Automata and Reactive Systems

Lecture WS 2002/2003
Prof. Dr. W. Thomas
RWTH Aachen

Preliminary version
(Last change March 20, 2003)
Translated and revised by S. N. Cho and
S. Wöhrle
German version by M. Gründler, C. Löding,
and W. Thomas

Note

These lecture notes have not yet undergone a thorough revision. They may contain mistakes of any kind. Please report any bugs you find, comments, and proposals on what could be improved to `skript@i7.informatik.rwth-aachen.de`.

Contents

Introduction

Reactive systems consist of *several components* which *continuously interact* with each other (and, most of the time, do not terminate). In the most basic case such a system would consist of two components, namely a controller program and its environment.

Example 0.1.

1. Signal-box:
   Controller program vs. Railway service (environment)

2. Operating system:
   Operating system program vs. User (environment)

$\boxtimes$

A reactive system is modeled by a two-player-game - Player 0 (*she*) vs. Player 1 (*he*). Infinite games are generated by alternate actions which do not need to be strictly rotational.

In order to decide on a winner, a winning condition for infinite games needs to be formulated (e.g. for Player 0). It's the goal of Player 0 to construct a winning strategy which, for every possible course of actions by Player 1, results in fulfilling the winning condition, and therefore in winning the game for Player 0.

Example 0.2. Modeling of an elevator control for 10 levels

Player 0: Elevator control

Player 1: User

The system state is described by the following properties:

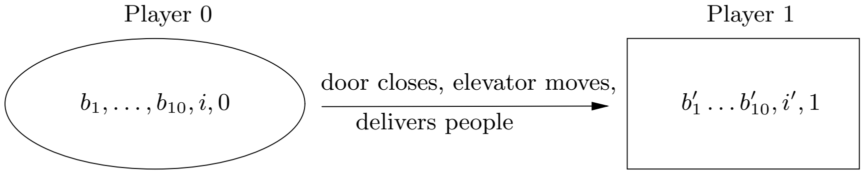
1. A set of level numbers that are requested by pushing a button (either on the respective floor or in the elevator). This set is represented by a bitvector $(b_1, \dots, b_{10})$ (with $b_i = 1 \Leftrightarrow$ level i is requested.)
2. A level number for the position of the elevator ($i \in \{1, \dots, 10\}$).
3. An indicator which $(0|1)$ shows whose turn it is.

State space: Let $\mathbb{B} = \{0, 1\}$. The state space of the system is

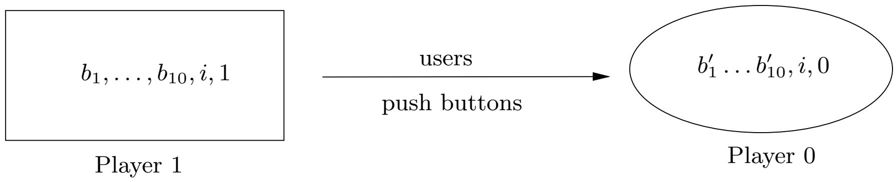
$$\mathbb{Z} = \mathbb{B}^{10} \times \{1, \dots, 10\} \times \{0, 1\}.$$

We note: $|\mathbb{Z}| \cong 20000$ states

Transitions: We define two different kinds of transition. They lead from the 0-states, where it is the turn of Player 0 (elevator controller), to 1-states, where it is the users turn, and vice versa.



with $i \neq i', b'_{i'} = 0, b'_j = b_j$ for $j \neq i'$



with $b_j \leq b'_j$ for every $j \in \{1, \ldots, 10\}$.

State space and transitions define the so called "system graph" or "game graph".

Examples for winning conditions:

1. Every requested floor is served at some time.
2. The elevator does not skip requested floors ($b_i = 1 \rightsquigarrow b_i = 0$), except on the way to level 10 (on the way to the top management :-)
3. On the way to level 10 the elevator stops at most one time.
4. The elevator always returns to level 1.
5. ...

$\boxtimes$

Important questions that need to be answered during the course of this lecture are:

- Can any controller program fulfill all demands? (Then we would have an implementation of a winning strategy.)
- Does a finite memory suffice and how large does it have to be?
- Can we automatically derive a controller program from the system graph and the winning conditions?

Chapter 1

Omega-Automata: Introduction

1.1 Terminology

Σ	denotes a finite *alphabet*.
$\mathbb{B} = \{0, 1\}$	is the Boolean alphabet.
$a, b, c, \ldots$	stand for *letters* of an alphabet.
Σ^*	is the set of finite *words* over Σ.
$u, v, w, \ldots$	stand for finite words.
ϵ	is the empty word.
$\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$	is the set of non-empty words over Σ.
$\alpha, \beta, \gamma, \ldots$	denote ω-*words* or *infinite words* where an ω-word over Σ is a sequence $\alpha = \alpha(0)\alpha(1)\ldots$ with $\alpha(i) \in \Sigma$ for all $i \in \mathbb{N}$.
Σ^ω	is the set of infinite words over Σ.
$\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$	
$U, V, W, \ldots$	denote sets of finite words ($*$-*languages*) $\subseteq \Sigma^*$.
$K, L, M, \ldots$	denote sets of infinite words (ω-*languages*) $\subseteq \Sigma^\omega$.

We write $u \cdot v$ or simply uv for the concatenation of the words u and v. Similarly, the concatenation of the word u and the ω-word α is the ω-word $u\alpha$.

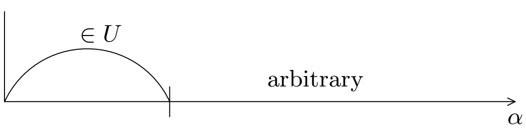
The concatenation of two languages is defined likewise:

$$\begin{aligned} U \cdot V &= \{uv \mid u \in U, v \in V\} \\ U \cdot L &= \{u\alpha \mid u \in U, \alpha \in L\} \end{aligned}$$

We consider three different transitions from a language $U \subseteq \Sigma^*$ to an ω-language, namely to $U \cdot \Sigma^\omega$, U^ω, and $\lim U$.

1. $U \cdot \Sigma^\omega := \{\alpha \in \Sigma^\omega \mid \alpha = u\beta \text{ with } u \in U, \beta \in \Sigma^\omega\}$

   Visualization:



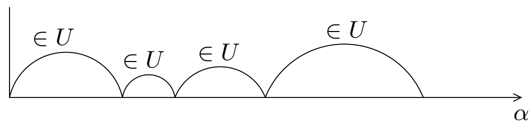
Example 1.1. Let $U_1 = 0110^* + (00)^+$. We obtain

$$U_1 \cdot \Sigma^\omega = \{\alpha \in \Sigma^\omega \mid \alpha \text{ starts with 00 or 011}\}$$

⊠

2. $U^\omega := \{\alpha \in \Sigma^\omega \mid \alpha = u_0u_1u_2\ldots \text{ with } u_i \in U\}$

   Visualization:



   Notice that $U^\omega = (U \setminus \{\epsilon\})^\omega$

Example 1.2. Let $\Sigma = \mathbb{B}$, U is given by the regular expression

$$0110^* + 00$$

Then U^ω contains the word

$$\alpha = 0001100110000000\ldots$$

Another word in U^ω

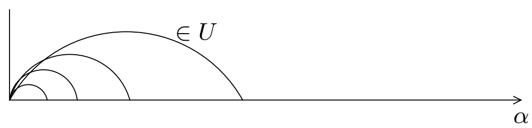
$$\alpha = 01100110001100001\ldots$$

⊠

3. $\lim U$ (or $\vec{U}$) $:= \{\alpha \in \Sigma^\omega \mid$ there exist infinitely many i with $\alpha(0)\ldots\alpha(i) \in U\}$.

   The expression "there exist infinitely many i with $\alpha(0)\ldots\alpha(i) \in U$" can also be written in short as "$\exists^\omega i\ \alpha[0,i] \in U$", where $\alpha[i,j] = \alpha(i)\ldots\alpha(j)$.

   Visualization:



Example 1.3. Claim: $\lim U_1$ contains just the two ω-words $0110000\ldots$ (in short 0110^ω) and $0000000\ldots$ (in short 0^ω).

The word 0110^ω is an element of $\lim U_1$, since $011, 0110, 011000, \cdots \in U_1$. The word 0^ω is an element of $\lim U_1$, since $00, 0000, 000000, \cdots \in U_1$.

Now, let $\alpha \in \lim U_1$, i.e. there exist infinitely many α-prefixes in U_1. Now look for the first α-prefix v in U_1.

Case 1: $v = 011$. Then all longer prefixes in U_1 have to be of the form 0110^*, thus $\alpha = 0110^\omega$.

Case 2: $v = 00$. Then every extension of v in U_1 has to be of the form $(00)^*$, thus $\alpha = 0^\omega$.

⊠

1.2 Büchi Automata

Definition 1.4. A *Büchi automaton* (to put it more precisely, a finite Büchi automaton) is of the form

$$\mathfrak{A} = (Q, \Sigma, q_0, \delta \text{ or } \Delta, F)$$

with a finite set of states Q, input alphabet Σ, initial state $q_0 \in Q$, a deterministic (and hence complete) transition function $\delta : Q \times \Sigma \to Q$ or a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set of final states F. In the case of δ we have a *deterministic Büchi automaton*, in the case of Δ a *nondeterministic Büchi automaton*.

Definition 1.5. (Run of a Büchi Automaton)

1. Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be a nondeterministic Büchi automaton.

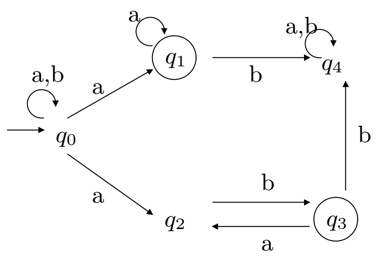
   A *run* of $\mathfrak{A}$ on α is a sequence of states $\rho = \rho(0)\rho(1)\cdots$ with $\rho(0) = q_0$ and $(\rho(i), \alpha(i), \rho(i+1)) \in \Delta$ for $i \geq 0$.

2. Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ be a deterministic Büchi automaton. As it is usual, we expand δ to $\delta^* : Q \times \Sigma^* \to Q$ by adding $\delta^*(q, \epsilon) = q$ and $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$.

   The unambiguous run of $\mathfrak{A}$ on α is the sequence of states ρ with $\rho(0) = q_0$, $\rho(1) = \delta(q_0, \alpha(0))$, $\rho(2) = \delta^*(q_0, \alpha(0)\alpha(1))$, in general $\rho(i) = \delta^*(q_0, \alpha(0) \ldots \alpha(i-1))$.

Deterministic Büchi automata can be seen as special cases of nondeterministic ones where $(p, a, q) \in \Delta \Leftrightarrow \delta(p, a) = q$. To simplify our notation, we just write $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ for a Büchi automaton if we don't care whether it is deterministic or not, and just speak of a Büchi automaton in this case.

Example 1.6. Given the following automaton $\mathfrak{A}_0$:



with $F = \{q_1, q_3\}$ and the ω-word $\alpha = abbaabababa\ldots$, some of the possible runs of $\mathfrak{A}_0$ on α are:

a	b	b	a	a	b	a	b	a	b	...
q_0	q_0	q_0	q_0	q_0	q_0	q_0	q_2	q_3	q_2	$q_3 \cdots$
q_0	q_0	q_0	q_0	q_1	q_1	q_4	q_4	q_4	q_4	$q_4 \cdots$
q_0	q_0	q_0	q_0	q_0	q_2	q_3	q_2	q_3	q_2	$q_3 \cdots$

⊠

Definition 1.7. Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be a Büchi automaton. We say, that

$$\mathfrak{A} \textit{ accepts } \alpha \Leftrightarrow \text{ex. a run } \rho \text{ of } \mathfrak{A} \text{ on } \alpha \text{ with } \exists^\omega i\ \rho(i) \in F.$$

Notice, that, for a deterministic Büchi automaton, the unambiguous run ρ has to fulfill this condition.

Definition 1.8. Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be a Büchi automaton. Then

$$L(\mathfrak{A}) \quad := \{\alpha \in \Sigma^\omega \mid \mathfrak{A} \text{ accepts } \alpha\}$$

is the *ω-language recognized* by $\mathfrak{A}$. An ω-language $L \subseteq \Sigma^\omega$ is *Büchi recognizable* (*deterministically Büchi recognizable*), if a corresponding Büchi automaton (deterministic Büchi automaton) $\mathfrak{A}$ with $L = L(\mathfrak{A})$ exists.

Example 1.9. Let $\mathfrak{A}_0$ be the nondeterministic Büchi automaton over $\Sigma = \{a, b\}$ as defined in example 1.6.

$$L(\mathfrak{A}_0) \quad = \quad \{\alpha \in \Sigma^\omega \mid \text{ from some point in } \alpha \text{ onwards, there is only the letter } a \text{ or the sequence } ab\ \}.$$

⊠

1.3 Elementary Constructions of Omega-Automata

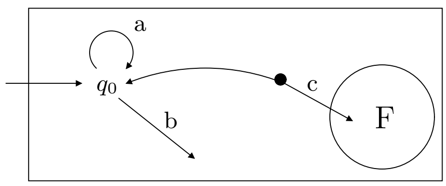
We will now, for the case of $U \subseteq \Sigma^*$ being regular, specify ω-automata for the ω-languages U^ω and $\lim U$.

Theorem 1.10. *$U \subseteq \Sigma^*$ is regular $\Rightarrow$ a) U^ω is Büchi recognizable*
b) $\lim U$ is deterministically Büchi recognizable

Proof

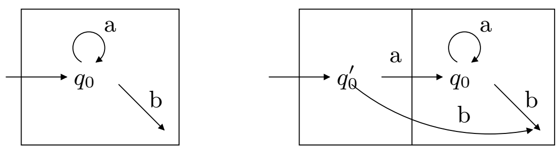
a) Consider an NFA $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ that recognizes U.

Idea: Instead of a transition to F, allow a return to q_0 and declare q_0 as a final state. But there will be a problem with this idea if a return to q_0 is already allowed in the original NFA.



Preparation: Transform $\mathfrak{A}$ into a standardized NFA $\mathfrak{A}'$ that has no transitions to the initial state.

Construction: Introduce a new initial state q_0' and add a transition (q_0', a, q) for every transition (q_0, a, q). The final states remain untouched. But if q_0 is a final state, add q_0' to F.



The construction of the Büchi automaton for U^ω for a given standardized NFA $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ is done in two steps:

- For every $q' \in F$ replace every transition (q, a, q') with a new transition (q, a, q_0).
- Fix the set of final states of the Büchi automaton to $\{q_0\}$.

We thereby obtain the Büchi automaton $\mathfrak{B}$. The automaton $\mathfrak{B}$ accepts $\alpha \Leftrightarrow (+)$ there exists a run of $\mathfrak{B}$ on α that enters q_0 infinitely often, e.g. after the segments $u_0, u_1, \ldots$. According to the construction, $u_i \in U$ holds and therefore $\alpha \in U^\omega$.

Conversely, let $\alpha \in U^\omega, \alpha = u_0 u_1 u_2 \ldots$ with $u_i \in U$. Then $\mathfrak{A} : q_0 \xrightarrow{u_i} F$ holds and according to the construction $\mathfrak{B} : q_0 \xrightarrow{u_i} q_0$. Thus there exists a run that fits $(+)$, and consequently $\mathfrak{B}$ accepts the ω-word α.

b) Let U be recognized by the DFA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$. Use $\mathfrak{A}$ as a deterministic Büchi automaton, now called $\mathfrak{B}$.

$$\begin{aligned} \mathfrak{B} \text{ accepts } \alpha \quad & \overset{\text{Def}}{\Longleftrightarrow} \text{The unambiguous run of } \mathfrak{B} \text{ on } \alpha \text{ enters } F \text{ infinitely often.} \\ & \Longleftrightarrow \exists^\omega i : \quad \mathfrak{A} \text{ reaches a state in } F \text{ after } \alpha(0) \ldots \alpha(i) \\ & \Longleftrightarrow \exists^\omega i : \quad \alpha(0) \ldots \alpha(i) \in U \text{ (according to the def. of } \mathfrak{A}) \\ & \Longleftrightarrow \alpha \in \lim U \end{aligned}$$

□

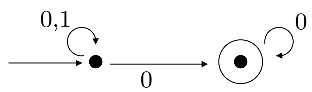
Note that the converse of Theorem 1.10(b) also holds. Every ω-language recognized by a deterministic Büchi automaton is of the form $\lim U$ for a regular language U.

Theorem 1.11. *There is an ω-language which is Büchi recognizable but not recognizable by any deterministic Büchi automaton.*

Proof Consider the language

$$L = \{\alpha \in \mathbb{B}^\omega \mid \text{from some point in } \alpha \text{ onwards only zeros}\},$$

thus $L = (0 + 1)^* 0^\omega$. A matching automaton could look like this:



Assume: L is recognized by det. Büchi Automata $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$. Then the following holds:

$\mathfrak{A}$ on 0^ω infinitely often enters final states, after 0^{n_1} for the first time. $\mathfrak{A}$ on $0^{n_1}10^\omega$ infinitely often enters final states, before the last 1 for the first time, and after processing $0^{n_1}10^{n_2}$ a second time. $\mathfrak{A}$ on $0^{n_1}10^{n_2}10^\omega$ infinitely often enters final states, before the last 1 for the first time, before the second 1 a second time, and a third time after processing $0^{n_1}10^{n_2}10^{n_3}$.

Continuing this we obtain an ω-word $0^{n_1}10^{n_2}10^{n_3}10^{n_4}\ldots$ which causes $\mathfrak{A}$ to enter final states after each 0-block. $\mathfrak{A}$ therefore accepts this ω-word, although it contains infinitely many 1s. Contradiction. □

1.4 Characterization of Büchi Recognizable Omega-Languages

Theorem 1.12. (Characterization of the Büchi recognizable ω-languages) $L \subseteq \Sigma^\omega$ *is Büchi recognizable* $\Leftrightarrow$ *L has a description of the form of*

$$L = \bigcup_{i=0}^{n} U_i \cdot V_i^\omega \text{ with } U_1, V_1, \ldots, U_n, V_n \subseteq \Sigma^* \text{ regular.}$$

Proof

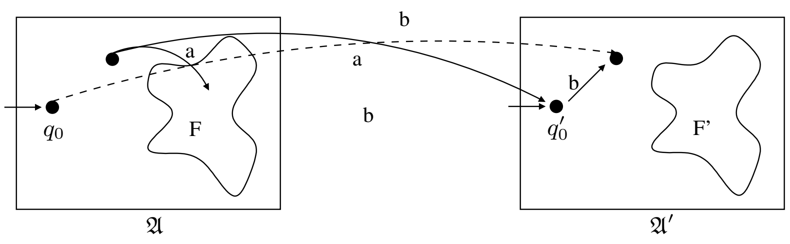
$\Leftarrow$ It suffices to show:

1. $U \subseteq \Sigma^*$ regular $\Rightarrow U^\omega$ Büchi recognizable.
2. $U \subseteq \Sigma^*$ regular, $K \subseteq \Sigma^\omega$ Büchi recognizable $\Rightarrow U \cdot K$ Büchi recognizable.
3. $L_1, L_2 \subseteq \Sigma^\omega$ Büchi recognizable $\Rightarrow L_1 \cup L_2$ Büchi recognizable.

For 1: Use Theorem 1.10(a).

For 2: Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be an NFA, which recognizes the language U, and let $\mathfrak{A}' = (Q', \Sigma, q_0', \Delta', F')$ be a Büchi automaton, which recognizes the language K. Now, construct a Büchi automaton $\mathfrak{B} = (Q \uplus Q', \Sigma, q_0, \Delta_{\mathfrak{B}}, F')$ for $U \cdot K$, where $\Delta_{\mathfrak{B}}$ contains, in addition to the transitions of Δ and Δ', the following:

- for every transition (p, a, q) with $q \in F$ the transition (p, a, q_0')
- if $q_0 \in F$, for every transition $(q_0', a, q') \in \Delta'$ the transition (q_0, a, q').



