_n (\text{first order kernel}).$$

In logical terms this means to obtain a second order Δ_2-definition of ω-regular languages.

If we work with alternating weak automata, then the first order kernel of the formula describes weak acceptance, which can be done with a boolean combination of first order $\exists$-formulas.

This is in some contrast to the classical logical definitions, which use Σ_1-formulas or Π_1-formulas with $\exists^\omega$-formulas in the kernel.

Nondeterministic Büchi automata can be transformed into second order Σ_1-formulas with a first order $\exists^\omega$-formula as kernel. Deterministic Rabin automata can be transformed into both, Σ_1 and Π_1 second order formulas, with a boolean combination of first order $\exists^\omega$-formulas as kernel. So the mode of the transition function of the automaton influences the depth of the second order quantifiers in the corresponding logical formula and the acceptance condition influences the depth of the first order quantifiers in the kernel of the formula.

Acknowledgment

I thank my supervisor Wolfgang Thomas for his advice during my work on this thesis. He proposed most of the present results. I would also like to thank Thomas Wilke for several helpful discussions.

Chapter 1

ω-Automata

1.1 Notations and Basic Definitions

In this section we introduce some terminology and notations, and the basic definitions. In our terminology an ω-automaton has an alternating transition functions in general. Deterministic, nondeterministic and universal automata are regarded as special cases of alternation.

Notations

For an arbitrary set X we denote set of infinite sequences (or infinite words) over X by X^ω and the set of finite words over X by X^*. For a sequence $\sigma \in X^\omega \cup X^*$ and for an $i \in \mathbb{N}$ the element on the ith position in σ is denoted by $\sigma(i)$, i.e., $\sigma = \sigma(0)\sigma(1)\sigma(2)\cdots$. The infix of σ from position i to position j is denoted by $\sigma[i,j]$. Round parentheses instead of the squared ones indicate that the position is excluded. For example $\sigma[i,j)$ is the same as $\sigma[i,j-1]$.

When quantifying over natural numbers, we use as abbreviations of "there are infinitely many" and "for almost all" the quantifiers $\exists^\omega$ and $\forall^\omega$. This means we write $\exists^\omega j(\phi(j))$ for $\forall i\ \exists j(j > i \wedge \phi(j))$ and we write $\forall^\omega j(\phi(j))$ for $\exists i\ \forall j(j > i \rightarrow \phi(j))$.

For $\sigma \in X^\omega$ we define $In(\sigma)$, the *infinity set* of σ, to be the set of elements from X that appear infinitely often in σ. The set of states that appear at least once in σ, the *existence* set of σ, is denoted by $Ex(\sigma)$. Formally this means.

$$\begin{aligned} In(\sigma) &= \{x \in X \mid \exists^\omega j \in \mathbb{N}(\sigma(j) = x)\}. \\ Ex(\sigma) &= \{x \in X \mid \exists j \in \mathbb{N}(\sigma(j) = x)\}. \end{aligned}$$

The set of boolean formulas over X, denoted by $\mathcal{B}(X)$, is inductively defined as follows.

1. Every $x \in X$, `true` and `false` are atomic formulas in $\mathcal{B}(X)$.

2. Let θ_1, θ_2 be formulas in $\mathcal{B}(X) \setminus \{\texttt{true},\texttt{false}\}$. Then $\theta_1 \wedge \theta_2$ and $\theta_1 \vee \theta_2$ are formulas in $\mathcal{B}(X)$.

3. Let $x \in X$. Then $\neg x$ is a formula in $\mathcal{B}(X)$.

The set $\mathcal{B}^+(X)$ of the positive boolean formulas over X is defined with the rules 1 and 2 from above. This means the formulas in $\mathcal{B}^+(X)$ are built up from the elements of X, `true`, `false`, $\wedge$ and $\vee$.

Definitions

Definition 1.1 A subset S of X is said to *satisfy* $\theta \in \mathcal{B}(X)$ iff the truth assignment that assigns `true` to the elements of S and `false` to the elements of $X \setminus S$ satisfies θ. We say S *exactly satisfies* θ iff S satisfies θ and every subset of S does not satisfy θ.

It is not very difficult to see that there is a one to one correspondence between the sets exactly satisfying a formula $\theta \in \mathcal{B}^+(X)$ and the conjunctive terms in the disjunctive normalform of θ. Formally we have the following remark.

Remark 1.2 Let $\theta \in \mathcal{B}^+(X)$ and let $\mathcal{S} = \{S \subseteq X \mid S \text{ exactly satisfies } \theta\}$. Then θ is equivalent to the formula $\bigvee_{S\in\mathcal{S}} \left(\bigwedge_{x\in S} x\right)$.

For example let $X = \{x_1, x_2, x_3, x_4\}$ and $\theta = (x_1 \wedge x_3) \vee (x_3 \wedge x_4)$. The sets $\{x_1, x_3\}$ and $\{x_3, x_4\}$ are the sets exactly satisfying θ. Every superset of one of these sets satisfies θ.

With these conventions we can define ω-automata and the language accepted by them.

Definition 1.3 An *ω-automaton* $\mathcal{A}$ is a tuple $(Q, \Sigma, q_0, \delta, Acc)$. The tuple (Q, Σ, q_0, δ) is called the *transition structure* of $\mathcal{A}$, where $Q \neq \emptyset$ is a finite set of states, Σ is a finite alphabet, q_0 is the initial state and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is the transition function. The last component Acc is the *acceptance condition*, a formula from $\mathcal{B}(Q)$.

For two states $q, q' \in Q$ we say a transition leads from q to q' if and only if there is an $a \in \Sigma$ such that q' appears in the formula $\delta(q, a)$.

In the following we use automata synonymously with ω-automata.

Definition 1.4 Let $\alpha \in \Sigma^\omega$ and let $G = (V, E)$ be a directed acyclic graph with the following properties.

- $V \subseteq Q \times \mathbb{N}$ with $(q_0, 0) \in V$.
- $E \subseteq \bigcup_{l>0} (Q \times l) \times (Q \times l + 1)$.
- For every $(q, l) \in V \setminus \{(q_0, 0)\}$ exists a $q' \in Q$, such that $((q', l-1), (q, l)) \in E$.

G is called a *run* of $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ on α, if for every $(q, l) \in V$ the set $\{q' \in Q \mid ((q, l), (q', l+1)) \in E\}$ exactly satisfies $\delta(q, \alpha(l))$.

Note that there is no run $G = (V, E)$ on α, such that $(q, l) \in V$ for a q with $\delta(q, \alpha(l)) =$ `false`, since there is no set satisfying `false`. Furthermore we point out that the notion of a run does not depend on the acceptance condition. Thus we also can talk of a run of a transition structure.

The *slice* or *level* n of a run $G = (V, E)$ is the set of nodes with n as second component, i. e., $\{v \in V \mid v = (q, n) \text{ with } q \in Q\}$.

Definition 1.5 Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton and let $\alpha \in \Sigma^\omega$. Let $G = (V, E)$ be a run of $\mathcal{A}$ on α. A *path* through G is an infinite sequence of states $\pi \in Q^\omega$, such that $((\pi(i), i), (\pi(i+1), i+1)) \in E$ for every $i \in \mathbb{N}$. We say π is *accepting* iff $In(\pi)$ satisfies Acc. We say G is *accepting* iff all paths π through G are accepting. If a path or a run is not accepting, then it is called *rejecting*. The language accepted by $\mathcal{A}$ is defined as

$$L(\mathcal{A}) = \{\alpha \in \Sigma \mid \text{ There exists an accepting run of } \mathcal{A} \text{ on } \alpha\}.$$

From the definition of run and path we get the following remark.

Remark 1.6 Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton without `true` and `false` in the transition function and let $\alpha \in \Sigma^\omega$. A sequence $\pi \in Q^\omega$ is a path through a run of $\mathcal{A}$ on α if and only if for every $j \in \mathbb{N}$ there exists an $S \subseteq Q$ exactly satisfying $\delta(\pi(j), \alpha(j))$ with $\pi(j+1) \in S$.

Sometimes, when talking of a path through a run, we do not just mean the sequence of states but the sequence of vertices. But this will always be understood from the context.

The infinity set of a path can also be called *cycle* because the automaton finally cycles through the states of the infinity set. So when talking of a cycle we mean a set of states that may occur as infinity set of a path through a run. Therefore a cycle is accepting if and only if it satisfies the acceptance condition.

In general an automaton with a transition function as defined above is called alternating. Deterministic, nondeterministic and universal automata are special cases of alternating automata. If we work with these kinds of automata, we will explicitly mention this. In the next section we introduce these types of restricted transition functions.

1.2 Transition Functions

Automata as defined in the previous section are called alternating in general. In most of the papers dealing with alternating automata, the notion of a run is defined in a slightly different way. In fact this other notion is not equivalent to our definition of acceptance. In the section on acceptance types we mention for which kinds of acceptance conditions our notion is equivalent to the general notion explained below.

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an automaton with $n = |Q|$ and let $\alpha \in \Sigma^\omega$. A run tree of $\mathcal{A}$ on α is a Q-labeled tree $r : T_r \to Q$ where $T_r \subseteq \mathbb{N}^*$ is the set of nodes. A run tree must have the following properties.

- $r(\varepsilon) = q_0$.
- Let $x \in T_r$ with $r(x) = q$. Let $x_1, \ldots x_k$ be the successors of x and let $S = \{r(x_1), \ldots, r(x_k)\}$. If $|x|$ denotes the distance of x from the root of the tree, then S must exactly satisfy $\delta(q, \alpha(|x|))$.

We call such a run tree *memoryless* iff for every two nodes $x, y \in T_r$ with $|x| = |y|$ and $r(x) = r(y)$ the subtrees with x and y as roots, respectively, are equal. The definitions of path, accepting and language are analogue.

Given an automaton acceptance with run trees is equivalent to acceptance as defined in the previous section iff for every accepting run tree there is an accepting memoryless run tree. This is the case for several types of acceptance conditions and since we are mainly dealing with alternating automata of these types, we defined acceptance as in the previous section.

From an accepting memoryless run tree $r : T_r \to Q$ we can construct an accepting run $G = (V, E)$ in a natural way. This construction is taken from [KV97].

- $V \subseteq Q \times \mathbb{N}$ such that $(q, l) \in V$ iff there exists an $x \in T_r$ with $|x| = l$ and $r(x) = q$.
- $E \subseteq \bigcup_{l \geq 0}(Q \times \{l\}) \times (Q \times \{l+1\})$ such that $((q, l), (q', l+1)) \in E$ iff there exists an $x \in T_r$ and a child x' of x, such that $|x| = l$, $r(x) = q$ and $r(x') = q'$.

The idea is to merge all the vertices in the tree that have the same label and the same distance from the root. If we apply this transformation, the paths through G are exactly the same as the paths through the r, since the run tree was memoryless.

As mentioned in the previous section there are different kinds of transition functions, namely deterministic, nondeterministic, and universal ones. These are introduced in the following.

Determinism

An ω-automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ is called deterministic, if for all $q \in Q$ and $a \in \Sigma$ one has $\delta(q, a) \in Q$. Thus, if $\mathcal{A}$ is deterministic, the transition function is of the form $\delta : Q \times \Sigma \to Q$.

For a word $\alpha \in \Sigma^\omega$ there exists exactly one run G_α of $\mathcal{A}$ on α and there is exactly one path through G_α. So for deterministic automata we can identify the run with this unique path, which is an infinite sequence over Q.

Nondeterminism

An ω-automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ is called nondeterministic if for all $q \in Q$ and $a \in \Sigma$ there exist $q_1, \ldots, q_j \in Q$, $j \geq 0$, such that $\delta(q, a) = q_1 \vee \cdots \vee q_j$. The empty disjunction in the case $j = 0$ is `false` by definition. Since the only operator that is used is $\vee$, we can identify such a disjunction of states with the set of states occurring in the disjunction. This allows us to write δ as a function $\delta : Q \times \Sigma \to 2^Q$.

In difference to the deterministic case there may be many runs of a $\mathcal{A}$ on α, but all these runs contain only one path because a set that exactly satisfies a disjunction of states contains exactly one state. Thus, as for deterministic automata, we can identify a run of a nondeterministic automata with this path.

Sometimes, if $T = (Q, \Sigma, q_0, \delta)$ is a nondeterministic transition structure and F is a subset of Q, we need the following notation. Let $w \in \Sigma^*$ with $|w| = m$ and $q, p \in Q$. We write $q \xrightarrow{w}_F p$ if p is reachable from q, touching F, while reading w. Formally this means there exists $p_0, \ldots, p_{m-1} \in Q$ with $q = p_0$, $p = p_{m-1}$, $\{p_0, \ldots, p_{m-1}\} \cap F \neq \emptyset$ and for $i \in \{1, \ldots, m-1\}$ one has $p_i \in \delta(p_{i-1}, w(i-1))$.

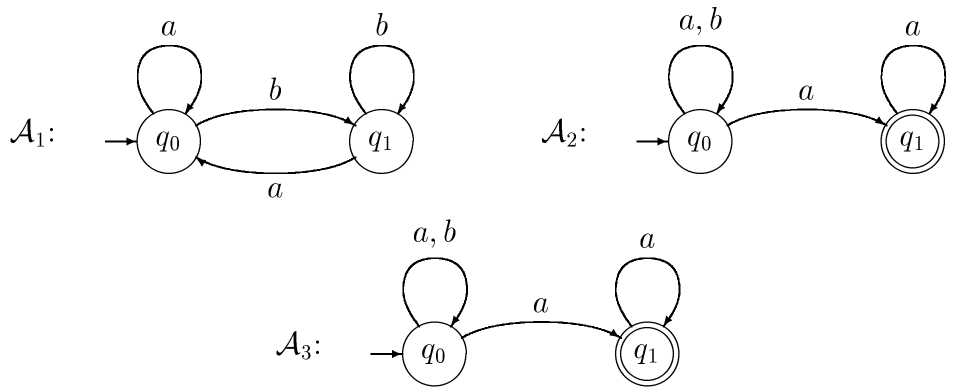


Figure 1.1: Transition Structures of the Example Automata

Universality

Universal automata are complementary to nondeterministic automata. An ω-automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ is called universal if for all $q \in Q$ and $a \in \Sigma$ there exist $q_1, \ldots, q_j \in Q$, $j \geq 0$, such that $\delta(q, a) = q_1 \wedge \cdots \wedge q_j$. The empty conjunction in the case $j = 0$ is `true` by definition. As for nondeterministic automata we can write δ as $\delta : Q \times \Sigma \to 2^Q$.

For universal automata there exists exactly one run on a word α because there exists only one set exactly satisfying a conjunction. But in this run there may be more than one path.

These different kinds of transition functions (deterministic, nondeterministic, universal and alternating) are called *modes* of transition functions. In chapter 2 we will give a detailed analysis of the relation between these different modes.

Examples

In all examples let $\Sigma = \{a, b\}$.

1. Let $\mathcal{A}_1 = (Q_1, \Sigma, q_0, \delta_1, Acc_1)$ with
   - $Q_1 = \{q_0, q_1\}$,
   - δ_1 defined by
     - $\delta_1(q_0, a) = q_0$,
     - $\delta_1(q_0, b) = q_1$,
     - $\delta_1(q_1, a) = q_0$,
     - $\delta_1(q_1, b) = q_1$ and
   - $Acc_1 = q_0 \wedge \neg q_1$.
2. Let $\mathcal{A}_2 = (Q_2, \Sigma, q_0, \delta_2, Acc_2)$ with
   - $Q_2 = \{q_0, q_1\}$,

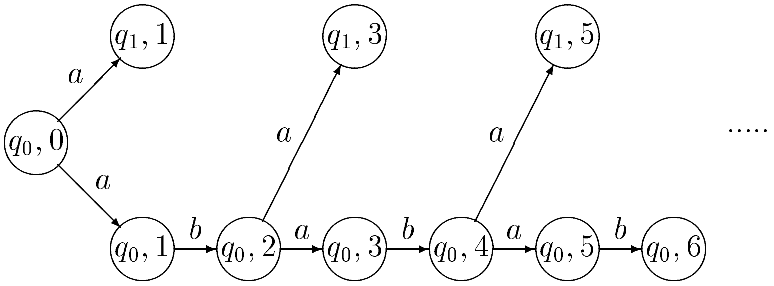


Figure 1.2: A Run of $\mathcal{A}_3$ on $ababab\cdots$

- δ_2 defined by
  - $\delta_2(q_0, a) = q_0 \vee q_1$,
  - $\delta_2(q_0, b) = q_0$,
  - $\delta_2(q_1, a) = q_1$,
  - $\delta_2(q_1, b) =$ `false` and
- $Acc_2 = q_1$.

3. Let $\mathcal{A}_3 = (Q_3, \Sigma, q_0, \delta_3, Acc_3)$ with
   - $Q_3 = \{q_0, q_1\}$,
   - δ_3 defined by
     - $\delta_3(q_0, a) = q_0 \wedge q_1$,
     - $\delta_3(q_0, b) = q_0$,
     - $\delta_3(q_1, a) = q_1$,
     - $\delta_3(q_1, b) =$ `true` and
   - $Acc_3 = \neg q_1$.

From the definitions above follows that $\mathcal{A}_1$ is deterministic, $\mathcal{A}_2$ is nondeterministic, and $\mathcal{A}_3$ is universal. The examples are illustrated by state graphs in figure 1.1. The automata $\mathcal{A}_1$ and $\mathcal{A}_2$ both accept all the words that contain only finitely many b's. The automaton $\mathcal{A}_3$ accepts all the words that contain infinitely many b's. The automata $\mathcal{A}_2$ and $\mathcal{A}_3$ are visualized in the same way. But since $\mathcal{A}_2$ is nondeterministic and $\mathcal{A}_3$ is universal, the interpretation of the transition structure is different.

In $\mathcal{A}_2$ and $\mathcal{A}_3$ the state q_1 is marked by a double circle, which will be explained in the next section.

Figure 1.2 shows the beginning of a run of $\mathcal{A}_3$ on the word $ababab\cdots$. Since $\mathcal{A}_3$ is universal, it is the only possible run.

In some sections we assume the used automata to be defined without `true` and `false` in their transition functions. This can be done without loss of generality, because for a given automaton one can introduce two new states q_{true} and q_{false} and replace every occurrence of `true`, `false` with q_{true}, q_{false}, respectively. For every letter of the alphabet q_{true} and q_{false} have themselves as successor.

To obtain an automaton that is equivalent to the given one, one has to adapt the acceptance condition such that every path with infinity set $\{q_{\text{true}}\}$ is accepting and every path with infinity set $\{q_{\text{false}}\}$ is rejecting.

1.3 Acceptance Types

In the previous section we have seen that there exist different modes of transition functions (alternating, nondeterministic, universal and deterministic). In this section we define certain acceptance types.

In an ω-automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ the acceptance condition is an arbitrary formula from $\mathcal{B}(Q)$. But in most cases one uses special normalforms of these formulas. This leads to different types of automata, which are named corresponding to their acceptance type. In these different types of automata the acceptance condition is not longer given as a boolean formula over the state set but as a combination of subsets of the state set. We use the following terminology. If an acceptance condition is, for example, of Rabin type, then we call it a Rabin condition.

In the following we list the types of acceptance conditions used in general.

Muller Conditions. In Muller conditions ([Mul63]) one directly specifies all the accepting cycles. Hence a Muller condition is represented by a system $\mathcal{F} \subseteq 2^Q$ of subsets of Q. The corresponding formula is of the form

$$\bigvee_{F\in\mathcal{F}} (\bigwedge_{q\in F} q \wedge \bigwedge_{q\in Q\setminus F} \neg q).$$

Since the acceptance condition can be represented by $\mathcal{F} \subseteq 2^Q$, we can write a Muller automaton as $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$. A run G on a word α is accepting iff for every path π through G one has $In(\pi) \in \mathcal{F}$. Obviously every acceptance condition can be transformed to a condition of Muller type, by transforming the formula into its disjunctive normalform.

Rabin Conditions. Rabin conditions [Rab69] can be represented by pairs of subsets of Q. Thus we can represent them by a set $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$, where $E_i, F_i \subseteq Q$ for $i \in \{1, \ldots, r\}$. The corresponding formula is of the form

$$\bigvee_{i\in\{1,\ldots,r\}} (\bigwedge_{q\in E_i} \neg q \wedge \bigvee_{q\in F_i} q).$$

A path π through a run G of a Rabin automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ with $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ is accepting iff there exists an $i \in \{1, \ldots, r\}$ such that $In(\pi) \cap E_i = \emptyset$ and $In(\pi) \cap F_i \neq \emptyset$.

Streett Conditions. Streett conditions ([Str82]) are dual to Rabin conditions. They also can be represented by pairs of subsets of Q but the interpretation of these pairs is complementary. So let $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ be as in a Rabin condition. The corresponding formula for a Streett condition is of the form

$$\bigwedge_{i \in \{1,\ldots,r\}} (\bigvee_{q \in E_i} q \vee \bigwedge_{q \in F_i} \neg q).$$

That is, a path π through a run G of a Streett automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ with $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ is accepting iff for all $i \in \{1, \ldots, r\}$ one has $In(\pi) \cap E_i \neq \emptyset$ or $In(\pi) \cap F_i = \emptyset$. This is equivalent to $In(\pi) \cap F_i \neq \emptyset$ implies $In(\pi) \cap E_i \neq \emptyset$.

Parity Conditions. Parity conditions [Mos84] are special kinds of Rabin conditions. They are also called Rabin-chain conditions and are represented and interpreted in the same way as Rabin conditions. The only difference is that the sets must form a chain. This means $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ can be interpreted as a parity condition iff $E_1 \subseteq F_1 \subseteq E_2 \subseteq F_2 \cdots \subseteq E_r \subseteq F_r$. The corresponding formula is built in the same way as for Rabin conditions.

Another way to represent a parity condition $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ is by means of a function $c : Q \to \{1, \ldots, 2r+1\}$. To define c we set $F_0 = \emptyset$ and $F_{r+1} = E_{r+1} = Q$. For $q \in Q$ we define

$$c(q) = \begin{cases} 2i & \text{if} \quad q \in F_i \setminus E_i, \\ 2i-1 & \text{if} \quad q \in E_i \setminus F_{i-1} \end{cases}$$

for $i \in \{1, \ldots, r+1\}$.

The elements from $\{1, \ldots, 2r+1\}$ are called colors and for a state $q \in Q$ the value $c(q)$ is the color of q. A path π through a run is accepting iff the smallest color of the states in $In(\pi)$ is even (this is why these conditions are called parity conditions). It is not difficult to see that this interpretation of a parity condition is equivalent to the one above.

Büchi Conditions. Büchi conditions [Büc62] can be represented by a subset of Q. For such a subset F the corresponding formula is of the form

$$\bigvee_{q \in F} q.$$

Hence, a path π through a run G of a Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ is accepting iff $In(\pi) \cap F \neq \emptyset$. The states in F are called final states. If we use state graphs to visualize automata of Büchi type, then the final states are marked by a double circle. The example automaton $\mathcal{A}_2$ from the previous section is of Büchi type. The only final state is q_1. Therefore it is marked with a double circle.

Co-Büchi Conditions. Co-Büchi conditions are dual to Büchi conditions and are also represented by an $F \subseteq Q$. The corresponding formula is of the form

$$\bigwedge_{q \in F} \neg q.$$

Muller	$\bigvee_{F \in \mathcal{F}} \Big(\forall^\omega i(\pi(i) \in F) \wedge \bigwedge_{q \in F} \exists^\omega i(\pi(i) = q)\Big)$	$\mathrm{Bool}(\exists^\omega)$
Rabin	$\bigvee_{k \in \{1,\dots,r\}} \Big(\exists^\omega i(\pi(i) \in F_k) \wedge \forall^\omega i(\pi(i) \notin E_k)\Big)$	$\mathrm{Bool}(\exists^\omega)$
Streett	$\bigwedge_{k \in \{1,\dots,r\}} \Big(\forall^\omega i(\pi(i) \notin F_k) \vee \exists^\omega i(\pi(i) \in E_k)\Big)$	$\mathrm{Bool}(\exists^\omega)$
Büchi	$\exists^\omega i\Big(\pi(i) \in F\Big)$	$\exists^\omega$
co-Büchi	$\forall^\omega i\Big(\pi(i) \notin F\Big)$	$\forall^\omega$
Weak	$\bigvee_{Q_k \subseteq F} \Big(\exists i(\pi(i) \in Q_k) \wedge \forall i(\pi(i) \notin \bigcup_{l<k} Q_l\Big)$	$\mathrm{Bool}(\exists)$

Table 1.1: Acceptance Conditions Expressed in First Order Logic

Hence, a path π through a run G of a co-Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ is accepting iff $In(\pi) \cap F = \emptyset$. In the state graphs of co-Büchi automata the states from F are also marked with a double circle as for example in $\mathcal{A}_3$ in figure 1.1.

Weak Conditions. Weak automata were introduced in [MSS86]. For a weak automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ it does not suffice to have a formula of a certain type as acceptance condition. The transition structure of a weak automaton must fulfill the following requirement. There is a partition $Q_1, \dots, Q_m$ of Q such that for every $q \in Q_i$ and $q' \in Q_j$ with a transition leading from q to q' one has $j \leq i$. It follows that for every path π through a run G of $\mathcal{A}$ there is an $i \in \{1, \dots, m\}$ such that $In(\pi) \subseteq Q_i$.

A weak acceptance condition can be represented by a subset F of Q such that there exists an $I \subseteq \{1, \dots, m\}$ with $F = \bigcup_{i \in I} Q_i$. The corresponding formula is of the form

$$\bigvee_{q \in F} q.$$

In a weak automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ a path π through a run G is accepting iff $In(\pi) \subseteq F$. The sets Q_i with $Q_i \subseteq F$ we call accepting and the other ones we call rejecting. We already said that $\mathcal{A}_3$ from figure 1.1 is a co-Büchi automaton. But with $Q_2 = \{q_0\}$, $Q_1 = \{q_1\}$ and $F = Q_2$ the automaton also is a weak automaton.

Weak acceptance can also be defined in terms of the states that are visited during a run instead of the states that are visited infinitely often. The automata syntactically stay the same but for a path π through a run we use $Ex(\pi)$ instead of $In(\pi)$ to evaluate the acceptance formula. Since the automaton can only move to lower sets in the partition, a path is accepting if it eventually visits an accepting set and never visits any of the lower sets in the partition. Thus the existence set of π satisfies the following formula iff π is accepting.

$$\bigvee_{Q_k \subseteq F} \Big(\bigvee_{q \in Q_k} q \wedge \bigwedge_{l<k} \bigwedge_{q \in Q_l} \neg q \Big)$$

In table 1.1 we described the different acceptance conditions with first order formulas. Expressing Muller, Rabin (and parity), and Streett acceptance conditions requires a boolean combination of formulas quantified with $\exists^\omega$ (Bool($\exists^\omega$)). For Büchi acceptance conditions it suffices to use just one formula quantified with $\exists^\omega$ and dual to this description we need a formula quantified with $\forall^\omega$ to express the co-Büchi condition. If we use the fact that the weak acceptance condition can also be expressed when using the existence set instead of the infinity set, then we can describe it in first order logic with a boolean combination of existential formulas (Bool($\exists$)).