For 3: Merge the Büchi automata $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ and $\mathfrak{A}' = (Q', \Sigma, q_0', \Delta', F')$ into a single automaton $\mathfrak{B} = (Q \dot\cup Q', \Sigma, q_0, \Delta_{\mathfrak{B}}, F)$, where $\Delta_{\mathfrak{B}}$ contains all transitions of Δ, Δ', as well as (q_0, a, q') for $(q_0', a, q') \in \Delta'$. In doing so, we assume w.l.o.g. that there are no transitions to q_0 in $\mathfrak{A}$.

$\Rightarrow$ Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be a Büchi automaton. Set $\mathfrak{A}_{qq'} = (Q, \Sigma, q, \Delta, \{q'\})$. Let $U_{qq'} \subseteq \Sigma^*$ be the language that is recognized by the NFA $\mathfrak{A}_{qq'}$. Notice that, consequently, $U_{qq'}$ is regular.

$\mathfrak{A}$ accepts $\alpha \Leftrightarrow$ ex. $q \in F$ which makes a segmentation of α into $\alpha = u_0 u_1 u_2 \ldots$, with $u_0 \in U_{q_0 q}$, $u_1 \in U_{qq}$, $u_2 \in U_{qq}, \ldots$, possible. Therefore the following holds.

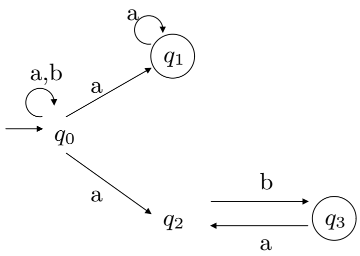
$$\mathfrak{A} \text{ accepts } \alpha \Leftrightarrow \text{ ex. } q \in F \text{ with } \alpha \in U_{q_0 q} \cdot U_{qq}^\omega \Leftrightarrow \alpha \in \bigcup_{q \in F} U_{q_0 q} \cdot U_{qq}^\omega$$

$\square$

Definition 1.13. An *ω-regular expressions* is of the form $r_1 s_1^\omega + \cdots + r_n s_n^\omega$ with standard regular expressions $r_1, s_1, \ldots, r_n, s_n$.

The meaning (semantics) of those expressions is defined in a manner analogous to standard regular expressions. We of course stipulate that for an expression s, which defines the language $U \subseteq \Sigma^*$, the expression s^ω defines the ω-language U^ω.

Example 1.14. Büchi automaton:



Defining ω-regular expression:

$$(a+b)^* a^\omega + (a+b)^*(ab)^\omega$$

⊠

From Theorem 1.12 we obtain:

Corollary 1.15. *An ω-language is Büchi recognizable iff it can be defined by an ω-regular expression.*

Definition 1.16. An ω-language L is called *regular* if it is definable by an ω-regular expression (or if it is nondeterministically Büchi recognizable).

Remark 1.17.

a) *Every nonempty regular ω-language contains an ω-word which is eventually periodic (in the form $uvvvvv\ldots$, with u, v finite).*

b) *A set $\{\alpha\}$ with exactly one element is regular $\Leftrightarrow$ α eventually periodic.*

Proof

a) Let $L = \bigcup_{i=1}^n U_i V_i^\omega$ be regular and nonempty. Then, for a suitable i, $U_i \cdot V_i^\omega \neq \emptyset$ holds. Therefore there are words $u \in U_i, v \in V_i$ with $v \neq \epsilon$. So $uvvv\ldots \in L$ is eventually periodic.

b) "$\Rightarrow$" is clear because of a)
"$\Leftarrow$" Let $\alpha = uvvvv\ldots$. Then $\{\alpha\} = \{u\} \cdot \{v\}^\omega$ holds, where $\{u\}$ and $\{v\}$ are regular.

□

Theorem 1.18. (Nonemptiness Problem) *The nonemptiness Problem for Büchi automata (with state set Q and transition relation Δ) is solvable in time $O(|Q| + |\Delta|)$.*

Proof Let $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ be a Büchi automaton. Define $E = \{(p,q) \in Q \times Q \mid \exists a \in \Sigma : (p,a,q) \in \Delta\}$ and call $G := (Q, E)$ the *transition graph* of $\mathfrak{A}$.

Therefore $L(\mathfrak{A}) \neq \emptyset$ iff in the transition graph there is a path from q_0 to a final state q, from which there is a path back to q.

This is the case iff in the transition graph of $\mathfrak{A}$ there is a strongly connected component (SCC) C such that C contains a final state and is reachable by a path from q_0.

Nonemptiness test

1. Apply depth-first search from q_0 in order to determine the set Q_0 of states reachable from q_0.

2. Apply Tarjan's SCC-algorithm to list all SCC's over Q_0, and check each SCC for the containment of a final state.

3. If such an SCC is encountered, answer $L(\mathfrak{A}) \neq \emptyset$, otherwise $L(\mathfrak{A}) = \emptyset$.

Items 1 and 2 require both time $O(|Q| + |\Delta|)$. (For details turn to Cormen, Leiserson, Rivest: Introduction to Algorithms.) □

1.5 Closure Properties of Büchi Recognizable Omega-Languages

We showed (in the exercises) that the union $L_1 \cup L_2$ of two Büchi recognizable ω-languages L_1, L_2 is in turn Büchi recognizable.

We will now verify closure under intersection:

Theorem 1.19. *The intersection $L_1 \cap L_2$ of two Büchi recognizable ω-languages L_1, L_2 is again Büchi recognizable.*

Proof Assume L_i is recognized by the Büchi automaton $\mathfrak{A}_i = (Q_i, \Sigma, q_{i0}, \Delta_i, F_i)$ for $i = 1, 2$.

First Idea: Form the product automaton

$$(Q_1 \times Q_2, \Sigma, (q_{10}, q_{20}), \Delta, F_1 \times F_2)$$

where $((p, q), a, (p', q')) \in \Delta$ iff $(p, a, p') \in \Delta_1$ and $(q, a, q') \in \Delta_2$.

Problem: We cannot assume that the final states in the two runs of $\mathfrak{A}_1, \mathfrak{A}_2$ are visited simultaneously

Solution: Repeatedly do the following steps

1. Wait for a final state $p \in F_1$ in the first component.

2. When a $p \in F_1$ is encountered, wait for a final state $q \in F_2$ in the second component.

3. When a $q \in F_2$ is encountered, signal "cycle completed" and go back to 1.

Hence work with the state space $Q_1 \times Q_2 \times \{1, 2, 3\}$.

Form the refined product automaton

$$\mathfrak{A} = (Q_1 \times Q_2 \times \{1, 2, 3\}, \Sigma, (q_{10}, q_{20}, 1), \Delta', Q_1 \times Q_2 \times 3)$$

with the following transitions in Δ', in each case assuming $(p, a, p') \in \Delta_1$ and $(q, a, q') \in \Delta_2$:

- $((p, q, 1), a, (p', q', 1))$ if $p' \notin F_1$

- $((p, q, 1), a, (p', q', 2))$ if $p' \in F_1$

- $((p, q, 2), a, (p', q', 2))$ if $q' \notin F_2$

- $((p, q, 2), a, (p', q', 3))$ if $q' \in F_2$
- $((p, q, 3), a, (p', q', 1)$

Then a run of $\mathfrak{A}$ simulates two runs ρ_1 of $\mathfrak{A}_1$ and ρ_2 of $\mathfrak{A}_2$: It has infinitely often 3 in the third component iff ρ_1 visits infinitely often F_1 and ρ_2 infinitely often F_2.

□

Note that up to now we do not know a general construction for the complement of a Büchi recognizable language.

1.6 Generalized Büchi Automata

Definition 1.20. (Generalized Büchi Automaton) A *generalized Büchi automaton* is of the form $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F_1, \ldots, F_k)$ with final state sets $F_1, \ldots, F_k \subseteq Q$. A run is successful if the automaton visits each of the sets F_i (i.e. the automaton enters a state in each F_i) infinitely often.

Remark 1.21. *For any generalized Büchi automaton one can construct an equivalent Büchi automaton.*

We will first give a proof idea before dealing with the exact construction: Work with the state set $Q \times \{1, \ldots, k, k+1\}$. For $i, \ldots, k$ a state (q, i) means "wait for visit to F_i". After visiting F_k proceed to $i = k + 1$ ("cycle completed") and go back to $i = 1$. Consequently, declare $Q \times \{k + 1\}$ as the set of final states.

Proof (Detailed construction)

Given a generalized Büchi automaton $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F_1, \ldots, F_k)$, construct the Büchi automaton

$$\mathfrak{A}' = (Q \times \{1, \ldots, k + 1\}, \Sigma, (q_0, 1), \Delta', Q \times \{k + 1\})$$

with the following transitions in Δ' (assuming that $(p, a, q) \in \Delta$):

- $((p, i), a, (q, i))$, if $i \leq k$ and $q \notin F_i$
- $((p, i), a, (q, i + 1))$, if $i \leq k$ and $q \in F_i$
- $((p, k + 1), a, (q, 1))$

□

1.7 Exercises

Exercise 1.1. Specify Büchi automata, which recognize the following ω-languages over $\Sigma = \{a, b, c\}$:

(a) The set of $\alpha \in \Sigma^\omega$, in which abc appears as an infix at least once.

(b) The set of $\alpha \in \Sigma^\omega$, in which abc appears as an infix infinitely often.

(c) The set of $\alpha \in \Sigma^\omega$, in which abc appears as an infix only finitely often.

Exercise 1.2. Find ω-regular expressions, which define the ω-languages in Exercise 1.1.

Exercise 1.3. Let the NFA $\mathcal{A}$ recognize the language $U \subseteq \Sigma^*$. Verify both inclusions of the equation $L(\mathcal{A}) = \lim U$.

Exercise 1.4. Prove or disprove the following equations (for $U, V \subseteq \Sigma^+$):

(a) $(U \cup V)^\omega = U^\omega \cup V^\omega$
(b) $\lim(U \cup V) = \lim U \cup \lim V$
(c) $U^\omega = \lim(U^+)$
(d) $\lim(U \cdot V) = U \cdot V^\omega$

Exercise 1.5. Consider the ω-language L over $\Sigma = \{a, b\}$ which is defined by the ω-regular expression $(a+b)^*a^\omega + (a+b)^*(ab)^\omega$. (see Example 1.14). Show that L cannot be described in the form $U \cdot V^\omega$, with $U, V \subseteq \Sigma^*$ regular (therefore one needs the union operation in order to generate all Büchi recognizable ω-languages).

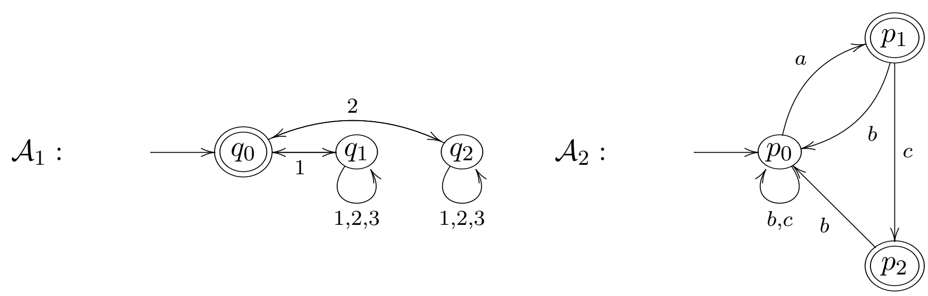
Hint: Assume that L is of the form $U \cdot V^\omega$ and consider the words in V.

Exercise 1.6. Let L_1, L_2 be the ω-languages recognized by Büchi automata $\mathcal{A}_1 = (Q_1, \Sigma, q_0, \Delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0, \Delta_2, F_2)$, respectively. Show (using some necessary assumptions on the structure of $\mathcal{A}_1$ and $\mathcal{A}_2$) that the language $L_1 \cup L_2$ is again Büchi recognizable by

(a) constructing a Büchi automaton $\mathcal{B}_1$ with state set $Q_1 \cup Q_2$, but without ϵ-transitions, that accepts $L_1 \cup L_2$.

(b) constructing a product Büchi automaton $\mathcal{B}_2$ with state set $Q_1 \times Q_2$ accepting $L_1 \cup L_2$. Show in particular how to combine the Büchi acceptance conditions of both automata $\mathcal{A}_1$ and $\mathcal{A}_2$ into a single one for $\mathcal{B}_2$.

Exercise 1.7. Given the following Büchi automata,



specify ω-regular expressions which define the ω-languages that are recognized by $\mathcal{A}_1$ and $\mathcal{A}_2$.

Exercise 1.8. Investigate the following question (whose answer is yet to be found): Is there an algorithm that, for a given Büchi automata $\mathcal{A}$ over the alphabet Σ, decides whether $L(\mathcal{A})$ is of the form U^ω for a regular language $U \subseteq \Sigma^*$?

Chapter 2

Temporal Logic and Model Checking

In this chapter we are going to discuss an automata theoretic approach to the model checking problem. The theory of Büchi automata, which we treated in the last chapter, will serve us in two ways:

On the one hand, Büchi automata obviously represent a model for systems with infinite runs. Such systems can be modeled by Büchi automata which have accepting runs only, i.e. every state is a final state.

On the other hand, Büchi automata can be used to specify properties and constraints for infinite state sequences, since they can be encoded by ω-words. For our purposes we need a logical language which can specify systems and be translated into Büchi automata as well.

Hence the model checking problem can be reduced to comparing two Büchi automata. This is where we will be using methods from the previous chapter.

2.1 The Model-Checking Problem and Sequence Properties

Starting the technical treatment, we will first recall the informal formulation of the model-checking problem from the introduction:

> Given a system *Sys* and a specification *Spec* on the runs of the system, decide whether *Sys* satisfies *Spec*.

There was an early example for this problem in the first lecture:

Example 2.1. *Sys* = MUX (Mutual exclusion) protocol, modeled by a transition system over the state-space $\mathbb{B}^5$.

```
Process 1:  Repeat
00:  non-critical section 1
01:  wait unless turn = 0
10:  critical section 1
11:  turn := 1
Process 2:  Repeat
00:  non-critical section 2
```

```
01:  wait unless turn = 1
10:  critical section 2
11:  turn := 0
```

A state is a bit-vector (line no. of process 1, line no. of process 2, value of turn). The system starts with the initial state (00000).

Spec = "a state (1010b) is never reached", and "always when a state ($01bcd$) is reached, then later a state ($10b'c'd'$) is reached" (similarly for states ($bc01d$), ($b'c'10d'$)). ⊠

This example is going to be used to introduce transition systems and system specification. After that, we will develop the general approach as follows:

1. **Kripke structures as system models:** Kripke structures provide a mathematical framework for transition systems. Their states give information about the properties of a system.
2. **Simple specifications:** We are going to model a simple example using Kripke structures and common language. Doing so we will see the need for a formal system specification language.
3. **Linear-time temporal logic LTL** is the logic we choose to set system constraints. It will enable us to express grammatical operators of common language.
4. **The automata theoretic approach to model-checking:** Having introduced the necessary tools we will sketch a way to solve the model-checking problem using (Büchi) automata theory.
5. **Translation of temporal logic formulas to Büchi automata:** At this stage we will lack just one method: bridging the gap between LTL and Büchi automata.

2.2 Kripke Structures

Kripke structures are a general framework for the case where state properties $p_1, \ldots, p_n$ are considered.

Definition 2.2. A *Kripke structure* over $p_1, \ldots, p_n$ has the form $\mathcal{M} = (S, R, \lambda)$ with

- a finite set S of "states"
- a "transition relation" $R \subseteq S \times S$
- a "labeling function" $\lambda : S \to 2^{\{p_1,\ldots,p_n\}}$, associating with $s \in S$ the set of those p_i which are assumed true in s

Usually we write a value $\lambda(s)$ as a bit vector $(b_1, \ldots, b_n)$ with $b_i = 1$ iff $p_i \in \lambda(s)$.

In a *pointed Kripke structure*, a state s is declared as initial; we write $(\mathcal{M}, s)$. All runs start in s.

Example 2.3. (MUX Protocol) State space: $S = \mathbb{B}^5$. We use the state properties

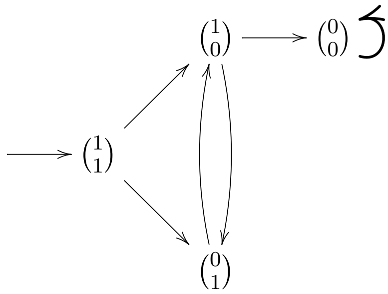
- p_1, p_2 for "being in wait instruction before the critical section of P_1, or P_2 respectively",

- p_3, p_4 for "being in critical section of P_1, respectively P_2".

The transition relation R is as defined by the transitions of the protocol. Example value of the label function: $\lambda(01101) = \{p_1, p_4\}$ [= (10010)]. ⊠

We have another example which we will use again and again to familiarize ourselves with the concept:

Example 2.4. (A toy example) Consider a system over two properties p_1 and p_2.



A *path* through a pointed Kripke structure $(\mathcal{M}, s)$ with $\mathcal{M} = (S, R, \lambda)$ is a sequence $s_0, s_1, s_2, \ldots$ where $s_0 = s$ and $(s_i, s_{i+1}) \in R$ for $i \geq 0$.

The corresponding *label sequence* is the ω-word over $\mathbb{B}^n$: $\lambda(s_0)\lambda(s_1)\lambda(s_2)\ldots$, for instance

$$\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\cdots$$

over the alphabet $\mathbb{B}^2 = \left\{\binom{0}{0}, \binom{1}{0}, \binom{0}{1}, \binom{1}{1}\right\}$.

We hereby obtain an ω-language that contains the corresponding label sequences of all possible runs of the Kripke structure. ⊠

Now that we have introduced Kripke structures, we will state the model-checking problem more precisely:

> Given a pointed Kripke structure over $p_1, \ldots, p_n$ and a condition ϕ on ω-words over $\mathbb{B}^n$, does every label sequence of $(\mathcal{M}, s)$ satisfy ϕ?

For the MUX protocol consider the following conditions ϕ:

- "Never p_3, p_4 are simultaneously true" which means for any label sequence: "there is no letter $(b_1, b_2, 1, 1)$".
- "Always when p_1 is true then sometime later p_3 is true" which means for any label sequence "when a letter $(1, b_2, b_3, b_4)$ occurs, later on a letter $(b_1, b_2, 1, b_4)$ occurs".

Basic sequence properties We consider state properties p_1, p_2. Label sequences are then ω-words over the alphabet $\mathbb{B}^2 = \left\{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\right\}$. We consider the following properties of label sequences over p_1 and p_2:

Guaranty property: "Sometime p_1 becomes true."

Safety property: "Always p_1 is true."

Periodicity property: "Initially p_1 is true, and p_1 is true precisely at every third moment."
Example sequence: $\binom{1}{0}\ \binom{0}{0}\ \binom{0}{1}\ \binom{1}{1}\ \binom{0}{1}\ \binom{0}{0}\ \binom{1}{0}\ \cdots$

Obligation property: "Sometime p_1 is true but p_2 is never true."

Recurrence property: "Again and again, p_1 is true."

Request-Response property: "Always when p_1 is true, p_2 will be true sometime later."

Until property: "Always when p_1 is true, sometime later p_1 will be true again and in the meantime p_2 is always true."

Fairness property: "If p_1 is true again and again, then so is p_2."

We reformulate these conditions by using the following temporal operators:

- $\mathrm{X}p$ for "p is true next time",
- $\mathrm{F}p$ for "eventually (sometime, including present) p is true",
- $\mathrm{G}p$ for "always (from now onwards) p is true",
- $p_1\mathrm{U}p_2$ for "p_1 is true until eventually p_2 is true".

Guaranty: "Sometime p_1 becomes true."

$$\mathrm{F}p_1$$

Safety: "Always p_1 is true."

$$\mathrm{G}p_1$$

Periodicity: "Initially p_1 is true, and p_1 is true at precisely every third moment."

$$p_1 \wedge \mathrm{X}\neg p_1 \wedge \mathrm{XX}\neg p_1 \wedge \mathrm{G}(p_1 \leftrightarrow \mathrm{XXX}p_1)$$

Obligation: "Sometime p_1 is true but p_2 is never true."

$$\mathrm{F}p_1 \wedge \underbrace{\neg \mathrm{F}p_2}_{\equiv \mathrm{G}\neg p_2}$$

Recurrence: "Again and again, p_1 is true."

$$\mathrm{GF}p_1$$

Request-Response: "Always when p_1 is true, p_2 will be true sometime later."

$$\mathrm{G}(p_1 \rightarrow \mathrm{XF}p_2)$$

Until Condition: "Always when p_1 is true, sometime later p_1 will be true again and in the meantime p_2 is always true."

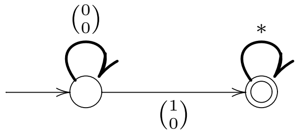
$$\mathrm{G}(p_1 \rightarrow \mathrm{X}(p_2\mathrm{U}p_1))$$

Fairness: "If p_1 is true again and again, then so is p_2."

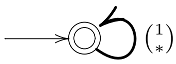
$$\mathrm{GF}p_1 \to \mathrm{GF}p_2$$

Example 2.5. (Translation of LTL-formulas to Büchi automata) By intuition one can construct corresponding Büchi automata for LTL-formula. These automata accept label sequences iff the corresponding LTL-formula are satisfied by them.

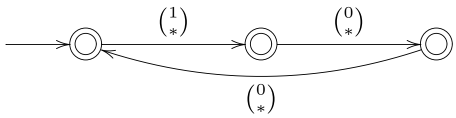
$\mathrm{F}p_1$:



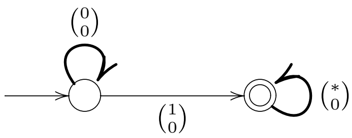
$\mathrm{G}P_1$:



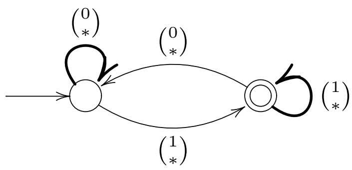
$p_1 \wedge \mathrm{X}\neg p_1 \wedge \mathrm{XX}\neg p_1 \wedge \mathrm{G}(p_1 \leftrightarrow \mathrm{XXX}p_1)$:



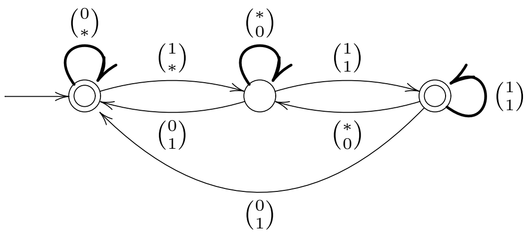
$\mathrm{F}p_1 \wedge \neg\mathrm{F}p_2$:



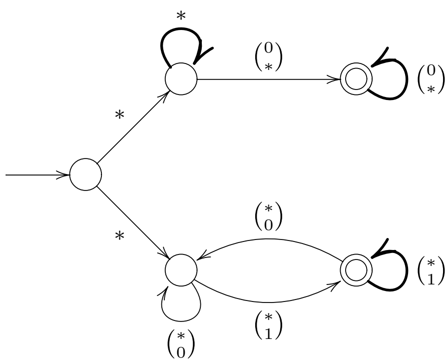
$\mathrm{GF}p_1$:



$\mathrm{G}(p_1 \to \mathrm{XF}p_2)$:



$\mathrm{GF}p_1 \to \mathrm{GF}p_2$:



We leave $\mathrm{G}(p_1 \to \mathrm{X}(p_2 \mathrm{U} p_1))$ as an exercise. ⊠

2.3 Linear-Time Temporal Logic LTL

We will now formally introduce the linear-time temporal logic.

Definition 2.6. (Syntax of LTL)
The LTL-formulas over atomic propositions $p_1, \ldots, p_n$ are inductively defined as follows:

- p_i is a LTL-formula.
- If φ, ψ are LTL-formulas, then so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$.
- If φ, ψ are LTL-formulas, then so are $\mathrm{X}\varphi$, $\mathrm{F}\varphi$, $\mathrm{G}\varphi$, $\varphi\mathrm{U}\psi$.

Example 2.7. For atomic propositions p_1, p_2 we consider

- $\mathrm{GF}p_1$: p_1 is true again and again.
- $\mathrm{XX}(p_1 \rightarrow \mathrm{F}p_2)$: if the p_1 is true in the moment after the next, then p_2 will eventually be true afterwards.
- $\mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$: Sometime p_1 will be true and from the next moment on p_2 will not be true until p_1 is true.

⊠

By convention we read "X" as "next", "F" as "eventually", "G" as "always", and "U" as "until".

LTL-formulas over $p_1, \ldots, p_n$ are interpreted in ω-words α over $\mathbb{B}^n$.

Notation If $\alpha = \alpha(0)\alpha(1)\ldots \in (\mathbb{B}^n)^\omega$, then

1. α^i stands for $\alpha(i)\alpha(i+1)\ldots$, so $\alpha = \alpha^0$.
2. $(\alpha(i))_j$ is the j-th component of $\alpha(i)$.

Definition 2.8. (Semantics of LTL)
Define the satisfaction relation $\alpha^i \models \varphi$ inductively over the construction of φ as follows:

- $\alpha^i \models p_j$ iff $(\alpha(i))_j = 1$.
- $\alpha^i \models \neg\varphi$ iff not $\alpha^i \models \varphi$.
- similarly for $\vee, \wedge, \rightarrow$.
- $\alpha^i \models \mathrm{X}\varphi$ iff $\alpha^{i+1} \models \varphi$.
- $\alpha^i \models \mathrm{F}\varphi$ iff for some $j \geq i$: $\alpha^j \models \varphi$.
- $\alpha^i \models \mathrm{G}\varphi$ iff for all $j \geq i$: $\alpha^j \models \varphi$.
- $\alpha^i \models \varphi \mathrm{~U~} \psi$ iff for some $j \geq i$, $\alpha^j \models \psi$ and for all $k = i, \ldots j-1$: $\alpha^k \models \varphi$.

Definition 2.9. An ω-language $L \subseteq (\{0,1\}^n)^\omega$ is *LTL-definable* if there is a LTL-formula ϕ with propositional variables $p_1, \ldots, p_n$ such that $L = \{\alpha \in (\{0,1\}^n)^\omega \mid \alpha \models \phi\}$.

Definition 2.10. (Satisfaction of LTL-Formulas by Kripke Structures)
A pointed Kripke structure $(\mathcal{M}, s)$ satisfies a LTL-formula ψ ($(\mathcal{M}, s) \models \psi$) if all words $\alpha = \lambda(q_0)\lambda(q_1)\ldots$, where $q_0, q_1, \ldots$ is a path through $\mathcal{M}$ with $q_0 = s$, satisfy ψ.

Example 2.11. We consider formulas over p_1, p_2.

1. $\alpha \models \mathrm{GF}p_1$

   iff for all $j \geq 0$: $\alpha^j \models \mathrm{F}p_1$

   iff for all $j \geq 0$ exists $k \geq j$: $\alpha^k \models p_1$

   iff for all $j \geq 0$ exists $k \geq j$: $(\alpha(k))_1 = 1$

   iff in α, infinitely often 1 appears in the first component.

2. $\alpha \models \mathrm{XX}(p_2 \rightarrow \mathrm{F}p_1)$

   iff $\alpha^2 \models p_2 \rightarrow \mathrm{F}p_1$

   iff if $(\alpha(2))_2 = 1$ then $\alpha^2 \models \mathrm{F}p_1$

   iff if $(\alpha(2))_2 = 1$ then $\exists j \geq 2$: $(\alpha(j))_1 = 1$

   iff "if second component of $\alpha(2)$ is 1, then the first component of some $\alpha(j)$ with $j \geq 2$ is 1".

   For example, this is true in: $\alpha = \binom{1}{0}\binom{0}{0}\binom{1}{1}\binom{0}{1}\binom{1}{0}\binom{0}{1}\cdots$

3. $\alpha \models \mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$

   iff for some $j \geq 0$: $\alpha^j \models p_1$ and $\alpha^{j+1} \models \neg p_2 \mathrm{U} p_1$

   iff for some $j \geq 0$: $\alpha^j \models p_1$ and there is a $j' \geq j+1$ with $\alpha^{j'} \models p_1$ such that for $k = j+1, \ldots, j'-1$: $\alpha^k \models \neg p_2$

   iff for some j and $j' > j$, $\alpha(j)$ and $\alpha(j')$ have 1 in first component such that for k strictly between j and j', $\alpha(k)$ has 0 in second component

   iff α has two letters $\binom{1}{*}$ such that in between only letters $\binom{*}{0}$ occur.

⊠

We have defined the semantics of LTL-formulas. Now we want to be able to determine whether a given sequence satisfies a formula.

Aim: Evaluation of a LTL-formula φ over a sequence $\alpha \in (\mathbb{B}^n)^\omega$.
Idea: Consider

- all subformulas ψ of φ in increasing complexity,
- the end sequences α^i for all $i \geq 0$.

This gives an infinite two-dimensional array of truth values: At array position (ψ, i) write 1 iff $\alpha^i \models \psi$. Then: $\alpha \models \varphi$ iff the value at position $(\varphi, 0)$ is 1.

Example 2.12. Let $\varphi = \mathrm{F}(\neg p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$. The corresponding array of truth values is:

$\alpha =$	$\binom{1}{0}$	$\binom{0}{1}$	$\binom{1}{1}$	$\binom{0}{0}$	$\binom{1}{0}$	$\binom{0}{1}$	$\cdots$
$\neg p_1$	0	1	0	1	0	1	$\ldots$
$\neg p_2$	1	0	0	1	1	0	$\ldots$
$\neg p_2 \mathrm{U} p_1$	1	0	1	1	1	0	$\ldots$
$\mathrm{X}(\neg p_2 \mathrm{U} p_1)$	0	1	1	1	0	.	$\ldots$
$\neg p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1)$	0	1	0	1	0	.	$\ldots$
$\underbrace{\mathrm{F}(\neg p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))}_{\phi}$	1	1	1	1	.	.	$\ldots$

$\boxtimes$

Definition 2.13. Given an ω-word α over $\mathbb{B}^n$ and a LTL-formula φ over $p_1, \ldots, p_n$, let m be the number of distinct subformulas of φ. The array of truth values for all subformulas is an ω-word $\beta \in \mathbb{B}^{n+m}$, called *the φ-expansion of α.*

2.4 LTL-Model-Checking Problem

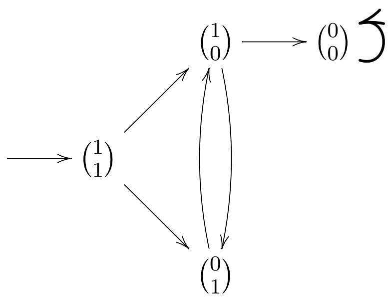
We have now met the technical requirements to reformulate the model-checking problem, using Kripke structures and LTL:

> A Kripke structure $(\mathcal{M}, s)$ is said to satisfy φ if each label sequence through $(\mathcal{M}, s)$ satisfies φ.

To write that more formally:

Definition 2.14. (LTL-Model-Checking Problem)
Given a pointed Kripke structure $(\mathcal{M}, s)$ and a LTL-formula φ (both over $p_1, \ldots, p_n$), decide whether $(\mathcal{M}, s)$ satisfies φ.

Example 2.15. Consider $\mathrm{GF}p_1$, $\mathrm{XX}(p_2 \to \mathrm{F}p_1)$, $\mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$, and the following Kripke structure:



We see that $\mathrm{GF}p_1$ fails, $\mathrm{XX}(p_2 \to \mathrm{F}p_1)$ is true, and $\mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$ fails. $\boxtimes$

How do we go about solving a given LTL model-checking problem? In the above example the answer was quite obvious. But for real world applications we need to do that algorithmically. This is the point where we will use Büchi automata for the following idea for LTL model-checking:

> Check for the *negative* answer: Is there a label sequence through $(\mathcal{M}, s)$ which does *not* satisfy φ?

Four steps are needed to implement this idea:

1. Define the ω-language of all label sequences through $(\mathcal{M}, s)$ by a Büchi automaton $\mathcal{A}_{\mathcal{M},s}$.

2. Define the ω-language of all label sequences, which do not satisfy φ by a Büchi automaton $\mathcal{A}_{\neg\varphi}$.

3. Construct a Büchi automaton $\mathcal{B}$ which recognizes $L(\mathcal{A}_{\mathcal{M},s}) \cap L(\mathcal{A}_{\neg\varphi})$, i.e. accepts all label sequences through $(\mathcal{M}, s)$ which violate φ.

4. Check $\mathcal{B}$ for nonemptiness; if $L(\mathcal{B}) \neq \emptyset$ then answer "$(\mathcal{M}, s)$ does not satisfy φ", otherwise "$(\mathcal{M}, s)$ satisfies φ".

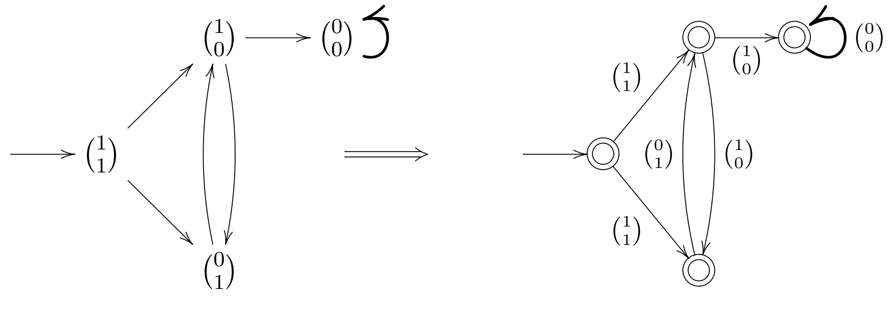
We already know algorithms for items 3. and 4. Items 1. and 2. still need to be taken care of.

From Kripke structures to Büchi automata This problem is straightforward and the solution is rather obvious: Given a pointed Kripke structure $(\mathcal{M}, s)$ with $\mathcal{M} = (S, R, \lambda)$, $\lambda : S \to \mathbb{B}^n$, construct a Büchi automaton $\mathcal{A}_{\mathcal{M},s} = (S, \mathbb{B}^n, s, \Delta, S)$ with

$$(s, (b_1 \dots b_n), s') \in \Delta \text{ iff } (s, s') \in R \text{ and } \lambda(s) = (b_1 \dots b_n).$$

So a transition gets the label of the source state.

Example 2.16. Consider the Kripke structure from Example 2.15:



⊠

The second item is not that easy to solve. We are going to dedicate a whole section to this problem.

2.5 From LTL to Büchi Automata

Idea: For a given LTL-formula φ construct a Büchi automaton, which, on input α, nondeterministically *guesses* the φ-expansion β of α and, while running, simultaneously *checks* that this guess is correct.

Consequently, a guess of β is correct, if the automaton accepts and the automaton will also ensure that the input α satisfies the corresponding LTL-formula by checking the entry at position $(\varphi, 0)$ of β. Recall that $\alpha \models \varphi$ iff $\beta(\varphi, 0) = 1$.

Therefore the automaton states are the bit vectors which are the "letters" ($\in \mathbb{B}^{n+m}$) of β.

To simplify the inductive structure of formulas, we only consider the temporal operators X and U. Eliminate F and G by the rules:

$$\begin{array}{lll} \mathrm{F}\varphi & \text{is equivalent to} & \mathtt{tt}\mathrm{U}\varphi \\ \mathrm{G}\varphi & \text{is equivalent to} & \neg\mathrm{F}\neg\varphi \end{array} \quad \text{with } \mathtt{tt} \equiv p_1 \vee \neg p_1.$$

Theorem 2.17. *For a LTL-formula φ over $p_1, \ldots, p_n$ let $\varphi_1, \ldots, \varphi_{n+m}$ be the list of all subformulas of φ in order of increasing complexity (such that $\varphi_1 = p_1, \ldots, \varphi_n = p_n, \ldots, \varphi_{n+m} = \varphi$). Then there is a generalized Büchi automaton $\mathcal{A}_\varphi$ with state-set $\{q_0\} \cup \mathbb{B}^{n+m}$, which is equivalent to φ (in the sense that $\alpha \models \varphi$ iff $\mathcal{A}_\varphi$ accepts α).*

In order to check the consistency (i.e. the correctness) of the φ-expansion that the automaton guesses, we need to come up with certain *compatibility conditions.* These are to assure that a state (that is, a letter of β) is consistent in itself and also consistent with the preceding state. Consider the following example:

Example 2.18. (Compatibility conditions) Let $\alpha \in (\mathbb{B}^n)^\omega$, $\varphi_1, \ldots, \varphi_n$ the list of subformulas of φ, and let β be the φ-expansion of α.

Illustration for $\varphi = p_1 \vee \mathrm{X}(\neg p_2 \mathrm{U} p_1)$:

$$\begin{array}{rccccccccccccc} p_1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \ldots \\ p_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \ldots \\ \neg p_2 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & \ldots \\ \neg p_2 \mathrm{U} p_1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \\ \mathrm{X}(\neg p_2 \mathrm{U} p_1) & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots \\ p_1 \vee \mathrm{X}(\neg p_2 \mathrm{U} p_1) & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \ldots \end{array}$$

Observe that the third line has to be exactly the inverse of the second line and the fourth line is equal to the fifth line, shifted to the right. The first and fifth line make up the input for the $\vee$-function which is the sixth line. ⊠

Under the assumptions of the previous example, the following holds:

$$\begin{array}{lcll} \varphi_j = \neg\varphi_{j_1} & \Rightarrow & (\beta(i))_j = 1 \text{ iff } & (\beta(i))_{j_1} = 0 \\ \varphi_j = \varphi_{j_1} \wedge \varphi_{j_2} & \Rightarrow & (\beta(i))_j = 1 \text{ iff } & (\beta(i))_{j_1} = 1 \text{ and } (\beta(i))_{j_2} = 1 \\ \varphi_j = \varphi_{j_1} \vee \varphi_{j_2} & \Rightarrow & (\beta(i))_j = 1 \text{ iff } & (\beta(i))_{j_1} = 1 \text{ or } (\beta(i))_{j_2} = 1 \\ \varphi_j = \mathrm{X}\varphi_{j_1} & \Rightarrow & (\beta(i))_j = 1 \text{ iff } & (\beta(i+1))_{j_1} = 1 \\ \varphi_j = \varphi_{j_1}\mathrm{U}\varphi_{j_2} & \Rightarrow & (\beta(i))_j = 1 \text{ iff } & (\beta(i))_{j_2} = 1 \text{ or } \big[(\beta(i))_{j_1} = 1 \\ & & & \text{and } (\beta(i+1))_j) = 1\big] \end{array}$$

For the last condition note: $\varphi\mathrm{U}\psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi\mathrm{U}\psi))$. To ensure the satisfaction of a subformula $\varphi_j = \varphi_{j_1}\mathrm{U}\varphi_{j_2}$ we have to add the condition

$$(*) \text{ there is no } k \text{ such that for every } l \geq k : (\beta(l))_j = 1 \text{ and } (\beta(l))_{j_2} = 0.$$

The first conditions are local (controllable by comparing successive column vectors of β). The last condition $(*)$ is non-local.

Proposition 2.19. *Assume $\beta \in \mathbb{B}^{n+m}$ satisfies all compatibility conditions for the given α. Then β is uniquely determined and in fact it is the φ-expansion of α.*

Proof by induction over the subformulas of φ:

For each subformula φ_j, the entry of the j-th component of β at position i is the truth value of φ_j over the sequence α^i. The cases of atomic formulas, Boolean connectives, and X-operator are clear.

For the case of $\varphi_j = \varphi_{j_1}\mathrm{U}\varphi_{j_2}$: If $(\beta(k))_{j_2} = 1$ then for all $i \leq k$ the entries for $(\beta(i))_j$ are correct. Recall that $\varphi\mathrm{U}\psi \equiv \psi \vee (\varphi \wedge \mathrm{X}(\varphi\mathrm{U}\psi))$. So if infinitely many k exist with $(\beta(k))_{j_2} = 1$, the entries $(\beta(i))_j$ are correct for all i. In the remaining case: Consider k with $(\beta(l))_{j_2} = 0$ for all $l \geq k$. Then show that for all $l \geq k$ the entry for $(\beta(l))_j$ is 0 (and hence correct). Otherwise $(\beta(l))_j = 1$ and $(\beta(l))_{j_2} = 0$ for all $l \geq k$, which poses a contradiction to $(*)$. Recall the definition of $(*)$: there is no k such that for all $l \geq k$: $(\beta(l))_j = 1$ and $(\beta(l))_{j_2} = 0$.
□

Proof of Theorem 2.17 The desired generalized Büchi automaton $\mathcal{A}_\varphi$ just has to check those compatibility conditions. It is defined as follows:

State set $Q := \{q_0\} \cup \mathbb{B}^{n+m}$, initial state q_0.