In Section 1.2 we mentioned that acceptance with run trees and acceptance with run graphs is not equivalent in general. If we consider the acceptance types from above the the only types where this fails are Streett and Muller. But in connection with these types of automata we will never use the fact that we defined acceptance via run graphs.

An automaton is characterized by the mode of its transition function and by its acceptance condition. We characterize ω-languages in terms of the automaton that recognizes this language. For example we say a language L is deterministic Büchi if there exists a deterministic Büchi automaton that recognizes L.

Furthermore we sometimes use abbreviations for the different types of automata. The transition modes alternating, universal, nondeterministic, and deterministic we abbreviate by A, U, N, and D, respectively. The acceptance types Muller, Rabin, Streett, parity, Büchi, co-Büchi, and weak we abbreviate by M, R, S, P, B, CB, and W, respectively. A nondeterministic Büchi automaton is abbreviated by NB for example.

In chapter 3 we give a detailed analysis of the relations between the different acceptance conditions.

1.4 Regular Expressions

The languages accepted by finite automata over finite words are called regular. These are exactly the languages that can be characterized by regular expressions. Regular expressions over a finite alphabet Σ are built up with the operators $+, \cdot, {}^*$. For example the regular expression

$$a \cdot a \cdot (b + c)^*$$

over $\Sigma = \{a, b, c\}$ characterizes the language that contains all words with aa at the beginning, followed by an arbitrary word containing only b and c. The operator $\cdot$ can

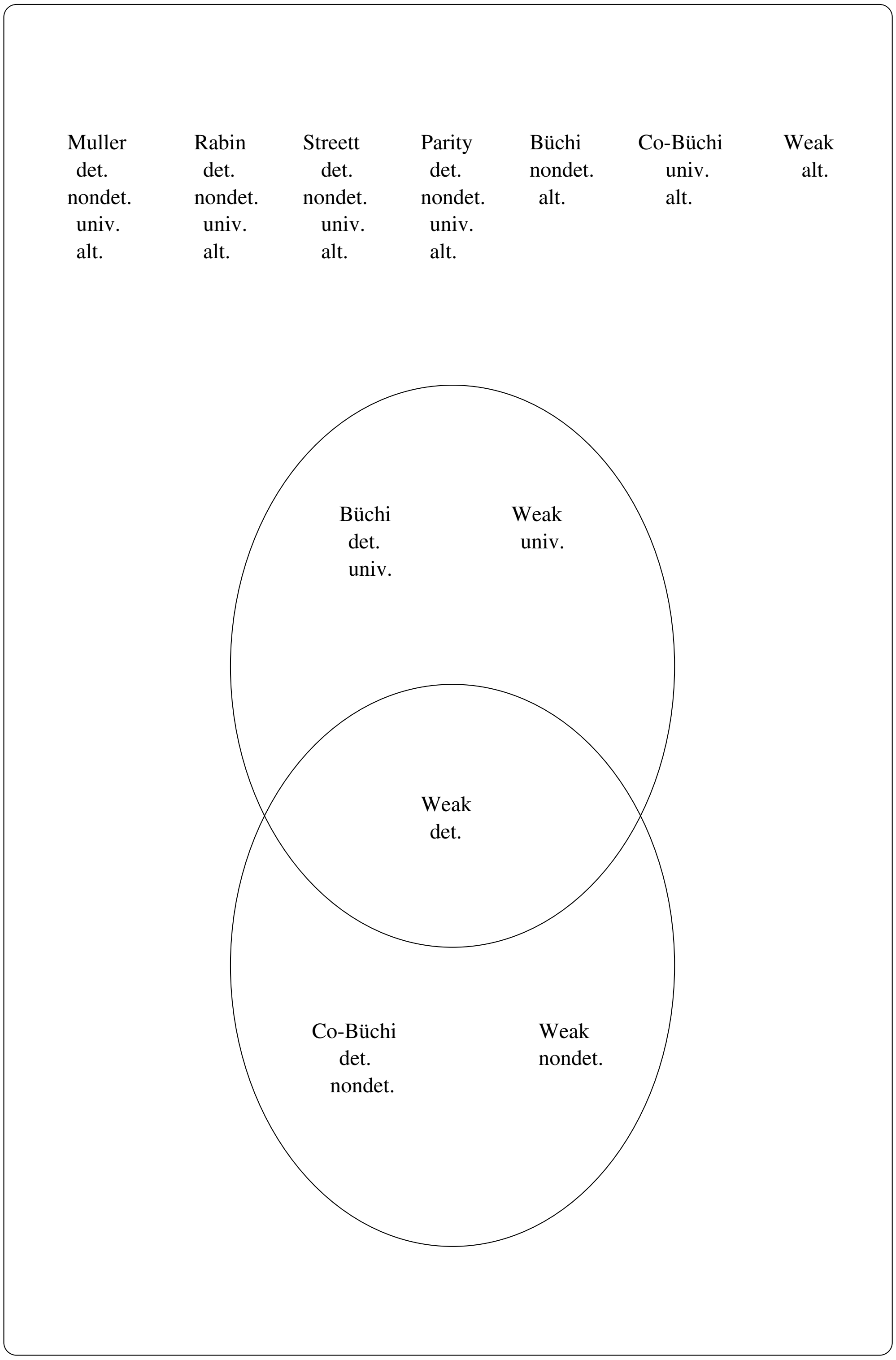


Figure 1.3: Hierarchy in the Expressiveness of Acceptance Conditions.

be omitted. For a regular expression α the language characterized by α is denoted by $L(\alpha)$.

In [McN66] McNaughton uses so called ω-regular expressions to characterize the Büchi recognizable languages. These ω-regular expressions can be defined from regular expressions by adding the operator $^\omega$. An ω-regular expression is of the form

$$\alpha_1\beta_1^\omega + \cdots + \alpha_n\beta_n^\omega,$$

where α_i, β_i are regular expressions for $i = 1, \ldots, n$, such that the languages characterized by the β_i do not contain the empty word. A word $\sigma \in \Sigma^\omega$ is in the language of an ω-regular expression $\gamma = \alpha_1\beta_1^\omega + \cdots + \alpha_n\beta_n^\omega$, if and only if there exists an $i \in \{1, \ldots, n\}$, and $u \in L(\alpha_i)$, $v_j \in L(\beta_i)$ for all $j \in \mathbb{N}$, such that $\sigma = uv_0v_1v_2\cdots$. These languages are called ω-regular.

It turns out, that the languages that are characterized by ω-regular expressions are exactly the languages that are accepted by nondeterministic Büchi automata.

Theorem 1.7 *Let Σ be a finite alphabet and let L be a language over Σ^ω. L is regular if and only if L can be recognized by a nondeterministic Büchi automaton.*

Proof Let $\gamma = \alpha_1\beta_1^\omega + \cdots + \alpha_n\beta_n^\omega$ and $L = L(\gamma)$. For $i = 1, \ldots, n$ let $\mathcal{A}_i$ be the deterministic $*$-automaton with $L(\alpha_i) = L(\mathcal{A}_i)$ and $\mathcal{B}_i$ be the deterministic $*$-automaton with $L(\beta_i) = L(\mathcal{B}_i)$. Now we merge $\mathcal{A}_i$ and $\mathcal{B}_i$ by adding for every transition that leads to a final state in $\mathcal{A}_i$ or $\mathcal{B}_i$ a transition that leads to the initial state of $\mathcal{B}_i$. The new final state is the initial state of $\mathcal{B}_i$. This procedure yields nondeterministic Büchi automata $\mathcal{C}_i$ with $L(\mathcal{C}_i) = L(\alpha_i\beta_i^\omega)$. From these automata we can construct a nondeterministic Büchi automaton $\mathcal{C}$ with $L(\mathcal{C}) = L(\mathcal{C}_1) \cup \cdots \cup L(\mathcal{C}_n)$

For the other direction let $\mathcal{A}$ be a nondeterministic Büchi automaton with $L(\mathcal{A}) = L$. Let $\{q_1, \ldots, q_n\}$ be the set of final states of $\mathcal{A}$. For q_i let $\mathcal{A}_i$ be the $*$-automaton with the same transition structure as $\mathcal{A}$ and q_i as the only final state. Let $\mathcal{B}_i$ be the same $*$-automaton as $\mathcal{A}_i$ with the only difference that q_i also is the initial state. From this construction follows $L(\mathcal{A}) = L(\mathcal{A}_1)L(\mathcal{B}_1)^\omega \cup \cdots \cup L(\mathcal{A}_n)L(\mathcal{B}_n)^\omega$. With regular expressions α_i, β_i such that $L(\alpha_i) = L(\mathcal{A}_i)$ and $L(\beta_i) = L(\mathcal{B}_i)$ for $i = 1, \ldots, n$ we obtain $L(\mathcal{A}) = L(\alpha_1\beta_1^\omega + \cdots + \alpha_n\beta_n^\omega)$. □

In figure 1.3 is illustrated what types of automata can recognize all the regular languages and what combinations of acceptance type and mode of transition function mean a proper restriction.

1.5 Games

The notion of an infinite game ([GS53],[Dav64]) is closely related to the theory of ω-automata. In this section we introduce games and explain their relation to automata.

We regard games consisting of a game graph and a winning condition. The nodes in the game graph are partitioned into two sets V_0 and V_1, the nodes of player 0 and the nodes of player 1. The edges go from V_0 to V_1 or vice versa. A match can be imagined as a marker that is moved along the edges. Every time it is on a node of V_0 player 0 has to choose the next edge, and every time it is on a node of V_1 player 1 has to choose the next edge. Thus, a game induces an infinite sequence of nodes. Player 0 wins the game if this sequence satisfies the winning condition and Player 1 wins otherwise.

Definition 1.8 A *game* is a tuple $G = (V_0, V_1, E, c, Win)$ with $V := V_0 \dot{\cup} V_1$, where V is a set of nodes, $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ is a set of edges such that $vE := \{w \in V \mid (v, w) \in E\}$ is finite and nonempty for every $v \in V$, $c : V \to C$ is a mapping from V into a finite set C and $Win \in \mathcal{B}(C)$ is a winning condition. The tuple (V_0, V_1, E, c) is called the *game graph*. A *match* starting at $v_0 \in V$ is an infinite sequence $\gamma = v_0 v_1 v_2 \cdots$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. Player 0 *wins* the match if and only if $In(c(\gamma))$ satisfies Win, where $c(\gamma) = c(v_0)c(v_1)c(v_2)\cdots$. Player 1 wins if and only if Player 0 does not win.

In such games there are starting points, such that one of the players can not loose if he plays with a certain strategy. Such a strategy is called winning strategy.

Definition 1.9 Let $G = (V_0, V_1, E, c, Win)$ be a game. A *match prefix* is a sequence $v_0 v_1 \cdots v_n$, such that $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \ldots, n-1\}$. A *strategy* for player i $(i = 0, 1)$ for a match starting at $v_0 \in V$ is a function $f_i : v_0 V^* \to V$, such that $(v_{n-1}, f_i(v_0 v_1 \cdots v_{n-1})) \in E$ for all match prefixes $v_0 v_1 \cdots v_{n-1}$ with $v_{n-1} \in V_i$. A *winning strategy* for player i for a match starting at v_0 is a strategy f_i for player i for a match starting at v_0, such that player i wins every match $\gamma = v_0 v_1 v_2 \cdots$, where $v_j = f_i(v_0 \cdots v_{j-1})$ for every $v_j \in V_{1-i}$. (This means player i wins all matches starting at v_0 if he plays according to his winning strategy.)

A node v belongs to the *winning area* of player i if and only if player i has a winning strategy for a match starting at v. It is obvious that, for a certain starting point v_0, only one of the players can have a winning strategy. From a theorem of Martin, where games are defined in a more general way, follows that in games as defined here for every starting point one of the players has a winning strategy. Such games are called *determined*.

Theorem 1.10 *If G is a game in the sense defined above, then G is determined.*

The theorem of Martin is a very deep theorem and we do not need it here in its full power. It states the determinacy of a very big class of games and we use it for a small subclass of these games.

In the following we will describe the relation between games and ω-automata. We assume that the automata are defined without `true` and `false` in their transition structures. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton and let $\alpha \in \Sigma^\omega$. A run of $\mathcal{A}$ on α is a directed acyclic graph as defined in section 1.1. We can view a path through a run of $\mathcal{A}$ on α as a match of the following game. The match starts at q_0. Then player 0 chooses a subset S_1 of Q exactly satisfying $\delta(q_0, \alpha(0))$. Player 1 chooses a state $q_1 \in S_1$. Now we repeat this procedure with q_1, $\alpha(1)$ instead of q_0, $\alpha(0)$ and so on. Thus a match of this game is an infinite sequence of the form $q_0 S_1 q_1 S_2 q_2 \cdots$. The sequence $\pi = q_0 q_1 q_2 \cdots$ is a path through a run of $\mathcal{A}$ on α. If we define the winning condition of the game to be the same as the acceptance condition of the automaton, then player 0 wins the match if and only if π is accepting. If we define this game formally, it does not suffice to take Q and 2^Q as nodes, but we also have to code the current index of the input word we are at.

Definition 1.11 Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton and let $\alpha \in \Sigma^\omega$. The game $G(\mathcal{A}, \alpha) = (V_0^{\mathcal{A}}, V_1^{\mathcal{A}}, E^{\mathcal{A},\alpha}, c^{\mathcal{A}}, Acc)$ is defined as follows.

- $V^{\mathcal{A}} = V_0^{\mathcal{A}} \cup V_1^{\mathcal{A}}$
- $V_0^{\mathcal{A}} = Q \times \mathbb{N}$.
- $V_1^{\mathcal{A}} = (2^Q \setminus \{\emptyset\}) \times \mathbb{N}$.
- Let $q \in Q$, $S \subseteq Q$ and $l \in \mathbb{N}$. $E^{\mathcal{A},\alpha}$ is defined by

$$\begin{array}{lll} ((q,l),(S,l+1)) \in E^{\mathcal{A},\alpha} & \text{iff} & S \text{ exactly satisfies } \delta(q,\alpha(l)), \\ ((S,l),(q,l)) \in E^{\mathcal{A},\alpha} & \text{iff} & q \in S \end{array}$$

- $c^{\mathcal{A}} : V^{\mathcal{A}} \to Q \cup 2^Q$, where $c^{\mathcal{A}}((q,l)) = q$ for $(q,l) \in V_0^{\mathcal{A}}$ and $c^{\mathcal{A}}((S,l)) = S$ for $(S,l) \in V_1^{\mathcal{A}}$.

Note that Acc is a formula from $\mathcal{B}(Q)$ and therefore it also is a formula from $\mathcal{B}(Q \cup 2^Q)$. If $T \subseteq Q \cup 2^Q$ satisfies Acc, then also $T \setminus 2^Q$ and $T \cup 2^Q$ do.

Since we want to relate games and automata, the only starting point in a match of $G(\mathcal{A}, \alpha)$ is $(q_0, 0) \in V_0^{\mathcal{A}}$. When we talk about a match of $G(\mathcal{A}, \alpha)$, we always mean a match starting at $(q_0, 0)$. As mentioned above, there is a relation between matches of $G(\mathcal{A}, \alpha)$ and paths through runs of $\mathcal{A}$ on α. From the definition of $G(\mathcal{A}, \alpha)$ and remark 1.6 we can deduce the following lemma.

Lemma 1.12 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an automaton, $\alpha \in \Sigma^\omega$ and let $\gamma = v_0 v_1 v_2 \cdots$ be a match of $G(\mathcal{A}, \alpha)$. The sequence $c^{\mathcal{A}}(v_0) c^{\mathcal{A}}(v_2) c^{\mathcal{A}}(v_4) \cdots$ from Q^ω is a path through a run of $\mathcal{A}$ on α.*

We will focus our interest on the relation between winning strategies and acceptance. Suppose that α is accepted by $\mathcal{A}$. Then there is an accepting run of $\mathcal{A}$ on α. Since player 0 chooses the sets exactly satisfying $\delta(q, \alpha(l))$, he can choose them like in the accepting run. It follows that, no matter what choices player 1 makes, the resulting match corresponds to a path through the accepting run and thus satisfies the winning condition. This means player 0 has a winning strategy. Vice versa we can construct an accepting run of $\mathcal{A}$ on α from a winning strategy of player 0. This leads to the following lemma.

Lemma 1.13 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an automaton and $\alpha \in \Sigma^\omega$.*

(i) *Player 0 has a winning strategy in $G(\mathcal{A}, \alpha)$ if and only if $\alpha \in L(\mathcal{A})$.*

(ii) *Player 1 has a winning strategy in $G(\mathcal{A}, \alpha)$ if and only if $\alpha \notin L(\mathcal{A})$.*

Proof (i): Let $\alpha \in L(\mathcal{A})$, $H = (V_H, E_H)$ an accepting run of $\mathcal{A}$ on α and $v_0 v_1 \cdots v_n$ be a match prefix of $G(\mathcal{A}, \alpha)$ with $v_n = (q, l) \in V_0^{\mathcal{A}}$. If $(q, l) \in V_H$, we define $f_0(v_0 \cdots v_n) = (S, l+1)$, where $S = \{p \in Q \mid ((q,l),(p,l+1)) \in E_H\}$. S exactly satisfies $\delta(q, \alpha(l))$. If $(q, l) \notin V_H$ we define $f_0(v_0 \cdots v_n) = (S, l+1)$, where S is an arbitrary set exactly satisfying $\delta(q, \alpha(l))$. The first observation, which can be easily proven by induction, is, if player 0 plays a match of $G(\mathcal{A}, \alpha)$ according to the strategy f_0, then all the nodes from $V_0^{\mathcal{A}}$ occurring in the match are also in V_H. The second observation is, that for every match γ of $G(\mathcal{A},\alpha)$ where player 0 plays according to f_0, the path corresponding to γ is a path through H. Thus $In(c(\gamma))$ satisfies Acc, because H is accepting.

For the other direction let f_0 be a winning strategy of player 0. Then we can built an accepting run $H = (V_H, E_H)$, by choosing the successors of a node (q, l) in V_H according to f_0. This can be done, if we regard a path from $(q_0, 0)$ to (q, l) as a match prefix. So we can built H stepwise, starting at $(q_0, 0)$. Since f_0 is a winning strategy, this results in an accepting run.

Part (ii) follows from (i), since $G(\mathcal{A},\alpha)$ is determined as stated in theorem 1.10. $\Box$

In the next section we use the formalism of games to prove the complementation theorem.

1.6 Duality and Complementation

In [MS87] Muller and Schupp gave an easy complementation procedure for alternating automata. They worked with tree automata and thus also covered the complementation of ω-automata as a special case. In this section we define the dual of a given automaton and prove that it accepts the complementary language. For this aim we need games, as introduced in section 1.5.

For simplicity we assume in this section that `true` and `false` do not appear in the transition functions of the automata. This can be avoided by using "sinking states".

For an arbitrary set X and $\theta \in \mathcal{B}^+(X)$ we define $\widetilde{\theta}$, the dual of θ, inductively by

$$\widetilde{\theta} = \begin{cases} \texttt{true} & \text{if } \theta = \texttt{false}, \\ \texttt{false} & \text{if } \theta = \texttt{true}, \\ x & \text{if } \theta = x \text{ with } x \in X, \\ \widetilde{\theta_1} \vee \widetilde{\theta_2} & \text{if } \theta = \theta_1 \wedge \theta_2 \text{ with } \theta_1, \theta_2 \in \mathcal{B}^+(X), \\ \widetilde{\theta_1} \wedge \widetilde{\theta_2} & \text{if } \theta = \theta_1 \vee \theta_2 \text{ with } \theta_1, \theta_2 \in \mathcal{B}^+(X). \end{cases}$$

Thus we get the dual of a formula by exchanging $\wedge$ with $\vee$ and `true` with `false`. We can state the following relation between sets exactly satisfying θ and sets satisfying $\widetilde{\theta}$.

Lemma 1.14 *Let $\theta \in \mathcal{B}^+(X)$. A set $S \subseteq X$ satisfies $\widetilde{\theta}$ if and only if $S \cap R \neq \emptyset$ for all R exactly satisfying θ.*

Proof Let $\mathcal{R} = \{R \subseteq X \mid R \text{ exactly satisfies } \theta\}$. From remark 1.2 and the definition of $\widetilde{\theta}$ it follows that $\widetilde{\theta}$ is equivalent to

$$\bigwedge_{R \in \mathcal{R}} \bigvee_{x \in R} x.$$

This is the conjunctive normalform of $\widetilde{\theta}$. S satisfies θ if and only if it contains at least one element from each of the disjunctive terms. $\Box$

For a formula $\theta \in \mathcal{B}(X)$ we define $\overline{\theta}$, the complement of θ, inductively by

$$\overline{\theta} = \begin{cases} \texttt{true} & \text{if } \theta = \texttt{false}, \\ \texttt{false} & \text{if } \theta = \texttt{true}, \\ \neg x & \text{if } \theta = x \text{ with } x \in X, \\ x & \text{if } \theta = \neg x \text{ with } x \in X, \\ \overline{\theta_1} \vee \overline{\theta_2} & \text{if } \theta = \theta_1 \wedge \theta_2 \text{ with } \theta_1, \theta_2 \in \mathcal{B}(X), \\ \overline{\theta_1} \wedge \overline{\theta_2} & \text{if } \theta = \theta_1 \vee \theta_2 \text{ with } \theta_1, \theta_2 \in \mathcal{B}(X). \end{cases}$$

With a simple induction one can show the following remark.

Remark 1.15 Let $\theta \in \mathcal{B}(X)$ for an arbitrary set X, and let $S \subseteq X$. S satisfies θ if and only if S does not satisfy $\overline{\theta}$.

Now we can define the dual of an automaton.

Definition 1.16 Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton. The *dual automaton* $\widetilde{\mathcal{A}}$ of $\mathcal{A}$ is defined as $\widetilde{\mathcal{A}} = (Q, \Sigma, q_0, \widetilde{\delta}, \overline{Acc})$, where $\widetilde{\delta}$ is defined by $\widetilde{\delta}(q, a) = \widetilde{\delta(q, a)}$ for all $q \in Q$ and $a \in \Sigma$.

From this definition we can see that the dual of a nondeterministic automaton is universal and vice versa. If an automaton has a deterministic transition structure, then also has its dual. As one can easily see, the complement of a Muller, parity and weak condition is again of the same type. The complement of a Rabin condition is a Streett condition and vice versa. The same holds for the Büchi and co-Büchi condition.

To prove the main theorem of this section we first show that a winning strategy of player 0 in $G(\mathcal{A}, \alpha)$ induces a winning strategy of player 1 in $G(\widetilde{\mathcal{A}}, \alpha)$ and vice versa.

Lemma 1.17 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton and $\alpha \in \Sigma^\omega$. Player 0 has a winning strategy in $G(\mathcal{A}, \alpha)$ if and only if player 1 has a winning strategy in $G(\widetilde{\mathcal{A}}, \alpha)$.*

Proof First, note that the winning condition in $G(\mathcal{A}, \alpha)$ only depends on the first component of the states in $V_0^{\mathcal{A}}$. So for these games it suffices to regard a winning strategy for player 0 as a mapping $f_0 : q_0 Q^* \to 2^Q \setminus \{\emptyset\}$. This means the next choice of player 0 only depends on the former choices of player 1. Such a mapping can be extended to a winning strategy in the sense of definition 1.9 by projecting a match prefix $v_0 \cdots v_{n-1}$ (with $n-1$ even) to $c(v_0)c(v_2)\cdots c(v_{n-1})$ and then applying f_0.

A winning strategy for player 1 also depends only on his former choices and in addition on the last choice of player 0, since he has to choose a state from the set chosen by player 0. Hence, we can regard a winning strategy for player 1 as a mapping $f_1 : q_0 Q^* \times (2^Q \setminus \{\emptyset\}) \to Q$.

Let f_0 be a strategy for player 0 in $G(\mathcal{A}, \alpha)$. Define $\tilde{f}_1 : q_0 Q^* \times (2^Q \setminus \{\emptyset\}) \to Q$ as follows. Let $q_0 \cdots q_{n-1} \in q_0 Q^*$ and $S \subseteq Q$, such that S exactly satisfies $\widetilde{\delta}(q_{n-1}, \alpha(n-1))$ (for sets not exactly satisfying $\widetilde{\delta}(q_{n-1}, \alpha(n-1))$ we do not have to define the strategy, since they never do appear in a match prefix). Let $q_n \in S \cap f_0(q_0 \cdots q_{n-1})$. Such a q_n exists by lemma 1.14, since $f_0(q_0 \cdots q_{n-1})$ exactly satisfies $\delta(q_{n-1}, \alpha(n-1))$. Now let $\tilde{f}_1((q_0 \cdots q_{n-1}, S)) = q_n$. This means player 1 always picks a state from the set that is determined by f_0. Thus, for a match γ of $G(\widetilde{\mathcal{A}}, \alpha)$ played according to $\tilde{f}_1$, there exists

Transition Mode of $\mathcal{A}$	Transition Mode of $\widetilde{\mathcal{A}}$
Deterministic	Deterministic
Nondeterministic	Universal
Universal	Nondeterministic
Alternating	Alternating
Acceptance Type of $\mathcal{A}$	**Acceptance Type of $\widetilde{\mathcal{A}}$**
Muller	Muller
Rabin	Streett
Streett	Rabin
Parity	Parity
Büchi	Co-Büchi
Weak	Weak

Table 1.2: Dualized Transition Modes and Acceptance Types

a match γ' of $G(\mathcal{A}, \alpha)$ played according to f_0, with $In(c^{\widetilde{\mathcal{A}}}(\gamma)) \cap Q = In(c^{\mathcal{A}}(\gamma')) \cap Q$. Therefore $\tilde{f}_1$ is a winning strategy of player 1 in $G(\widetilde{\mathcal{A}}, \alpha)$.

Now let $f_1 : q_0Q^* \times (2^Q \setminus \{\emptyset\}) \to Q$ be a winning strategy of player 1 in $G(\mathcal{A}, \alpha)$. If we show that f_1 induces a winning strategy for player 0 in $G(\widetilde{\mathcal{A}}, \alpha)$, then we are done, since the dual of $\widetilde{\mathcal{A}}$ is again $\mathcal{A}$. We define a winning strategy $\tilde{f}_0 : q_0Q^* \to 2^Q \setminus \{\emptyset\}$ of player 0 in $G(\widetilde{\mathcal{A}}, \alpha)$. Let $q_0 \cdots q_{n-1} \in q_0Q^*$ and let $S_1, \ldots, S_l$ be the sets exactly satisfying $\delta(q_{n-1}, \alpha(n-1))$. The set $S = \{q \in Q \mid q = f_1((q_0 \cdots q_{n-1}, S_i)) \text{ for an } i\}$ satisfies $\widetilde{\delta}(q_{n-1}, \alpha(n-1))$ by lemma 1.14. Let S' be a subset of S that exactly satisfies $\widetilde{\delta}(q_{n-1}, \alpha(n-1))$ and define $\tilde{f}_0(q_0 \cdots q_{n-1}) = S'$. This means player 0 in $G(\widetilde{\mathcal{A}}, \alpha)$ chooses a set of states that consists of possible choices of player 1 in $G(\mathcal{A}, \alpha)$ according to f_1. When player 0 plays a match γ of $G(\widetilde{\mathcal{A}}, \alpha)$ according to this strategy, then there exists a match γ' of $G(\mathcal{A}, \alpha)$ that is played according to f_1 by player 1, such that $In(c^{\widetilde{\mathcal{A}}}(\gamma)) \cap Q = In(c^{\mathcal{A}}(\gamma')) \cap Q$. As a consequence of this player 0 wins γ. □

With this lemma we can easily prove the complementation theorem.