Transitions (for $\vec{b} = (b_1, \ldots, b_n)$ and $\vec{c} = (c_1, \ldots, c_m)$):

$q_0 \xrightarrow{\vec{b}} (\vec{b}\ \ \vec{c})$, where $(\vec{b}\ \ \vec{c})$ satisfies the Boolean compatibility conditions, and $c_m = 1$ (φ should be checked to be true).

$(\vec{b}\ \ \vec{c}) \xrightarrow{\vec{b}'} (\vec{b}'\ \ \vec{c}')$, where $\vec{b}, \vec{c}, \vec{b}', \vec{c}'$ satisfy all compatibility conditions except of $(*)$.

Final state sets For the until-subformula $\varphi_j = \varphi_{j_1}\mathrm{U}\varphi_{j_2}$ the *final state-set* F_j contains all states with j-component 0 or j_2-component 1. If there is no until-subformula, then every state is a final state.

This definition ensures that $\mathcal{A}_\varphi$ accepts α iff for some $\mathcal{A}_\varphi$-run $\rho \in (\mathbb{B}^{n+m})^\omega$, *each* F_j is visited infinitely often (i.e. the j-component $= 0$ or the corresponding j_2-component $= 1$ infinitely often).

This means it does not happen that from some time k onwards, the j-component stays 1 and the j_2-component stays 0.

Therefore $(*)$ is guaranteed. Recall $(*)$: there is no k such that for all $l \geq k$: $(\beta(l))_j = 1$ and $(\beta(l))_{j_2} = 0$. Consequence:

$\mathcal{A}_\varphi$ accepts α

iff the (unique) accepting run β of $\mathcal{A}_\varphi$ on α is the φ-expansion of α, and moreover at time 0 the $(n+m)$-th component of the state is 1 (signaling $\varphi_{n+m} = \varphi$ to be true)

iff $\alpha \models \varphi$.

Summary of LTL-Model-Checking Check whether a pointed Kripke structure $(\mathcal{M}, s)$ satisfies the LTL-formula φ:

1. Transform the given pointed Kripke structure $(\mathcal{M}, s)$ into a Büchi automaton $\mathcal{A}_{\mathcal{M},s}$.
2. Transform the formula $\neg\varphi$ into an equivalent generalized (and then standard) Büchi automaton $\mathcal{A}_{\neg\varphi}$.
3. Construct a Büchi automaton $\mathcal{B}$ which recognizes $L(\mathcal{A}_{\mathcal{M},s}) \cap L(\mathcal{A}_{\neg\varphi})$, i.e. accepts all label sequences through $(\mathcal{M}, s)$ which violate φ.
4. Check $\mathcal{B}$ for nonemptiness; if $L(\mathcal{B}) \neq \emptyset$ then answer "$(\mathcal{M}, s)$ does not satisfy φ", otherwise "$(\mathcal{M}, s)$ satisfies φ".

Note that items 1., 3., and 4. are all done in polynomial time.

Item 2. needs exponential time in the size of the formula (number of occurring atomic formulas, connectives, and operators)

Summary: The LTL-model-checking problem "$(\mathcal{M}, s) \models \varphi$?" is solvable in polynomial time in the size of $\mathcal{M}$ and in exponential time in the size of φ.

Further questions:

1. Is this exponential complexity avoidable?
2. Given that LTL-formulas are translatable into Büchi automata, what about the converse? (Answer: No)
3. Is there a logic which is equivalent in expressive power to Büchi automata (the logic $S1S$ over ω-sequences)?

Theorem 2.20. *The LTL-model-checking problem LTL-MC "$(\mathcal{M}, s) \models \varphi$?" is NP-hard.*

Remark 2.21. *One can even show PSPACE-completeness of LTL-MC.*

Proof of Theorem 2.20 For the NP-complete problem SAT(3) we show:

$$\text{SAT}(3) \ \leq_P \ \text{LTL-MC}$$

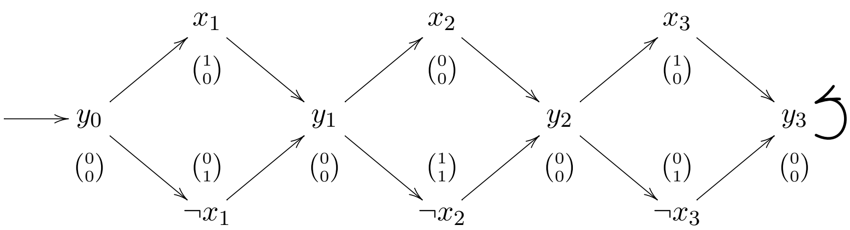
More precisely: A propositional formula ψ in conjunctive normal form with three literals per clause can be transformed in polynomial time into a pointed Kripke structure $(\mathcal{M}, s)_\psi$ and a LTL-formula φ_ψ such that

$$\psi \text{ is satisfiable iff } \textit{not } (\mathcal{M}, s)_\psi \models \varphi_\psi.$$

First, let us consider an example of constructing an equivalent LTL-model-checking problem for a SAT(3) formula.

Example 2.22. $\psi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ (satisfiable with the assignment $x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 0$)

Model $(\mathcal{M}, s)_\psi$:



LTL-formula $\varphi_\psi(p_1, p_2) := \mathrm{G}\neg p_1 \vee \mathrm{G}\neg p_2$ $\boxtimes$

General construction Given $\psi = C_1 \wedge \ldots \wedge C_n$ (the C_i are clauses), with $C_i = \chi_{i1} \vee \chi_{i2} \vee \chi_{i3}$, where χ_{ij} is a literal, i.e. either x_k or $\neg x_k$, $x_k \in \{x_1, \ldots, x_m\}$.

Define $(\mathcal{M}, s)_\psi$ over $p_1, \ldots, p_n$, $\mathcal{M} = (S, R, \lambda)$ with

$$S = \{y_0, \ldots, y_m, x_1, \ldots x_m, \neg x_1, \ldots \neg x_m\}$$

and R with the edges $(y_i, x_{i+1}), (y_i, \neg x_{i+1}), (x_i, y_i), (\neg x_i, y_i)$ and (y_m, y_m). The labeling function $\lambda : S \to \mathbb{B}^n$ is given by

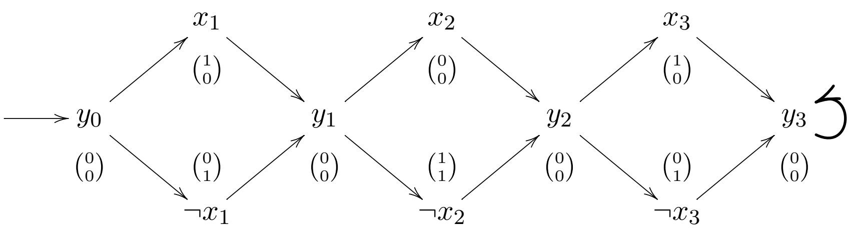
$\lambda(y_i) = 0^n$,

$(\lambda(x_i))_j = 1$ iff x_i is literal of C_j, and

$(\lambda(\neg x_i))_j = 1$ iff $\neg x_i$ is literal of C_j.

The LTL-formula is $\varphi_\psi = \mathrm{G}\neg p_1 \vee \ldots \mathrm{G}\neg p_n$.

We have to show that ψ is satisfiable iff not $(\mathcal{M}, s)_\psi \models \varphi_\psi$. Take the example $\psi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$.



An assignment $A : \{x_1, \ldots, x_m\} \to \mathbb{B}$ defines a path through $\mathcal{M}$. Therefore

ψ is satisfiable

iff some assignment makes each C_i true

iff some path through $\mathcal{M}$ meets a 1 in each component

iff not for all paths there is a component which is constantly 0

iff not $(\mathcal{M}, s)_\psi \models \mathrm{G}\neg p_1 \vee \mathrm{G}\neg p_2 \quad (= \varphi_\psi)$. $\square$

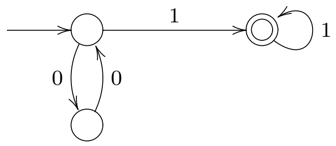
The translation from LTL to Büchi automata showed:

Each LTL-definable ω-language is Büchi recognizable.

We show that Büchi automata are (strictly) more expressive than LTL-formulas:

Theorem 2.23. *There are ω-languages which are Büchi recognizable but not LTL-definable.*

The general idea for proving this theorem is to show that LTL-formulas cannot describe "modulo-counting". As an example language we take $L = (00)^*1^\omega$. L is obviously Büchi recognizable:



We will proceed as follows:

1. Introduce the language property "non-counting".
2. Show that $L = (00)^*1^\omega$ does not have this property.
3. Show that each LTL-definable ω-language has this property.

Definition 2.24. Call $L \subseteq \Sigma^\omega$ *non-counting* if

$$\exists n_0\ \forall n \geq n_0\ \forall u, v \in \Sigma^*\ \forall \beta \in \Sigma^\omega : uv^n\beta \in L \ \Leftrightarrow\ uv^{n+1}\beta \in L.$$

This means for $n \geq n_0$ either all $uv^n\beta$ are in L, or none is. L is *not* non-counting (short: L is *counting*) iff

$$\forall n_0\ \exists n \geq n_0\ \exists u, v, \beta : (uv^n\beta \in L \text{ and } uv^{n+1}\beta \notin L) \text{ or } (uv^n\beta \notin L \text{ and } uv^{n+1}\beta \in L).$$

Claim: $L = (00)^*1^\omega$ is counting.
Given n_0 take n = next even number $\geq n_0$ and $u = \epsilon, v = 0, \beta = 1^\omega$. Then $uv^n\beta = 0^n1^\omega$ ($\in L$), but $uv^{n+1}\beta = 0^{n+1}1^\omega$ ($\notin L$). □

Proposition 2.25. *Each LTL-definable ω-language L is non-counting:*

$$\exists n_0\ \forall n \geq n_0\ \forall u, v \in \Sigma^*\ \forall \beta \in \Sigma^\omega : uv^n\beta \in L \ \Leftrightarrow\ uv^{n+1}\beta \in L$$

Proof by induction on LTL-formulas φ.

$\varphi = p_i$: Take $n_0 = 1$. Whether $uv^n\beta \in L$ only depends on first letter. This is the same letter as in $uv^{n+1}\beta$. So $uv^n\beta \in L$ iff $uv^{n+1}\beta \in L$.

$\varphi = \neg\psi$: The claim is trivial. [$uv^n\beta \notin L \ \Leftrightarrow\ uv^{n+1}\beta \notin L$]

$\varphi = \psi_1 \wedge \psi_2$: ψ_1, ψ_2 define non-counting L_1, L_2 (with n_1, n_2) by induction hypothesis. Take $n_0 = \max(n_1, n_2)$. Then the claim is true for $L_1 \cap L_2$, defined by $\psi_1 \wedge \psi_2$.

$\varphi = \mathrm{X}\psi$: By induction hypothesis assume ψ defines non-counting L with n_1.

Take $n_0 := n_1 + 1$, at least $n_0 \geq 2$.

For $n \geq n_0$ we have to show: $uv^n\beta \models \mathrm{X}\psi$ iff $uv^{n+1}\beta \models \mathrm{X}\psi$.

If $u \neq \epsilon$, say $u = au'$, then use the above induction hypothesis:

$$u'v^n\beta \models \psi \text{ iff } u'v^{n+1}\beta \models \psi.$$

If $u = \epsilon$ and $v = av'$ then use (for $n \geq n_0$)

$$v^n\beta \models X\psi \text{ iff } v'v^{n-1}\beta \models \psi \text{ iff } v'v^n\beta \models \psi \text{ iff } v^{n+1}\beta \models X\psi.$$

$\varphi = \psi_1 U \psi_2$: ψ_1, ψ_2 defining non-counting L_1, L_2 (with n_1, n_2) by induction hypothesis. Take $n_0 := 2 \cdot \max(n_1, n_2)$. We have to show: for all $n \geq n_0$:

$$uv^n\beta \models \psi_1 U \psi_2 \text{ iff } uv^{n+1}\beta \models \psi_1 U \psi_2.$$

More precisely:

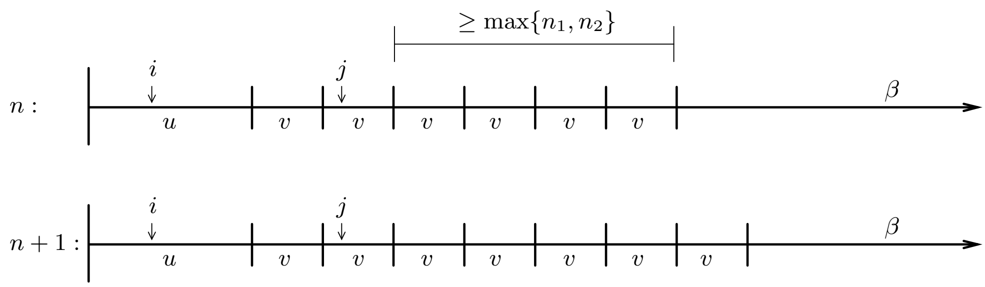
$$\begin{aligned} &\text{for some } j\text{: } (uv^n\beta)^j \models \psi_2 \text{ and for every } i < j\text{: } (uv^n\beta)^i \models \psi_1 \\ \text{iff } &\text{for some } j\text{: } (uv^{n+1}\beta)^j \models \psi_2 \text{ and for every } i < j\text{: } (uv^{n+1}\beta)^i \models \psi_1. \end{aligned}$$

Since both sides of the equivalence are symmetric, we only consider the proof from left to right. Therefore we have to show:

if for some j: $(uv^n\beta)^j \models \psi_2$ and for every $i < j$: $(uv^n\beta)^i \models \psi_1$

then for some j: $(uv^{n+1}\beta)^j \models \psi_2$ and for every $i < j$: $(uv^{n+1}\beta)^i \models \psi_1$

Case 1: $(uv^n\beta)^j$ contains $\geq \max\{n_1, n_2\}$ v-segments



Then for every $i \leq j$ $(uv^n\beta)^i$ also contains $\geq \max\{n_1, n_2\}$ v-segments. Hence by the induction hypothesis we know that

$$(uv^n\beta)^i \models \psi_1 \Leftrightarrow (uv^{n+1}\beta)^i \models \psi_1$$

for every $i < j$ and

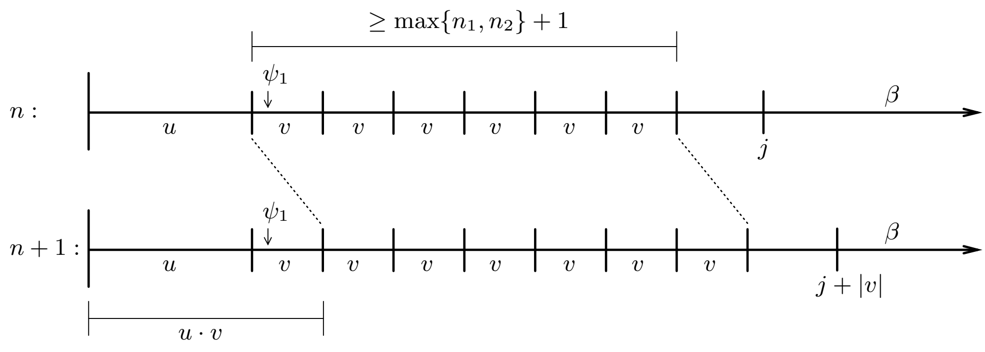
$$(uv^n\beta)^j \models \psi_2 \Leftrightarrow (uv^{n+1}\beta)^j \models \psi_2.$$

Therefore

$$uv^n\beta \models \psi_1 U \psi_2 \Leftrightarrow uv^{n+1}\beta \models \psi_1 U \psi_2.$$

Case 2: $(uv^n\beta)^j$ contains $< \max\{n_1, n_2\}$ v-segments

Then by the choice of n_0 $(uv^n\beta)[0 \ldots j]$ has $\geq \max\{n_1, n_2\} + 1$ v-segments.



Consider now for $i \leq |uv|$ the words $(uv^n\beta)^i$: Since each of these words contains $\geq \max\{n_1, n_2\}$ v-segments, we know by the induction hypothesis

$$(uv^n\beta)^i \models \psi_1 \Leftrightarrow (uv^{n+1}\beta)^i \models \psi_1.$$

For $|uv| < i < j + |v|$ $(uv^n\beta)^i = (uv^{n+1}\beta)^{i+|v|}$ and hence

$$(uv^{n+1}\beta)^i \models \psi_1 \text{ and } (uv^{n+1}\beta)^{j+|v|} \models \psi_2.$$

Therefore we obtain

$$uv^n\beta \models \psi_1 \mathrm{U} \psi_2 \Leftrightarrow uv^{n+1}\beta \models \psi_1 \mathrm{U} \psi_2.$$

□

We have proven that LTL-formulas are less expressive than Büchi automata. Now we are going to introduce a logic which can define the same class of languages as Büchi automata.

2.6 S1S (Second-Order Theory of One Successor)

The idea is to use the following elements

- variables $s, t, \ldots$ for time-points (positions in ω-words),
- variables $X, Y, \ldots$ for sets of positions,
- the constant 0 for position 0, the successor function $'$, equality $=$, and the less-than relation $<$,
- the usual Boolean connectives and the quantifiers $\exists$, $\forall$.

For clarification we compare LTL-formulas to S1S-formulas.

Example 2.26. (LTL-formulas and their translation to S1S)

$$\begin{array}{lcl}
\mathrm{GF}p_1 & : & \forall s \exists t (s \leq t \wedge X_1(t)) \\
\mathrm{XX}(p_2 \rightarrow \mathrm{F}p_1) & : & X_2(0'') \rightarrow \exists t (0'' \leq t \wedge X_1(t)) \\
\mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1)) & : & \exists t_1 (X_1(t_1) \wedge \exists t_2 (t_1' \leq t_2 \wedge X_1(t_2) \wedge \\
 & & \forall t ((t_1' \leq t \wedge t < t_2) \rightarrow \neg X_2(t))))
\end{array}$$

Let us define a counting language using S1S: $L = (00)^*1^\omega$

$$\exists X\ \exists t(X(0) \wedge \forall s(X(s) \leftrightarrow \neg X(s')) \wedge X(t) \wedge \forall s(s < t \rightarrow \neg X_1(s)) \wedge \forall s(t \leq s \rightarrow X_1(s)))$$

⊠

There are three points that we need to address, in order to prove equality in expressiveness between S1S and Büchi automata.

1. Syntax and semantics of S1S.
2. Expressive power: Büchi recognizable ω-languages are S1S-definable.
3. S1S-definable ω-languages are Büchi recognizable (Preparation).

Syntax and Semantics of S1S

Definition 2.27. (Syntax of S1S) S1S-formulas are defined over *variables*:

- *first-order* variables $s, t, \ldots, x, y, \ldots$ (ranging over natural numbers, i.e. positions in ω-words),
- *second-order* variables $X, X_1, X_2, Y, Y_1, \ldots$ (ranging over sets of natural numbers).

Terms are

- the constant 0 and first-order variables,
- for any term τ also τ' (the successor of τ).

For instance, consider the terms: $t, t', t'', 0, 0', 0''$. We can now define four classes of S1S-formulas:

- *Atomic formulas*: $X(\tau)$, $\sigma < \tau$, $\sigma = \tau$ for terms σ, τ. Note that the atomic formula $X(\tau)$ is also denoted by $\tau \in X$.
- *First-order formulas* ($\mathsf{S1S}_1$-formulas) are built up from atomic formulas using Boolean connectives and quantifiers $\exists, \forall$ over first-order variables.
- *S1S-formulas* are built up from atomic formulas using Boolean connectives and quantifiers $\exists, \forall$ over first-order variables and second-order variables.
- *Existential S1S-formulas* are $\mathsf{S1S}_1$-formulas preceded by a block $\exists Y_1 \ldots \exists Y_m$ of existential second-order quantifiers.

Example 2.28. First-order formulas:

$$\begin{aligned} \varphi_1(X) &: \forall s \exists t(s < t \wedge X(t)) \\ \varphi_2(X_1, X_2) &: X_2(0'') \rightarrow \exists t(0'' \leq t \wedge X_1(t)) \\ \varphi_3(X_1, X_2) &: \exists t_1(X_1(t_1) \wedge \exists t_2(t_1' \leq t_2 \wedge X_1(t_2) \wedge \\ &\quad \forall t((t_1' \leq t \wedge t < t_2) \rightarrow \neg X_2(t)))) \end{aligned}$$

An existential second-order formula:

$$\begin{aligned} \varphi_4(X_1) &: \exists X\ \exists t(X(0) \wedge \forall s(X(s) \leftrightarrow \neg X(s')) \wedge X(t) \\ &\quad \wedge \forall s(s < t \rightarrow \neg X_1(s)) \wedge \forall s(t \leq s \rightarrow X_1(s))) \end{aligned}$$

⊠

Notation: $\varphi(X_1, \ldots, X_n)$ indicates that at most the variables $X_1, \ldots, X_n$ occur freely in φ, i.e. are not in the scope of a quantifier.

Definition 2.29. (Semantics of S1S) We need a mathematical structure over which S1S-formulas can be interpreted. We will

- use $\mathbb{N}$ as the universe for the first-order variables,
- use $2^{\mathbb{N}}$ (the powerset of $\mathbb{N}$) as the universe for the second-order variables,
- apply the standard semantics for Boolean connectives and quantifiers.

We write $(\mathbb{N}, 0, +1, <, P_1, \ldots, P_n) \models \varphi(X_1, \ldots, X_n)$ if φ is true in this semantics, with $P_1 \subseteq \mathbb{N}, \ldots, P_n \subseteq \mathbb{N}$ as interpretations of $X_1, \ldots, X_n$. Therefore we need only specify $\overline{P} = P_1, \ldots, P_n$. $\overline{P}$ can be coded by the ω-word $\alpha(\overline{P}) \in ((\mathbb{B}^n)^\omega$ defined by

$$i \in P_k \iff (\alpha(i))_k = 1.$$

Then we simply write: $\alpha(\overline{P}) \models \varphi(X_1, \ldots, X_n)$.

Example 2.30. (Satisfaction of a S1S-formula)

$$\begin{aligned} \varphi_3(X_1, X_2) \quad : \quad & \exists t_1(X_1(t_1) \wedge \exists t_2(t_1' \leq t_2 \wedge X_1(t_2) \wedge \\ & \forall t(\underbrace{(t_1' \leq t \wedge t < t_2)}_{t_1 < t < t_2} \rightarrow \neg X_2(t)))) \end{aligned}$$

Let P_1 be the set of even numbers, P_2 be the set of prime numbers.

$$\alpha(P_1, P_2) : \begin{array}{c|cccccccc} \text{t} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\ \hline P_1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \ldots \\ P_2 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & \ldots \\ & t_1 & & t_2 & & & & & \end{array}$$

The time t_1 and t_2 instances fulfill φ_3 for P_1 and P_2: $\alpha \models \varphi_3(X_1, X_2)$ ⊠

Definition 2.31. (S1S-definable languages) An ω-language $L \subseteq (\mathbb{B}^n)^\omega$ is *S1S-definable* if for some S1S-formula $\varphi(X_1, \ldots, X_n)$ we have

$$L = \{\alpha \in (\mathbb{B}^n)^\omega \mid \alpha \models \varphi(X_1, \ldots, X_n)\}.$$

We similarly define *first-order definable*, *existential second-order definable.*

Example 2.32. (Some ω-languages defined by S1S)

1. $L = \{\alpha \in \mathbb{B}^\omega \mid \alpha \text{ has infinitely many } 1\}$ is first-order definable by

$$\forall s \exists t(s < t \wedge X_1(t)).$$

2. $(00)^*1^\omega$ is existential second-order definable by

$$\begin{aligned} \varphi_4(X_1) \quad : \quad & \exists X\ \exists t(X(0) \wedge \forall s(X(s) \leftrightarrow \neg X(s')) \wedge X(t) \\ & \wedge \forall s(s < t \rightarrow \neg X_1(s)) \wedge \forall s(t \leq s \rightarrow X_1(s))). \end{aligned}$$

⊠

From Büchi automata to S1S Before showing that Büchi automata can be translated to S1S-formulas, we prove the latter for LTL.

Theorem 2.33. *A LTL-definable ω-language is* $\mathsf{S1S}_1$*-definable.*

For an illustration of the proof let us recall the example translations from the beginning of the section:

$$\begin{array}{lcl} \mathrm{GF}p_1 & : & \forall s \exists t (s \leq t \wedge X_1(t)) \\ \mathrm{XX}(p_2 \rightarrow \mathrm{F}p_1) & : & X_2(0'') \rightarrow \exists t (0'' \leq t \wedge X_1(t)) \\ \mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1)) & : & \exists t_1 (X_1(t_1) \wedge \exists t_2 (t_1' \leq t_2 \wedge X_1(t_2) \wedge \\ & & \forall t ((t_1' \leq t \wedge t < t_2) \rightarrow \neg X_2(t)))) \end{array}$$

In general, the idea is to describe the semantics of the temporal operators in S1S. Once this is done, Theorem 2.33 can be proven inductively (Exercise).

Theorem 2.34. *A Büchi-recognizable ω-language is S1S-definable.*

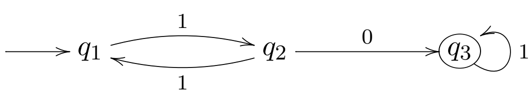
Idea: For Büchi automaton $\mathcal{A}$ over the input alphabet $\mathbb{B}^n$ find a S1S-formula $\varphi(X_1, \ldots, X_n)$ such that

$$\mathcal{A} \text{ accepts } \alpha \ \text{ iff } \ \alpha \models \varphi(X_1, \ldots, X_n).$$

We express in $\varphi(X_1, \ldots, X_n)$: "There is a successful run of $\mathcal{A}$ on the input given by $X_1, \ldots, X_n$". But how to express the existence of a run? Assume $\mathcal{A}$ has m states $q_1, \ldots, q_m$ (q_1 initial) Then a run $\rho(0)\rho(1)\ldots$ is coded by m sets $Y_1, \ldots, Y_m$ with

$$i \in Y_k \iff \rho(i) = q_k.$$

Example 2.35. (Transformation of a Büchi automaton to a S1S-formula)



Input		1	1	1	0	1	1	1
Run	Y_1	1$*$	0	1$*$				
	Y_2	0	1$*$	0	$*$			
	Y_3	0	0	0		$*$	$*$	$*$

The stars mark the state at the given point of the input word. Naturally the automaton can only be in one state for each point of time. Therefore there is just one 1 in every column of the run. How can we describe a successful run? That is, how do we set constraints to $X_1, \ldots, X_n$? Consider the formula

$$\begin{array}{lcl} \varphi(X_1) & = & \exists Y_1 Y_2 Y_3 \ (\mathrm{Partition}(Y_1, \ldots, Y_m) \wedge Y_1(0) \wedge \\ & & \forall t ((Y_1(t) \wedge X_1(t) \wedge Y_2(t')) \vee (Y_2(t) \wedge X_1(t) \wedge Y_1(t')) \\ & & \vee (Y_2(t) \wedge \neg X_1(t) \wedge Y_3(t')) \vee (Y_3(t) \wedge X_1(t) \wedge Y_3(t'))) \\ & & \wedge \forall s \exists t (s < t \wedge Y_3(t))). \end{array}$$

Partition is an expression for the above mentioned unambiguous of the automaton state. Since there is just one 1 in every Y-bitvector, Y_1, Y_2, Y_3 have to form a partition of $\mathbb{N}$.

$Y_1(0)$ states that the automaton starts in q_1. The following subformulas in the scope of the first $\forall$-quantifier represent the transition relation. The last subformula demands that the automaton enters the final state infinitely often. ⊠

Proof of Theorem 2.34 In order to be able to translate an Büchi automata with m states, some formulas, which are needed, have to be prepared:

Preparation 1: Partition$(Y_1,\ldots,Y_m) := \forall t\,(\bigvee_{i=1}^{m} Y_i(t)) \quad \wedge \quad \forall t\big(\neg \bigvee_{i\neq j}(Y_i(t)\wedge Y_j(t))\big)$.

Preparation 2: For $a \in \mathbb{B}^n$, say $a = (b_1,\ldots,b_n)$, we write $X_a(t)$ as an abbreviation for

$$(b_1)X_1(t) \wedge (b_2)X_2(t) \wedge \ldots \wedge (b_n)X_n(t)$$

where $(b_i) = \neg$ if $b_i = 0$, and b_i is empty if $b_i = 1$. For instance $a = (1,0,1) : X_a(t) = X_1(t) \wedge \neg X_2(t) \wedge X_3(t)$.

Now we can translate any Büchi automaton to an equivalent S1S-formula: Given the Büchi automaton $\mathcal{A} = (Q, \mathbb{B}^n, 1, \Delta, F)$ with $Q = \{1,\ldots,m\}$, define

$$\begin{aligned}\varphi(X_1,\ldots,X_n) \quad = \quad & \exists Y_1\ldots Y_m \Big(\text{Partition}(Y_1,\ldots,Y_m) \wedge Y_1(0) \\ & \wedge \forall t\Big(\bigvee_{(i,a,j)\in\Delta} (Y_i(t) \wedge X_a(t) \wedge Y_j(t'))\Big) \\ & \wedge \forall s \exists t\big(s < t \wedge \bigvee_{i\in F} Y_i(t)\big)\Big).\end{aligned}$$

Obviously this is just a generalization of Example 2.35. The first line gives the partitioning of $\mathbb{N}$ and the start state 1. Line 2 describes all transitions of $\mathcal{A}$ and line 3 the acceptance condition.

We conclude: A Büchi recognizable ω-language is existential second-order definable (within S1S). $\square$

In order to prove the reverse direction, we need more automata theory, which we will develop in the next chapter.

2.7 Exercises

Exercise 2.1. Consider the lift system from the introduction, now with only 4 floors. Present a set of propositions (10 are enough) needed to describe the following properties as LTL-formulas, and give the corresponding LTL-formulas:

(a) Every requested floor will be served sometime.

(b) Again and again the lift returns to floor 1.

(c) When the top floor is requested, the lift serves it immediately and does not stop on the way there.

(d) While moving in one direction, the lift will stop at every requested floor, unless the top floor is requested.

Exercise 2.2. Construct a Büchi automaton, which recognizes the set of ω-words $\alpha \in (\{0,1\}^2)^\omega$ with

$$\alpha \models G(p_1 \rightarrow X(p_2 U p_1)).$$

Exercise 2.3. Show that there is no Büchi automaton with less than three states that recognizes the set of ω-words $\alpha \in (\{0,1\}^2)^\omega$ with $\alpha \models G(p_1 \rightarrow XFp_2)$.

Exercise 2.4. Let ϕ, ψ and χ be LTL-formulas. Consider the following equivalences:

(a) $FG\phi \equiv GF\phi$,

(b) $X(\phi \wedge \psi) \equiv X\phi \wedge X\psi$,

(c) $(\phi \vee \psi)U\chi \equiv \phi U\chi \vee \psi U\chi$, and

(d) $(\phi U\psi)U\chi \equiv \phi U(\psi U\chi)$.

Prove or disprove their correctness.

Exercise 2.5. Consider the LTL-formula $\phi = p_1 U(Xp_2)$.

(a) Let $\alpha \in (\{0,1\}^2)^\omega$. Formulate the compatibility conditions for the ϕ-expansion of α in the present case.

(b) Construct, using the procedure from Theorem 3.1, a generalized Büchi automaton $\mathcal{A}$ which is equivalent to ϕ. First derive from (a) the set of compatible states, and then the transition graph of $\mathcal{A}$. What are the final states of $\mathcal{A}$?

(c) Construct directly a Büchi automaton recognizing $L := \{\alpha \in (\{0,1\}^2)^\omega \mid \alpha \models \phi\}$.

Exercise 2.6.

(a) Show that the ω-language $L_1 := (01)^\omega$ is non-counting.

(b) Show that the ω-language $L_2 := 01(0101)^*0^\omega$ is counting.

Exercise 2.7. An ω-language $L \subseteq \Sigma^\omega$ is called *strictly Büchi recognizable* if there is a Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ such that

$$L = \{\alpha \in \Sigma^\omega \mid \text{ there is a run of } \mathcal{A} \text{ on } \alpha \text{ visiting precisely the states in } F \text{ infinitely often}\}.$$

Prove, or give a counter-example, for each direction of the following equivalence:

$$L \text{ is Büchi recognizable} \iff L \text{ is strictly Büchi recognizable.}$$

Exercise 2.8. Let ϕ, ψ be LTL-formulas. We define new operators for LTL:

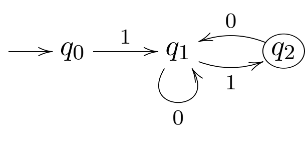
(a) "at next" $\phi AX\psi$: At the next time where ψ holds, also ϕ does.

(b) "while" $\phi W\psi$: ϕ holds at least as long as ψ does.

(c) "before" $\phi B\psi$: If ψ holds sometime, ϕ does so before.

Show that adding these operators to LTL does not increase the expressive power, i.e. find for every formula from above an equivalent (ordinary) LTL-formula.

Exercise 2.9. Let $\mathcal{A}$ be the following Büchi automaton:



Construct, using the method from the lecture, a S1S-formula $\phi(X)$ such that $\alpha \in \{0,1\}^\omega$ satisfies ϕ iff $\mathcal{A}$ accepts α.

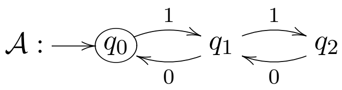
Exercise 2.10. Give S1S-formulas $\phi_1(X_1, X_2)$ and $\phi_2(X_1, X_2)$ for the following ω-languages:

(a) $L_1 := \binom{1}{1}\binom{1}{0}^*\left(\begin{smallmatrix}1 & 0\\ 1 & 0\end{smallmatrix}\right)^\omega$

(b) $L_2 := \left(\begin{smallmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{smallmatrix}\right)^*\binom{0}{1}^\omega$

Explain the purpose of the main subformulas of $\phi_1(X_1, X_2)$ and $\phi_2(X_1, X_2)$.

Exercise 2.11. Consider the following Büchi automaton:



(a) Construct a $S1S_1$-formula equivalent to $\mathcal{A}$.

(b) Construct a LTL-formula equivalent to $\mathcal{A}$.

Chapter 3

Theory of Deterministic Omega-Automata

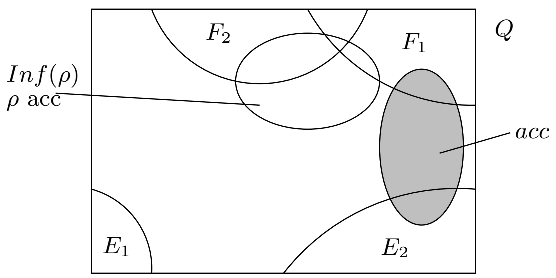
3.1 Deterministic Omega-Automata

In this chapter we are going to deal with the theory of deterministic ω-automata, as it was developed in the 1960s by Muller, McNaughton, and Rabin. The crucial point of this chapter is the transformation of nondeterministic Büchi automata into deterministic Muller automata. We will follow the construction discovered by Safra in 1988.

Definition 3.1. Let $\mathrm{Inf}(\rho) = \{q \in Q \mid q \text{ occurs infinitely often in } \rho\}$. A deterministic ω-automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathrm{Acc})$ is called

Muller automaton if Acc is of the form $\mathcal{F} = \{F_1, \ldots, F_k\}$ with $F_i \subseteq Q$, and a run ρ is successful if $\mathrm{Inf}(\rho) \in \mathcal{F}$.

Rabin automaton if Acc is of the form $\Omega = ((E_1, F_1), (E_2, F_2), \ldots, (E_k, F_k))$ with $E_i, F_i \subseteq Q$, and a run ρ is successful if $\bigvee_{i=1}^{k} (\mathrm{Inf}(\rho) \cap E_i = \emptyset \wedge \mathrm{Inf}(\rho) \cap F_i \neq \emptyset)$.



A Büchi automaton is a special case of a Rabin automaton. That Rabin automaton would have $\Omega = ((E_1, F_1))$ with $E_1 = \emptyset$ and F_1 = set of final states.

Lemma 3.2. *$L \subseteq \Sigma^\omega$ is deterministically Muller recognizable $\Leftrightarrow$ L is a Boolean combination of deterministically Büchi recognizable ω-languages.*

Proof Let the Muller automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ recognize L. Then the following holds:

$$\begin{aligned}
\alpha \in L \Leftrightarrow\ & \mathfrak{A} \text{ accepts } \alpha \\
\Leftrightarrow\ & \text{ex. } F \in \mathcal{F} : \mathfrak{A} \text{ on } \alpha \text{ visits the } F\text{-states infinitely often.} \\
\Leftrightarrow\ & \bigvee_{F \in \mathcal{F}} \Big(\bigwedge_{q \in F} \underbrace{\exists^\omega i : \delta(q_0, \alpha(0)\ldots\alpha(i)) = q}_{\substack{\alpha \text{ satisfies this condition iff} \\ \text{the Büchi automaton} \\ (Q,\Sigma,q_0,\delta,\{q\}) \text{ accepts } \alpha.}} \wedge \bigwedge_{q \in Q \setminus F} \neg \underbrace{\exists^\omega i : \delta(q_0, \alpha(0)\ldots\alpha(i)) = q}_{\text{ditto}} \Big)
\end{aligned}$$

Therefore L is a Boolean combination of deterministically Büchi recognizable ω-languages.

The reverse direction can be shown by induction over the composition of Boolean combinations. The beginning of the induction is clear since every deterministic Büchi automaton is a special case of a deterministic Muller automaton. For the induction step, we need to show that the class of deterministically Muller recognizable languages is closed under complement, intersection, and union.

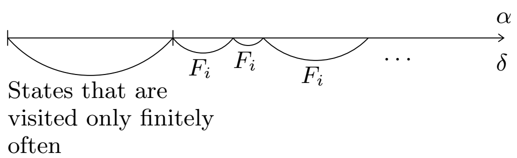
A part of the induction step: $L \subseteq \Sigma^\omega$ det. Muller recognizable $\Rightarrow \Sigma^\omega \setminus L$ det. Muller recognizable. If L is recognized by $(Q, \Sigma, q_0, \delta, \mathcal{F})$ then $\Sigma^\omega \setminus L$ will be recognized by $(Q, \Sigma, q_0, \delta, 2^Q \setminus \mathcal{F})$. $\square$

3.2 McNaughton's Theorem, Safra Construction

We show the equivalence between nondeterministic Büchi and deterministic Muller automata. This was first shown by McNaughton in 1966. One direction is easy:

Theorem 3.3. *L Muller recognizable $\Rightarrow$ L nondeterministically Büchi recognizable.*

Proof Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ recognize L with $\mathcal{F} = \{F_1, \ldots, F_k\}$. The structure of an accepting run looks like the following:



Idea for the Büchi automaton $\mathfrak{B}$: $\mathfrak{B}$ guesses the position on the input from which onwards $\mathfrak{A}$ only enters states in $\mathrm{Inf}(\rho)$. $\mathfrak{B}$ also guesses the index i of the final set F_i und asserts whether F_i is entered again and again.