Theorem 1.18 [MS87] *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an ω-automaton. The dual automaton of $\mathcal{A}$ accepts the complement of $L(\mathcal{A})$, i.e., $L(\widetilde{\mathcal{A}}) = \overline{L(\mathcal{A})}$.*

Proof If $\alpha \in L(\mathcal{A})$, then player 0 has a winning strategy in $G(\mathcal{A}, \alpha)$ and by lemma 1.17 player 1 has a winning strategy in $G(\widetilde{\mathcal{A}}, \alpha)$. By lemma 1.13 this is equivalent to $\alpha \notin L(\widetilde{\mathcal{A}})$. The other direction follows in the same way. □

In table 1.2 we listed for every mode of transition function and every acceptance type the dualized version.

Using theorem 1.18 we can define acceptance of a word by an automaton in a second way.

Definition 1.19 Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, Acc)$ be an automaton and $\alpha \in \Sigma^\omega$. A *dual run* of $\mathcal{A}$ on α is a run of $\widetilde{\mathcal{A}}$ on α. A dual run G of $\mathcal{A}$ is called accepting for $\mathcal{A}$ if and only if there exists a path π through G such that $In(\pi)$ satisfies Acc.

Theorem 1.18 implies the following lemma.

Lemma 1.20 *A word $\alpha \in \Sigma^\omega$ is in $L(\mathcal{A})$ if and only if every dual run of $\mathcal{A}$ is accepting for $\mathcal{A}$.*

We use this second definition of acceptance in chapter 4 to give two different characterizations of alternating automata in second order logic.

Chapter 2

Modes of Transition Functions

In the theory of $*$-automata the equivalence of deterministic and nondeterministic automata is shown by means of the powerset construction of Rabin and Scott (see for example [HU79]). For ω-automata the situation turns out to be much more difficult because of the infinite inputs. Besides there are many different acceptance types. In this chapter we introduce some extensions of the powerset construction for transformations between automata with different modes of transition functions.

2.1 Safra's Determinization Construction

In section 3.4 we will see that nondeterministic Büchi automata are stronger than deterministic Büchi automata. Therefore we can not find a construction transforming nondeterministic Büchi automata into equivalent deterministic Büchi automata. Figure 1.3 shows that in a determinization procedure the resulting automaton must have the acceptance type Muller, Rabin, Streett, or parity. In [McN66] McNaughton gave a construction to transform ω-regular expressions into deterministic Rabin automata.

In this subsection we present an extension of the usual powerset construction suggested by Safra in [Saf88]. In this construction a nondeterministic Büchi automaton is transformed into a deterministic Rabin automaton. The resulting automaton is singly exponential in the size of the original automaton with an almost linear exponent. Before we explain Safra's construction, analyze its complexity and show its optimality, we will give a short explanation why the usual powerset construction does not work.

In the usual powerset construction one collects the states that are reachable in the given automaton after having read a certain prefix of the input. The new set of final states consists of the state sets that contain a final state. For $*$-automata this construction suffices to determinize a nondeterministic automaton. For ω-automata there may be infinitely many runs, every visiting only finitely many times a final state. Merging these runs in the powerset construction results in an accepting run of the deterministic automaton. Take For example the automaton from figure 2.1. The input $(ab)^\omega$ is rejected. But in the powerset automaton this input would have the accepting run $(\{0\}\{0,1\})^\omega$.

The idea in Safra's extension of the powerset construction is to branch a new computation path every time the given automaton reaches a final state. Therefore the states

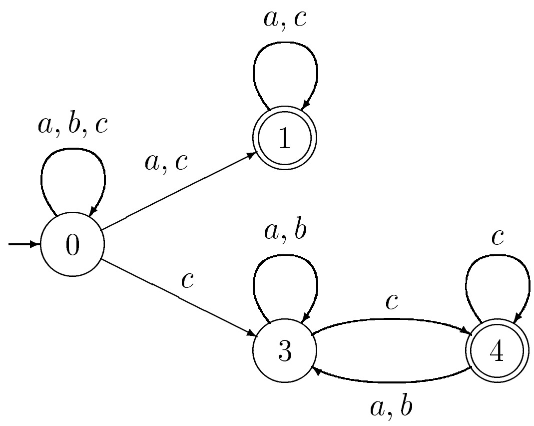


Figure 2.1: Example Automaton to Illustrate Safra's Powerset Construction

in the resulting automaton are not just sets but finite ordered trees, called Safra trees. In these Safra trees the nodes are labeled with sets of states.

We illustrate the idea by an example. Let $\mathcal{A}$ be the resulting automaton if we apply Safra's construction to the automaton from figure 2.1. The rightmost trees of each row in figure 2.2 give the sequence of $\mathcal{A}$-states induced by the input *ccbcb*.

The initial tree is the one just consisting of a root labeled with $\{0\}$. Since 0 is not a final state, we just apply the powerset construction on 0 with input c. The result is the tree with one node, labeled with $\{0, 1, 3\}$. Now we branch a new node because 1 is a final state. After having done this we again apply the powerset construction to every node.

In the next step we again branch new nodes for the final states in every set. The set $\{0, 1, 4\}$ contains two final states. Hence we branch a new node and label it with $\{1, 4\}$. The set $\{1\}$ contains one final state and thus we branch a new node and label it with $\{1\}$. Then we apply the powerset construction to every node. In the resulting tree there are nodes labeled with the empty set. These nodes are removed in the next step. After all these actions the resulting tree is the next Safra tree in our sequence.

To get the next Safra tree we just have to apply the powerset construction because none of the labels contains a final state.

The first two steps in the last row of our example are again the branching of new nodes for final states and the powerset construction. In the resulting tree are two nodes with the same parent which both have 3 in their label. Hence we remove the state 3 from the youngest child. Then the nodes with empty labels are removed. Now the tree contains a node such that the union of the labels of its children equals its own label. Here the node and its only child both have label $\{3\}$. In this situation we remove all descendants of the node and mark it (in the picture we used an exclamation mark).

The last row in the example shows all necessary actions to define the transition function of the new automaton. But before we come to the construction we formally define the notion of a Safra tree.

Definition 2.1 A *Safra tree* over a finite, nonempty set Q with $|Q| = n$ is a finite ordered tree. The nodes have names from $\{1, \ldots, 2n\}$ and labels from $2^Q \setminus \{\emptyset\}$. Every

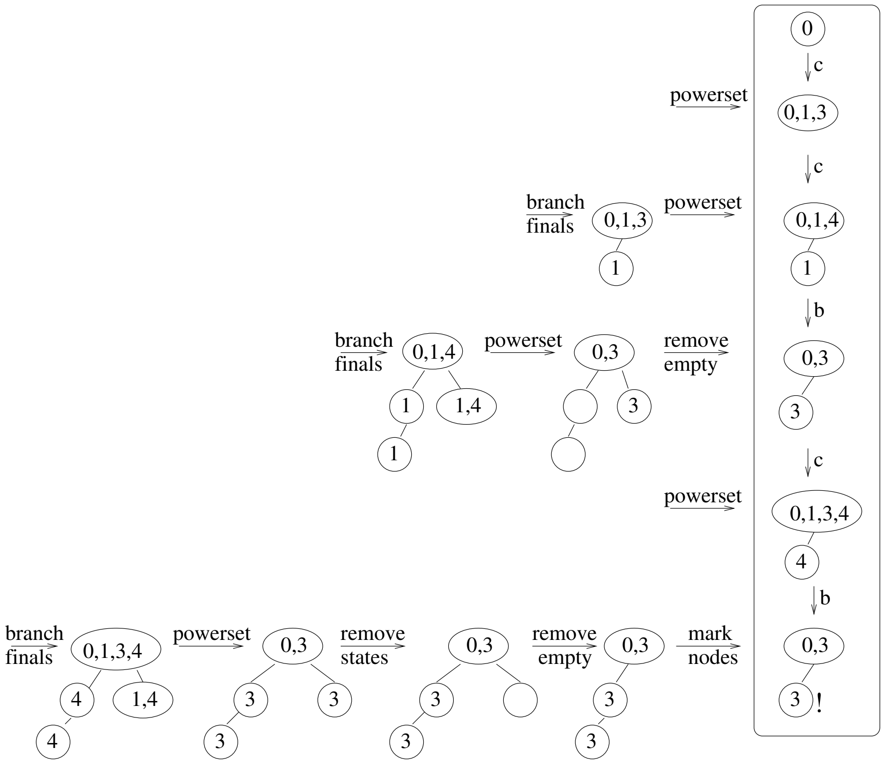


Figure 2.2: Sequence of Safra Trees Induced by *ccbcb*

node can be marked or unmarked. The label of a node is a proper superset of the union of the labels of its children. Two nodes with the same parent have disjoint labels.

Since Q is finite, the number of Safra trees over Q is finite too. More precisely we can bound the number of nodes in a Safra tree as follows.

Lemma 2.2 *A Safra tree over a set Q with $|Q| = n$ has at most n nodes.*

Proof We prove the lemma by induction over the depth of the tree.

If the depth is 1, then the root is the only node in the tree.

Let $h+1$ be the depth of the tree. Let $i_1, \ldots, i_k$ be the children of the root with labels $Q_1, \ldots, Q_k$. For every $j \in \{1, \ldots, k\}$ the subtree with i_j as root is a Safra tree over Q_j with depth at most h. By the induction hypothesis this subtree has at most $n_j = |Q_j|$ nodes. The Q_j are disjoint and their union is a proper subset of Q. Therefore $\sum_{j=1}^{k} n_j < n$. This means the number of nodes in the whole Safra tree is at most $1 + \sum_{j=1}^{k} n_j < 1 + n \leq n$. $\square$

Theorem 2.3 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a nondeterministic Büchi automaton with n states. There exists an equivalent deterministic Rabin automaton $\mathcal{A}' = (Q', \Sigma, q_0', \delta', \Omega)$ with $2^{\mathcal{O}(n \cdot \log n)}$ states and $\mathcal{O}(n)$ pairs.*

Proof Define $\mathcal{A}'$ as follows.

- Q' = The set of all Safra trees over Q.
- q_0' = The Safra tree just consisting of the node 1, labeled with $\{q_0\}$.
- The transition function δ' transforms a Safra tree, given an input $a \in \Sigma$, using the following steps.
  1. Unmark all nodes.
  2. For every node with label S such that $S \cap F \neq \emptyset$ create a new youngest child with label $S \cap F$. The name of this node may be any unused name from $\{1, \dots, 2n\}$.
  3. Apply the powerset construction to every node. This means replace every label S by $\bigcup_{q \in S} \delta(q, a)$.
  4. For every two nodes with the same parent such that $q \in Q$ is contained in both labels remove q from the label of the younger child and all its children.
  5. Remove all nodes with empty label.
  6. For every node whose label equals the union of the labels of its children remove all descendants of this node and mark it.
- The acceptance condition $\Omega = \{(E_1, F_1), \dots, (E_{2n}, F_{2n})\}$ is defined by

  $$\begin{aligned} F_i &= \text{set of all Safra trees with node } i \text{ marked,} \\ E_i &= \text{set of all Safra trees without node } i, \end{aligned}$$

  for $i \in \{1, \dots, 2n\}$.

First note that the transition function in fact is a mapping from $Q' \times \Sigma$ into Q'. The resulting tree after applying the first three steps not necessarily is a Safra tree. But as we can see from the definition of a Safra tree the required structure is reestablished in steps 4 to 6. From lemma 2.2 follows that the number of different node names is sufficient because a Safra tree has at most n nodes and every node gets at most one new child in step 2. Therefore the tree after step 2 has at most $2n$ nodes.

It may happen that all nodes must be removed. In this case there is no successor. Formally in a deterministic automaton there must be a successor for every pair of state and letter. To avoid this situation we can insert a sinking state representing the empty tree.

Now we turn to the correctness proof.

$L(\mathcal{A}) \subseteq L(\mathcal{A}')$: Let $\alpha \in L(\mathcal{A})$ and let $\sigma \in Q^\omega$ be a successful run of $\mathcal{A}$ on α. The root will never be removed, since it always includes the state of σ. If the root is marked infinitely often, then we are done. Otherwise, consider the next time σ visits a final state after the last time the root was marked. A new child of the root is branched. From now on the state of σ always is included in one of the children of the root. The state of σ may only move to older children of the root. Therefore there eventually exists a child of the root that is never removed again. Now we can repeat the arguments. If this node is marked infinitely often, then we are done. Otherwise, as above, we obtain a child of this node that is never removed again. Since the depth of the Safra trees is bounded by n, we finally will find a node that is marked infinitely often and will never be removed from some point on.

$L(\mathcal{A}') \subseteq L(\mathcal{A})$: Let $\alpha \in L(\mathcal{A}')$ and let $\sigma' \in Q'^{\omega}$ be the successful run of $\mathcal{A}'$ on α. Then there exists a node v that occurs in every tree of σ' from some point on and that is marked infinitely often. Let x be the position such that the node v occurs in every tree $\sigma'(y)$ for $y \geq x$. Now let $x \leq i_1, i_2, \ldots$ be the positions in σ' where node v is marked and let $Q_1, Q_2, \ldots$ be the labels of v at these positions. Furthermore let $i_0 = 0$ and $Q_0 = \{q_0\}$. From the definition of δ' (step 6) follows that for every $j \in \mathbb{N}$ and $q \in Q_{j+1}$ there exists a $p \in Q_j$ such that $p \xrightarrow{\alpha[i_j, i_{j+1})}_F q$. Define a tree with the nodes (q, j) where $j \in \mathbb{N}$ and $q \in Q_j$. The parent of a node $(q, j+1)$ is any node (p, j) with $p \xrightarrow{\alpha[i_j, i_{j+1})}_F q$. This is a well formed infinite tree. Since every node has less than n children, there exists an infinite path in the tree by König's Lemma. From the construction of the tree one can see that there is an accepting run of $\mathcal{A}$ on α corresponding to this path.

Complexity:

The number of pairs in the acceptance condition is the same as the number of possible names for nodes. Therefore $\mathcal{A}'$ has $2n$ pairs. To bound the number of states by $2^{\mathcal{O}(n \cdot \log n)}$ we use the following representation of the Safra trees.

We say a node v is characteristic for a state q if and only if q occurs in the label of v but in none of the labels of v's children. There is exactly one characteristic node for every state occurring in the root. The other states do not have a characteristic node. We can describe this relation by a mapping $Q \to \{0, \ldots, 2n\}$. This mapping contains the information which nodes are in the tree and which labels they have. For a complete description of the tree we additionally need a mapping $\{1, \ldots, 2n\} \to \{0, 1\}$ for the marked nodes, a mapping $\{1, \ldots, 2n\} \to \{0, \ldots, 2n\}$ for the parent relation and a mapping $\{1, \ldots, 2n\} \to \{0, \ldots, 2n\}$ to describe the ordering of nodes with the same parent.

Using this description we can bound the number of Safra trees by the number of different quadruples of such functions. This means

$$Q' \leq (2n+1)^n \cdot 2^{2n+1} \cdot (2n+1)^{2n} \cdot (2n+1)^{2n} \leq (2n+1)^{8n} \in 2^{\mathcal{O}(n \cdot \log n)}.$$

□

Theorem 2.3 also allows us to transform nondeterministic Rabin, Streett, and Muller automata into deterministic Rabin automata. This follows from the results presented in section 3.4, where we transform these automata into Büchi automata.

For the determinization of Muller automata the concatenation of these two constructions yields a doubly exponential blow up. This can not be improved, as shown in [SV89].

If we concatenate the two constructions to determinize Rabin automata, then the resulting automaton has $2^{\mathcal{O}(rn\log(rn))}$ states (n number of states, r number of pairs). But for Streett automata the blow up is doubly exponential in the number of pairs, when concatenating the two constructions. This can be avoided as shown by Safra in [Saf92]. The construction suggested there transforms a nondeterministic Streett automaton with n states and r pairs into a deterministic Rabin automaton with $2^{\mathcal{O}(rn\log(rn))}$ states.

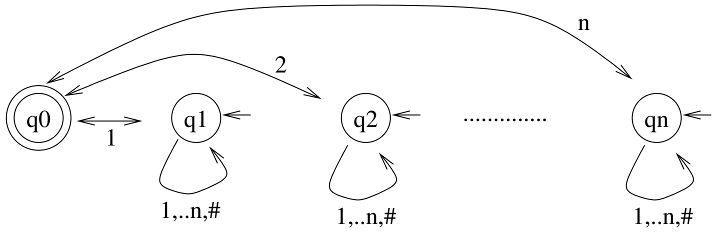


Figure 2.3: A complementary Büchi automaton needs at least $n!$ states.

2.2 Optimality of Safra's Construction

We will show that Safra's construction is of optimal complexity for the transformation of nondeterministic Büchi automata into deterministic Rabin automata. The proof was suggested by Michel in 1988. It uses a family of languages L_n which can be recognized by nondeterministic Büchi automata with $\mathcal{O}(n)$ states but not by deterministic Rabin automata with less than $n!$ states. For better understanding we define the Büchi automaton $\mathcal{A}_n$ over the alphabet $\Sigma_n = \{1, \ldots, n, \#\}$. As for $*$-automata, one can allow more than one initial state in nondeterministic ω-automata.

Define the automaton $\mathcal{A}_n = (Q_n, \Sigma_n, Q_0^n, \delta_n, F_n)$ as follows.

- $Q_n = \{q_0, q_1, \ldots, q_n\}$.
- $\Sigma_n = \{1, \ldots, n, \#\}$.
- $Q_0^n = \{q_1, \ldots, q_n\}$.
- The transition function δ_n is defined by

$$\begin{array}{lcll} \delta_n(q_0, a) & = & \{q_a\} & \text{for } a \in \{1, \ldots, n\}, \\ \delta_n(q_0, \#) & = & \emptyset, & \\ \delta_n(q_i, a) & = & \{q_i\} & \text{for } a \in \Sigma_n, i \in \{1, \ldots, n\}, a \neq i, \\ \delta_n(q_i, i) & = & \{q_i, q_0\} & \text{for } i \in \{1, \ldots, n\}. \end{array}$$

- $F_n = \{q_0\}$

The automaton looks like shown in figure 2.3. The ideas can be adjusted to automata with the constant alphabet $\{a, b, \#\}$ by coding $i \in \{1, \ldots, n-1\}$ with $a^i b$, n with $a^n a^* b$ and $\#$ with $\#$. The resulting automaton is shown in figure 2.4 and still has $\mathcal{O}(n)$ states.

As an abbreviation we define $L_n = L(\mathcal{A}_n)$. The following lemma characterizes the language L_n

Lemma 2.4 *Let $n \in \mathbb{N}$ and $\alpha \in \Sigma_n^\omega$. Then the following two statements are equivalent.*

(i) $\alpha \in L_n$.

(ii) *There exist letters $i_1, \ldots, i_k \in \{1, \ldots, n\}$ such that $i_1 i_2, \ldots, i_{k-1} i_k, i_k i_1$ appear infinitely often in α.*

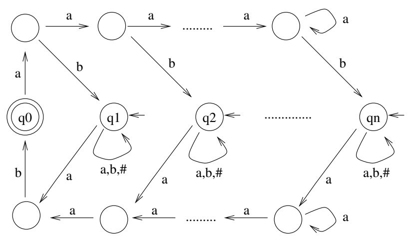


Figure 2.4: The Automaton from Figure 2.3 over a Constant Alphabet

Proof Let $i_1, \ldots, i_k \in \{1, \ldots, n\}$ be as in (ii). A successful run of $\mathcal{A}_n$ on α moves from q_{i_1} to q_0 and then to q_{i_2} the first time $i_1 i_2$ occurs. Then it waits for the next occurrence of $i_2 i_3$ and moves from q_{i_2} to q_0 and then to q_{i_3} and so on. The resulting run is infinite and accepting because it infinitely often visits q_0.

Now suppose that property (ii) is not satisfied. Let $i_1 i'_1, \ldots i_m i'_m \in \{1, \ldots, n\}^2$ be the tuples that appear infinitely often in α and let $x \in \mathbb{N}$ be the index such that from this point on only these tuples appear in α. Formally this means if $y > x$ and $\alpha(y)\alpha(y+1) \in \{1, \ldots, n\}^2$, then $\alpha(y)\alpha(y+1) \in \{i_1 i'_1, \ldots, i_m i'_m\}$.

Assume by contradiction that $\alpha \in L_n$. Let σ be an accepting run of $\mathcal{A}_n$ on α. Choose $y_1 > x$ in such a way that $\sigma(y_1) = q_i \neq q_0$ and $y_2 > y_1$ such that $\sigma(y_2) = q_0$. Since (ii) does not hold, there exists no $y > y_2$ with $\sigma(y) = q_i$. If we repeat this n times, we obtain an index z such that $\sigma(y) \neq q_j$ for all $y > z$ and for all $j \in \{1, \ldots, n\}$. Since q_0 can not be the only state that is visited from some point on, we have a contradiction. Therefore α is not in L_n. $\square$

With this characterization of L_n we can prove the following lemma.

Lemma 2.5 *Let $n \in \mathbb{N}$. The complement $\overline{L_n}$ of L_n can not be recognized by a nondeterministic Streett automaton with less than $n!$ states.*

Proof Let $\mathcal{A}'_n$ be a Streett automaton with $L'_n := L(\mathcal{A}'_n) = \overline{L_n}$ and let $(i_1 \ldots i_n)$, $(j_1 \ldots j_n)$ be different permutations of $(1 \ldots n)$. Define $\alpha = (i_1 \ldots i_n \#)^\omega$ and $\beta = (j_1 \ldots j_n \#)^\omega$. From lemma 2.4 follows $\alpha, \beta \notin L'_n$. Thus we have two successful runs $r_\alpha, r_\beta \in Q'^{\omega}_n$ of $\mathcal{A}'_n$ on α, β. Define $R = In(r_\alpha)$ and $S = In(r_\beta)$. If we can show $R \cap S = \emptyset$, then we are done because there are $n!$ permutations of $(1 \ldots n)$.

Assume $R \cap S \neq \emptyset$. Under this assumption we can construct $\gamma \in \Sigma_n^\omega$ with $\gamma \in L_n \cap L'_n$. This is a contradiction, since L'_n is the complement of L_n.

Let $q \in R \cap S$ and let u be a prefix of α with $r_\alpha(|u|) = q$ such that we can choose $v \in \Sigma_n^*$ with the following properties.

- The word uv is a prefix of α.
- The word $i_1 \ldots i_n$ is an infix of v.
- $r_\alpha(|uv|) = q$.
- The segment $r_\alpha[|u|, |uv|]$ contains exactly the states from R.

Let u' be a prefix of β with $r_\beta(|u'|) = q$ such that we can choose $w \in \Sigma_n^*$ with the following properties.

- The word $u'w$ is a prefix of β.
- The word $j_1 \dots j_n$ is an infix of w.
- $r_\beta(|u'w|) = q$.
- The segment $r_\beta[|u'|, |u'w|]$ contains exactly the states from S.

We define $\gamma = u(vw)^\omega$.

From the definition of γ one can see that there is a run r_γ of $\mathcal{A}'_n$ on γ with infinity set $R \cup S$. Because R and S satisfy the Streett condition, $R \cup S$ also does and we have $\gamma \in L'_n$.

To show $\gamma \in L_n$ we first note that, if k is the lowest index with $i_k \neq j_k$, then there exist $l, m > k$ with $j_k = i_l$ and $i_k = j_m$. By the choice of the words v and w one can see that γ contains infinitely often the segments $i_1 \dots i_n$ and $j_1 \dots j_n$. Thus γ also contains the segments $i_k i_{k+1}, \dots, i_{l-1} i_l, j_k j_{k+1}, \dots, j_{m-1} j_m$. Now, using lemma 2.4, we can conclude $\gamma \in L_n$. $\square$

With this lemma it is simple to prove the following theorem.

Theorem 2.6 *There is no conversion of Büchi automata with $\mathcal{O}(n)$ states into deterministic Rabin automata with $2^{\mathcal{O}(n)}$ states.*

Proof Let $n \in \mathbb{N}$. Assume there exists a deterministic Rabin automaton with $2^{\mathcal{O}(n)}$ states recognizing L_n. The dual of this automaton would be a deterministic Streett automaton with $2^{\mathcal{O}(n)}$ states recognizing $\overline{L_n}$. This is a contradiction to lemma 2.5. $\square$

This example demonstrates the optimality of Safra's construction for Rabin automata. In the following we will see that the example can not be used to show the same lower bound for Muller automata. For that aim we construct Muller automata $\mathcal{M}_n$, $n \in \mathbb{N}$, with $\mathcal{O}(n^2)$ states recognizing the language L_n.

For $n \in \mathbb{N}$ define the Muller automaton $\mathcal{M}_n = (Q'_n, \Sigma_n, q'_0, \delta'_n, \mathcal{F}_n)$ by

- $Q'_n = \Sigma_n \times \Sigma_n$,
- $q'_0 = (\#, \#)$,
- $\delta'_n((i, j), a) = (j, a)$,
- $F \in \mathcal{F}_n$ iff there exist $i_1, \dots, i_k \in \{1, \dots, n\}$ such that $(i_1, i_2), \dots, (i_{k-1}, i_k), (i_k, i_1) \in F$.

The automaton just collects all pairs of letters occurring in the input word and then decides, using the Muller acceptance condition, if the property from lemma 2.4 is satisfied. Thus we have $L(\mathcal{M}_n) = L_n$.

It is unknown if Safra's construction is of optimal complexity for the transformation of nondeterministic Büchi automata into deterministic Muller automata. The known lower bound for this transformation is $2^{\mathcal{O}(n)}$ ([SV89]).

But we can use the example from above to solve the question of the complexity for the transformation of nondeterministic Büchi automata into deterministic Büchi automata for languages that are deterministic Büchi.

For $n \in \mathbb{N}$ the Muller automaton $\mathcal{M}_n$ has the property that every accepting cycle is superset closed. This means, if $F \subset Q'_n$ is in $\mathcal{F}_n$, then every subset of Q'_n containing F also is in $\mathcal{F}_n$. From a theorem of Landweber follows that $L(\mathcal{M}_n)$ is deterministic Büchi. Since Büchi automata are special kinds of Streett automata, we can conclude with lemma 2.5 that the lower bound for a transformation from nondeterministic Büchi automata into deterministic Büchi automata is $2^{\mathcal{O}(n \cdot \log n)}$.

Furthermore it is known that every deterministic Rabin automaton recognizing a language that is deterministic Büchi can be transformed into an equivalent Büchi automaton with the same transition structure. Thus we can use Safra's construction for the transformation of nondeterministic Büchi automata into deterministic Büchi automata, meeting the lower bound.