- $Q_\mathfrak{B} = Q \cup (Q \times 2^Q \times \{1, \ldots, k\})$
- $q_0^\mathfrak{B} = q_0$
- $F_\mathfrak{B} = \{(p, \emptyset, j) | p \in Q, j \in \{1, \ldots, k\}\}$
- $\Delta_\mathfrak{B}$ contains (for all $j \in \{1, \ldots, k\}$)

$$\begin{array}{ll}
(p, a, q) \text{ and } (p, a, (q, \emptyset, j)) & \text{if } \delta(p, a) = q, \\
((p, P, j), a, (q, P \cup \{q\}, j)) & \text{if } \delta(p, a) = q \text{ and } P \cup \{q\} \subsetneqq F_j, \\
((p, P, j), a, (q, \emptyset, j)) & \text{if } \delta(p, a) = q \text{ and } P \cup \{q\} = F_j.
\end{array}$$

$\square$

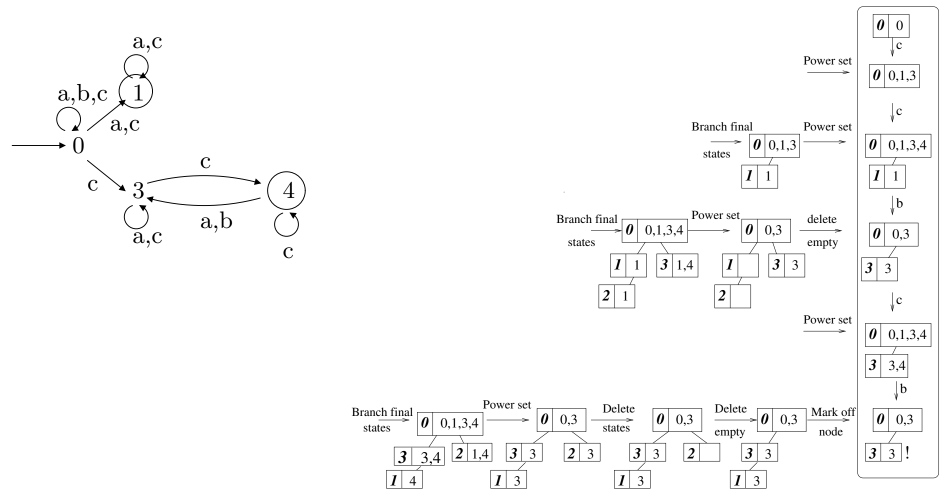
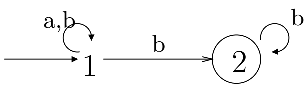


Figure 3.1: The column on the right, read top down, is the sequence of Safra trees for the given automaton on *ccbcb*. The intermediate steps are shown on the left. Within a node, the name of the node is on the left and the label on the right.

Theorem 3.4. (McNaughton's Theorem) *L nondeterministically Büchi recognizable $\Rightarrow$ L is deterministically Muller recognizable.*

Before proving the theorem, let us consider an example which shows that the powerset construction, as known from finite automata theory, does not work.

A "macro state" is a set of states of the given Büchi automaton $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$. If we apply the powerset construction on the following Büchi automaton, the new automaton will also accept the word $(ab)^\omega$, since some macro state which contains a final state is entered infinitely often.



Input	a	b	a	b	a	b	...
macro state	1	1	1,2	1	1,2	1	1,2 ...

To show McNaughton's Theorem we use a generalized powerset construction that is based on a construction by Safra (1988). In this construction, the macro states are not sets of states but rather trees, whose nodes are labeled with sets of states of the Büchi automaton. The powerset construction is performed on each node and new child nodes are branched off for final states. A state that is contained in several childs of a node, will remain in the oldest child only. Nodes with empty labels are removed (except the root node). If the union of the labels of the childs of a node is equal to the label of that node, then all children and their descendants are deleted und that node will be marked with "!". An example can be seen in Figure 3.1.

Definition 3.5. A *Safra tree* over Q is an ordered finite tree with node names in $\{1, \ldots, 2|Q|\}$, whose nodes are each labeled with a nonempty subset R of Q ($R = \emptyset$ is only allowed in the root node) or with a pair $(R, !)$. The state sets of brother nodes are disjoint and the union of the labels of child nodes is a proper subset of the label of the parent node.

Remark 3.6. *Since Q is finite, the set of Safra trees over Q is also finite.*

Notation: $P \overset{w}{\rightsquigarrow} R$ ($P, R \subseteq Q$) denotes: $\forall r \in R : \exists p \in P\ \mathfrak{A} : p \xrightarrow{w} r$.

Remark 3.7. *Let*

$$\begin{array}{ccccccccccccc}
R_0 & \overset{u_1}{\rightsquigarrow} & P_1 & \overset{v_1}{\rightsquigarrow} & R_1! & \overset{u_2}{\rightsquigarrow} & P_2 & \overset{v_2}{\rightsquigarrow} & R_2! & \dots & P_i & \overset{v_i}{\rightsquigarrow} & R_i! \\
 & & \cup\!| & & \| & & \cup\!| & & \| & & \cup\!| & & \| \\
 & & F_1 & \overset{v_1}{\rightsquigarrow} & Q_1 & & F_2 & \overset{v_2}{\rightsquigarrow} & Q_2 & & F_i & \overset{v_i}{\rightsquigarrow} & Q_i
\end{array}$$

where F_i = set of final states of P_i.

Then $\forall r \in R_i\ \exists p \in R_0 : \mathfrak{A}$ reaches from p via input $u_1v_1u_2v_2 \dots u_iv_i$ state r with $\geq i$ visits in final states.

This is made clear by retracing a run from $R_i!$ over the stages $Q_i, F_i, P_i, R_{i-1}!, Q_{i-1}, \dots$.

Lemma 3.8. (König's Lemma) *A finitely branching, infinite tree contains an infinite path.*

Proof Let t be a finitely branching, infinite tree. Define a path π that ensures the following property for every node v of π: there are infinitely many children of v in t.

The root node fulfills this by definition. This property can be transferred to a child node v' of v, because the tree is finitely branching (at v). $\square$

Lemma 3.9. *Let $R_0 \overset{u_1v_1}{\rightsquigarrow} R_1! \overset{u_2v_2}{\rightsquigarrow} R_2! \dots R_i! \overset{u_{i+1}v_{i+1}}{\rightsquigarrow} \dots$ as defined in Remark 3.7. Then there is a successful run of the nondeterministic Büchi automaton $\mathfrak{A}$ on $u_1v_1u_2v_2\dots$, beginning with a state in R_0.*

Proof Consider the tree of states that is formed by runs from R_0 to r via $u_1v_1 \dots u_iv_i$, for each state $r \in R_i$. These runs form an infinite and finitely branching tree. Then by König's Lemma there is an infinite path in this tree. This path describes an infinite (successful) run of $\mathfrak{A}$ during which $\mathfrak{A}$ enters a final state after each prefix $u_1v_1 \dots u_iv_i$. $\square$

Proof of Theorem 3.4: Definition of the desired Muller automaton $\mathfrak{B}$ for a given Büchi automaton $\mathfrak{A}$:

- $Q_{\mathfrak{B}}$:= Set of Safra trees over Q.
- $q_{0\mathfrak{B}}$:= Safra tree consisting of just the root with label $\{q_0\}$.
- For the definition of $\delta_{\mathfrak{B}}$: Compute $\delta_{\mathfrak{B}}(s, a)$ for the Safra tree $s, a \in A$ in four stages:
  1. For every node with a label that contains final states, introduce a new child node with a label that only consists of these final states. Take a free number from $2|Q|$ as the name for that node. We will show in the next section that a Safra tree has got at most $|Q|$ nodes. Since at most one child node is introduced for every node, $2|Q|$ node names suffice.
  2. Apply the powerset construction to each node label for the input letter a: $R \to \{r' \mid \exists(r, a, r') \in \Delta$, with $r \in R\}$.

3. Cancel the state q from a node and from all nodes in its subtree if it also occurs in an older brother node. Cancel a node and its descendants if it carries the label $\emptyset$ (unless it is the root).
4. Cancel all sons and their descendants if the union of their labels is the parent label. In this case mark the parent node with "**!**".

- Definition of the system $\mathcal{F}$ of final state sets:

  A set S of Safra trees is in $\mathcal{F} \Leftrightarrow$ there exists a node name that appears in each $s \in S$, and if in some tree $s \in S$, the label of this node name carries the marker "**!**".

Now we need to show: $L(\mathfrak{A}) = L(\mathfrak{B})$.

$\supseteq$ Let the constructed Muller automaton $\mathfrak{B}$ accept α. Consider the run of Safra trees of $\mathfrak{B}$ on α. Then there is a node k which, by definition of $\mathcal{F}$, occurs in every Safra tree from some point onwards and is marked with "!" infinitely often. Hence, for a suitable R, $R!$ occurs again and again as a label of k.

Then we have, according to Remark 3.9, a successful run of $\mathfrak{A}$ on α. Therefore α is accepted by $\mathfrak{A}$.

$\subseteq$ Let the Büchi automaton $\mathfrak{A}$ accept α. Trace a successful run of $\mathfrak{B}$ on α, in which a final state q is visited infinitely often. Observe in which Safra trees (of the unambiguous run of $\mathfrak{B}$) this state q occurs.

If the root is labeled with "!" infinitely often, then α will be accepted by $\mathfrak{B}$.

Otherwise consider the first occurence of a final state in the Büchi run after the root was marked off with "!" for the last time. From this point onwards the current state is always in the label of one of the child nodes of the root. That will eventually be a fixed child node k_1, since states can only be transferred to older child nodes. Now we can apply the same line of reasoning to the node k_1 as we did for the root: Either k_1 will be marked infinitely often, or we will find a child k_2 of k_1 that will from some point onwards contain the current state of the Büchi run. Thus we obtain a sequence of nodes $k_1, k_2, \ldots$.

As Safra trees are limited in depth, a node k_i must eventually be marked infinitely often and therefore $\mathfrak{B}$ accepts.

□

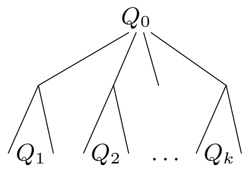
3.3 Complexity Analysis of the Safra Construction

Remark 3.10. *Let $|Q| = n$. Then every Safra tree over Q has got at most n nodes.*

Proof by induction over the height of Safra trees.

Height 0: The Safra tree has got one node ($\leq n$). Assumption clear.

Height h+1: Safra tree



$Q_0 \subseteq Q$, and $Q_1, \dots Q_k$ are disjoint and the union of them is a proper subset of Q_0. The subtrees are Safra trees over $Q_1, \dots Q_k$ (say $|Q_i| = n_i$), at each case with $\leq n_1, \dots, \leq n_k$ nodes by induction hypothesis. The number of nodes of the Safra tree of height $h+1$ is therefore $\leq n_1 + \cdots + n_k + 1 \leq |Q|$.

□

To simplify the description of Safra trees we introduce the notion of the *characteristic node* of a state $q \in Q$. This is the node with q in its label and whose children are not labeled with a set containing q. The labeling of a Safra tree is uniquely determined by the assignment $q \mapsto$ name of the characteristic node of q.

Consequently, a Safra tree s is specified by four functions:

1. Assignment of the characteristic nodes $Q \to \{0, \dots, 2n\}$, where $q \mapsto 0 \Leftrightarrow q$ is not contained in the tree.

2. "!"-Marking: $\{1, \dots, 2n\} \to \{0, 1\}$ (value $= 1$ iff label has "!").

3. Father function: $\{1, \dots, 2n\} \to \{0, \dots, 2n\}$, where Father$(i) = 0 \Leftrightarrow i$ is not contained in s.

4. Brother function: $\{1, \dots, 2n\} \to \{0, \dots, 2n\}$, where Brother$(i) = 0 \Leftrightarrow i$ is not contained in s.

The number of Safra trees is therefore $\leq$ number of quadrupels of those functions

$$\leq (2n+1)^n \cdot 2^{2n} \cdot (2n+1)^{2n} \cdot (2n+1)^{2n}$$
$$\leq (2n+1)^{7n} \in 2^{O(n \log n)}.$$

We obtain "more states" than by using the powerset construction (with 2^n states).

The Muller acceptance condition, defined in the proof of Theorem 3.4, can be transformed into an equivalent Rabin acceptance condition $((E_1, F_1), \dots, (E_m, F_m))$, where

$E_k :=$ Set of all Safra trees without the node k,
$F_k :=$ Set of all Safra trees with the node k marked with "!".

Then the following holds:

$$\begin{aligned} \mathrm{Inf}(\rho) \in \mathcal{F} \Leftrightarrow & \text{ for a node name } k: \\ & \mathrm{Inf}(\rho) \cap E_k = \emptyset \quad (k \text{ must occur in every } s \in \mathrm{Inf}(\rho) \text{ then}) \\ & \mathrm{Inf}(\rho) \cap F_k \neq \emptyset \quad (\text{Marker ! occurs with } k \text{ in a } s \in \mathit{In}(\rho)) \end{aligned}$$

We can infer Safra's Theorem:

Theorem 3.11. (Safra 1988) *A Büchi automaton with* n *states is transformed by the Safra construction into a deterministic Rabin automaton with* $2^{O(n \log n)}$ *states and* $O(n)$ *accepting pairs* (E_k, F_k) *(more precisely:* $2n$ *pairs).*

One can show that this construction is optimal:

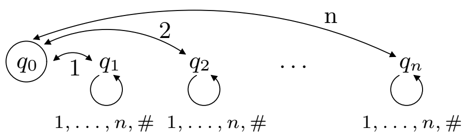
Theorem 3.12. (M. Michel 1988, C. Löding 1998) *There is no translation of nondeterministic Büchi automata with $O(n)$ states into deterministic Rabin automata with $2^{O(n)}$.*

The upper bound of the powerset construction is always exceeded. Proof strategy:

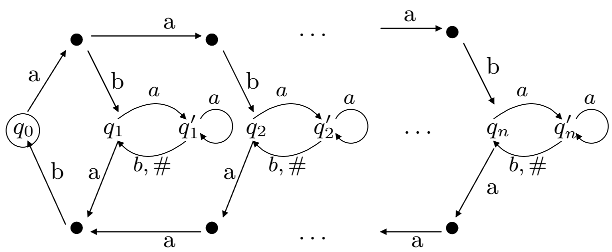
1. Specify a family $(L_n)_{n\geq 1}$ of ω-languages $L_n \subseteq \{1,\ldots,n,\#\}^\omega$, which is recognized by a Büchi automaton with $O(n)$ states.

2. Prove that L_n cannot be recognized by a deterministic Rabin automaton with $2^{O(n)}$ states.

For 1.: Define L_n by the Büchi automaton $\mathfrak{B}_n$, alphabet $\Sigma = \{1,\ldots,n,\#\}$. All states of $\mathfrak{B}_n$ are initial states.



Remark 3.13. *The alphabet depends on n. We can change it into a fixed alphabet $\{a,b,\#\}$ by the correspondence $1 \to ab$, $2 \to a^2b$, $\ldots$, $n \to a^nb$, $\# \to \#$. The following Büchi automaton, where the states $q_0, q_1, q_2, \ldots, q_n$ are initial, recognizes L_n.*



Lemma 3.14. $\alpha \in L_n \Leftrightarrow (*)$ *there are pairwise distinct letters $i_1,\ldots,i_k \in \{1,\ldots,n\}$, such that the segments made up of letter pairs $i_1i_2, i_2i_3, \ldots, i_{k-1}i_k, i_ki_1$ occur infinitely often in α.*

Proof

$\Leftarrow$ Let $(*)$ hold for $i_1,\ldots,i_k$. Find the successful run of $\mathfrak{B}_n$ on α in the following way:

Go to q_{i_1} and stay there until i_1i_2 occurs for the first time. Then do the following: $q_{i_1} \xrightarrow{i_1} q_0 \xrightarrow{i_2} q_{i_2}$. Similarly with $i_2i_3, i_3i_4, \ldots$ in the cycle $i_1, i_2, \ldots, i_k, i_1$. Thereby we obtain infinitely many visits to q_0 and $\mathfrak{B}_n$ accepts.

$\Rightarrow$ Assume $\mathfrak{B}_n$ accepts α but $(*)$ fails. Pick a position p in α such that the letter pairs i_1i_2 occuring later will in fact occur infinitely often.

If the state $q_i \neq q_0$ is visited after p and q_0 later than that, then no return to q_i is possible, since otherwise we would get a cycle as in $(*)$.

Since $q_i \neq q_0$ was arbitrary, the run would eventually stay in q_0. Contradiction. $\square$

Lemma 3.15. (Permutation Lemma) *For every permutation* $(i_1 \dots i_n)$ *of* $(1, \dots, n)$ *the* ω-*Word* $(i_1 \dots i_n \#)^\omega$ *is not in* L_n.

To prove 2. we just need a remark on Rabin automata.

Lemma 3.16. (Union Lemma) *Let* $\mathfrak{R} = (Q, \Sigma, q_0, \delta, \Omega)$ *be a Rabin automaton with* $\Omega = \{(E_1, F_1), \dots, (E_k, F_k)\}$. *Let* $\rho_1, \rho_2, \rho \in Q^\omega$ *be runs of* $\mathfrak{R}$ *with* $\mathrm{Inf}(\rho_1) \cup \mathrm{Inf}(\rho_2) = \mathrm{Inf}(\rho)$. *If* ρ_1 *and* ρ_2 *are not successful, then* ρ *is not successful, either.*

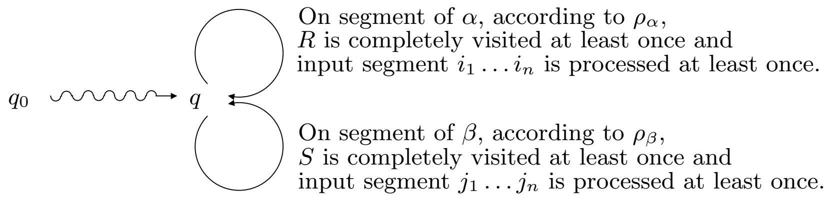
Proof Assume ρ_1, ρ_2 are not successful and ρ is successful. Then there exists an $i \in \{1, \dots, k\}$ with $\mathrm{Inf}(\rho) \cap E_i = \emptyset$ and $\mathrm{Inf}(\rho) \cap F_i \neq \emptyset$. Because of $\mathrm{Inf}(\rho_1) \cup \mathrm{Inf}(\rho_2) = \mathrm{Inf}(\rho)$, $\mathrm{Inf}(\rho_1) \cap E_i = \mathrm{Inf}(\rho_2) \cap E_i = \emptyset$ holds, and also $\mathrm{Inf}(\rho_x) \cap F_i \neq \emptyset$ for a $x \in \{1, 2\}$. Thus ρ_x is successful. Contradiction. □

Proof of Theorem 3.12 Let the deterministic Rabin automaton $\mathfrak{C}_n$ recognize L_n. Claim: $\mathfrak{C}_n$ has got $\geq n!$ states.

Consider two different permutations $(i_1, \dots, i_n), (j_1 \dots, j_n)$ of $1, \dots, n$. Then the ω-words $\underbrace{(i_1 \dots i_{n\#})^\omega}_{\alpha}$, $\underbrace{(j_1 \dots j_{n\#})^\omega}_{\beta}$ are not accepted by $\mathfrak{C}_n$. Let ρ_α, ρ_β be the non-accepting runs of $\mathfrak{C}_n$ on α and β. Set $R := \mathrm{Inf}(\rho_\alpha)$ and $S := \mathrm{Inf}(\rho_\beta)$.

Claim: $R \cap S = \emptyset$. From this follows (since there are $n!$ permutations) that $\mathfrak{C}_n$ has got at least $n!$ states and we are finished.

Assume $q \in R \cap S$: From ρ_α, ρ_β construct a new run of $\mathfrak{C}_n$, on the new input, that has the following structure:



Repeating these two loops in alternation, we get a new input word γ and a new run of $\mathfrak{C}_n$ on γ with $\mathrm{Inf}(\rho_\gamma) = R \cup S$. According to Lemma 3.16, $\mathfrak{C}_n$ does not accept γ.

Both $i_1 \dots i_n$ and $j_1 \dots j_n$ occur infinitely often in γ. Since $i_1 \dots i_n \neq j_1 \dots j_n$ choose the smallest k with $i_k \neq j_k$. Then we have the following situation:

$$\begin{array}{ccc} i_1 \; \dots & i_{k-1} & i_k \\ \| & \| & \nparallel \\ j_1 & j_{k-1} & j_k \end{array}$$

There has to be an i_l, $l > k$, with $i_l = j_k$, as well as a j_r, $r > k$, with $j_r = i_k$. We therefore obtain a cycle that corresponds to the characterization of L_n.

$$i_k i_{k+1}, \dots, i_{l-1} \underset{\substack{\| \\ j_k}}{i_l}, \; j_k j_{k+1}, \dots, j_{r-1} \underset{\substack{\| \\ i_k}}{j_r}, \; i_k i_{k+1}, \dots$$

Thus $\gamma \in L_n$ which is a contradiction to our choice of $\mathfrak{C}_n$. Therefore Theorem 3.12 has been proved. $\square$

It is an open question whether there are ω-languages L_n that can be recognized by nondeterministic Büchi automata with $O(n)$ states and only by deterministic *Muller* automata with $\geq n!$ states.

Remark 3.17. *The example languages L_n as defined for the proof of Theorem 3.12 are recognized by deterministic Muller automata with $O(n^2)$ states.*

3.4 Logical Application: From S1S to Büchi Automata

In the last chapter we tried to show the equivalence of the logic S1S and Büchi automata. Now we have the tools ready to prove that every S1S definable language is Büchi recognizable. As a consequence of McNaughton's Theorem we see:

Theorem 3.18. *The class of Büchi recognizable ω-languages is closed under complement.*

Proof Given a Büchi automaton $\mathcal{B}$, construct a Büchi automaton for the complement ω-language as follows:

1. From $\mathcal{B}$ obtain an equivalent deterministic Muller automaton $\mathcal{M}$ by Safra's construction.
2. In $\mathcal{M}$ declare the non-accepting state sets as accepting and vice versa and thus obtain $\mathcal{M}'$.
3. From $\mathcal{M}'$ obtain an equivalent Büchi automaton $\mathcal{B}'$.

$\square$

We showed that a Büchi-recognizable ω-language is S1S definable. Now we prove the converse:

Theorem 3.19. *An S1S-definable ω-language is Büchi recognizable.*

There will be two stages in the proof:

1. Reduction of S1S to a simpler formalism $\mathsf{S1S}_0$.
2. Construction of an equivalent Büchi automata by induction on $\mathsf{S1S}_0$-formulas.

From S1S to $\mathsf{S1S}_0$ For simplification we eliminate some constructs from S1S:

- The *constant* 0 can be eliminated: Instead of $X(0)$ write
$$\exists t(X(t) \wedge \neg\exists s(s < t)).$$
- The *relation symbol* $<$ can be eliminated: Instead of $s < t$ write
$$\forall X(X(s') \wedge \forall y(X(y) \rightarrow X(y')) \ \rightarrow \ X(t))$$
(each set which contains s' and is closed under successors must contain t).

- The *successor function* only occurs in formulas of type $x' = y$: Instead of $X(s'')$ write

$$\exists y \exists z (s' = y \wedge y' = z \wedge X(z)).$$

- Eliminate the use of first-order variables by using different atomic formulas:

$$X \subseteq Y, \quad \mathrm{Sing}(X), \quad \mathrm{Succ}(X, Y),$$

  meaning: "X is subset of Y", "X is a singleton set", and "$X = \{x\}$, $Y = \{y\}$ are singleton sets with $x + 1 = y$". Now one can write $X(y)$ as $\mathrm{Sing}(Y) \wedge Y \subseteq X$ and $x' = y$ as $\mathrm{Succ}(X, Y)$.

Example 3.20. Translation example: $\forall x \, \exists y (x' = y \wedge Z(y))$ is written as

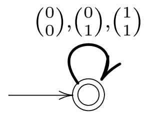
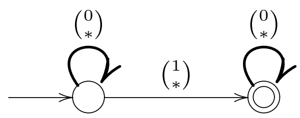
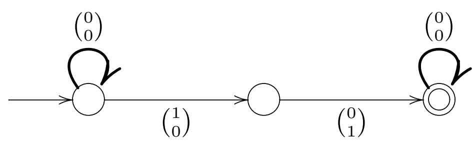
$$\forall X (\mathrm{Sing}(X) \rightarrow \exists Y (\mathrm{Sing}(Y) \wedge \mathrm{Succ}(X, Y) \wedge Y \subseteq Z)).$$

⊠

Proof of Theorem 3.19 We can assume that S1S-formulas $\varphi(X_1, \ldots, X_n)$ are rewritten as $\mathsf{S1S}_0$-formulas. We show the claim by induction on $\mathsf{S1S}_0$-formulas. It suffices to treat

- the atomic formulas
  $X_1 \subseteq X_2, \quad \mathrm{Sing}(X_1)\ , \quad \mathrm{Succ}(X_1, X_2),$
- the connectives $\vee$ and $\neg$, and the existential set quantifier $\exists$.

We can easily specifiy Büchi automata for the atomic formulas (induction basis):

Atomic formula	Corresponding Büchi automaton	Recognized example word
$X_1 \subseteq X_2$		X_1=001101... X_2=010101...
$\mathrm{Sing}(X_1)$		$X_1 = 000010000\ldots$
$\mathrm{Succ}(X_1, X_2)$		X_1=0001000... X_2=0000100...

Induction step:

1. Or connective: Consider $\varphi_1(X_1, \ldots, X_n) \vee \varphi_2(X_1, \ldots, X_n)$.
   By induction hypothesis we have Büchi automata $\mathcal{A}_1, \mathcal{A}_2$ that are equivalent to φ_1, φ_2. Take the Büchi automaton for the union as the one equivalent to $\varphi_1 \vee \varphi_2$.

2. Negation: Consider $\neg\varphi(X_1, \ldots, X_n)$.
   By induction hypothesis there is a Büchi automaton equivalent to φ.

   Apply the closure of Büchi recognizable ω-languages under complement, to obtain a Büchi automaton equivalent to $\neg\varphi$.

3. Existential quantifier: Consider $\exists X \varphi(X, X_1, \ldots, X_n)$.
   Assume $\mathcal{A}$ is a Büchi automaton equivalent to $\varphi(X, X_1, \ldots, X_n)$. In $\mathcal{A}$, change each transition label $(b, b_1, \ldots, b_n)$ into $(b_1, \ldots, b_n)$; thus obtain $\mathcal{A}'$. Then a transition via $\overline{b}$ exists in $\mathcal{A}'$ if there is a transition via $(0, \overline{b})$ or $(1, \overline{b})$ in $\mathcal{A}$.

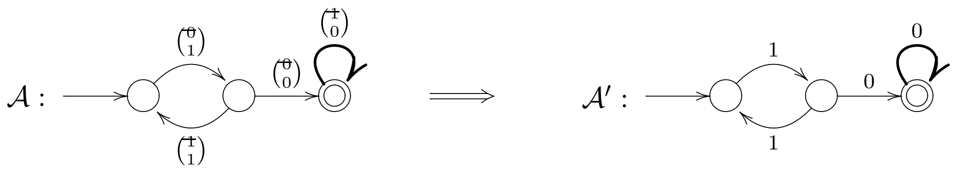
   $\mathcal{A}'$ accepts $\alpha \in (\mathbb{B}^n)^\omega$

   iff there exists a bit sequence $c_0 c_1 \ldots$ such that $(c_0, \alpha(0)), (c_1, \alpha(1)) \ldots$ is accepted by $\mathcal{A}$

   iff $\exists c_0 c_1 \ldots$ such that $\mathcal{A}$ accepts $(c_0, \alpha(0)), (c_1, \alpha(1)) \ldots$

   iff $\alpha \models \exists X \varphi(X, X_1, \ldots, X_n)$.

   So the Büchi automaton $\mathcal{A}'$ is equivalent to $\exists X \varphi(X, X_1, \ldots, X_n)$. An example:



□

3.5 Complexity of Logic-Automata Translations

We have translated LTL- and S1S-formulas into Büchi automata. The complexity bounds are very different. We define the k-fold exponential function g_k by

$$g_0(n) = n, \quad g_{k+1}(n) = 2^{g_k(n)}.$$

Theorem 3.21. (Translation complexity of LTL and S1S)

1. *An LTL formula of size n (measured in the number of subformulas) can be translated into a Büchi automaton with 2^n states.*

2. *There is no k such that each S1S formula of size n (measured in the number of subformulas) can be translated into a Büchi automaton with $g_k(n)$ states.*

The case of sentences We will briefly mention a historical application of the translation from S1S to Büchi automata. Consider *sentences*, which are formulas without free variables.

The translation of a sentence φ into a Büchi automaton $\mathcal{A}_\varphi$ yields an automaton with unlabeled transitions.

As we can now see, the sentence φ is true in the structure $(\mathbb{N}, +1, <, 0)$ iff the automaton $\mathcal{A}_\varphi$ has a successful run. The latter condition can be checked with the nonemptiness test. Consequently, one can decide, for any given S1S-sentence φ, whether φ is true in $(\mathbb{N}, +1, <, 0)$ or not.

The *monadic second-order theory of* $(\mathbb{N}, +1, <, 0)$ is the set of S1S-sentences that are true in $(\mathbb{N}, +1, <, 0)$. This is written as $\mathrm{MTh}_2(\mathbb{N}, +1, <, 0)$.

Some example sentences:

$$\begin{array}{ll} \forall X\, \exists Y (\forall t(X(t) \to Y(t))) & \text{true} \\ \forall X\, \exists t\, \forall s(X(s) \to s < t) & \text{false} \\ \forall X(X(0) \wedge \forall s(X(s) \to X(s')) \;\to\; \forall t X(t)) & \text{true} \end{array}$$

By applying the above mentioned translation into Büchi automata and by testing for nonemptiness, we immediately see :

Theorem 3.22. (Büchi 1960) *The theory* $\mathrm{MTh}_2(\mathbb{N}, +1, <, 0)$ *is decidable.*

3.6 Classification of Omega-Regular Languages and Sequence Properties

Up to now we have treated general logical and automata theoretical methods to describe sequence properties (i.e. system properties). However the last section showed that taking too broad a view results in computationally infeasible results.

In Section 2.2 we mentioned several interesting sequence properties, e.g. "safety", "guaranty" and so on. We will now narrow our view of ω-languages to automata models which correspond to those properties. Using those models we will prove certain relationships between those properties, i.e. can some property be expressed by some other property? This will give us the tools to solve infinite games in the second part of this course.

So what are we going to do in this section?

1. Definition of a natural classification scheme based on deterministic automata.
2. Comparison of the levels of this classification.
3. Decision to which level a given property belongs.

The four basic types of sequence properties We have already seen a wide variety of sequence properties in Section 2.2. The following four basic properties can be described intuitively:

- **Guaranty condition** requires that *some* finite prefix has a certain property.
- **Safety condition** requires that *all* finite prefixes have a certain property.
- **Recurrence condition** requires that *infinitely many* finite prefixes have a certain property.
- **Persistence condition** requires that *almost all (i.e. from a certain point onwards all)* finite prefixes have a certain property.

We shall describe the prefix properties by deterministic automata.

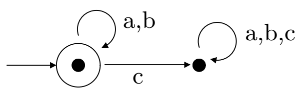
Definition 3.23. Given a deterministic automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$,

- $\mathfrak{A}$ *E-accepts* $\alpha \Leftrightarrow$ exists a run ρ of $\mathfrak{A}$ on α with $\exists i : \rho(i) \in F$.
- $\mathfrak{A}$ *A-accepts* $\alpha \Leftrightarrow$ exists a run ρ of $\mathfrak{A}$ on α, so that $\forall i : \rho(i) \in F$.

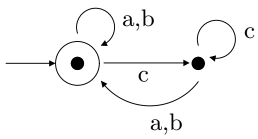
- $\mathfrak{A}$ *Büchi-accepts* $\alpha \Leftrightarrow$ exists a run, so that $\forall j \exists i \geq j : \rho(i) \in F$.
- $\mathfrak{A}$ *co-Büchi-accepts* $\alpha \Leftrightarrow$ exists a run ρ of $\mathfrak{A}$ on α, so that for almost all i (except of finitely many, written: $\forall^\omega i$) $\rho(i) \in F$ holds, i.e. from some point onwards *only* final states will be visited.

The notions *A*-, *E*-, and *co-Büchi automaton* and *A*-, *E*-, and *co-Büchi recognizable* are defined accordingly.

Example 3.24. Let $\Sigma = \{a, b, c\}$.
$L_1 = \{\alpha \in \Sigma^\omega \mid \text{ no } c \text{ in } \alpha\}$. L_1 is A-recognizable by



$L_2 = \{\alpha \in \Sigma^\omega \mid c \text{ only finitely often in } \alpha\}$. L_2 is co-Büchi recognizable by



$\boxtimes$

By intuition we can summarize some relationships between acceptance conditions on the one side and sequence properties on the other side, in Table 3.1.

We want to show the following connections for specifications by automata:

- Guaranty *and* safety properties can be rewritten as recurrence and persistence properties.
- Guaranty properties cannot be described as safety properties (and vice versa).
- The same holds for recurrence and persistence properties.

These claims can be proven within the precise framework of ω-automata.

Theorem 3.25. *Let $L \subseteq \Sigma^\omega$.*

a) L deterministically E-recognizable $\Leftrightarrow L = U \cdot \Sigma^\omega$ for a regular $U \subseteq \Sigma^*$.

b) L deterministically Büchi recognizable $\Leftrightarrow L = \lim(U)$ for a regular $U \subseteq \Sigma^*$.

Proof Item (b) was shown earlier in the proof of Theorem 1.10 b). Proof of (a): Similar to the proof of Theorem 1.10 b): Let U be recognized by the DFA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$. Use $\mathfrak{A}$ as a deterministic E-automaton, now called $\mathfrak{B}$.

$$\begin{aligned}
\mathfrak{B} \text{ accepts } \alpha \quad & \stackrel{\text{Def}}{\Longleftrightarrow} \text{The unambiguous run of } \mathfrak{B} \text{ on } \alpha \text{ enters } F \text{ at least once} \\
& \Longleftrightarrow \exists i : \quad \mathfrak{A} \text{ reaches a state in } F \text{ after } \alpha(0)\ldots\alpha(i) \\
& \Longleftrightarrow \exists i : \quad \alpha(0)\ldots\alpha(i) \in U \text{ (according to the def. of } \mathfrak{A}) \\
& \Longleftrightarrow \alpha \in U \cdot \Sigma^\omega.
\end{aligned}$$

$\square$

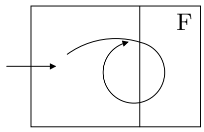
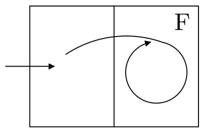
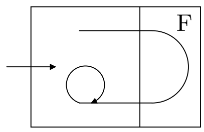
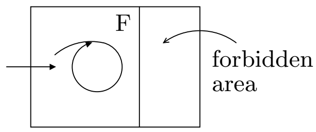
Büchi acceptance	**co-Büchi acceptance**
grasps "recurrence properties" Illustration:  System assumes desired states again and again	grasps "persistence properties" Illustration:  System finally assumes desired states only
E-acceptance grasps "guaranty properties" Illustration:  System assumes desired state sometime	**A-acceptance** grasps "safety properties" Illustration:  System is always in a desired state

Table 3.1: Overview

Lemma 3.26. (Complement Lemma) *Let $L \subseteq \Sigma^\omega$. Then the following holds:*

a) L is deterministically E-recognizable $\Leftrightarrow$ the complement language $\Sigma^\omega \setminus L$ is deterministically A-recognizable.

b) L is deteterministically Büchi recognizable $\Leftrightarrow$ the complement language $\Sigma^\omega \setminus L$ is deterministically co-Büchi recognizable.

Proof Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ recognize L.

a) $\alpha \in \Sigma^\omega \setminus L$ $\Leftrightarrow$ F is never reached during the unambiguous run ρ of $\mathfrak{A}$ on α
$\Leftrightarrow$ Only states from $Q \setminus F$ are assumed during the unambiguous run ρ of $\mathfrak{A}$ on α.

Thus $\mathfrak{A}' := (Q, \Sigma, q_0, \delta, Q \setminus F)$ A-accepts $\Sigma^\omega \setminus L$. "$\Leftarrow$" can be shown analogously.

b) $\alpha \in \Sigma^\omega \setminus L$ $\Leftrightarrow$ F is visited only finitely often during the unambiguous run ρ of $\mathfrak{A}$ on α
$\Leftrightarrow$ from some point onwards only states in $Q \setminus F$ are assumed.

Thus $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, Q \setminus F)$ co-Büchi accepts $\Sigma^\omega \setminus L$. "$\Leftarrow$" can be shown analogously.

□

Theorem 3.27. *Let* $L \subseteq \Sigma^\omega$.

a) L deterministically E-recognizable $\Rightarrow$ L is deterministically Büchi recognizable.

b) The converse does not hold in general.

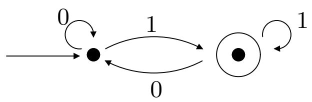
Proof

a) Given $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ construct a deterministic Büchi automaton by adding a state q_f. We are going to "divert" all transitions to F to the newly created state q_f. We define a new transition function δ':

$$\begin{aligned} \delta'(q,a) &= \delta(q,a) \text{ if } q \notin F \\ \delta'(q,a) &= q_f \text{ if } q \in F \\ \delta'(q_f,a) &= q_f \end{aligned}$$

Set $\mathfrak{B} := (Q \cup \{q_f\}, \Sigma, q_0, \delta', \{q_f\})$. Then the new automaton $\mathfrak{B}$ Büchi accepts the ω-word α iff $\mathfrak{A}$ E-accepts α.

b) $\not\Leftarrow$: Consider $L = \{\alpha \in \mathbb{B}^\omega \mid 1 \text{ appears infinitely often in } \alpha\}$. A deterministic Büchi automaton which recognizes this language could look like this:



Assume: A deterministic E-automaton $\mathfrak{A}$ recognizes L. According to Theorem 3.25 $L = U \cdot \Sigma^\omega$ for a regular $U \subseteq \Sigma^*$. Since L is nonempty, U is also nonempty. Let $u \in U$. Then $u0^\omega \in U \cdot \Sigma^\omega$ but $u0^\omega \notin L$.

□

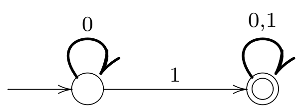
Lemma 3.28. *There are languages which separate the above mentioned language classes:*

1. $\mathbb{B}^* \cdot 1 \cdot \mathbb{B}^\omega$ *is E-recognizable, but not A-recognizable.*
2. $\{0^\omega\}$ *is A-recognizable but not E-recognizable.*
3. $(0^*1)^\omega$ *is Büchi recognizable but not co-Büchi recognizable.*
4. $\mathbb{B}^*0^\omega$ *is co-Büchi recognizable but not Büchi recognizable.*

Note that $\{0^\omega\} = \mathbb{B}^\omega \setminus (\mathbb{B}^* \cdot 1 \cdot \mathbb{B}^\omega)$, $\quad \mathbb{B}^*0^\omega = \mathbb{B}^\omega \setminus (0^*1)^\omega$.

Proof

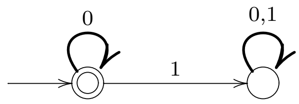
1. E-recognizability is clear.



Assume $\mathbb{B}^* \cdot 1 \cdot \mathbb{B}^\omega$ is A-recognizable, say by $\mathcal{A}$ with n states.

Consider $\mathcal{A}$ on 0^n10^ω; all states of the run are final. Before input letter 1 there is a state repetition (loop of final states). So with this loop $\mathcal{A}$ also accepts the input word 0^ω, contradiction.

2. $\{0^\omega\}$ being A-recognizable but not E-recognizable follows from the Complement Lemma, since $\{0^\omega\} = \mathbb{B}^\omega \setminus (\mathbb{B}^* \cdot 1 \cdot \mathbb{B}^\omega)$.

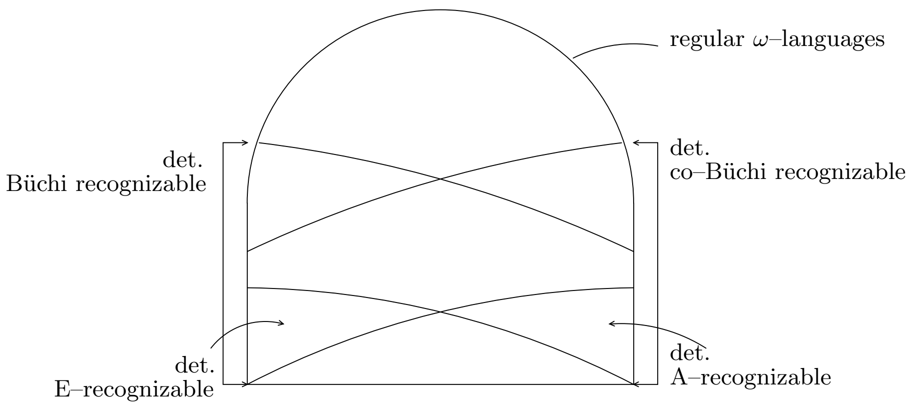


3. $(0^*1)^\omega$ being Büchi recognizable was shown for Theorem 3.27(b). It is easy to show that this language is not co-Büchi recognizable

4. $\mathbb{B}^*0^\omega$ being co-Büchi recognizable but not Büchi recognizable then follows from the Complement Lemma and 3.