2.3 The Breakpoint Construction

The breakpoint construction is a simple variant of the usual powerset construction, suggested by Miyano and Hayashi in [MH84]. A simplified version (which we present here) can also be found in [KV97]. The aim of the construction is to transform alternating Büchi automata into nondeterministic Büchi automata. Applying the complementation theorem 1.18 this also yields a transformation of alternating co-Büchi automata into universal co-Büchi automata.

We call the construction breakpoint construction because it identifies breakpoints in an accepting run of an alternating Büchi automaton. The basic idea is to mark all the vertices that have visited a final state before. If in a slice of the run all vertices are marked, then we have reached a breakpoint. In this situation we remove the marks and start again. The run is accepting if and only if we infinitely often pass such breakpoints. This is formalized as follows.

Lemma 2.7 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an alternating Büchi automaton with n states. There exists an equivalent nondeterministic Büchi automaton $\mathcal{A}' = (Q', \Sigma, q'_0, \delta', F')$ with $2^{\mathcal{O}(n)}$ states.*

Proof The states of $\mathcal{A}'$ are pairs of subsets of Q. The first component, after having read a prefix of length l, contains the states from level l in a possible run of $\mathcal{A}$. The second component contains the unmarked states. If it is empty, then a breakpoint is reached and the marking of the states starts again. Formally we define $\mathcal{A}'$ as follows.

- $Q' = 2^Q \times 2^Q$.
- $q_0 = (\{q_0\}, \emptyset)$.

- For $S, R \subseteq Q$ and $a \in \Sigma$ we define the transition function as follows.
  - If $R \neq \emptyset$, then

$$\begin{array}{rll}\delta'((S,R),a) = & \{(S', R' \setminus F) \mid & S' \text{ exactly satisfies } \bigwedge_{q\in S} \delta(q,a), \\ & & R' \subseteq S', \text{ and} \\ & & R' \text{ exactly satisfies } \bigwedge_{q\in R} \delta(q,a)\}.\end{array}$$

  - If $R = \emptyset$, then

$$\delta'((S,R),a) = \{(S', S' \setminus F) \mid \ S' \text{ exactly satisfies } \bigwedge_{q\in S} \delta(q,a)\}.$$

- $F' = 2^Q \times \{\emptyset\}$.

Obviously $\mathcal{A}'$ is a nondeterministic Büchi automaton with $2^{\mathcal{O}(n)}$ states.

$L(\mathcal{A}') \subseteq L(\mathcal{A})$: Let $\alpha \in L(\mathcal{A}')$ and let $\sigma \in Q'^{\omega}$ be an accepting run of $\mathcal{A}$ on α. Every state in σ consists of two components. With σ_1 (resp. σ_2) we denote the infinite sequence of the first (resp. second) components in σ. From the definition of δ' follows that we can construct a run G of $\mathcal{A}$ on α such that level l of G contains exactly the states from $\sigma_1(l)$. It remains to show that G is accepting. Let $i_0 < i_1 < i_2 < \cdots$ be the indices with $\sigma(i_j) \in F'$ for all $j \in \mathbb{N}$. Let π be a path through G. From the definition of δ' follows that for each $j \in \mathbb{N}$ the segment $\pi[i_j, i_{j+1})$ contains at least one state from F. Therefore π is accepting. Since every path through G is accepting, G is an accepting run.

$L(\mathcal{A}) \subseteq L(\mathcal{A}')$: Let $\alpha \in L(\mathcal{A})$ and let G be an accepting run of $\mathcal{A}$ on α. Let S be the states in level l of G and let S' be the states in level $l+1$ in G. From the definition of run follows that S' exactly satisfies $\bigwedge_{q\in S} \delta(q, \alpha(l))$. Hence there exists a run $\sigma \in Q'^{\omega}$ such that for each $l \in \mathbb{N}$ the first component of $\sigma(l)$ contains exactly the states from level l in G. In this run the second components are determined by δ'. Assume that σ is not accepting and let $i \in \mathbb{N}$ be the maximal number such that $\sigma(i) \in F'$. Since the second component never becomes empty again, there is an infinite sequence $q_{i+1}, q_{i+2}, q_{i+3}, \ldots$ of non-final states such that $((q_j, j), (q_{j+1}, j+1))$ is an edge in G for all $j \geq i+1$. Hence there is a rejecting path through G. This is a contradiction since G is an accepting run. Thus σ is an accepting run of $\mathcal{A}'$ on α. □

In the next subsection we give a lower bound for the determinization of alternating automata. The family of languages we use there can also be used to show the optimality of the breakpoint construction.

2.4 A Lower Bound for the Determinization of Alternating Automata

In the former sections we transformed nondeterministic automata into deterministic ones and alternating into nondeterministic ones. Both constructions where of optimal complexity. To transform alternating Büchi automata into deterministic ones, we can concatenate the breakpoint construction and Safra's construction. This yields an automaton with a number of states that is doubly exponential in the number of states of the given automaton:

$$n \overset{\text{breakpoint}}{\longrightarrow} 2^{\mathcal{O}(n)} \overset{\text{Safra}}{\longrightarrow} 2^{\mathcal{O}(2^{\mathcal{O}(n)}\log(2^{\mathcal{O}(n)}))} = 2^{\mathcal{O}(2^{\mathcal{O}(n)}\cdot\mathcal{O}(n))} = 2^{2^{\mathcal{O}(n)}}.$$

In this section we show that the complexity of this transformation is optimal. We give a family $\mathcal{A}_n$ of weak alternating automata with $\mathcal{O}(n)$ states, such that deterministic Muller automata recognizing the languages $L(\mathcal{A}_n)$ need at least 2^{2^n} states. Since the weak acceptance type is a special case of every other acceptance type, and every acceptance type is a special case of the Muller acceptance type, we then know that every transformation for alternating automata into deterministic ones is inherently doubly exponential. From the proof one will see that we do not need the infinity of the input and thus the same result holds for $*$-automata.

Lemma 2.8 *There exists a family $(L_n)_{n\geq 1}$ of languages such that, for every n, L_n can be recognized by a weak alternating automaton with $\mathcal{O}(n)$ states but not by a deterministic Muller automaton with less than 2^{2^n} states.*

Proof The intuitive definition of the language is as follows. In the first part of the word are different subsets $S_1, \ldots, S_m$ of $\{1, \ldots, n\}$, represented by n-bit vectors. Then a delimiter (a special character) appears and after that an n-bit vector representing a set S. The word is in the language L_n if and only if the set S also appeared in front of the delimiter, i.e., $S \in \{S_1, \ldots, S_m\}$. The idea is that any deterministic automaton has to remember all the sets that appear in front of the delimiter and therefore needs 2^{2^n} states.

In the formal definition of the language we do not demand that in the prefix up to the delimiter only correctly coded sets appear. We just demand that at least one correctly coded set appears that equals the set behind the delimiter.

Let $\Sigma = \{0, 1, 2, \#\}$ and $\alpha \in \Sigma^\omega$. For $n \geq 1$ the language L_n is defined by

$$\alpha \in L_n \text{ iff } \alpha = v_1 2u2v_2\#u\beta \text{ with } v_1, v_2 \in \{0,1,2\}^*, u \in \{0,1\}^n, \text{ and } \beta \in \Sigma^\omega.$$

We first show that L_n can not be recognized by a deterministic Muller automaton with less than 2^{2^n} states.

Let $n \geq 1$. Assume by contradiction that the deterministic Muller automaton $\mathcal{M}$ recognizes L_n and has less than 2^{2^n} states. Then there exist $u_1, \ldots, u_k, v_1, \ldots, v_l \in \{0,1\}^n$ with $\{u_1, \ldots, u_k\} \neq \{v_1, \ldots, v_l\}$, such that $\mathcal{M}$ is, after having read the word $2u_1 2 \cdots 2u_k 2$, in the same state as after having read $2v_1 2 \cdots 2v_l 2$. Without loss of generality we can assume that there exists an $i \in \{1, \ldots, k\}$ such that $u_i \notin \{v_1, \ldots, v_l\}$. If we choose $\beta = 0^\omega$, then the words $\alpha_1 = 2u_1 2 \cdots 2u_k 2\#u_i\beta$ and $\alpha_2 = 2v_1 2 \cdots 2v_l 2\#u_i\beta$ have the same run in $\mathcal{M}$, but $\alpha_1 \in L_n$ and $\alpha_2 \notin L_n$. Therefore, contradicting our assumption, $\mathcal{M}$ does not recognize L_n.

It remains to show that L_n can be recognized by an alternating weak automaton with $\mathcal{O}(n)$ states. Intuitively this automaton guesses one set in front of the delimiter, then spawns of a new path in its run for every bit (distinguishing between 0's and 1's),

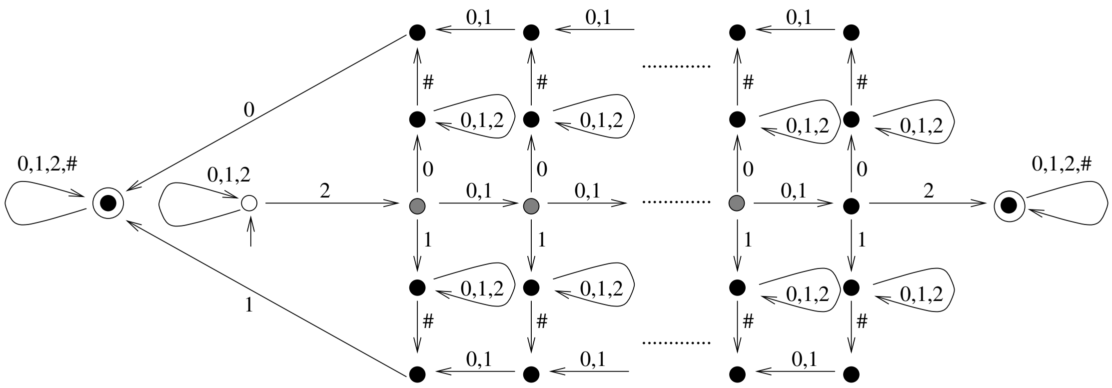


Figure 2.5: An equivalent DM automaton needs at least 2^{2^n} states.

then waits for the delimiter, and then tests if the following set is the same as the one guessed before.

It is difficult to visualize the transition structure of an alternating automaton, because there are nondeterministic and universal branches. But in this case the formulas in the definition of the transition function do not contain both, $\wedge$ and $\vee$. In figure 2.5 the black states only have one outgoing transition for every letter. If a gray state has more than one outgoing transition for one letter, then they are concatenated with $\wedge$. If a white state has more than one outgoing transition for one letter, then they are concatenated with $\vee$. If, for a state and a letter no successor is specified, then the corresponding formula in the transition function is `false`.

Formally the automaton $\mathcal{A}_n = (Q_n, \Sigma, q_{\mathrm{I}}, \delta_n, \{q_{\mathrm{F}}, q_{\mathrm{true}}\})$ is defined as follows.

- $Q_n = \{q_{\mathrm{I}}, q_1, \ldots, q_n, q_1^0, \ldots, q_n^0, q_1^1, \ldots, q_n^1, q_1^{0,\#}, \ldots, q_n^{0,\#}, q_1^{1,\#}, \ldots, q_n^{1,\#}, q_{\mathrm{true}}, q_{\mathrm{F}}\}$. We identify these states with the states from figure 2.5. The leftmost state is q_{F}. The initial state is q_{I}. The rightmost state is q_{true}. The middle row contains the states $q_1, \ldots, q_n$ from the left to the right. The row over the middle one contains the states $q_1^0, \ldots, q_n^0$ from the left to the right. The upper row contains the states $q_1^{0,\#}, \ldots, q_n^{0,\#}$ from the left to the right. The row below the middle one contains the states $q_1^1, \ldots, q_n^1$ from the left to the right. The lowest row contains the states $q_1^{1,\#}, \ldots, q_n^{1,\#}$ from the left to the right.

- The definition of the transition function can be seen in figure 2.5. To avoid misunderstandings we define the transition function for the white and the gray states.
  - $\delta(q_{\mathrm{I}}, 0) = \delta(q_{\mathrm{I}}, 1) = q_{\mathrm{I}}$.
    $\delta(q_{\mathrm{I}}, 2) = q_{\mathrm{I}} \vee q_1$.
  - For $i \in \{1, \ldots, n-1\}$:
    $\delta(q_i, 0) = q_{i+1} \wedge q_i^0$,
    $\delta(q_i, 1) = q_{i+1} \wedge q_i^1$,
    $\delta(q_i, 2) = \delta(q_i, \#) = \mathtt{false}$.

This automaton is weak. As partition of the state set one can take $Q_1 = \{q_{\mathrm{F}}, q_{\mathrm{true}}\}$ and $Q_2 = Q \setminus Q_1$. The number of states is $|Q| = 5n + 3$ and therefore in $\mathcal{O}(n)$.

One can easily verify $L_n = L(\mathcal{A})$. This completes the proof of the theorem. $\square$

We can use the same family of languages to show with a similar argument that the breakpoint construction is optimal.

Chapter 3

Transforming Acceptance Types

This chapter describes different constructions to transform automata of a certain acceptance type into equivalent automata of another acceptance type.

3.1 Adjustment of Acceptance Types

From the definition of the different acceptance types one can see that some of them are simply special cases of others. An acceptance type A is a special case of an acceptance type B if we can transform any given automaton with acceptance type A into an equivalent automaton with acceptance type B and the same transition structure as the given automaton.

If an acceptance type A is a special case of the acceptance type B, then we denote this by $B \supseteq A$. This leads to the following picture.

$$\begin{array}{ccccccccc} & & \text{Rabin} & & & & \text{Büchi} & & \\ \text{Muller} & \supseteq & & \supseteq & \text{Parity} & \supseteq & & \supseteq & \text{Weak} \\ & & \text{Streett} & & & & \text{Co-Büchi} & & \end{array}$$

All of these inclusions directly follow from the definitions of the different acceptance types, except the inclusion Streett $\supseteq$ Parity. In the following we explain this transformation in more detail.

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be a parity automaton. As mentioned in subsection 1.3 we can interpret the parity condition as a set of pairs $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ with $E_1 \subseteq F_1 \subseteq E_2 \ldots \subseteq E_r \subseteq F_r$. A path σ through a run of $\mathcal{A}$ is accepting if and only if there exists an $i \in \{1, \ldots, r\}$ such that $In(\sigma) \cap E_i = \emptyset$ and $In(\sigma) \cap F_i \neq \emptyset$. Since the E's and F's form a chain, we can also say: "If we infinitely often touch E_i then we also have to touch infinitely often F_{i-1}". Formally this means $In(\sigma) \cap E_i \neq \emptyset$ implies $In(\sigma) \cap F_{i-1} \neq \emptyset$ for all $i \in \{1, \ldots, r+1\}$, where we define $F_0 = \emptyset$ and $E_{r+1} = Q$. This implication can be transformed into the disjunction $In(\sigma) \cap E_i = \emptyset$ or $In(\sigma) \cap F_{i-1} \neq \emptyset$ for all $i \in \{1, \ldots, r+1\}$.

Now we can define a Streett automaton $\mathcal{A}'$, equivalent to $\mathcal{A}$, as follows. $\mathcal{A}' =$

$(Q, \Sigma, q_0, \delta, \Omega')$ with $\Omega' = \{(E'_1, F'_1), \ldots, (E'_{r+1}, F'_{r+1})\}$ where

$$\begin{array}{lcl} E'_1 & = & \emptyset, \\ E'_i & = & F_{i-1} \text{ for } i \in \{2, \ldots, r+1\}, \\ F'_i & = & E_i \text{ for } i \in \{1, \ldots, r\}, \\ F'_{r+1} & = & Q. \end{array}$$

3.2 Later Appearance Records

The aim of this section is to transform Muller, Rabin, and Streett automata into parity automata. In connection with the relations stated in section 3.1, we then can transform every Muller, Rabin, Streett, and parity automaton into an equivalent automaton with any of these acceptance types.

The main idea for the transformations introduced in this section is based on a data structure called later appearance record (LAR). Here we use two forms of LAR, namely state appearance records and index appearance records. The state appearance records serve to transform Muller automata into parity automata and the index appearance records serve to transform Streett automata into parity automata. Using the complementation theorem we are also able to transform Rabin automata into parity automata.

Another essential fact about these transformations is that they do not affect the mode of the transition function, i.e., the transition function of the resulting automaton has the same mode as the transition function of the original automaton.

3.2.1 State Appearance Records

The state appearance record (SAR) is a kind of memory to save information about a state sequence over a finite set of states Q. An SAR consists of two components. The first component is a permutation of the states in Q and the second component is a pointer to one of the states in this permutation. (Büchi called the permutation order vector and the pointer hit. Thus he talked of order vectors with hit instead of state appearance records.) Without loss of generality we can assume that $Q = \{1, \ldots, n\}$. An SAR is of the form $([i_1 \ldots i_n], h)$, where $[i_1 \ldots i_n]$ is a permutation of $\{1, \ldots, n\}$ and $h \in \{1, \ldots, n\}$. Let us give an example illustrating the usage of these state appearance records.

Example Let $Q = \{1, 2, 3, 4\}$. The state sequence

$$\sigma = 1 \to 2 \to 3 \to 2 \to 1 \to 3 \to 4 \to 3 \to 4 \to \cdots$$

induces the following sequence of SAR's.

$$\begin{array}{c} ([4321], 4) \to ([4312], 3) \to ([4123], 2) \to ([4132], 3) \to ([4321], 2) \\ \to ([4213], 2) \to ([2134], 1) \to ([2143], 3) \to ([2134], 3) \to \cdots \end{array}$$

The first SAR is an arbitrary SAR with 1 at the end of the permutation. The next state in σ is 2. Hence state 2 is moved to the end of the permutation and the pointer is set to 3, the index where 2 stood before. The next state is 3. Thus we move 3 to the end of the permutation. Since 3 was the second state in the former permutation, we set the index in the new SAR to 2. This procedure is repeated for every state in the sequence.

As we will see later this structure suffices to transform a Muller automaton into a parity automaton. The idea is to collect the states that are visited infinitely often at the right side of the permutation. If we extend the sequence σ from the example by repeating the states 3 and 4, then these two states will always be at the last two positions of the permutation. The lowest value of the pointer that is taken infinitely often will be 3. Thus we can conclude that the states in the permutation from position 3 on to the right are exactly those forming the infinity set of the original state sequence. This idea is formalized in the following.

First we define a mapping which we use to transform a state sequence into the corresponding sequence of SAR's. Since, in this subsection, we work with natural numbers as states, we will assume in the following that $Q = \{1, \ldots, n\}$ and $q_0 = 1$.

The set of permutations of $\{1, \ldots, n\}$ is

$$\mathrm{Perm}(\{1, \ldots, n\}) = \{[i_1 \ldots i_n] \in \{1, \ldots, n\}^n \mid \forall j, k \in \{1, \ldots, n\}(j \neq k \to i_j \neq i_k)\}.$$

The set of SAR's over Q is defined as $Q^{\mathrm{SAR}} = \mathrm{Perm}(Q) \times \{1, \ldots, n\}$. Let $([i_1 \ldots i_n], h) \in Q^{\mathrm{SAR}}$, $q \in Q$, and let k be the index with $i_k = q$. The function $\phi^{\mathrm{SAR}} : Q^{\mathrm{SAR}} \times Q \to Q^{\mathrm{SAR}}$ is defined by

$$\phi^{\mathrm{SAR}}(([i_1 \ldots i_n], h), q) = ([i'_1 \ldots i'_n], h'),$$

where

$$i'_j = \begin{cases} i_j & \text{if} \quad j < k, \\ i_{j+1} & \text{if} \quad k \leq j < n, \\ i_k & \text{if} \quad j = n \end{cases}$$

and

$$h' = k.$$

With this function we can formally define how to transform a state sequence into the corresponding sequence of SAR's.

Definition 3.1 Let $\sigma \in q_0 Q^\omega$. The corresponding sequence σ^{SAR} of SAR's is defined inductively by

$$\begin{array}{rcl} \sigma^{\mathrm{SAR}}(0) & = & ([n \ldots 1], n) \text{ and} \\ \sigma^{\mathrm{SAR}}(i) & = & \phi^{\mathrm{SAR}}(\sigma^{\mathrm{SAR}}(i-1), \sigma(i)) \text{ for all } i \geq 1. \end{array}$$

If we project every SAR in σ^{SAR} to the last state of the permutation, then we obviously get σ again. To show that the states that are finally collected at the end of the permutation are exactly those forming the infinity set of σ we need the following terminology.

Definition 3.2 Let $(\pi, h) \in Q^{\mathrm{SAR}}$ and $F \subseteq Q$ with $|F| = m \geq 1$. We say (π, h) *displays* F if and only if $\{\pi(n-m+1), \ldots, \pi(n)\} = F$. We say (π, h) *separates* F if and only if (π, h) displays F and $h = n - m + 1$. Let $\sigma \in q_0 Q^\omega$. The sequence σ^{SAR} of SAR's is said to display (or separate) F if and only if there exists an $i \in \mathbb{N}$ such that $\sigma^{\mathrm{SAR}}(i)$ displays (or separates) F. Furthermore σ^{SAR} is said to display (or separate) F infinitely often if and only if there are infinitely many $i \in \mathbb{N}$ such that $\sigma^{\mathrm{SAR}}(i)$ displays (or separates) F.

Note that every $(\pi, h) \in Q^{\mathrm{SAR}}$ displays more than one set but separates exactly one set.

Example. The SAR $([4213], 3)$ displays the sets $\{1, 2, 3, 4\}$, $\{1, 2, 3\}$, $\{1, 3\}$, $\{3\}$ and separates $\{1, 3\}$.

With the following two lemmas we will give a characterization of σ^{SAR} that suffices to transform a Muller condition over Q into an equivalent parity condition over the SAR's as state set.

Lemma 3.3 *Let $\sigma \in q_0 Q^\omega$ and $F \subseteq Q$, $F \neq \emptyset$.*

(1) *If $k, l \in \mathbb{N}$ with $k \leq l$, then σ^{SAR} displays $\{\sigma(k), \ldots, \sigma(l)\}$.*

(2) *Let $l \in \mathbb{N}$. If $\sigma^{\mathrm{SAR}}(l)$ displays F and if $\sigma(l+1) \in F$, then $\sigma^{\mathrm{SAR}}(l+1)$ displays F and separates a subset of F.*

(3) *If σ^{SAR} infinitely often separates F, then F is a subset of $In(\sigma)$.*

(4) *$In(\sigma)$ is separated infinitely often by σ^{SAR}.*

Proof (1) and (2) directly follow from the definition of σ^{SAR} and ϕ^{SAR}.

(3): Choose $k \leq l$ such that $\sigma(i) \in In(\sigma)$ for all $i \geq k$ and $\{\sigma(k), \ldots, \sigma(l)\} = In(\sigma)$. By (1) $\sigma^{\mathrm{SAR}}(l)$ displays $In(\sigma)$ and by repeatedly applying (2) we get $\sigma^{\mathrm{SAR}}(i)$ displays $In(\sigma)$ and separates a subset of $In(\sigma)$ for all $i > l$.

(4): Let $m = |In(\sigma)|$. Let l be as in the proof of (3) and let $l_1 \in \mathbb{N}$ with $l_1 \geq l$. There exists an $l_2 \geq l_1$ such that $\{\sigma(l_1), \ldots, \sigma(l_2)\} = F'$ for an F' with $F' \subset In(\sigma)$ and $|F'| = |F| - 1$. We can choose l_2 in such a way that $\sigma(l_2 + 1) \in In(\sigma) \setminus F'$. Let $q = \sigma(l_2 + 1)$ and $m = |In(\sigma)|$. Since $l_2 > l$ we know that $\sigma^{\mathrm{SAR}}(l_2)$ displays $In(\sigma)$ and from (1) we can derive that $\sigma^{\mathrm{SAR}}(l_2)$ displays F'. Thus q is on position $n - m + 1$ in the permutation of $\sigma(l_2)$. But this means that the pointer in $\sigma^{\mathrm{SAR}}(l_2 + 1)$ has the value $n - m + 1$ and hence $\sigma^{\mathrm{SAR}}(l_2 + 1)$ separates $In(\sigma)$. Since l_1 was an arbitrary number bigger than l, we can conclude that $In(\sigma)$ is separated infinitely often by σ^{SAR}. $\square$

Lemma 3.4 *Let $\sigma \in q_0 Q^\omega$ and $F \subseteq Q$.*

$$In(\sigma) = F \quad \Leftrightarrow \quad \begin{array}{l} \text{(i) } \sigma^{\mathrm{SAR}} \text{ infinitely often separates } F \text{ and} \\ \text{(ii) if } \sigma^{\mathrm{SAR}} \text{ infinitely often separates } F' \neq F, \text{ then } |F'| < |F|. \end{array}$$

Proof "$\Rightarrow$": Property (i) follows from lemma 3.3 (4) and property (ii) follows from lemma 3.3 (3).
"$\Leftarrow$": Assume $In(\sigma) \neq F$. Since σ^{SAR} infinitely often separates $In(\sigma)$ by lemma 3.3 (4), we have $|In(\sigma)| < |F|$. This is a contradiction to lemma 3.3 (3) because F is separated infinitely often by σ^{SAR}. Thus we have $In(\sigma) = F$. $\square$

This relation between σ and σ^{SAR} allows us to define a parity acceptance condition such that the resulting parity automaton is equivalent to the original Muller automaton.

Theorem 3.5 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a deterministic Muller automaton with $Q = \{1, \ldots, n\}$ and $q_0 = 1$. There exists a deterministic parity automaton $\mathcal{A}^{\mathrm{SAR}}$ with $n \cdot n!$ states and $L(\mathcal{A}) = L(\mathcal{A}^{\mathrm{SAR}})$.*

Proof The parity automaton $\mathcal{A}^{\mathrm{SAR}} = (Q^{\mathrm{SAR}}, \Sigma, q_0^{\mathrm{SAR}}, \delta^{\mathrm{SAR}}, \Omega_{\mathcal{F}}^{\mathrm{SAR}})$ is defined as follows.

- The new state space Q^{SAR} was defined above.
- $q_0^{\mathrm{SAR}} = ([n \dots 1], n)$.
- For $([i_1 \dots i_n], h) \in Q^{\mathrm{SAR}}$ and $a \in \Sigma$ let

$$\delta^{\mathrm{SAR}}(([i_1 \dots i_n], h), a) = \phi^{\mathrm{SAR}}(([i_1 \dots i_n], h), \delta(i_n, a)).$$

  This means we take the a-successor of the last state in the permutation and then apply ϕ^{SAR}.

- The new acceptance condition $\Omega_{\mathcal{F}}^{\mathrm{SAR}} = \{(E_1, F_1), \dots, (E_n, F_n)\}$ is defined by

$$\begin{aligned} F_i &= \{q \in Q^{\mathrm{SAR}} \mid q \text{ separates an } F \subseteq Q \text{ with } |F| \geq n - i + 1\}, \\ E_i &= F_i \setminus \{q \in Q^{\mathrm{SAR}} \mid q \text{ separates an } F \in \mathcal{F} \text{ with } |F| = n - i + 1\}. \end{aligned}$$

First note that the sets in $\Omega_{\mathcal{F}}^{\mathrm{SAR}}$ form a chain $E_1 \subseteq F_1 \subseteq E_2 \subseteq \cdots \subseteq E_n \subseteq F_n$ and thus $\mathcal{A}^{\mathrm{SAR}}$ in fact is a parity automaton. Obviously $\mathcal{A}^{\mathrm{SAR}}$ has $n \cdot n!$ states.

Since $\mathcal{A}$ and $\mathcal{A}^{\mathrm{SAR}}$ are deterministic automata, we can identify the runs of these automata with the infinite state sequences.

Let $\alpha \in \Sigma^\omega$ and $\sigma \in Q^\omega$ be the run of $\mathcal{A}$ on α. From the definition of the initial state and the transition function of $\mathcal{A}^{\mathrm{SAR}}$ we can see that σ^{SAR} is the run of $\mathcal{A}^{\mathrm{SAR}}$ on α. Thus, to complete the proof, we have to show that $In(\sigma)$ satisfies $\mathcal{F}$ if and only if $In(\sigma^{\mathrm{SAR}})$ satisfies $\Omega_{\mathcal{F}}^{\mathrm{SAR}}$.

Assume $In(\sigma)$ satisfies $\mathcal{F}$, i.e., $In(\sigma) \in \mathcal{F}$. Choose i such that $|In(\sigma)| = n - i + 1$. From lemma 3.4 follows that σ^{SAR} infinitely often separates $In(\sigma)$. Hence we have $In(\sigma) \cap F_i \neq \emptyset$. For all $F \neq In(\sigma)$ that are infinitely often separated by σ^{SAR} one has $|F| < |In(\sigma)|$ (lemma 3.4). Therefore $F_j \cap In(\sigma^{\mathrm{SAR}}) = \emptyset$ and $E_j \cap In(\sigma^{\mathrm{SAR}}) = \emptyset$ for all $j < i$ and, since $In(\sigma) \in \mathcal{F}$, $E_i \cap In(\sigma^{\mathrm{SAR}}) = \emptyset$. This means $In(\sigma^{\mathrm{SAR}})$ satisfies $\Omega_{\mathcal{F}}^{\mathrm{SAR}}$.

Now assume $In(\sigma^{\mathrm{SAR}})$ satisfies $\Omega_{\mathcal{F}}^{\mathrm{SAR}}$, i.e., there exists an $i \in \{1, \dots, n\}$ such that $In(\sigma^{\mathrm{SAR}}) \cap F_i \neq \emptyset$ and $In(\sigma^{\mathrm{SAR}}) \cap E_i = \emptyset$. Thus there exists an $F \in \mathcal{F}$ with $|F| = n - i + 1$ such that σ^{SAR} infinitely often separates F. Since $In(\sigma^{\mathrm{SAR}}) \cap E_i = \emptyset$, the intersection of $In(\sigma^{\mathrm{SAR}})$ and F_j is empty for all j with $j < i$. Thus for every $F' \neq F$ that is separated infinitely often by σ^{SAR} one has $|F'| \leq |F|$. Since only subsets of $In(\sigma)$ are separated infinitely often by σ^{SAR} (lemma 3.3 (3)), we can conclude that $F = In(\sigma)$ and thus $In(\sigma) \in \mathcal{F}$. $\square$

We proved that the SAR construction transforms a deterministic Muller automaton into a deterministic parity automaton. Now we will explain that this construction can also be used to transform any Muller automaton into an equivalent parity automaton with the same mode of the transition function. For that aim we extend the mapping ϕ^{SAR} to a mapping $\phi_+^{\mathrm{SAR}} : Q^{\mathrm{SAR}} \times \mathcal{B}^+(Q^{\mathrm{SAR}}) \to \mathcal{B}^+(Q)$. Let $q \in Q$, $q' \in Q^{\mathrm{SAR}}$ and $\theta_1, \theta_2 \in \mathcal{B}^+(Q)$.

$$\begin{aligned}
\phi_+^{\mathrm{SAR}}(q', \mathtt{true}) &= \mathtt{true}, \\
\phi_+^{\mathrm{SAR}}(q', \mathtt{false}) &= \mathtt{false}, \\
\phi_+^{\mathrm{SAR}}(q', q) &= \phi^{\mathrm{SAR}}(q', q), \\
\phi_+^{\mathrm{SAR}}(q', \neg q) &= \neg\phi^{\mathrm{SAR}}(q', q), \\
\phi_+^{\mathrm{SAR}}(q', \theta_1 \wedge \theta_2) &= \phi_+^{\mathrm{SAR}}(q', \theta_1) \wedge \phi_+^{\mathrm{SAR}}(q', \theta_2) \text{ and} \\
\phi_+^{\mathrm{SAR}}(q', \theta_1 \vee \theta_2) &= \phi_+^{\mathrm{SAR}}(q', \theta_1) \vee \phi_+^{\mathrm{SAR}}(q', \theta_2).
\end{aligned}$$

Now we can extend the construction in the proof of theorem 3.5 to general Muller automata by replacing ϕ^{SAR} with ϕ_+^{SAR}. It is not difficult to see that the resulting parity automaton is equivalent to the original Muller automaton.

3.2.2 Index Appearance Records

The transformation from Muller into parity automata can also be used to transform Rabin and Streett automata into parity automata, since Rabin and Streett automata are special cases of Muller automata. In this subsection we present a different construction. The complexity of this construction is better than the complexity of the SAR construction of the previous subsection if the number of pairs in the acceptance condition of the Rabin or Streett automaton is smaller than the number of states. If there are more pairs than states, then the SAR construction is better.

The idea of the construction we present here is similar to the idea of the SAR construction. In a Muller automaton acceptance depends on the states that are visited infinitely often. In a Rabin or Streett automaton it suffices to know which E_i's and F_i's from the pairs of the acceptance condition are visited infinitely often. This is why we use a permutation of $\{1, \ldots, r\}$, if there are r pairs in the acceptance condition, instead of a permutation of the states as in the SAR construction. If we know which of the F_i's in a Streett automaton are visited infinitely often, then we know that all the corresponding E_i's must also be visited infinitely often. With this information we can decide if a word is accepted or not.

Let Q be a set of states and let $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$, where $E_i, F_i \subseteq Q$ for $i = 1, \ldots, r$. The set of index appearance records is defined as $Q_r^{\mathrm{IAR}} = Q \times \mathrm{Perm}(\{1, \ldots, r\}) \times \{1, \ldots, r+1\} \times \{1, \ldots, r+1\}$. As in the previous subsection we give an example how to work with these IAR's.

Example. Let $Q = \{1, \ldots, 8\}$ and $\Omega = \{(E_1, F_1), \ldots, (E_4, F_4)\}$ with $F_i = \{2i-1\}$ and $E_i = \{2i\}$ for $i = 1, \ldots, 4$. An IAR over Q and Ω is of the form $(q, [i_1 \ldots i_4], f, e)$, where $q \in Q$ is the component to simulate the original automaton, $[i_1 \ldots i_4]$ is a permutation of $\{1, \ldots, 4\}$, and f, e are pointers from $\{1, \ldots, 5\}$ (similar to the h in the SAR's). In the permutation of $\{1, \ldots, 4\}$ the indices of the E_i's that are visited infinitely often are collected at the right side. The pointer e indicates from which position the last index was moved to the end of the permutation. In every step the leftmost index i_j in the permutation with $q \in E_{i_j}$ is moved to the right. If there is no j with $q \in E_{i_j}$, then no index is moved and e is set to $r + 1 = 5$. The pointer f points to the leftmost position

j such that $q \in F_{i_j}$. If there is no such position, then f is set to $r+1=5$. Here is an example how to convert a sequence over Q into a sequence over Q_4^{IAR}.

$$\sigma = 1 \to 3 \to 5 \to 4 \to 2 \to 1 \to 4 \to 1 \to 4 \cdots$$

induces the following sequence of IAR's

$$\begin{gathered}(1,[1234],5,5) \to (3,[1234],2,5) \to (5,[1234],3,5) \to (4,[1342],5,2) \to (2,[3421],5,1) \\ \to (1,[3421],4,5) \to (4,[3412],5,3) \to (1,[3412],3,5) \to (4,[3412],5,4) \cdots.\end{gathered}$$

The first IAR is an arbitrary IAR with 1 in the first component. The second state in σ is 3. So the first component in the IAR is updated to 3. The state 3 is in F_2. Hence the pointer f (the third component) is set to the position where 2 stood in the former IAR. In this case this is also 2. None of the E_i's contains state 3. So the pointer e is set to 5 and the permutation stays the same. The first state from one of the E_i's occurring in σ is 4. This is the first time we change the permutation. Since 4 is in E_2, we move 2 to the end of the permutation. The pointer e is set to 2, because 2 stood on position 2 before. The pointer f is set to 5, because 4 is in none of the F_i's.

It turns out that, similar to the SAR construction, the indices of the E_i's that are visited infinitely often are collected at the right side of the permutation. If there are k such indices, then the lowest value of e that is taken infinitely often is $r-k+1$. If f takes a value lower than $r-k+1$ infinitely often, then there must be an F_i such that F_i is visited infinitely often but not so the corresponding E_i. In the other direction, if all values of f that are taken infinitely often are bigger than $r-k+1$, then for every F_i that is visited infinitely often also the corresponding E_i is visited infinitely often.

Now we formalize the idea from the example by means of a function $\phi_\Omega^{\mathrm{IAR}}$. Let Q be a set of states and $\Omega = \{(E_1,F_1),\ldots,(E_r,F_r)\}$, where $E_i, F_i \subseteq Q$ for $i=1,\ldots,r$. Let $(q,[i_1\ldots i_r],f,e) \in Q_r^{\mathrm{IAR}}$ and $q' \in Q$. Let $k = \min\{j \in \{1,\ldots,r\} \mid q' \in E_{i_j}\}$ if there exists a j such that $q' \in E_{i_j}$ and $k = r+1$ otherwise. Let $l = \min\{j \in \{1,\ldots,r\} \mid q' \in F_{i_j}\}$ if there exists a j such that $q' \in F_{i_j}$ and $l = r+1$ otherwise. The function $\phi_\Omega^{\mathrm{IAR}} : Q_r^{\mathrm{IAR}} \times Q \to Q_r^{\mathrm{IAR}}$ is defined by

$$\phi_\Omega^{\mathrm{IAR}}((q,[i_1\ldots i_r],f,e),q') = (q',[i'_1\ldots i'_r],f',e'),$$

where

$$i'_j = \begin{cases} i_j & \text{if} \quad j<k, \\ i_{j+1} & \text{if} \quad k \le j < r, \\ i_k & \text{if} \quad j = r, \end{cases}$$

and

$$e' = k, f' = l.$$

Now we can define how to transform a sequence over Q into a sequence over Q_r^{IAR}.

Definition 3.6 Let $\sigma \in Q^\omega$. The corresponding sequence $\sigma_\Omega^{\mathrm{IAR}}$ of IAR's is inductively defined by

$$\begin{aligned} \sigma_\Omega^{\mathrm{IAR}}(0) &= (\sigma(0),[1\ldots r],r+1,r+1), \\ \sigma_\Omega^{\mathrm{IAR}}(i) &= \phi_\Omega^{\mathrm{IAR}}(\sigma_\Omega^{\mathrm{IAR}}(i-1),\sigma(i)) \text{ for } i \ge 1. \end{aligned}$$

Similar to the previous subsection we need some terminology to prove some properties of the sequence $\sigma_\Omega^{\mathrm{IAR}}$.

Definition 3.7 Let $(q, \pi, f, e) \in Q_r^{\mathrm{IAR}}$ and $S \subseteq \{1, \dots, r\}$ with $|S| = m$. We say (q, π, f, e) *displays* S if and only if $\{\pi(r - m + 1), \dots, \pi(r)\} = S$.

Let $\sigma \in Q^\omega$. We are interested in the E_i's and F_i's that are visited infinitely often and thus define $Ind_E(\sigma) = \{j \in \{1, \dots, r\} \mid E_j \cap In(\sigma) \neq \emptyset\}$ and $Ind_F(\sigma) = \{j \in \{1, \dots, r\} \mid F_j \cap In(\sigma) \neq \emptyset\}$.

Lemma 3.8 *Let $\sigma \in Q^\omega$. For simplicity we use the notation $\sigma_\Omega^{\mathrm{IAR}}(i) = (q_i, \pi_i, f_i, e_i)$.*

(1) $\forall_i^\omega(\sigma_\Omega^{\mathrm{IAR}}(i)$ *displays* $Ind_E(\sigma))$.

(2) $|Ind_E(\sigma)| \leq k \Leftrightarrow \forall_i^\omega(e_i \geq r - k + 1)$.

(3) $Ind_F(\sigma) \subseteq Ind_E(\sigma) \Leftrightarrow \forall_i^\omega(f_i \geq r - |Ind_E(\sigma)| + 1)$.

Proof (1): For all $i \in \mathbb{N}$ let s_i be the biggest number such that $\pi_i(s_i) \notin Ind_E(\sigma)$. From this definition follows that $\sigma_\Omega^{\mathrm{IAR}}(i)$ displays $Ind_E(\sigma)$ if and only if $s_i = r - |Ind_E(\sigma)| + 1$. Choose $m \in \mathbb{N}$ such that $\{j \in \{1, \dots, r\} \mid \sigma(i) \in E_j\} \subseteq Ind_E(\sigma)$ for all $i \geq m$. The choice of m implies that $s_i \geq s_{i+1}$ for all $i \geq m$. If $s_i \neq r - |Ind_E(\sigma)| + 1$, then there must be an $s < s_i$ such that $\pi_i(s) \in Ind_E(\sigma)$. If we choose s as small as possible, then the next time $E_{\pi(s)}$ is visited $\pi(s)$ is moved to the end of the permutation and s_i decreases. This can be repeated until we reach an l such that $s_l = r - |Ind_E(\sigma)| + 1$. Since s_i can never increase again, $\sigma_\Omega^{\mathrm{IAR}}(i)$ always displays $Ind_E(\sigma)$ from this index onwards.

(2): First assume $Ind_E(\sigma) \leq k$. We know that there is an l such that $\sigma_\Omega^{\mathrm{IAR}}(i)$ displays $Ind_E(\sigma)$ for all $i \geq l$. If for some $i > l$ one has $e_i < r - k + 1$, then there must be an element not from $Ind_E(\sigma)$ at the end of the permutation, a contradiction.

Now assume $\forall_i^\omega(e_i \geq r - k + 1)$. This implies that there are at most k elements in the permutation that are moved infinitely often. Hence $Ind_E(\sigma)$ can not contain more than k elements.

(3): First assume $Ind_F(\sigma) \subseteq Ind_E(\sigma)$. Let l be the index such that $\sigma_\Omega^{\mathrm{IAR}}(i)$ displays $Ind_E(\sigma)$ for all $i \geq l$. Now let $m \geq l$ such that $\{j \in \{1, \dots, r\} \mid \sigma(i) \in F_j\} \subseteq Ind_F(\sigma)$ for all $i \geq m$. Since $Ind_F(\sigma) \subseteq Ind_E(\sigma)$ and $\sigma_\Omega^{\mathrm{IAR}}(i)$ displays $Ind_E(\sigma)$ for all $i \geq m$, we can conclude that $f_i \geq r - |Ind_E(\sigma)| + 1$ for all $i \geq m$.

The other direction also follows from the fact that $\sigma_\Omega^{\mathrm{IAR}}(i)$ continuously displays $Ind_E(\sigma)$ from some point on. $\square$

With these preparations we can now prove the main theorem of this subsection.

Theorem 3.9 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be a deterministic Streett automaton with $|Q| = n$ and $\Omega = \{(E_1, F_1), \dots, (E_r, F_r)\}$. There exists a deterministic parity automaton $\mathcal{A}^{\mathrm{IAR}}$ with $n \cdot (r+1)^2 \cdot r!$ states and $L(\mathcal{A}) = L(\mathcal{A}^{\mathrm{IAR}})$.*

Proof Define $\mathcal{A}^{\mathrm{IAR}} = (Q_r^{\mathrm{IAR}}, \Sigma, q_0^{\mathrm{IAR}}, \delta^{\mathrm{IAR}}, \Omega^{\mathrm{IAR}})$ as follows.

- Q_r^{IAR} was defined above.
- $q_0^{\mathrm{IAR}} = (q_0, [1 \dots r], r + 1, r + 1)$.

- For $(q, \pi, f, e) \in Q_r^{\mathrm{IAR}}$ and $a \in \Sigma$ let
$$\delta^{\mathrm{IAR}}((q, \pi, f, e), a) = \phi_\Omega^{\mathrm{IAR}}((q, \pi, f, e), \delta(q, a)).$$
- The acceptance condition $\Omega^{\mathrm{IAR}} = \{(E'_1, F'_1), \ldots, (E'_r, F'_r)\}$ is defined by
$$\begin{aligned} E'_1 &= \emptyset, \\ F'_1 &= \{(q, \pi, f, e) \mid e = 1\}, \\ E'_i &= F_{i-1} \cup \{(q, \pi, f, e) \mid e > i-1 \text{ and } f = i-1\}, \\ F'_i &= E_i \cup \{(q, \pi, f, e) \mid f \geq i \text{ and } e = i\} \end{aligned}$$
for $i = 2, \ldots, r+1$.

Obviously $\mathcal{A}^{\mathrm{IAR}}$ is a deterministic parity automaton with $n \cdot (r+1)^2 \cdot r!$ states. Again we identify a run of a deterministic automaton with the induced sequence of states.

For the correctness proof first note that if σ is the run of $\mathcal{A}$ on a word α, then $\sigma_\Omega^{\mathrm{IAR}}$ is the run of $\mathcal{A}^{\mathrm{IAR}}$ on α. As in lemma 3.8 we use the notation $\sigma_\Omega^{\mathrm{IAR}}(i) = (q_i, \pi_i, f_i, e_i)$.

Assume that $In(\sigma)$ satisfies Ω. Since $\mathcal{A}$ is a Streett automaton, this is equivalent to $Ind_F(\sigma) \subseteq Ind_E(\sigma)$. Let $k \in \{1, \ldots, r\}$ such that $|Ind_E(\sigma)| = r - k + 1$. From lemma 3.8 (3) follows that $f_i \geq k$ for almost all i. To show $In(\sigma_\Omega^{\mathrm{IAR}}) \cap F'_k \neq \emptyset$ we have to prove that $e_i = k$ for infinitely many i. This follows from lemma 3.8 (2). From lemma 3.8 (2) also follows that $e_i \geq k$ for almost all i. Now $\forall_i^\omega(f_i \geq k)$ and $\forall_i^\omega(e_i \geq k)$ imply $In(\sigma_\Omega^{\mathrm{IAR}}) \cap E_k = \emptyset$. Thus $In(\sigma_\Omega^{\mathrm{IAR}})$ satisfies Ω^{IAR}.

Now assume that $In(\sigma_\Omega^{\mathrm{IAR}})$ satisfies Ω^{IAR}. Then there exists a k such that $In(\sigma_\Omega^{\mathrm{IAR}}) \cap E'_k = \emptyset$ and $In(\sigma_\Omega^{\mathrm{IAR}}) \cap F'_k \neq \emptyset$. First we will show that $\forall_i^\omega(e_i \geq k)$. Assume by contradiction that there exist infinitely many i such that $e_i = l < k$. Then we can fix an m such that there are infinitely many i with $e_i = l$ and $f_i = m$. In the case $m \geq l$ we have $In(\sigma_\Omega^{\mathrm{IAR}}) \cap F'_l \neq \emptyset$. This is a contradiction, since $F'_l \subseteq E'_k$. In the case $m < l$ we have $In(\sigma_\Omega^{\mathrm{IAR}}) \cap E'_{m+1} \neq \emptyset$. This also is a contradiction, since $m + 1 \leq l < k$. Thus we know that $e_i \geq k$ for almost all i and from $In(\sigma_\Omega^{\mathrm{IAR}}) \cap F'_k \neq \emptyset$ we know that there are infinitely many i such that $e_i = k$. So we can conclude with lemma 3.8 (2) that $|Ind_E(\sigma)| = r - k + 1$. From $In(\sigma_\Omega^{\mathrm{IAR}}) \cap E'_k = \emptyset$ and $In(\sigma_\Omega^{\mathrm{IAR}}) \cap F'_k \neq \emptyset$ follows that $f_i \geq k$ for almost all i. Since $k = r - |Ind_E(\sigma)| + 1$, lemma 3.8 (3) implies $Ind_F(\sigma) \subseteq Ind_E(\sigma)$. Therefore $In(\sigma)$ satisfies Ω and we are done. □

As in the previous subsection the construction only works for deterministic automata. But analogue to the extension of ϕ^{SAR} we can extend $\phi_\Omega^{\mathrm{IAR}}$ and then modify the construction in such a way that it works for automata with any mode of transition function.

In table 3.1 the constructions for the transformations between the acceptance types Muller, Rabin, Streett, and parity are summarized.

3.2.3 Optimality of the LAR Constructions

In this section we will show that the LAR constructions are of optimal complexity. The basic idea of the proof is taken from [DJW97], where the optimality of the SAR as memory for winning strategies in Muller games was shown. Here we show the optimality of both, SAR and IAR, for the transformations of acceptance types presented in the former subsections. Note that our proof does not replace the proof from [DJW97] because it can only by applied to automata but not to games.

	Muller	Rabin	Streett	Parity
Muller		SAR	SAR	SAR
Rabin	adjustment of condition		IAR	IAR
Streett	adjustment of condition	IAR		IAR
Parity	adjustment of condition	adjustment of condition	adjustment of condition	

Table 3.1: Transforming the Acceptance Conditions of ω-Automata

Lemma 3.10 *There exists a family $(L_n)_{n\geq 2}$ of languages such that for every n the language L_n can be recognized by a deterministic Streett automaton with $\mathcal{O}(n)$ states and $\mathcal{O}(n)$ pairs but can not be recognized by a deterministic Rabin automaton with less than $n!$ states.*

Proof We define the languages L_n via deterministic Streett automata $\mathcal{A}_n$ over the alphabet $\{1, \ldots, n\}$. Later we will explain how we can adapt the proof for an alphabet of constant size. The transition structure of $\mathcal{A}_n$ is shown schematically in figure 3.1. Formally, for $n \geq 2$, we define the Streett automaton $\mathcal{A}_n = (Q_n, \Sigma_n, q_0, \delta_n, \Omega_n)$ as follows.

- $Q_n = \{-n, \ldots, -1, 1, \ldots, n\}$.
- $\Sigma_n = \{1, \ldots, n\}$.
- $q_0 = -1$.
- For $i, j \in \{1, \ldots, n\}$ let
  - $\delta_n(i, j) = -j$ and
  - $\delta_n(-i, j) = j$.
- $\Omega_n = \{(E_1, F_1), \ldots, (E_n, F_n)\}$ with $E_i = \{i\}$ and $F_i = \{-i\}$.

To characterize the words in L_n we use the following notation. For a word $\alpha \in \Sigma_n^\omega$ let $even(\alpha)$ be the set containing the letters that infinitely often occur on an even position in α and let $odd(\alpha)$ be the set containing the letters that infinitely often occur on an odd position in α. This means $even(\alpha) = In(\alpha(0)\alpha(2)\alpha(4)\cdots)$ and $odd(\alpha) = In(\alpha(1)\alpha(3)\alpha(5)\cdots)$. From the definition of $\mathcal{A}_n$ follows that a word $\alpha \in \Sigma_n^\omega$ is in L_n if and only if $odd(\alpha) \subseteq even(\alpha)$. As a consequence of this, for $\alpha \in \Sigma_n^\omega$ and $u \in \Sigma_n^*$ with $|u|$ even, the word $u\alpha$ is in L_n if and only if α is in L_n. Therefore, in a deterministic automaton recognizing L_n, every state that can be reached by reading a prefix of even length can be used as initial state without changing the accepted language.

We will prove by induction that every deterministic Rabin automaton recognizing L_n needs at least $n!$ states.

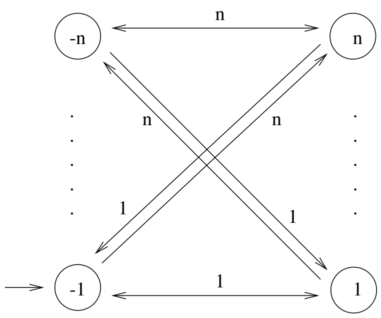


Figure 3.1: The Transition Structure of $\mathcal{A}_n$

We show the base case of the induction for $n = 2$. An automaton recognizing a nonempty proper subset of Σ_2^ω needs at least 2 states. Therefore the base case of the induction holds.

Now let $n > 2$ and let $\mathcal{B} = (Q, \Sigma_n, q_0, \delta, \Omega)$ be a deterministic Rabin automaton with $L(\mathcal{B}) = L_n$. Let Q_{even} be the states that can be reached from q_0 by reading a prefix of even length.

For every $i \in \{1, \ldots, n\}$ and every $q \in Q_{\text{even}}$ we construct a deterministic Rabin automaton $\mathcal{B}_i^q$ over $\Sigma_n \setminus \{i\}$ by removing all i-transitions from $\mathcal{B}$. Furthermore q is the initial state of $\mathcal{B}_i^q$. Since q can be reached in $\mathcal{B}$ after having read a prefix of even length, the language recognized by $\mathcal{B}_i^q$ is L_{n-1} (if $i \neq n$ then the names of the letters are different but the language essentially equals L_{n-1}). Thus, by the induction hypothesis, $\mathcal{B}_i^q$ has at least $(n-1)!$ states.

We can strengthen this statement as follows. In every $\mathcal{B}_i^q$ ($i \in \{1, \ldots, n\}$ and $q \in Q_{\text{even}}$) is a strongly connected component with at least $(n-1)!$ states. Just take a strongly connected component S in $\mathcal{B}_i^q$ such that there is no other strongly connected component reachable from S in $\mathcal{B}_i^q$. Let p be a state that is reachable from q in $\mathcal{B}_i^q$ by reading a prefix of even length. As we have seen above, we can use p as initial state in $\mathcal{B}_i^q$ without changing the accepted language. Therefore, by the induction hypothesis, S must contain at least $(n-1)!$ states.

Now, for $i \in \{1, \ldots, n\}$, we construct words $\alpha_i \in \Sigma_n^\omega$ with runs σ_i of $\mathcal{B}$ such that $|In(\sigma_i)| \geq (n-1)!$ and $In(\sigma_i) \cap In(\sigma_j) = \emptyset$ for $i \neq j$. Then we are done because $|Q'| \geq \sum_{i=1}^{n} |In(\sigma_i)| \geq n \cdot (n-1)! = n!$.

For $i \in \{1, \ldots, n\}$ construct the word α_i as follows. First take a $u_0 \in (\Sigma_n \setminus \{i\})^*$ such that u_0 has even length and contains every letter from $\Sigma_n \setminus \{i\}$ on an even and on an odd position. Furthermore $\mathcal{B}_i^{q_0}$ should visit at least $(n-1)!$ states while reading u_0. This is possible since $\mathcal{B}_i^{q_0}$ contains a strongly connected component with $\geq (n-1)!$ states. Let q_1 be the state reached by $\mathcal{B}_i^{q_0}$ after having read the word $u_0 ij$, where j is different from i. Then we choose a word $u_1 \in (\Sigma_n \setminus \{i\})^*$ with the same properties as u_0, using $\mathcal{B}_i^{q_1}$ instead of $\mathcal{B}_i^{q_0}$. This means u_1 has even length, contains every letter from $\Sigma_n \setminus \{i\}$ on an even and on an odd position, and $\mathcal{B}_i^{q_1}$ visits at least $(n-1)!$ states while reading u_1.

Repeating this procedure we get a word $\alpha_i = u_0 ij u_1 ij u_2 ij \cdots$. From the construction follows that $even(\alpha_i) = \{1, \ldots, n\} \setminus \{i\}$ and $odd(\alpha_i) = \{1, \ldots, n\}$ and therefore $\alpha_i \notin L_n$.

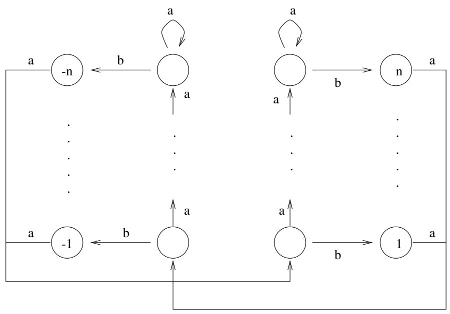


Figure 3.2: The Transition Structure of $\mathcal{A}_n$ over the Alphabet $\{a, b\}$

For the run σ_i of $\mathcal{A}'$ on α_i we have $|In(\sigma_i)| \geq (n-1)!$. Hence it remains to show $In(\sigma_i) \cap In(\sigma_j) = \emptyset$ for $i \neq j$.

Assume by contradiction that there exist $i \neq j$ with $In(\sigma_i) \cap In(\sigma_j) \neq \emptyset$. Then we can construct a word α with run σ such that $even(\alpha) = even(\alpha_i) \cup even(\alpha_j) = \{1, \ldots, n\}$, $odd(\alpha) = odd(\alpha_i) \cup odd(\alpha_j) = \{1, \ldots, n\}$ and $In(\sigma) = In(\sigma_i) \cup In(\sigma_j)$, by cycling alternatingly through the infinity sets of σ_i and σ_j (as in the proof of lemma 2.5). This is a contradiction since in Rabin automata the union of rejecting cycles is rejecting, but α is in L_n. $\square$

To adapt the proof for an alphabet of constant size we can code every letter $i \in \{1, \ldots, n-1\}$ with $a^i b$ and n with $a^n a^* b$. The resulting automaton looks like shown in figure 3.2 and still has $\mathcal{O}(n)$ states.

This lemma shows the optimality of the IAR construction for the transformation of Streett into Rabin and parity automata. Using the complementation theorem 1.18, we can also conclude that the IAR construction is optimal for the transformation of Rabin automata into Streett and parity automata.

Streett automata are special cases of Muller automata. Therefore the SAR construction is optimal for the transformation of Muller automata into Rabin and parity automata. Again, using the complementation theorem, we can conclude that the SAR construction is optimal for the transformation of Muller into Streett automata.

Note that from this lemma follows the optimality of all LAR (SAR and IAR) constructions listed in table 3.1.

3.3 Weak Automata

In this section we transform (co-)Büchi automata into weak automata. In [MSS86] Muller, Saoudi and Schupp introduce weak alternating automata. Using the full power of alternation this acceptance condition suffices to recognize all ω-regular languages. In [KV97] Kupferman and Vardi described a quadratic transformation of co-Büchi automata into weak automata, the rank construction. In this section we present their results.

3.3.1 The Rank Construction

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a co-Büchi automaton with $|Q| = n$. We assign to every vertex in a successful run G of $\mathcal{A}$ a unique rank in $\{0, \ldots, 2n\}$. For this aim we define a sequence $G_0, G_1, G_2, \ldots$ of subgraphs of G.

Let $G = (V, E)$ be a run of $\mathcal{A}$ on $\alpha \in \Sigma^\omega$ and $G' = (V', E')$ be a subgraph of G. A vertex (q', l') in G' is a descendant of a vertex (q, l) in G' if it is reachable from (q, l), i.e., $l' > l$ and there exist states $p_0, \ldots, p_m \in Q$, $m = l' - l$, such that $q = p_0$, $q' = p_m$ and $((p_i, l + i), (p_{i+1}, l + i + 1)) \in E'$ for $i = 0, \ldots, m - 1$. Furthermore we say a vertex (q, k) is final if and only if $q \in F$.

Now we define inductively for $i \in \mathbb{N}_{>0}$

- $G_0 = G$,
- $G_{2i-1} = G_{2i-2} \setminus \{(q, l) \mid (q, l)$ has only finitely many descendants in $G_{2i-2}\}$ and
- $G_{2i} = G_{2i-1} \setminus \{(q, l) \mid (q, l)$ is not final and has no final descendants in $G_{2i-1}\}$.

The rank of a vertex $(q, l) \in V$ is the unique $i \in \mathbb{N}$ such that (q, l) is in G_i but not in G_{i+1}. Above we mentioned that we assign to every vertex in G a rank from $\{0, \ldots, 2n\}$. This is stated in the following lemma.

Lemma 3.11 *G_{2n+1} is empty.*

Proof We proof by induction that for $i \in \{0, \ldots, n\}$ there exists a $k_i \in \mathbb{N}$ such that all levels k in G_{2i} with $k \geq k_i$ have at most $n - i$ vertices. Then we are done because G_{2n} has a level with 0 vertices. This means G_{2n} is finite and to obtain G_{2n+1} all vertices are removed.

Since $|Q| = n$, every level of $G = G_0$ has at most n vertices. This proves the base case of the induction with $k_0 = 0$.

Let $i \in \{0, \ldots, n - 1\}$. From the induction hypothesis follows that there is a k_i such that every level higher than k_i has at most $n - i$ vertices.

Suppose every vertex in G_{2i} has at least one final descendant. Let (q_1, l_1) be a final descendant of the root. Now define inductively (q_j, l_j) to be a final descendant of (q_{j-1}, l_{j-1}) for $j > 1$. There exists a path through all the (q_j, l_j). This path infinitely often visits a final state and therefore is rejecting. This is a contradiction to the fact that G is an accepting run. Hence there exists a vertex (q, l) in G_{2i} with no final descendants. We can choose (q, l) such that it is not a final vertex. In G_{2i+1} the vertex (q, l) and all its descendants are removed. This means every level higher than l in G_{2i} has at least one vertex more than in G_{2i+1}. If we choose $k_{i+1} = \max\{k_i, l\}$, then every level higher than k_{i+1} in G_{2i+1} has at most $n - (i + 1)$ vertices. This also holds for $G_{2(i+1)}$, since we only remove states from G_{2i+1} to obtain $G_{2(i+1)}$. $\square$

It is easy to see that for every vertex in G its rank can not be smaller than the rank of one of its descendants. Another obvious fact is that final vertices can only be removed in the step from G_{2i} to G_{2i+1}. Therefore all final vertices have an even rank.

Remark 3.12 (i) Let (q, l) be a vertex in G and (q', l') be a descendant of (q, l). The rank of (q', l') is less or equal to the rank of (q, l).

(ii) Let (q, l) be a final vertex in G. The rank of (q, l) is even.

Now we can characterize the paths going through an accepting run.

Lemma 3.13 *In every infinite path through G exists a vertex (q,l) with an odd rank such that all descendants of (q,l) lying on this path have the same rank as (q,l).*

Proof From remark 3.12 (i) follows that on every infinite path through G exists a vertex (q,l) such that all descendants of (q,l) lying on this path have the same rank i. All these vertices are removed in one step when constructing G_{i+1} from G_i. This means (q,l) lies on an infinite path in G_i and therefore has infinitely many descendants. Therefore i must be odd, otherwise (q,l) would not have been removed in the step from G_i to G_{i+1}. □

As we will see in the construction, the property from lemma 3.13 can be tested by a weak automaton.

Theorem 3.14 ([KV97]) *Let $\mathcal{A} = (Q,\Sigma,q_0,\delta,F)$ be a co-Büchi automaton with n states. There exists an equivalent weak automaton $\mathcal{A}' = (Q',\Sigma,q_0',\delta',F')$ with $\mathcal{O}(n^2)$ states.*

Proof The idea in the construction is to guess the rank of the vertices in a successful run of $\mathcal{A}$ on α. Therefore we make $2n+1$ copies of the automaton, corresponding to the ranks, and start with $(q_0, 2n)$. The first component simulates a run of $\mathcal{A}$ and in the second component the rank of the vertices is guessed. From lemma 3.13 follows that every path must eventually get trapped in a copy of the automaton with an odd index. Therefore these copies become the final states. The automaton $\mathcal{A}'$ is defined as follows.

- $Q' = Q \times \{0,\ldots,2n\}$.
- $q_0' = (q_0, 2n)$.
- δ' is defined by means of a function *release*: $\mathcal{B}^+(Q) \times \{0,\ldots,2n\} \longrightarrow \mathcal{B}^+(Q')$. For a formula $\theta \in \mathcal{B}^+(Q)$ and $i \in \{0,\ldots,2n\}$ the formula *release*(θ,i) is obtained by replacing every element q from Q in θ by $\bigvee_{j\le i}(q,j)$. Now we can define for $q \in Q$, $i \in \{0,\ldots,2n\}$ and $a \in \Sigma$

$$\delta'((q,i),a) = \begin{cases} \mathit{release}(\delta(q,a),i) & \text{if} \quad i \text{ is even or } q \notin F, \\ \texttt{false} & \text{if} \quad i \text{ is odd and } q \in F. \end{cases}$$

- $F' = Q \times \{1,3,\ldots,2n-1\}$.

$\mathcal{A}'$ is a weak automaton. If we define $Q_i = Q \times \{i\}$ for $i \in \{0,\ldots,2n\}$, then all the transitions starting in Q_i lead to a Q_j with $j \le i$. Furthermore the set of final states is $\bigcup_{i\in\{1,\ldots,n\}} Q_{2i-1}$.

Before we prove the equivalence of $\mathcal{A}$ and $\mathcal{A}'$ first note the following relation between δ and δ'. Let $q \in Q$, $i \in \{0,\ldots,2n\}$ and $a \in \Sigma$ such that $q \notin F$ or i is even. Let $q_1,\ldots,q_k \in Q$ and $i_1,\ldots,i_k \in \{0,\ldots,i\}$. The set $\{q_1,\ldots,q_k\}$ exactly satisfies $\delta(q,a)$ if and only if $\{(q_1,i_1),\ldots,(q_k,i_k)\}$ exactly satisfies $\delta'((q,i),a)$. This follows from the definition of the function *release*.

$L(\mathcal{A}) \subseteq L(\mathcal{A}')$: Let $\alpha \in L(\mathcal{A})$ and let G be an accepting run of $\mathcal{A}$ on α. Let G' be the graph obtained from G by replacing $(q_0, 0)$ with $((q_0, 2n), 0)$ and every other vertex (q, l) is replaced with $((q, r_{q,l}), l)$, where $r_{q,l}$ is the rank of (q, l) in G. Consider that there is no final vertex with an odd rank. Thus we can conclude from the relation of δ and δ', mentioned above, that G' is a run of $\mathcal{A}'$ on α. From lemma 3.13 follows that every path eventually gets trapped in a Q_i with i odd. Therefore G' is an accepting run of $\mathcal{A}'$ on α.

$L(\mathcal{A}') \subseteq L(\mathcal{A})$: Let $\alpha \in L(\mathcal{A}')$ and let G' be an accepting run of $\mathcal{A}'$ on α. If we replace every vertex $((q, i), l)$ in G' by (q, l), then we get a run of $\mathcal{A}$ on α. This follows from the relation between δ and δ' and the fact that there exists no vertex $((q, i), l)$ in G' with $q \in F$ and i odd.

Every path in G' eventually gets trapped in a Q_i with i odd. Therefore every path in G eventually stays in $Q \setminus F$, since $\delta'((q, i), a) = \texttt{false}$ if $q \in F$ and i is odd. □

Since the dual of a weak automaton also is a weak automaton, we can use the construction from theorem 3.14 to transform Büchi automata into weak automata. If we are given a Büchi automaton with n states, then we can construct a complementary co-Büchi automaton with n states using the complementation theorem 1.18. Then we apply theorem 3.14 and complement again. The resulting automaton is a weak automaton with $\mathcal{O}(n^2)$ states that is equivalent to the Büchi automaton we started with. Therefore we have the following corollary.

Corollary 3.15 *For every Büchi automaton with n states exists an equivalent weak automaton with $\mathcal{O}(n^2)$ states.*

In [KV98] Kupferman and Vardi transform parity and Rabin automata into weak automata. The main idea of these constructions is also based on the rank construction and hence we do not present them here.

Optimality Considerations

Before we turn to nondeterministic, universal and deterministic weak automata, we will discuss possible improvements for the transformation of (co-)Büchi automata into weak automata. Furthermore we give an example showing that in the rank construction the number of different ranks has to be linear in the number of states.

The rank construction is of quadratic complexity in the number of states. Kupferman and Vardi already showed in [KV97] that this construction can not be improved to a linear construction. Let us explain why the lower bound is $\mathcal{O}(n \cdot \log n)$.

In section 2.2 we proved that there exist languages L_n that can be recognized by nondeterministic Büchi automata with n states, but the complement of L_n can not be recognized by nondeterministic Streett automata with less than $n!$ states. Since Büchi automata are special cases of Streett automata, we know that a complementation construction for nondeterministic Büchi automata must be of complexity $2^{\mathcal{O}(n \cdot \log n)}$. We can obtain a complementation construction by concatenating the transformation of Büchi automata into weak automata, the complementation of weak automata and the transformation of weak automata into nondeterministic Büchi automata (using the breakpoint construction). Since the complementation of weak automata can be done

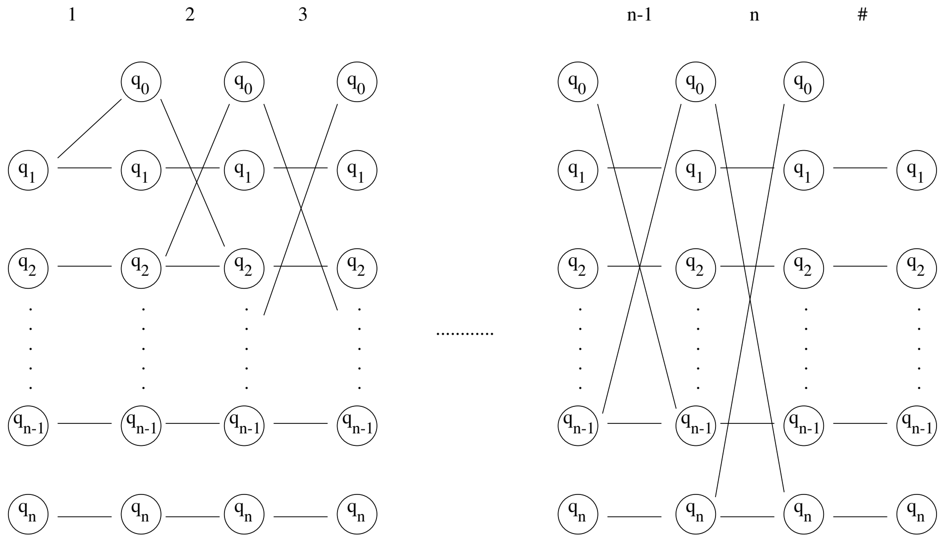


Figure 3.3: Run Segment of $\mathcal{A}_n$ on $(12\cdots n\#)^\omega$

without changing the set of states and the transformation of weak automata into nondeterministic Büchi automata is singly exponential, the complexity of the transformation of Büchi automata into weak automata must be at least $\mathcal{O}(n \cdot \log n)$.

It is not possible to meet this lower bound with the rank construction, i.e., the number of ranks can not be reduced to $\mathcal{O}(\log n)$.

Remember the automaton from figure 2.3. In the following we work with the dual of this automaton with q_0 as the only initial state. The resulting universal co-Büchi automaton $\mathcal{A}_n = (Q_n, \Sigma_n, q_0, \delta_n, F_n)$ is defined as follows.

- $Q_n = \{q_0, q_1, \ldots, q_n\}$.
- $\Sigma_n = \{1, \ldots, n, \#\}$.
- The transition function δ_n is defined by

$$\begin{array}{lcll}
\delta_n(q_0, a) & = & q_a & \text{for } a \in \{1, \ldots, n\}, \\
\delta_n(q_0, \#) & = & \texttt{true}, & \\
\delta_n(q_i, a) & = & q_i & \text{for } a \in \Sigma_n, i \in \{1, \ldots, n\}, a \neq i, \\
\delta_n(q_i, i) & = & q_i \wedge q_0 & \text{for } i \in \{1, \ldots, n\}.
\end{array}$$

- $F_n = \{q_0\}$

Regard the run G_n of $\mathcal{A}_n$ on the word $\alpha_n = (12\cdots n\#)^\omega$. After having read the prefix $(12\cdots n\#)^n$ of α_n the run consists of a repetition of the segment shown in figure 3.3. The main observation about this run is the following. Let $l \in \mathbb{N}$ be the smallest index such that $(q_n, l) \in G_n$. If we remove (q_n, l) and all its descendants, then we obtain the run of $\mathcal{A}_{n-1}$ on α_{n-1}. We will use this fact to prove by induction on n that the vertex $(q_0, 0)$ has rank $2n$ in the run G_n.

For $n = 1$ all the vertices of the form (q_0, l) with $l > 0$ have no successors and therefore rank 0. All the vertices of the form (q_1, l) have infinitely many descendants, namely all vertices (q_1, k) or (q_0, k) with $k > l$. After having removed all vertices with rank 0, the vertices (q_1, l) still have infinitely many descendants but no more final descendants. Therefore these vertices have rank 1. The only vertex that is left is $(q_0, 0)$ which has rank 2. This proves the base case of the induction.

Now let $n \geq 2$. The only vertices with finitely many descendants are the vertices (q_0, l) such that $(q_n, l-1)$ is a predecessor of (q_0, l). These vertices have rank 0. After having removed these, the vertices of the form (q_n, l) are exactly those having no final descendants. Thus they have rank 1. As mentioned above we obtain G_{n-1} after having removed all these vertices. From the induction hypothesis we know that $(q_0, 0)$ has rank $2(n-1)$ in G_{n-1}. Thus, to determine the rank of $(q_0, 0)$ in G_n, we only have to add 2 to its rank in G_{n-1}. Therefore the rank of $(q_0, 0)$ in G_n is $2n$.

3.3.2 Nondeterministic and Universal Weak Automata

From figure 1.3 one can see that nondeterministic weak automata are as expressive as nondeterministic co-Büchi automata and universal weak automata are as expressive as universal Büchi automata. These relations are explained in the following.

From the construction we used in the proof of theorem 3.14 follows that if we start with a nondeterministic co-Büchi automaton, then the resulting weak automaton is nondeterministic too. Since a run of a nondeterministic automaton contains at most one path, we also know that the states in such a run have at most rank 2. Therefore the number of states in the resulting weak automaton is linear in the number of states in the co-Büchi automaton.

Using the complementation theorem 1.18 we can conclude that if we start with a universal Büchi automaton, then the resulting weak automaton is universal too.

Lemma 3.16 (i) *For every nondeterministic co-Büchi automaton with n states exists an equivalent nondeterministic weak automaton with $\mathcal{O}(n)$ states.*

(ii) *For every universal Büchi automaton with n states exists an equivalent universal weak automaton with $\mathcal{O}(n)$ states.*

3.3.3 Deterministic Weak Automata

In figure 1.3 we can see that deterministic weak automata recognize exactly those languages that are deterministic Büchi and deterministic co-Büchi. This is stated in the following lemma.

Lemma 3.17 *Let $L \subseteq \Sigma^\omega$ be an ω-language. L is deterministic weak if and only if it is deterministic Büchi and deterministic co-Büchi.*

Proof First note that a deterministic weak automaton in particular is a deterministic Büchi and a deterministic co-Büchi automaton.

For the other direction let $\mathcal{A}_1 = (Q_1, \Sigma, q_0^1, \delta_1, F_1)$ be a deterministic Büchi automaton with $L(\mathcal{A}_1) = L$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0^2, \delta_2, F_2)$ be a deterministic co-Büchi automaton with $L(\mathcal{A}_2) = L$. Now we construct the product transition structure $T = (Q, \Sigma, q_0, \delta)$ in the following way.

- $Q = Q_1 \times Q_2$.
- $q_0 = (q_0^1, q_0^2)$.
- For $q_1 \in Q_1$, $q_2 \in Q_2$ and $a \in \Sigma$ let $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.

Obviously T is a deterministic transition structure. Let $\mathcal{B}$ be the Büchi automaton with transition structure T and acceptance condition $F = F_1 \times Q_2$ and let $\mathcal{C}$ be the co-Büchi automaton with transition structure T and acceptance condition $F' = Q_1 \times F_2$. From the construction of T follows $L(\mathcal{B}) = L(\mathcal{C}) = L$. This means for every cycle $C \subseteq Q$ one has $C \cap F \neq \emptyset$ if and only if $C \cap F' = \emptyset$.

Let $S \subseteq Q$ be a strongly connected component of T and let $C_1, C_2 \subseteq S$ be two cycles. Since S is strongly connected, we can conclude that $C_1 \cap F \neq \emptyset$ if and only if $C_2 \cap F \neq \emptyset$ (otherwise there would be a cycle touching F and F'). This means in a strongly connected component every cycle touches F or no cycle touches F.

Let $Q_1, \ldots, Q_m$ be the strongly connected components of T ordered in such a way that if Q_j is reachable from Q_i, then $j \leq i$. From the facts above follows that the automaton with transition structure T and Büchi acceptance condition $\bigcup_{Q_i \cap F \neq \emptyset} Q_i$ is a weak automaton recognizing L. $\square$

Using the above lemma we can conclude that deterministic weak automata exactly recognize the Staiger-Wagner hierarchy [SW74]. Summarizing the results from this section, we can characterize all levels of the Landweber hierarchy (as shown in figure 1.3) with weak automata.

Deterministic Muller	$\leftrightarrow$	Alternating Weak
Deterministic Büchi	$\leftrightarrow$	Universal Weak
Deterministic Co-Büchi	$\leftrightarrow$	Nondeterministic Weak
DB and DCB	$\leftrightarrow$	Deterministic Weak

The directions from left to the right follow from the results of this section. The directions from right to the left are easy to prove. An alternating weak automaton can be transformed into a deterministic Muller automaton by concatenating the breakpoint and Safra's construction. A universal weak automaton can be transformed into a deterministic Büchi automaton, using the breakpoint construction (applying the breakpointconstruction to a universal automaton yields a deterministic one). Due to the complementation theorem it is also possible to transform a nondeterministic weak automaton into a deterministic co-Büchi automaton. A deterministic weak automaton in particular is a deterministic Büchi automaton and a deterministic co-Büchi automaton.

3.4 Büchi and Co-Büchi Automata

Theorem 1.7 stated that every ω-regular language can be recognized by a nondeterministic Büchi automaton. In this section we will prove that deterministic Büchi automata do not suffice to recognize all the ω-regular languages. As a consequence of this there exists no construction transforming a Muller, Rabin, or Streett automaton into a Büchi automaton with the same mode of transition structure. Hence, in this section, we focus our interest on transformations from nondeterministic Muller, Rabin, or Streett automata into nondeterministic Büchi automata. With the complementation theorem we can conclude that every such transformation can be used to transform universal Muller, Rabin, or Streett automata into universal co-Büchi automata.

Lemma 3.18 *There exists a language L that can be recognized by a nondeterministic Büchi automaton but can not be recognized by a deterministic Büchi automaton.*

Proof Let L be the language over $\Sigma = \{a, b\}$ containing all the words with only finitely many b's. Then L is described by the regular expression $(a+b)^*a^\omega$ and thus can be recognized by a nondeterministic Büchi automaton, following theorem 1.7.

Assume there is a deterministic Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ with $L(\mathcal{A}) = L$. Let $\sigma_1 \in Q^\omega$ be the run of $\mathcal{A}$ on $\alpha_1 = a^\omega$. Since a^ω is accepted by $\mathcal{A}$ there is a $k_1 \in \mathbb{N}$ such that $\sigma_1(k_1) \in F$. Let $\sigma_2 \in Q^\omega$ be the run of $\mathcal{A}$ on $\alpha_2 = a^{k_1}ba^\omega$. There exists a $k_2 > k_1$ such that $\sigma_2(k_2) \in F$. Because $\mathcal{A}$ is deterministic, one has $\sigma_2(k_1) = \sigma_1(k_1)$. This procedure is repeated until there are i, j with $i < j$ and $\sigma_j(k_i) = \sigma_j(k_j) \in F$. From the construction of α_j follows that there is a k with $k_i < k \leq k_j$ such that $\alpha_j(k) = b$. Therefore the word $\alpha = \alpha_j(0) \ldots \alpha_j(k_i)(\alpha_j(k_i+1) \ldots \alpha_j(k_j))^\omega$ contains infinitely many b's and the run of $\mathcal{A}$ on α contains infinitely many final states. This is a contradiction to $L(\mathcal{A}) = L$. $\square$

The following constructions are not very difficult. So we explain the idea but do not give a formal correctness proof.

3.4.1 Muller to Büchi

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a nondeterministic Muller automaton with $\mathcal{F} = \{F_1, \ldots, F_r\}$. An equivalent Büchi automaton guesses for an accepting run of $\mathcal{A}$ the accepting set F_k and the index from which on only the states in F_k are visited. To test if all the states in F_k are visited, the automaton accumulates the states in a set R and resets this set to $\emptyset$ in the case of $R = F_k$.

Define the Büchi automaton $\mathcal{A}' = (Q', \Sigma, q_0, \delta', F)$ as follows.

- $Q' = Q \cup (Q \times 2^Q \times \{1, \ldots, r\})$.
- For $a \in \Sigma$, $q \in Q$, $P \subseteq Q$ and $k \in \{1, \ldots, r\}$ define

$$\begin{array}{lcl} \delta'(q,a) & = & \delta(q,a) \cup \{(p, \emptyset, j) \mid j \in \{1, \ldots, r\} \text{ and } p \in \delta(q,a)\}, \\ \delta'((q,P,k),a) & = & \left\{ \begin{array}{ll} \{(p, P \cup \{q\}, k) \mid p \in \delta(q,a)\} & \text{if } \; P \cup \{q\} \neq F_k, \\ \{(p, \emptyset, k) \mid p \in \delta(q,a)\} & \text{if } \; P \cup \{q\} = F_k. \end{array} \right. \end{array}$$

- $F = \{(q, \emptyset, j) \mid q \in Q \text{ and } j \in \{1, \ldots, r\}\}$.

If the Muller automaton has n states, then the resulting Büchi automaton has $2^{\mathcal{O}(n)}$ states. This bound was shown to be optimal in [SV89].

3.4.2 Streett to Büchi

Of course we can use the previous construction to transform nondeterministic Streett automata into nondeterministic Büchi automata. The complexity of the construction we present here mainly depends on the number of pairs in the acceptance condition. If this number is smaller than the number of states, then the complexity of this construction is better than the previous one.

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be a nondeterministic Streett automaton with acceptance condition $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$. An accepting run σ of $\mathcal{A}$ on a word $\alpha \in \Sigma^\omega$ has the property that there exists a k such that for every $l_1 > k$ with $\sigma(l_1) \in F_i$, there is an $l_2 \geq l_1$ with $\sigma(l_2) \in E_i$.

Hence the Büchi automaton can guess this k and collect from this point on the indices of the visited E's and F's in two sets I and J. Every time one has $I \subseteq J$ both sets are reset to $\emptyset$. If I and J are reset infinitely often to $\emptyset$, then the run is accepting.

Define the Büchi automaton $\mathcal{A}' = (Q', \Sigma, q_0, \delta', F)$ as follows.

- $Q' = Q \cup (Q \times 2^{\{1,\ldots,r\}} \times 2^{\{1,\ldots,r\}})$.
- For $a \in \Sigma$, $q \in Q$, $I, J \subseteq \{1, \ldots, r\}$ and $I' = I \cup \{i \mid q \in F_i\}$, $J' = J \cup \{j \mid q \in E_j\}$ define

$$\begin{array}{lcl} \delta'(q,a) & = & \delta(q,a) \cup \{(p,\emptyset,\emptyset) \mid p \in \delta(q,a)\}, \\ \delta'((q,I,J),a) & = & \left\{ \begin{array}{ll} \{(p,I',J') \mid p \in \delta(q,a)\} & \text{if } \; I' \not\subseteq J', \\ \{(p,\emptyset,\emptyset) \mid p \in \delta(q,a)\} & \text{if } \; I' \subseteq J'. \end{array} \right. \end{array}$$

- $F = Q \times \{\emptyset\} \times \{\emptyset\}$.

If the Streett automaton has n states, then the resulting Büchi automaton has $n \cdot 2^{\mathcal{O}(r)}$ states. This bound also was shown to be optimal in [SV89].

3.4.3 Rabin to Büchi

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be a nondeterministic Rabin automaton with acceptance condition $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$. An equivalent Büchi automaton guesses for an accepting run of $\mathcal{A}$ the accepting pair (E_k, F_k) and the index from which on the states in E_k are not visited anymore.

Define the Büchi automaton $\mathcal{A}' = (Q', \Sigma, q_0, \delta', F)$ as follows.

- $Q' = Q \cup (Q \times \{1, \ldots, r\})$.
- For $a \in \Sigma$, $q \in Q$ and $k \in \{1, \ldots, r\}$ define

$$\begin{array}{lcl} \delta'(q,a) & = & \delta(q,a) \cup \{(p,j) \mid p \in \delta(q,a) \text{ and } j \in \{1,\ldots,r\}\}, \\ \delta'((q,k),a) & = & \left\{ \begin{array}{ll} \emptyset & \text{if } q \in E_k, \\ \{(p,k) \mid p \in \delta(q,a)\} & \text{otherwise.} \end{array} \right. \end{array}$$

- $F = \bigcup_{i=1}^{r} F_i \times \{i\}$.

If the Rabin automaton has n states, then the resulting Büchi automaton has $\mathcal{O}(n \cdot r)$ states.

3.4.4 Complementation of Büchi Automata

With the complementation theorem from section 1.6 we can complement a nondeterministic Büchi automaton, yielding a universal co-Büchi automaton. In this subsection we will investigate the problem how to complement a nondeterministic Büchi automaton, yielding a nondeterministic Büchi automaton again. Such a construction can also be viewed as a transformation of acceptance types, namely transforming universal co-Büchi automata into nondeterministic Büchi automata (due to the complementation theorem 1.18). From lemma 2.5 follows that in such a complementation construction the resulting automaton must have $2^{\mathcal{O}(n \cdot \log n)}$ states if the given automaton has n states.

Büchi gave a direct complementation procedure [Büc62]. For a nondeterministic Büchi automaton $\mathcal{A}$ he described $L(\mathcal{A})$ and $\overline{L(\mathcal{A})}$ by regular expressions, using a congruence over finite words, defined in terms of the automaton $\mathcal{A}$. From a regular expression describing a language L one can construct an automaton recognizing the same language.

McNaughton's approach uses deterministic automata and the fact that they are closed under complementation [McN66]. Thus, a determinization construction (e.g. Safra's construction from section 2.1) is involved.

$$\mathrm{NB} \to \mathrm{DR} \stackrel{\text{Comp.}}{\longrightarrow} \mathrm{DS} \to \mathrm{NB}$$

Applying Safra's construction to a nondeterministic Büchi automaton with n states yields a deterministic Rabin automaton with $2^{\mathcal{O}(n \cdot \log n)}$ states and $\mathcal{O}(n)$ pairs in the acceptance condition. The complementation does not affect the number of states or pairs. The transformation of Streett automata into Büchi automata is exponential in the number of pairs. Thus, concatenating these constructions yields a complementation construction for nondeterministic Büchi automata, meeting the lower bound.

In [KV97] Kupferman and Vardi gave a complementation procedure using the rank construction and the breakpoint construction.

$$\mathrm{NB} \stackrel{\text{Comp.}}{\longrightarrow} \mathrm{UCB} \to \mathrm{AW} \to \mathrm{NB}$$

Concatenating these constructions straightforward yields an automaton with $2^{\mathcal{O}(n^2)}$ states. But, taking advantage of the special structure of the alternating weak automaton, the complexity can be cut to $2^{\mathcal{O}(n \cdot \log n)}$.

Here we present a complementation construction similar to the one of Kupferman and Vardi. But in our construction the complementation step is done on the level of the alternating weak automata. This yields a more 'symmetric' construction, using the fact that alternating weak automata are closed under complementation.

$$\mathrm{NB} \to \mathrm{AW} \stackrel{\text{Comp.}}{\longrightarrow} \mathrm{AW} \to \mathrm{NB}$$

Again a straightforward concatenation yields a blow up of $2^{\mathcal{O}(n^2)}$ but similar to [KV97] we cut the complexity to $2^{\mathcal{O}(n \cdot \log n)}$.

The construction transforming nondeterministic Büchi automata into alternating weak automata is primary the dual to the rank construction. But the correctness proof is different and so we present this construction here.

Lemma 3.19 *For every nondeterministic Büchi automaton with n states exists an equivalent alternating weak automaton with $\mathcal{O}(n^2)$ states.*

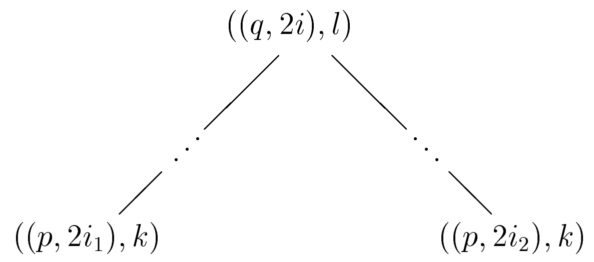


Figure 3.4: Illustration of the notion double descendant.

Proof Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a nondeterministic Büchi automaton with n states. Define the alternating weak automaton $\mathcal{A}' = (Q', \Sigma, q_0', \delta', F')$ as follows.

- $Q' = Q \times \{0, \ldots, 2n\}$.
- $q_0' = (q_0, 2n)$.
- For $q \in Q$, $i \in \{0, \ldots, 2n\}$ and $a \in \Sigma$ let

$$\delta'((q,i),a) = \begin{cases} \bigvee_{p \in \delta(q,a)} (p,i) & \text{if } \; i = 0, \\ \bigvee_{p \in \delta(q,a)} (p,i) \wedge (p, i-1) & \text{if } \; i > 0, \; i \text{ is even}, \\ \bigvee_{p \in \delta(q,a)} (p,i) & \text{if } \; i \text{ is odd and } q \notin F, \\ \bigvee_{p \in \delta(q,a)} (p,i-1) & \text{if } \; i \text{ is odd and } q \in F. \end{cases}$$

- $F' = Q \times \{0, 2, 4, \ldots, 2n\}$.

Obviously $\mathcal{A}'$ is a weak automaton with $\mathcal{O}(n^2)$ states.

For a vertex $((q,i),l)$ in a run of $\mathcal{A}'$ we call q its Q-component, i its rank and l its level.

$L(\mathcal{A}) \subseteq L(\mathcal{A}')$: Let $\alpha \in L(\mathcal{A})$ and let σ be an accepting run. There exists a run G' of $\mathcal{A}'$ on α such that every path through this run projected to its Q-component equals σ. Since σ contains infinitely many final states, no path through G' gets trapped in an odd copy of the automaton. Therefore G' is accepting.

$L(\mathcal{A}') \subseteq L(\mathcal{A})$: Let $\alpha \in L(\mathcal{A}')$ and let G' be an accepting run. To show that there is an accepting run of $\mathcal{A}$ on α, we need some terminology.

Consider the following situation in G'. Let $v = ((q,2i),l)$ be a vertex with a descendant $u = ((p,2j),k)$ and $j < i$. Then the path leading from v to u must visit a vertex with odd rank and, to leave this odd rank again, it must pass a vertex with a Q-component from F. We can conclude that $q \xrightarrow{\alpha[l,k]}_F p$ in $\mathcal{A}$. This motivates the following definition. Let $v = ((q,2i),l)$ be a vertex in G'. A descendant $((p,2i_1),k)$ of v is called a *double descendant* of v if and only if there exists a descendant $((p,2i_2),k)$ of v with $i_2 < i_1$. Since descendants of v have a rank lower or equal to $2i$, we also know $i_2 < i$ and therefore $q \xrightarrow{\alpha[l,k]}_F p$ in $\mathcal{A}$.

We call a vertex $((p, 2i_1), k)$ double descendant iff it is a double descendant of any vertex in G'.

From the observations made above, we can conclude that, if there exists a path in G' leading through infinitely many vertices $v_0, v_1, v_2, \ldots$ such that v_{j+1} is a double descendant of v_j for all $j \in \mathbb{N}$, then there is an accepting run of $\mathcal{A}$ on α because of

$$q_0 \xrightarrow{\alpha[l_0,l_1)}_F q_1 \xrightarrow{\alpha[l_1,l_2)}_F q_2 \cdots$$

with $v_j = ((q_j, 2i_j), l_j)$ for $j \in \mathbb{N}$.

We have to show that G' contains such an infinite sequence of double descendants.

Let v be a vertex of even rank $2i$. We say that a level l saturates v iff it contains for each $j < i$ a descendant of v with rank $2j$ and claim that there is a level l saturating v: The automaton $\mathcal{A}$ is nondeterministic and therefore has no $\top$ in its transition function. Hence, $\mathcal{A}'$ also has no $\top$ in its transition function. Since G' is an accepting run of $\mathcal{A}'$, all paths through G' must be infinite. If a vertex of even rank is visited in G', then all the following levels also contain a vertex with this rank (this follows from the infinity of the paths through G' and the definition of δ'). If $i = 0$, then we are done. In the other case we additionally show that v has a descendant of rank $2(i-1)$. Then we are done, because the rank $2i$ was chosen arbitrarily. From the definition of δ' follows that v has a successor u of rank $2i - 1$. Since G' is accepting, u must have a descendant of rank $2(i-1)$. Otherwise there is an infinite path never leaving rank $2i - 1$. This is a contradiction because G' is accepting and $2i - 1$ is odd.

Using similar arguments as above, it is easy to see that even for a finite number of vertices $v_0, \ldots, v_m$ there exists a level l saturating all these vertices.

Now suppose in the accepting run G' of $\mathcal{A}'$ exists no infinite sequence of double descendants. Then there exist nodes $v_0, \ldots, v_m$ in G' such that v_0 has rank $2n$, v_i is a double descendant of v_{i-1} with maximal rank for all $i \in \{1, \ldots, m\}$, and v_m does not have any double descendants. The existence of vertices with these properties is guaranteed from the fact that there is no infinite sequence of double descendants in G'. Furthermore we can choose $v_0, \ldots, v_m$ such that the rank of v_i is strictly smaller than the rank of v_{i-1} for all $i \in \{1, \ldots, m\}$, just by removing all v_i with the same rank as v_{i+1}. Let $j_0, \ldots, j_m$ denote the ranks of $v_0, \ldots, v_m$ and let $j_{m+1} = 0$.

Let l be a level that saturates all vertices $v_0, \ldots, v_m$. For every $j \in \{0, \ldots, n\}$ we pick a vertex u_j with rank $2j$ from level l such that it is a descendant of the unique v_i with $j_i \geq j > j_{i+1}$. Then u_j can not be a descendant of v_{i+1}, because u_j has a higher rank. Let $p_0, \ldots, p_n$ be the Q-components of $u_0, \ldots, u_n$. Since there are only n states in $\mathcal{A}$, there exists $s, t \in \{0, \ldots, n\}$ with $s > t$ and $p_s = p_t$. Let i be such that u_s is a descendant of v_i but no descendant of v_{i+1}. The vertex u_t also is a descendant of v_i and therefore u_s is a double descendant of v_i with a bigger rank than v_{i+1}. But v_{i+1} was chosen as a double descendant of v_i with maximal rank. This gives us a contradiction. $\square$

The next step in our complementation procedure for Büchi automata is the complementation of alternating weak automata. For the proof of the complementation theorem one needs that, for an automaton $\mathcal{A}$ and a word α, the game $G(\mathcal{A}, \alpha)$ is determined. In section 1.5 we used the theorem of Martin to obtain this result. But Martin's theorem is a very deep result, stating that every game with a winning condition that is Borel is determined. For a game $G(\mathcal{A}, \alpha)$, where $\mathcal{A}$ is a weak automaton, this result is much more easy to prove.

The idea is to find the sets of nodes in the game where either player 0 can force the match to move to an accepting component and stay there, or player 1 can force the match to move to a rejecting component and stay there. From these sets of nodes one of the players has a winning strategy. If the union of these two sets covers the whole game graph, then the game is determined. The set of nodes from which player i can force the match to move into a set of nodes T is called the i-attractor of T.

Definition 3.20 Let $G = (V_0, V_1, E, c, Win)$ with $V = V_0 \cup V_1$ be a game, let $i \in \{0, 1\}$, and let $T \subseteq V$. The *i-attractor of* T, denoted $Attr_i(T)$, is defined as follows.

$$\begin{array}{rcl} Attr_i^0(T) & = & T, \\ v \in Attr_i^{k+1}(T) & \Leftrightarrow & v \in Attr_i^k(T) \text{ or} \\ & & v \in V_i^{\mathcal{A}} \text{ and } \exists (v, w) \in E^{\mathcal{A}} (w \in Attr_i^k(T)) \text{ or} \\ & & v \in V_{1-i}^{\mathcal{A}} \text{ and } \forall (v, w) \in E^{\mathcal{A}} (w \in Attr_i^k(T)), \\ Attr_i(T) & = & \bigcup_{k \in \mathbb{N}} Attr_i^k(T). \end{array}$$

Similar to weak automata we define the notion of a weak game.

Definition 3.21 A game $G = (V_0, V_1, E, c, Win)$ with $V = V_0 \cup V_1$ is called *weak* iff there exists a partition $U_1, \ldots, U_m$ of V and a set $I \subseteq \{1, \ldots, m\}$ with the following properties.

- Let $C_i = \{c(v) \mid v \in U_i\}$. For $i, j \in \{1, \ldots, m\}$ with $i \neq j$ the sets C_i and C_j are disjoint.
- If $v \in U_i$, $u \in U_j$, and $(v, u) \in E$, then $j \leq i$.
- Player 0 wins a match γ iff there exists an $i \in I$ such that $In(c(\gamma)) \subseteq C_i$.

Corresponding to the terminology for weak automata, we call a set U_i accepting if $i \in I$ and rejecting otherwise.

To show that a game $G(\mathcal{A}, \alpha)$, where $\mathcal{A}$ is a weak automaton, is determined, we first show that weak games are determined and then show that $G(\mathcal{A}, \alpha)$ is a weak game if $\mathcal{A}$ is a weak automaton.

Lemma 3.22 *Weak games are determined.*

Proof Let $G = (V_0, V_1, E, c, Win)$, $V = V_0 \cup V_1$, be a weak game with partition $U_1, \ldots, U_m$ and index set $I \subseteq \{1, \ldots, m\}$ as in the definition above. We prove, by induction on m, the number of sets in the partition, that G is determined.

For $m = 1$ the whole set V is either accepting or rejecting. Obviously G is determined in this case.

Let $m \geq 2$. In a match of G the players can only move to lower sets in the partition or stay in the same set. Thus, if U_1, the lowest set in the partition, is accepting, then player 0 wins if he can force the match to move to U_1. If U_1 is rejecting, then player

1 wins if he can force the match to move to U_1. We only deal with the case that U_1 is accepting. The other case is totally symmetric to this case.

Let $A = Attr_0(U_1)$. To apply the induction hypothesis we define a new game $G' = (V_0', V_1', E', c', Win)$, $V' = V_0' \cup V_1'$, by removing all the nodes A from V and all the corresponding edges. We have to show that G' still is a game and that the winning area of player i in G' belongs to the winning area of player i in G. Then we are done, because U_1 belongs to the 0-attractor of U_1, and therefore the game G' has a partition, as demanded in definition 3.21, with less than m sets. Hence G' is determined by the induction hypothesis. The winning area of player 0 in G then is the union of A and the winning area of player 0 in G'. The winning area of player 1 in G is the same as in G'. Thus the winning areas cover the whole set of nodes and G is determined.

Without loss of generality we can assume that V' is not empty. Otherwise the game obviously is determined. To show that G' is still a game, we have to show that every node in G' has at least one successor. Suppose there is a node v in V' such that v has no successor. Let $v_1, \ldots, v_l$ be the successors of v in G (by the definition of game every node only has finitely many successors). Then there exists a $k \in \mathbb{N}$ such that $v_j \in Attr_0^k(U_1)$ for each $j \in \{1, \ldots, l\}$. But then v belongs to $Attr_0^{k+1}(U_1)$. This is a contradiction, since $v \notin A$.

Let $v \in V'$ belong to the winning area of player 0 in G'. If a match starting at v stays in V', then player 0 has a winning strategy, since v belongs to the winning area of player 0 in G'. If the match moves to $V \setminus V' = A$, then player 0 can force the game to reach U_1 and also wins. Therefore v also belongs to the winning area of player 0 in G.

Now let $v \in V'$ belong to the winning area of player 1 in G'. Assume that v does not belong to the winning area of player 1 in G. Since player 1 wins the match if it stays in V', there must exists a node $u \in V_0'$ and a node $w \in A$ such that $(u, w) \in E$. But since $w \in A$, there must be a $k \in \mathbb{N}$ such that $w \in Attr_0^k(U_1)$. From the definition of $Attr_0^{k+1}(U_1)$ follows that $u \in Attr_0^{k+1}(U_1)$, contradicting $u \in V'$. Therefore v also belongs to the winning area of player 1 in G. □

Lemma 3.23 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an alternating weak automaton and let $\alpha \in \Sigma^\omega$. The game $G(\mathcal{A}, \alpha)$ is determined.*

Proof As mentioned above we show that $G(\mathcal{A}, \alpha)$ is a weak game and then apply lemma 3.22.

Let $Q_1, \ldots, Q_m$ be the partition of Q. The nodes of $G(\mathcal{A}, \alpha)$ are of the form (q, l) or (S, l), for $q \in Q$, $S \subseteq Q$, and $l \in \mathbb{N}$. From the restricted transition structure of $\mathcal{A}$ and the definition of $G(\mathcal{A}, \alpha)$ follows that every node (q, l) with $q \in Q_i$ can only have successors of the form (S, l), where S is a subset of the the union of the Q_j with $j \leq i$. And nodes (S, l) can only have successors of the form (q, l) with $q \in Q_i$, where $i \leq max\{j \in \{1, \ldots, m\} \mid S \cap Q_j \neq \emptyset\}$. Therefore we define the partition of $V^{\mathcal{A}}$ for $i \in \{1, \ldots, m\}$ as follows.

$$U_i = (Q_i \times \mathbb{N}) \cup \{(S, l) \in v^{\mathcal{A}} \mid max\{j \in \{1, \ldots, m\} \mid S \cap Q_j \neq \emptyset\} = i\}.$$

Obviously $U_1, \ldots, U_m$ is a partition of $V^{\mathcal{A}}$. From the properties of weak automata follows that this partition also fulfills the properties demanded in the definition of a weak game. □

Now we prove the main theorem of this subsection.

Theorem 3.24 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a nondeterministic Büchi automaton with n states. There exists a nondeterministic Büchi automaton $\mathcal{A}' = (Q', \Sigma, q_0', \delta', F')$ with $2^{\mathcal{O}(n \log n)}$ states such that $L(\mathcal{A}') = \overline{L(\mathcal{A})}$.*

Proof Let $\mathcal{B} = (Q \times \{0, \ldots, 2n\}, \Sigma, (q_0, 2n), \delta_{\mathcal{B}}, F_{\mathcal{B}})$ be the resulting automaton after applying the construction from lemma 3.19 to $\mathcal{A}$ and then dualizing it. To complete our complementation procedure, we want to use the breakpoint construction from section 2.3. Applying this construction straightforward to $\mathcal{B}$ would yield a nondeterministic automaton with $2^{\mathcal{O}(n^2)}$ states. Let us explain (similar to [KV97]) how we can take advantage of the structure of $\mathcal{B}$ to yield an automaton with $2^{\mathcal{O}(n \log n)}$ states.

The states in $\mathcal{B}$ are from $Q \times \{0, \ldots, 2n\}$. For $q \in Q$ and $i \in \{0, \ldots, 2n\}$ let $\mathcal{B}_{(q,i)}$ be the same automaton as $\mathcal{B}$ with the only difference that the initial state is replaced with (q, i). From the construction in lemma 3.19 follows that, in the dualized automaton $\mathcal{B}$, if an $\alpha \in \Sigma^\omega$ is accepted by $\mathcal{B}_{(q,i)}$, then it is also accepted by $\mathcal{B}_{(q,j)}$ for all $j \in \{i, \ldots, 2n\}$.

Therefore, in the breakpoint construction, we only need to consider sets of the form $\{(q_1, i_1), \ldots, (q_m, i_m)\}$ with $q_k \neq q_l$ for $k \neq l$. These sets we call *consistent*. Let $S \subseteq Q \times \{0, \ldots, 2n\}$ with $S = \{(q_1, i_1), \ldots, (q_m, i_m)\}$. We obtain the *consistent version* of S by removing each (q, k) from S iff there exists an $l < k$ with $(q, l) \in S$.

We define $\mathcal{A}'$ as follows.

- $Q' = \{(S, R) \mid S, R \subseteq Q \times \{0, \ldots, 2n\}, R \subseteq S, \text{ and } S, R \text{ are consistent}\}$.
- $q_0' = (\{(q_0, 2n)\}, \emptyset)$.
- Let $(S, R) \in Q'$ and $a \in \Sigma$.

  If $R \neq \emptyset$, then $(S', R' \setminus F_{\mathcal{B}})$ is in $\delta'((S, R), a)$ iff S' is the consistent version of a set exactly satisfying $\bigwedge_{q \in S} \delta(q, a)$, $R' \subseteq S'$, and R' is the consistent version of a set exactly satisfying $\bigwedge_{q \in R} \delta(q, a)$.

  If $R = \emptyset$, then $(S', S' \setminus F_{\mathcal{B}}$ is in $\delta'((S, R), a)$ iff S' is the consistent version of a set exactly satisfying $\bigwedge_{q \in S} \delta(q, a)$.
- $F' = Q' \cap (2^{Q \times \{0, \ldots, 2n\}} \times \{\emptyset\})$.

The correctness of this construction follows from the correctness of the breakpoint construction and the explanations from above.

It remains to show that $\mathcal{A}'$ has $2^{\mathcal{O}(n \log n)}$ states. We can represent every consistent set from $Q \times 2^{\{0, \ldots, 2n\}}$ by a mapping $Q \longrightarrow \{-1, 0, \ldots, 2n\}$. There are $(2n+2)^n$ such mappings and therefore

$$|Q'| \leq (2n+2)^n \cdot (2n+2)^n \in 2^{\mathcal{O}(n \log n)}.$$

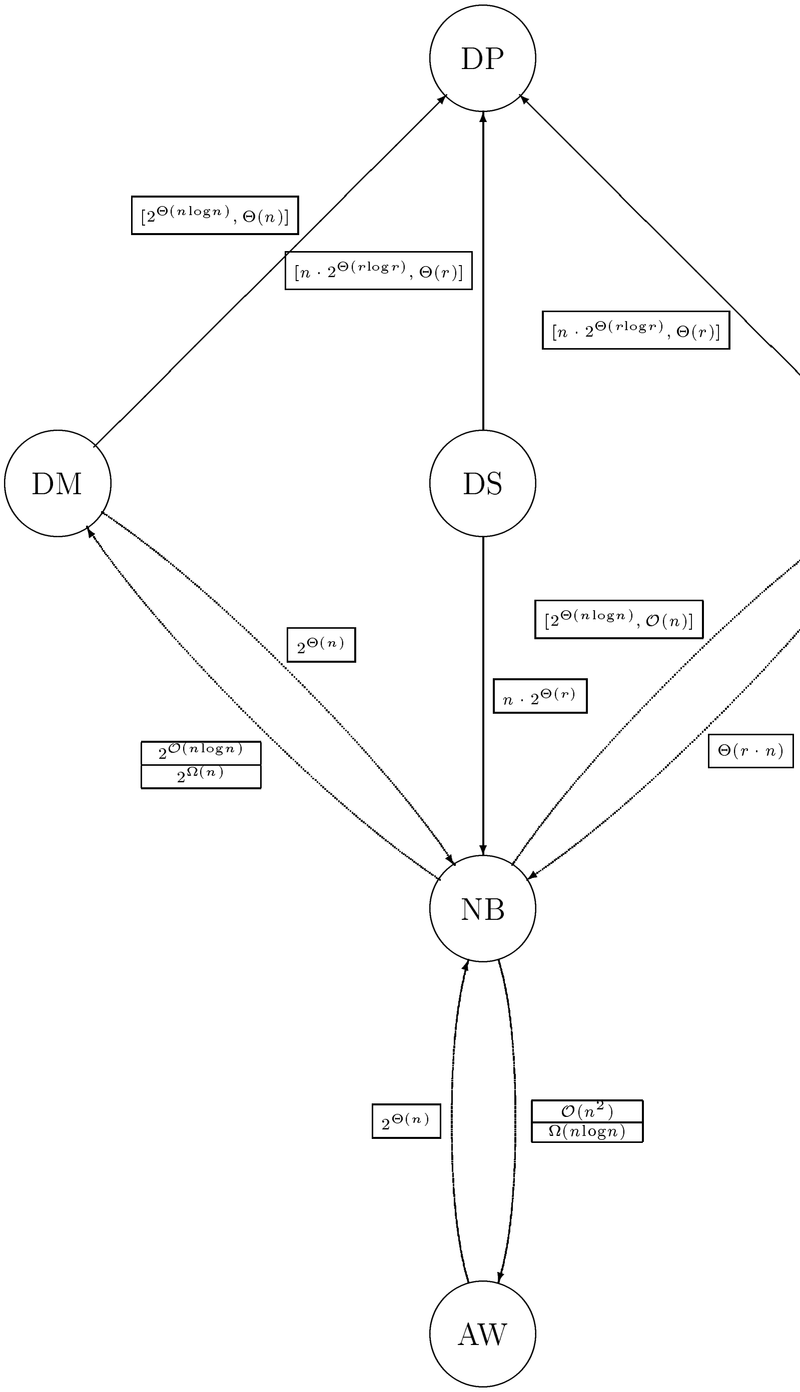
□

3.5 Concluding Remarks

The main types of automata we investigated are DM, DR, DS, DP, NB, and AW. Figure 3.5 gives an overview over the complexity of the transformations discussed in chapters 2 and 3. We used the following notation: n denotes the number of states of the automaton before the transformation. If we start with a Rabin, Streett, or parity automaton, then r denotes the number of pairs. If the resulting automaton is a Muller, Büchi, or weak automaton, then we only give the number of states. If the resulting automaton is a Rabin, Streett, or parity automaton, then we give the number of states and the number of pairs in the following notation *[number of states, number of pairs]*. If an arrow is labeled with only one box, then the transformation is of optimal complexity. If an arrow is labeled with two boxes, then the upper one gives the upper bound for such a transformation and the lower one gives the lower bound for such a transformation. We omitted the transformations that can be done by an adjustment of the acceptance condition (section 3.1).

For most of the transformations we proved that they are of optimal complexity. There are two questions left.

1. Is Safra's construction optimal for the transformation of NB into DM ?
2. Is the rank construction optimal for the transformation of NB into AW ?



Legend: Given automaton: $n\widehat{=}$ number of states, $r\widehat{=}$ number of pairs
Resulting automaton: number of [states, pairs] or number of states

Figure 3.5: Diagram of the Discussed Constructions

Chapter 4

Automata and Logic

Büchi [Büc62] used ω-automata to prove the decidability of the second order theory of one successor (S1S). He worked with nondeterministic Büchi automata. These can be transformed into existential S1S formulas. Vice versa S1S formulas can be transformed into automata. Therefore S1S is decidable and furthermore we can conclude that every S1S formula is equivalent to an existential S1S formula.

In this chapter we focus on the translation from automata into formulas and analyze the effects of different modes of transition functions and different acceptance conditions on the resulting formula.

In section 1.6 we have seen how to complement alternating automata. We will use this result to give two dual characterizations of alternating automata in S1S.

The remainder of this chapter is structured as follows. First we introduce Büchi's logic S1S. In the following we describe alternating, nondeterministic, universal, and deterministic automata by S1S formulas. For every mode of transition function we use an acceptance condition that allows to describe the whole set of ω-regular languages. That is the weak condition for alternating automata, the Büchi condition for nondeterministic automata, the co-Büchi condition for universal automata, and the Rabin condition for deterministic automata. At the end we will give an overview of the effects of the different accepting conditions and modes of transition functions.

4.1 S1S

The second order theory of 1 successor is built up from the following symbols.

x, y, z	(first order) variables for natural numbers
X, Y, Z	(second order) variables for sets of natural numbers
0	natural number zero
$+1$	successor function of the natural numbers
$=, <, \in$	equals, less than, element of
$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$	boolean connectives
$\exists, \forall$	exists, for all

Both types of variables may also occur indexed. We can interpret the sets of natural numbers as predicates, and abbreviate $x \in X$ with Xx. We also use natural abbreviations as $\leq, >, \geq, +2, +3, \ldots$ and $\exists^\omega$, $\forall^\omega$.

Terms and formulas in this formalism are built up with the following rules.

1. Atomic terms: 0 is an atomic term and every variable x is an atomic term.
2. Terms: Atomic terms are terms and if t is a term, then $t+1$ is a term.
3. Atomic formulas: If t and t' are terms, then $t = t'$ and $t < t'$ are atomic formulas and Xt ($t \in X$) is an atomic formula for every variable X.
4. Formulas: Atomic formulas are formulas, and if ϕ and ϕ' are formulas, then $\neg\phi$, $\phi \wedge \phi'$, $\phi \vee \phi'$, $\phi \rightarrow \phi'$, $\phi \leftrightarrow \phi'$, $\exists x(\phi(x))$, $\exists X(\phi(X))$, $\forall x(\phi(x))$, and $\forall X(\phi(X))$ are formulas.

The notation $\phi(x_1, \ldots x_n, X_1, \ldots, X_m)$ for a formula ϕ indicates that only the variables $x_1, \ldots x_n, X_1, \ldots, X_m$ may occur as free (not in the scope of a quantifier) variables in ϕ.

The formulas are interpreted in the structure $(\mathbb{N}, <, +1, 0)$. As models we take tuples of subsets of $\mathbb{N}$. Such a tuple $\underline{\alpha} = (Q_1, \ldots Q_n)$, with $Q_1, \ldots, Q_n \subseteq \mathbb{N}$, is a model of a formula $\phi(X_1, \ldots X_n)$, denoted by $\underline{\alpha} \models \phi(X_1, \ldots X_n)$, if and only if ϕ evaluates to true when we substitute each X_i by Q_i. Thus we regard only formulas with second order variables as free variables.

Examples.

1. $\phi_1(X_0, X_1) = \forall x(X_0 x \rightarrow \exists y(x < y \wedge X_1 y))$. This formula is satisfied if and only if for every number that is contained in X_0 there is a bigger number in X_1.
2. $\phi_2(X_0) = \forall x(X_0 x \rightarrow X_0 x + 1)$. This formula is satisfied if and only if the set X_0 contains with every number also its successor. It follows, if $x \in X_0$, then $y \in X_0$ for all $y > x$.

The number of alternating blocks of the second order quantifiers $\exists$ and $\forall$ gives a measure of the complexity of the formula. In the following definition the notation $\exists\overline{X}$ or $\forall\overline{X}$ indicates that there is a sequence of one or more second order quantifiers of the same type.

Definition 4.1 A Σ_n-formula is of the form

$$\exists\overline{X}_1\forall\overline{X}_2\cdots{}^{\exists}/_{\forall}\overline{X}_n\ \phi(\overline{X}, \overline{X}_1, \ldots, \overline{X}_n)$$

and a Π_n-formula is of the form

$$\forall\overline{X}_1\exists\overline{X}_2\cdots{}^{\forall}/_{\exists}\overline{X}_n\ \phi(\overline{X}, \overline{X}_1, \ldots, \overline{X}_n),$$

where $\phi(\overline{X}, \overline{X}_1, \ldots, \overline{X}_n)$ is a first order formula, called the *kernel* of the formula.

With formulas in S1S we can describe properties of the used structures. When characterizing automata by S1S formulas for example, the described property of the structures is that they belong to a language recognized by the automaton. We call such a property Σ_n-property if it can be described by a formula from Σ_n, we call it Π_n-property if can be described by a formula from Π_n and we call it Δ_n-property if can be described by a formula from Σ_n and by a formula from Π_n.

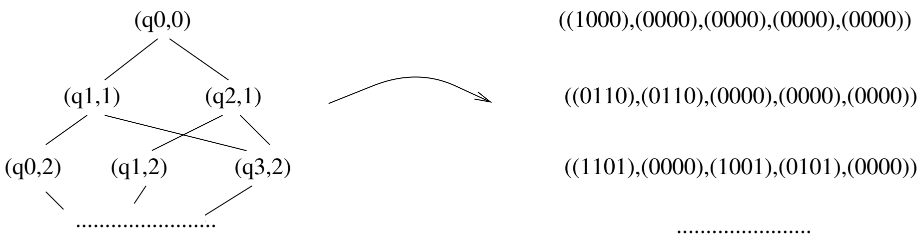


Figure 4.1: Coding of a Run

4.2 Characterizing Automata in S1S

To characterize automata in S1S, we first have to find a correspondence between infinite words α over an alphabet and the models $\underline{\alpha}$ we use in S1S. Then we will construct a formula $\phi_{\mathcal{A}}$ such that the automaton $\mathcal{A}$ accepts a word α if and only if the corresponding tuple $\underline{\alpha}$ is a model of $\phi_{\mathcal{A}}$. In addition, for alternating and deterministic automata, we construct a second formula $\widetilde{\phi}_{\mathcal{A}}$ equivalent to $\phi_{\mathcal{A}}$, such that $\neg\widetilde{\phi}_{\mathcal{A}} = \phi_{\widetilde{\mathcal{A}}}$. The connection between these two formulas is analogue to the connection between an automaton and its dual.

Before we turn to the different types of automata we first explain how we represent infinite words in S1S. There is a natural correspondence between words from Σ^ω and tuples of subsets of $\mathbb{N}$. First, without loss of generality, we can assume that $\Sigma = \{0,1\}^k$. With this convention a word $\alpha \in \Sigma^\omega$ is a sequence of vectors. We can also view this word as k words $\alpha_1, \ldots, \alpha_k \in \{0,1\}^\omega$. If we code α_i with the set X_i, defined by

$$x \in X_i \Leftrightarrow \alpha_i(x) = 1,$$

then the tuple $\underline{\alpha} = (X_1, \ldots, X_k)$ is a unique coding of α. Since the numbers in the sets $X_1, \ldots, X_k$ code the positions in α, we will also refer to natural numbers as positions. Now we can express the fact that a word has a certain letter a at the position x.

Remark 4.2 Let $a = (a_1, \ldots, a_k) \in \Sigma$, and let $O = \{i | a_i = 1\}$ and $Z = \{i | a_i = 0\}$. The formula

$$\mathrm{POS}_a(x, X_1, \ldots, X_k) = \bigwedge_{i \in O} X_i x \wedge \bigwedge_{i \in Z} \neg X_i x$$

is satisfied if and only if the word coded by $X_1, \ldots X_k$ has the letter a at the xth position.

With these preparations we can turn to the characterization of the different types of automata. For simplicity we assume in this section that the transition functions are defined without `true` and `false`.

4.2.1 Alternating Automata

For alternating automata the weak condition suffices to describe all ω-regular languages. Therefore we work with weak automata in this subsection. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an alternating weak automaton with partition $Q = \dot{\bigcup}_{i \in \{1,\ldots,m\}} Q_i$.

First we have to code the runs of an automaton on a word with subsets of $\mathbb{N}$. Let $Q = \{1, \dots, n\}$ with $q_0 = 1$ and let $m = n^2 + n$. We code every level l of a run and the successor relation with a vector $v_l \in \{0,1\}^m$. The first n entries code the active states. This means entry i is 1 iff (i, l) is a vertex of the run. For every $i \in \{1, \dots n\}$ the entries from $i \cdot n + 1$ to $i \cdot n + n$ code the successors of the vertex $(i, l-1)$. This means entry $i \cdot n + j$ is 1 iff (j, l) is a successor of $(i, l-1)$. This idea is illustrated in figure 4.1.

This coding yields an infinite sequence $v_0, v_1, \dots$ of vectors from $\{0,1\}^m$ which can also be represented by $Y_1, \dots, Y_m \subseteq \mathbb{N}$ in the same way as the words from Σ^ω.

Now we can express the fact that a tuple $Y_1, \dots, Y_m \subseteq \mathbb{N}$ is the coding of a run of $\mathcal{A}$ on a word α. For notational simplicity we define for every $i \in Q$ and $a \in \Sigma$

- $S \in \mathcal{S}_i^a \Leftrightarrow S$ exactly satisfies $\delta(i, a)$,
- $S \in \widetilde{\mathcal{S}}_i^a \Leftrightarrow S$ exactly satisfies $\widetilde{\delta}(i, a)$.

Lemma 4.3 *The first order formula* $\mathrm{RUN}_{\mathcal{A}}(X_1, \dots, X_k, Y_1, \dots, Y_m) =$

$$
\begin{aligned}
& (Y_1 0 \wedge \bigwedge_{i=2}^{m} \neg Y_i 0) && (1)\\
\wedge\ & \forall x \Big[\bigwedge_{i=1}^{n} (x > 0 \wedge Y_i x \rightarrow (\bigvee_{j=1}^{n} Y_{j \cdot n + i} x)) && (2)\\
\wedge\ & \bigwedge_{i=1}^{n} (\neg Y_i x \rightarrow (\bigwedge_{j=1}^{n} \neg Y_{i \cdot n + j} x + 1)) && (3)\\
\wedge\ & \bigwedge_{(i,a) \in Q \times \Sigma} (Y_i x \wedge \mathrm{POS}_a(x, X_1, \dots, X_k) && (4)\\
& \qquad \rightarrow \bigvee_{S \in \mathcal{S}_i^a} (\bigwedge_{j \in S} Y_j x + 1 \bigwedge_{j \in S} Y_{i \cdot n + j} x + 1 \wedge \bigwedge_{j \notin S} \neg Y_{i \cdot n + j} x + 1)) \Big] && (5)
\end{aligned}
$$

is satisfied if and only if $Y_1, \dots, Y_m$ *code a run of* $\mathcal{A}$ *on the input coded by* $X_1, \dots, X_k$*.*

Proof We only give an informal description of what the different parts of the formula state.

Part (1) states that $(0,0)$ is a vertex in the run and that it has no predecessors. Part (2) states that every vertex in level x with $x > 0$ must have at least one predecessor. With part (3) we ensure that only vertices that are in level x may have a successor. If $\neg Y_i x$, i.e., vertex (i, x) is not in the run, then (i, x) can not have any successor. Thus part (2) and (3) test if the coded relation is a successor relation. Whether this relation is correct with respect to the input and the transitions is tested in part (4) and (5). For every vertex i in level x and the actual input letter $a = \alpha(x)$ is tested, whether the set S of its successors exactly satisfies $\delta(i, a)$ (for all $j \in S : Y_j x + 1$) and whether the successor relation is coded correctly. $\square$

For a dual run of $\mathcal{A}$ we have an analogue formula.

Lemma 4.4 *The first order formula* $\mathrm{DUALRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_m) =$

$$\begin{aligned}
& (Y_1 0 \wedge \bigwedge_{i=2}^{m} \neg Y_i 0) \\
\wedge \quad & \forall x \Big[\bigwedge_{i=1}^{n} (x > 0 \wedge Y_i x \rightarrow (\bigvee_{j=1}^{n} Y_{j\cdot n+i} x)) \\
\wedge \quad & \bigwedge_{i=1}^{n} (\neg Y_i x \rightarrow (\bigwedge_{j=1}^{n} \neg Y_{i\cdot n+j} x+1)) \\
\wedge \quad & \bigwedge_{(i,a)\in Q\times\Sigma} (Y_i x \wedge \mathrm{POS}_a(x, X_1,\ldots,X_k) \\
& \qquad \rightarrow \bigvee_{S\in\widetilde{\mathcal{S}}_i^a} (\bigwedge_{j\in S} Y_j x+1 \bigwedge_{j\in S} Y_{i\cdot n+j} x+1 \wedge \bigwedge_{j\notin S} \neg Y_{i\cdot n+j} x+1))\Big]
\end{aligned}$$

is satisfied if and only if $Y_1,\ldots,Y_m$ *code a dual run of* $\mathcal{A}$ *on the input coded by* $X_1,\ldots,X_k$.

From the definitions of $\mathcal{S}_i^a$ and $\widetilde{\mathcal{S}}_i^a$ we get the following remark.

Remark 4.5 $\mathrm{RUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_m) = \mathrm{DUALRUN}_{\widetilde{\mathcal{A}}}(X_1,\ldots,X_k,Y_1,\ldots,Y_m)$.

To express the acceptance of a word by an automaton we now have to code the paths of a run. A path through a run $G = (V,E)$ can be viewed as a subgraph $G' = (V',E')$ of G ($V' \subseteq V$ and $E' \subseteq E$) such that every vertex has exactly one successor and every vertex except the initial vertex has exactly one predecessor. We can code a path by $Z_1,\ldots Z_n \subseteq \mathbb{N}$. These Z_i must have the following properties (with i and j ranging over $\{1,\ldots,n\}$).

- $Z_i \subseteq Y_i$ for every i ($V' \subseteq V$).
- $Z_i \cap Z_j = \emptyset$ for every $i \neq j$ (in every level is at most one vertex).
- $\forall x \exists i (x \in Z_i)$ (in every level is at least one vertex).
- For all x with $Z_i x$ and $Z_j x+1$ one has $Y_{i\cdot n+j} x+1$ (the vertex in level $x+1$ is a successor of the vertex in level x).

Formally we have the following lemma.

Lemma 4.6 *Let* $Y_1,\ldots,Y_m \subseteq \mathbb{N}$, *such that there exist* $X_1,\ldots,X_k \subseteq \mathbb{N}$ *satisfying* $\mathrm{RUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_m)$ *or* $\mathrm{DUALRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_m)$. *The first order formula* $\mathrm{PATH}_{\mathcal{A}}(Y_1,\ldots,Y_m,Z_1,\ldots,Z_n) =$

$$\begin{aligned}
& \forall x (\bigwedge_{i\in\{1,\ldots,n\}} Z_i x \rightarrow Y_i x) \\
\wedge \quad & \neg \exists x (\bigvee_{i\neq j} (Z_i x \wedge Z_j x)) \\
\wedge \quad & \forall x (\bigvee_{i\in\{1,\ldots,n\}} Z_i x) \\
\wedge \quad & \forall x (\bigwedge_{i\neq j} (Z_i x \wedge Z_j x+1 \rightarrow Y_{i\cdot n+j} x+1))
\end{aligned}$$

is satisfied iff $Z_1,\ldots,Z_n$ *code a path through the run coded by* $Y_1,\ldots,Y_m$.

The last fact we have to express is that a path satisfies the acceptance condition. In table 1.1 we gave a first order formula expressing the weak acceptance condition. We just have to adapt this formula to the formalism S1S.

Lemma 4.7 *Let $Z_1, \ldots, Z_n \subseteq \mathbb{N}$ be the coding of a path through a run or a dual run. The first order formula*

$$\mathrm{WEAK}_{\mathcal{A}}(Z_1, \ldots, Z_n) = \bigvee_{Q_k \subseteq F} \Big(\exists x (\bigvee_{i \in Q_k} Z_i x) \wedge \forall x (\bigwedge_{i \in \bigcup_{l<k} Q_l} \neg Z_i x) \Big)$$

is satisfied if and only if the path coded by $Z_1, \ldots, Z_n$ satisfies the acceptance condition of $\mathcal{A}$.

Now we can translate an automaton into two equivalent S1S formulas.

Theorem 4.8 *The following two formulas are satisfied if and only if $X_1, \ldots X_k$ is the coding of a word $\alpha \in L(\mathcal{A})$.*

$$\begin{aligned}
\phi_{\mathcal{A}}(X_1, \ldots, X_k) \;=\; \exists Y_1, \ldots, Y_m \;\; & \forall Z_1, \ldots, Z_n(\\
& \mathrm{RUN}_{\mathcal{A}}(X_1, \ldots X_k, Y_1, \ldots, Y_m) \\
& \wedge (\mathrm{PATH}_{\mathcal{A}}(Y_1, \ldots, Y_m, Z_1, \ldots, Z_n) \rightarrow \\
& \quad \mathrm{WEAK}_{\mathcal{A}}(Z_1, \ldots, Z_n))).
\end{aligned}$$

$$\begin{aligned}
\widetilde{\phi}_{\mathcal{A}}(X_1, \ldots, X_k) \;=\; \forall Y_1, \ldots, Y_m \;\; & \exists Z_1, \ldots, Z_n(\\
& \mathrm{DUALRUN}_{\mathcal{A}}(X_1, \ldots X_k, Y_1, \ldots, Y_m) \\
& \rightarrow (\mathrm{PATH}_{\mathcal{A}}(Y_1, \ldots, Y_m, Z_1, \ldots, Z_n) \\
& \quad \wedge \mathrm{WEAK}_{\mathcal{A}}(Z_1, \ldots, Z_n))).
\end{aligned}$$

With remark 4.5 one can easily transform $\neg\phi_{\mathcal{A}}(X_1, \ldots, X_k)$ into $\widetilde{\phi}_{\widetilde{\mathcal{A}}}(X_1, \ldots, X_k)$.

4.2.2 Nondeterministic and Universal Automata

For nondeterministic automata a run just consists of a path. This means the coding of the run and the path can be made in one step. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a nondeterministic Büchi automaton with $Q = \{1, \ldots, n\}$ states. A run can be viewed as in infinite sequence of states $\sigma \in Q^\omega$. In this situation we just need n sets $Y_1, \ldots, Y_n \subseteq \mathbb{N}$ to code the run. The set Y_i contains position x if and only if $\sigma(x) = i$. Let $\mathrm{NONDETRUN}_{\mathcal{A}}(X_1, \ldots, X_k, Y_1, \ldots, Y_n)$ be the first order formula that is satisfied if and only if $Y_1, \ldots, Y_n$ code a run of $\mathcal{A}$ on the input coded by $X_1, \ldots, X_k$. The formula is similar to the description of a run of an alternating automaton but with a simpler structure.

In the nondeterministic case we have to work with the Büchi condition. Again we have to adapt the formula from table 1.1 to S1S.

Lemma 4.9 *Let $Y_1, \ldots, Y_n \subseteq \mathbb{N}$ be the coding of a run of $\mathcal{A}$. The first order formula*

$$\mathrm{BÜCHI}_{\mathcal{A}}(Y_1, \ldots, Y_n) = \exists^\omega x (\bigvee_{i \in F} Y_i x)$$

is satisfied if and only if the path coded by $Y_1, \ldots, Y_n$ satisfies the acceptance condition of $\mathcal{A}$.

The formula characterizing nondeterministic automata in S1S looks as follows.

Lemma 4.10 *The formula*

$$\begin{array}{rcl}\psi_{\mathcal{A}}(X_1,\ldots,X_k) & = & \exists Y_1,\ldots,Y_n(\\ & & \text{NONDETRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_n) \\ & & \wedge\ \text{BÜCHI}_{\mathcal{A}}(Y_1,\ldots,Y_n)).\end{array}$$

is satisfied if and only if $X_1,\ldots,X_k$ code a word $\alpha \in L(\mathcal{A})$.

For universal co-Büchi automata we just have to negate the formula $\psi_{\mathcal{A}}(X_1,\ldots,X_k)$. This follows from the complementation theorem 1.18. Dualizing a universal co-Büchi automaton yields a nondeterministic Büchi automaton. This automaton can be described with $\psi_{\mathcal{A}}(X_1,\ldots,X_k)$. Therefore complementing this formula results in a description for the co-Büchi automaton.

4.2.3 Deterministic Automata

Deterministic automata a special cases of nondeterministic automata. Therefore we characterize runs of deterministic automata in the same way as runs of nondeterministic automata. Let $\mathcal{A} = (Q,\Sigma,q_0,\delta,\Omega)$ be a deterministic Rabin automaton with $\Omega = \{(E_1,F_1),\ldots,(E_r,F_r)\}$. We define

$$\text{DETRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_n) = \text{NONDETRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_n).$$

As for the other types of automata we give a first order formula for the acceptance condition.

Lemma 4.11 *Let $Y_1,\ldots,Y_n \subseteq \mathbb{N}$ be the coding of a run of $\mathcal{A}$. The first order formula*

$$\text{RABIN}_{\mathcal{A}}(Y_1,\ldots,Y_n) = \bigvee_{k\in\{1,\ldots,r\}} \Big(\exists^{\omega}x(\bigvee_{i\in F_k} Z_i x) \wedge \forall^{\omega}x(\bigwedge_{i\in E_k} \neg Z_i x)\Big)$$

is satisfied if and only if the path coded by $Y_1,\ldots,Y_n$ satisfies the acceptance condition of $\mathcal{A}$.

The choice of the Rabin condition is arbitrary. In section 1.3 we have seen that the description of the Muller, Rabin, parity, and Streett condition in first order logic requires a Bool($\exists^{\omega}$) formula.

In a deterministic automaton exists exactly one run for every input. Hence, if there exists a path satisfying the acceptance condition, then all paths do. Thus we can deterministic automata as alternating automata with two dual formulas.

Lemma 4.12 *The formulas*

$$\begin{array}{rcl}\theta_{\mathcal{A}}(X_1,\ldots,X_k) & = & \exists Y_1,\ldots,Y_n(\\ & & \text{DETRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_n) \\ & & \wedge\ \text{RABIN}_{\mathcal{A}}(Y_1,\ldots,Y_n)), \\ \widetilde{\theta}_{\mathcal{A}}(X_1,\ldots,X_k) & = & \forall Y_1,\ldots,Y_n(\\ & & \text{DETRUN}_{\mathcal{A}}(X_1,\ldots,X_k,Y_1,\ldots,Y_n) \\ & & \rightarrow\ \text{RABIN}_{\mathcal{A}}(Y_1,\ldots,Y_n))\end{array}$$

are satisfied if and only if $X_1,\ldots,X_k$ code a word $\alpha \in L(\mathcal{A})$.

Automaton Type	Second Order Complexity	Kernel
Deterministic Rabin	Δ_1	Bool($\exists^\omega$)
Nondeterministic Büchi	Σ_1	$\exists^\omega$
Universal Co-Büchi	Π_1	$\forall^\omega$
Alternating Weak	Δ_2	Bool($\exists$)

Table 4.1: Description of Different Automata Types in Second Order Logic

4.3 Synopsis

From the transformations of the different types of automata into S1S formulas we have seen that the mode of the transition function affects the second order quantifiers needed in the formula. The different acceptance types affect the first order quantifiers in the kernel of the formula. The descriptions of the runs and paths can be neglected because these formulas just contain one first order quantifier. If we focus on types of automata recognizing all ω-regular languages (deterministic Rabin, nondeterministic Büchi, universal co-Büchi, and alternating weak), then the kernel of the formula becomes simpler if we allow more possibilities for the second order quantifiers. Table 4.1 gives an overview of the second order complexity of the formulas and the complexity of their kernel.

Bibliography

[Büc62] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. International Congress on Logic, Method and Philos. Sci. 1960*, pages 1–11, 1962.

[Dav64] M. Davis. Infinite games of perfect information. *Advances in Game Theory*, pages 85–101, 1964.

[DJW97] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games? In *Proc. IEEE, LICS*, 1997.

[GS53] D. Gale and F.M. Stewart. Infinite games with perfect information. *Contributions to the Theory of Games*, pages 245–266, 1953.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, 1979.

[KV97] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symposium on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.

[KV98] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. 30th ACM Symp. on the Theory of Computing*, 1998.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[Mos84] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. *Lecture Notes in Computer Science*, 208:157–168, 1984.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MSS86] D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th ICALP, LNCS 226*, pages 275–283, 1986.

[Mul63] D.E. Muller. Infinite sequences and finite machines. In *Proc. 4th IEEE Symposium on Switching Circuit Theory and Logical design*, pages 3–16, 1963.

[Rab69] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[Saf92] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *Proc. 24th ACM Symp. on the Theory of Computing*, pages 275–282, 1992.

[Str82] R.S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54:121–141, 1982.

[SV89] S. Safra and M. Y. Vardi. On ω-automata and temporal logic. In *Proc. 21th ACM Symp. on the Theory of Computing*, 1989.

[SW74] L. Staiger and K. Wagner. Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regulärer folgenmengen. *Elektron. Informationsverarbeitung und Kybernetik EIK*, 10:379–392, 1974.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 135–191, 1990.

[Tra62] B.A. Trakhtenbrot. Finite automata and the logic of one-place predicates. *Sibirian Mathmatical Journal* (English translation in: *AMS Transl.* 59 (1966) 23-55), 3:103–131, 1962.