$\square$

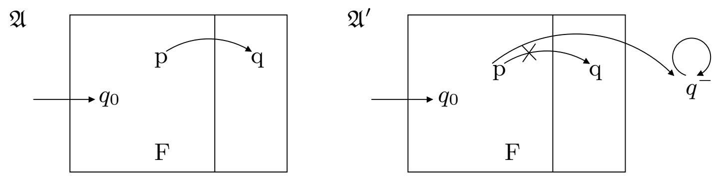
Theorem 3.29. (Hierarchy Theorem) *The following diagram of inclusions holds for the classes of ω-languages that can be recognized by deterministic automata with E-, A-, Büchi, and co-Büchi acceptance conditions:*



Proof (Inclusions)

$$\left.\begin{array}{l} L \text{ is det. E-recogn.} \\ L \text{ is det. A-recogn.} \end{array}\right\} \Rightarrow L \text{ is det. Büchi recogn.} \Rightarrow L \text{ is nondet. Büchi recogn.}$$

The implication L is det. E-recognizable $\Rightarrow$ L is det. Büchi recognizable was already shown (Theorem 3.27). Show: L is deterministically A-recognizable $\Rightarrow$ L is deterministically Büchi recognizable. Consider the automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$, which A-recognizes L and modify it as follows:



We replace every $\delta(p, a) = q$, where $p \in F, q \notin F$, by $\delta(p, a) = q^-$, and add $\delta(q^-, a) = q^-$ for every $a \in \Sigma$. Then the $\mathfrak{A}$-run ρ only assumes final states on α

$\Leftrightarrow$ the corresponding $\mathfrak{A}'$-run ρ' on α only assumes final states

$\Leftrightarrow$ $(*)$ the corresponding $\mathfrak{A}'$-run ρ' on α infinitely often assumes final states.

For $(*)$: $\Leftarrow$ If $\mathfrak{A}'$ infinitely often assumes a final state, then $\mathfrak{A}'$ follows no transition leading out of F. Therefore $\mathfrak{A}'$ only enters final states, i.e. $\mathfrak{A}'$ Büchi recognizes L.

The claims

$$\begin{array}{ll} \left.\begin{array}{l} L \text{ is det. E-recognizable} \Rightarrow \\ L \text{ is det. A-recognizable} \Rightarrow \end{array}\right\} & L \text{ is det. co-Büchi recognizable,} \\ L \text{ is det. co-Büchi-recognizable} \Rightarrow & L \text{ is nondet. Büchi recognizable} \end{array}$$

will be proved in the exercises.

In order to show that these inclusions are *proper*, we need to consider seven different cases. These are depicted in Figure 3.2. 4, 5, 6, and 7 have already been proved to be nonempty by

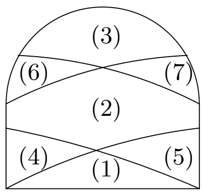
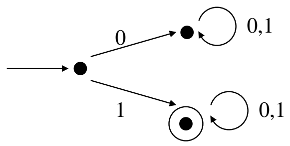


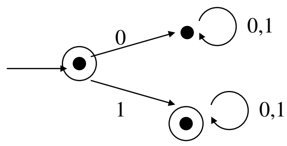
Figure 3.2: Seven inclusions

the languages $\mathbb{B}^* \cdot 1 \cdot \mathbb{B}^\omega$, $\{0^\omega\}$, $(0^*1)^\omega$, and $\mathbb{B}^*0^\omega$ respectively in Remark 3.28.

(1) $L_1 := \{1(0+1)^\omega\}$ is det. E-recognizable and det. A-recognizable:

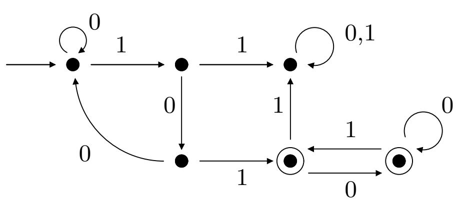


E-recognizes L_1



A-recognizes L_1

(2) $L_2 := \{\alpha \in \mathbb{B}^\omega \mid$ 11 never occurs in α but 101 at least once$\}$. This language is recognized by the following det. Büchi automaton:



This det. Büchi automaton for L_2 is at the same time the co-Büchi automaton for L_2.

Assume 1: L_2 is E-recognizable, say by $\mathfrak{A}$.

Consider $\mathfrak{A}$ on the word 1010^ω. This ω-word is accepted by the automaton. A final state is visited not later than after the prefix 1010^n. Therefore the ω-word 1010^n110^ω is accepted. Contradiction.

Assume 2: L_2 is A-recognizable, say by $\mathfrak{A}$ with n states.

Consider $\mathfrak{A}$ on 0^n1010^ω. The automaton accepts, i.e. it visits final states only. Because of the repetition of states on 0^n only final states are assumed on 0^ω. Thus 0^ω is accepted but $0^\omega \notin L_2$. Contradiction.

(3) $L_3 := \{\alpha \in \mathbb{B}^\omega \mid$ 00 occurs only finitely often in α, but 11 only finitely often$\}$.

We will show in the exercises that L_3 is nondeterministically Büchi recognizable but neither deterministically Büchi nor deterministically co-Büchi recognizable.

□

3.7 Deciding the Level of Languages

For a given regular ω-language L (defined by, say, a Muller automaton) one can decide whether L is det. Büchi recognizable or det. E-recognizable.

Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a Muller automaton that has (w.l.o.g.) only reachable states. A set $S \subseteq Q$ is named *loop* if $S \neq \emptyset$ and $\forall s, s' \in S\ \exists w \in \Sigma^+\ \delta(s, w) = s'$. Thus loops are the sets of states which can occur as $\mathrm{Inf}(\rho)$ of a run ρ. Let (w.l.o.g.) $\mathcal{F}$ consist of loops only.

Definition 3.30. Call $\mathcal{F}$ *closed under reachable loops* iff each loop S' reachable from a loop $S \in \mathcal{F}$ also belongs to $\mathcal{F}$. Call $\mathcal{F}$ *closed under superloops* iff each loop $S' \supseteq S$ for a loop $S \in \mathcal{F}$ also belongs to $\mathcal{F}$.

$$\begin{aligned}
\mathcal{F}_1 &:= \{S \subseteq Q \mid S \text{ is a loop, } S \text{ is reachable from a loop in } \mathcal{F}\}\\
\mathcal{F}_2 &:= \mathcal{F} \cup \{F \cup E \mid F \cup E \text{ is a loop with at least one state more than in } F \in \mathcal{F}\}\\
&= \text{“proper superloop of } \mathcal{F}\text{-loops”}
\end{aligned}$$

Remark 3.31.

1. *$\mathcal{F}$ is closed under reachable loops iff $\mathcal{F} = \mathcal{F}_1$.*
2. *$\mathcal{F}$ is closed under superloops iff $\mathcal{F} = \mathcal{F}_2$.*
3. *Each superloop of an $\mathcal{F}$-loop is also reachable from an $\mathcal{F}$-loop; so if $\mathcal{F}$ is closed under reachable loops then it is also closed under superloops. So obviously $\mathcal{F} \subseteq \mathcal{F}_2 \subseteq \mathcal{F}_1$ holds.*

Theorem 3.32. (Landweber's Theorem)

a) $\mathcal{F} = \mathcal{F}_1 \Leftrightarrow L(\mathfrak{A})$ is deterministically E-recognizable.

b) $\mathcal{F} = \mathcal{F}_2 \Leftrightarrow L(\mathfrak{A})$ is deterministically Büchi recognizable.

Proof of a)

$\Rightarrow$ Let $\mathcal{F} = \mathcal{F}_1$. Define the E-automaton $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, \bigcup \mathcal{F})$.

$$\begin{aligned}
\mathfrak{A} \text{ accepts } \alpha &\iff \mathfrak{A} \text{ eventually stays in a loop } S \in \mathcal{F}_1 \text{ on } \alpha\\
&\overset{\mathrm{Def}\mathcal{F}_1}{\iff} \text{at some point } \mathfrak{A} \text{ reaches a loop from } \mathcal{F}_1 \text{ on } \alpha\\
&\iff \mathfrak{A}' \text{ } E\text{-accepts } \alpha.
\end{aligned}$$

Thus $\mathfrak{A}$ and $\mathfrak{A}'$ are equivalent.

$\Leftarrow$ Let the deterministic E-automaton $\mathfrak{B}$ recognize $L(\mathfrak{A})$, w.l.o.g. let $L(\mathfrak{A}) \neq \emptyset$.

Show: $\mathcal{F}_1 \subseteq \mathcal{F}$

Consider $q \in S \in \mathcal{F}$. Show that all loops reachable from q are already in $\mathcal{F}$.

Choose $u \in \Sigma^*$ with $\delta_{\mathfrak{A}}(q_0, u) = q$. Choose $\gamma \in \Sigma^\omega$, so that $\mathfrak{A}$ on $u\gamma$ assumes the loop S. Since $S \in \mathcal{F}$, $u\gamma \in L(\mathfrak{A})$ holds. The automaton $\mathfrak{B}$ at some time reaches a final state on $u\gamma$, say after uv. In $\mathfrak{A}$ extend uv with w so that $\delta_{\mathfrak{A}}(q_0, uvw) = q$.

Let S' be a loop, reachable from q, say via the input word $uvw\gamma'$. Since this ω-word has got the prefix uv, $\mathfrak{B}$ accepts $uvw\gamma'$. Therefore $\mathfrak{A}$ also accepts $uvw\gamma'$. Thus the loop S' is also in $\mathcal{F}$. □

Proof of b)

$\Rightarrow$ Let $\mathcal{F} = \mathcal{F}_2$.

$\mathfrak{A}$ accepts $\alpha \overset{\text{Def}\mathcal{F}_2}{\iff} \mathfrak{A}$ eventually assumes a superloop of an $\mathcal{F}$-loop on α.

Construct a Büchi automaton $\mathcal{A}'$ with the state set $Q \times 2^Q$ and start state $(q_0, \emptyset)$. The automaton accumulates the visited states in (q, R) until a $\mathcal{F}$-loop is reached or outnumbered. Then we reset $R := \emptyset$. The final states are all $(q, \emptyset)$. So

$\mathcal{A}'$ accepts α

iff of input α, $\mathcal{A}$ infinitely often passes through loops $S' \supseteq S$ where $S \in \mathcal{F}$

iff (since only finitely many such S' exist) for some $S' \supseteq S$ with $S \in \mathcal{F}$, precisely the states of S' are visited infinitely often

iff (since $\mathcal{F}$ is closed under superloops) for some $S \in \mathcal{F}$, precisely the states of S are visited infinitely often

iff $\mathcal{A}$ accepts α.

$\Leftarrow$ Let the det. Büchi automaton $\mathfrak{B}$ with final state set F recognize $L(\mathfrak{A})$.

Show: The system of accepting loops of $\mathfrak{A}$ is closed under superloops ($\mathcal{F}_2 \subseteq \mathcal{F}$).

So we have to find $\alpha \in L(\mathfrak{A})$ which finally lets $\mathfrak{A}$ cycle through S'. For that matter pick $q \in S$, reached by $\mathfrak{A}$ via w. Continue w by γ such that $\mathfrak{A}$ loops through S and hence accepts. So $\mathfrak{B}$ on $w\gamma$ infinitely often visits F, say first after wu_1. Continuation via v_1 through S leads $\mathfrak{A}$ back to q, then a travel through the superloop S' via x_1 again back to q.

Repetition yields $wu_1v_1x_1u_2v_2x_2\ldots$ such that $\mathfrak{B}$ assumes a final state after each u_i; so $\mathfrak{A}$ accepts, and due to the x_i, $\mathfrak{A}$ visits the S'-states again and again. □

3.8 Staiger-Wagner Automata

For a run $\rho \in Q^\omega$ let $\text{Occ}(\rho) := \{q \in Q \mid \exists i : \rho(i) = q\}$. Guaranty and safety conditions can be described with $\text{Occ}(\rho)$.

Guaranty condition: $\exists i \rho(i) \in F \Leftrightarrow \mathrm{Occ}(\rho) \cap F \neq \emptyset$

Safety condition: $\forall i \rho(i) \in F \Leftrightarrow \mathrm{Occ}(\rho) \subseteq F$

Definition 3.33. A *Staiger-Wagner automaton* (SW-automaton) is of the form $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathrm{Acc})$ with Q, Σ, q_0, δ as defined earlier and Acc is a family $\mathcal{F}$ of sets of states (Notation: $\mathcal{F} = \{F_1, \ldots, F_k\}, F_i \subseteq Q$).

$\mathfrak{A}$ accepts α $:\Leftrightarrow$ $\mathrm{Occ}(\rho) \in \mathcal{F}$ (i.e. $\mathrm{Occ}(\rho) = F_1$ or ... or $\mathrm{Occ}(\rho) = F_k$) holds for the unambiguous run ρ of $\mathfrak{A}$ on α.

Idea: The Staiger-Wagner condition grasps options for state sets in accepting runs.

Remark 3.34. *The accepting component $\mathcal{F}$ of a Staiger-Wagner automaton only needs to include sets F which consist of*

- *a strongly connected component (SCC) P,*
- *a path from q_0 to P.*

Remark 3.35. *Deterministic E- and A-automata are special cases of SW-automata.*

Proof

a) Let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ be an E-automaton. Then the SW-automaton $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, \mathcal{F})$ with $\mathcal{F} = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$ is equivalent to $\mathfrak{A}$.

b) Let $\mathfrak{A}$ be an A-automaton as above. Then the SW-automaton $\mathfrak{A}'' = (Q, \Sigma, q_0, \delta, \mathcal{F}')$ with $\mathcal{F}' = \{P \subseteq Q \mid P \subseteq F\}$ is equivalent to $\mathfrak{A}$.

□

Question: Why is it not sufficient to define the SW-automaton as $\mathfrak{A}'' = (Q, \Sigma, q_0, \delta, \{F\})$? That would not be correct, because then a visit to every state in F would be mandatory, which is not always necessary.

Theorem 3.36. *The acceptance conditions, made up of Boolean combinations of guaranty conditions (or safety conditions), are exactly those which can be described by SW-conditions.*

Proof Consider the state space Q.

$\Leftarrow$ Consider the condition $\mathrm{Occ}(\rho) \in \mathcal{F}$, say for $\mathcal{F} = \{F_1, \ldots, F_k\}$, i.e. $\mathrm{Occ}(\rho) = F_1 \vee \cdots \vee \mathrm{Occ}(\rho) = F_k$.

$$\mathrm{Occ}(\rho) = F_j \text{ is equivalent to } \bigwedge_{q \in F_j} \underbrace{\exists i \rho(i) = q}_{\exists i \rho(i) \in \{q\}} \wedge \bigwedge_{q \in Q \setminus F_j} \neg \underbrace{\exists i \rho(i) = q}_{\exists i \rho(i) \in \{q\}}$$

We obtain a Boolean combination of guaranty conditions of the form $\exists i \rho(i) \in \{q\}$.

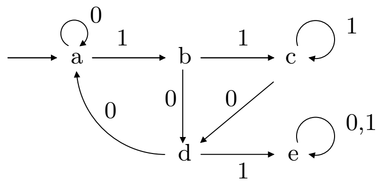
$\Rightarrow$ Consider a Boolean combination of guaranty conditions $\exists \rho(i) \in P_k$ (or $\mathrm{Occ}(\rho) \cap P_k \neq \emptyset$) for suitable sets $P_k \subseteq Q$. The DNF yields disjunction of conditions of the following kind (we denote the jth element of the disjunction with $(*)_j$):

$$\mathrm{Occ}(\rho) \cap P_{j1} \neq \emptyset \wedge \cdots \wedge \mathrm{Occ}(\rho) \cap P_{jm_j} \neq \emptyset \wedge \mathrm{Occ}(\rho) \cap P_{jm_{j+1}} = \emptyset \wedge \cdots \wedge \mathrm{Occ}(\rho) \cap P_{jn_j} = \emptyset$$

Call $F \subseteq Q$ *good* for the index j, if F, substituted for $\mathrm{Occ}(\rho)$, fulfills the condition $(*)_j$. Set $\mathcal{F} := \{F \subseteq Q \mid F \text{ is good for an index } j\}$. The SW-automaton with this $\mathcal{F}$ accepts iff the given Boolean combination is fulfilled.

□

Example 3.37. Let $L_4' = \{\alpha \in \mathbb{B}^\omega \mid$ 11 never occurs in α, or 101 occurs $\geq$ one time$\}$. We want to define the acceptance condition of $\mathfrak{A}$, so that L_4' is recognized.



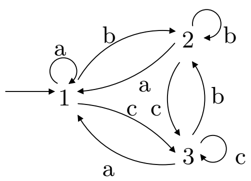
$\mathfrak{A}$ has got the following properties:

- If 101 occurs, then e is reached.
- If 101 does not occur, then the occurence of 11 is signaled by reaching c.

So we need to require that either e is visited or that c is never visited. Thus the system $\mathcal{F}$ of accepting sets precisely contains $\{a\}, \{a, b, d\}, \{a, b, d, e\}, \{a, b, c, d, e\}$. ⊠

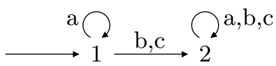
Remark 3.38. *There are SW-recognizable languages which cannot be recognized by a SW-automaton with only one set in its accepting component.*

Before proving the remark we give an example for which the reduction to an accepting component $\{F\}$ succeeds. Let $\Sigma = \{a, b, c\}$, $L = \{\alpha \in \Sigma^\omega \mid b$ or c occur in $\alpha\}$.



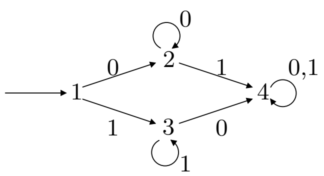
$\mathcal{F} = \{\{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$

In this case there are more than just one set F. But another SW-automaton only requires an simpler $\mathcal{F}$:



$\mathcal{F} = \{\{1, 2\}\}$

Proof of Remark 3.38 Consider $L = \{0^\omega, 1^\omega\}$.



$\mathcal{F} = \{\{1, 2\}, \{1, 3\}\}$

Assume: The SW-automaton $\mathfrak{A} = (Q, \mathbb{B}, q_0, \delta, \{F\})$ recognizes L, say with n states. Consider the run ρ_0 on 0^ω, which visits exactly the F-states. After 0^n, $p \in F$ is reached, and every state that is visited on 0^ω has already been visited. 1^ω is also accepted. Therefore exactly the F-states are visited, i.e. also p. From the state $p := \delta(q_0, 0^n)$ on the word 1^ω a subset of F is visited. Consider $0^n 1^\omega$. For this word precisely F is visited and therefore the word is accepted. Contradiction. □

Theorem 3.39. (Staiger, Wagner 1977) *An ω-language $L \subseteq \Sigma^\omega$ is SW-recognizable iff it is deterministically Büchi recognizable and deterministically co-Büchi recognizable.*

From Staiger-Wagner to Büchi Proof idea: Given a Staiger-Wagner automaton with state-set Q and acceptance component $\mathcal{F} = \{F_1, \dots, F_k\}$, we introduce an automaton $\mathcal{A}'$ with state space $Q \times 2^Q$.

In the first component, $\mathcal{A}'$ simulates $\mathcal{A}$. In the second component, $\mathcal{A}'$ accumulates the visited states. If this set coincides with some F_i, the state is declared final. Formally, a state (q, R) is declared final in $\mathcal{A}'$ if for some i we have $R = F_i$. We show

$$\mathcal{A} \text{ accepts } \alpha \text{ iff } \mathcal{A}' \text{ Büchi-accepts } \alpha \text{ iff } \mathcal{A}' \text{ co-Büchi-accepts } \alpha.$$

$\mathcal{A}'$ accepts α

iff $\mathcal{A}'$ on α visits infinitely often a final state

iff in the run of $\mathcal{A}'$ on α, infinitely often there is some i such that the visited states form the set F_i

iff for some i, infinitely often the visited states form the set F_i

iff $\mathcal{A}$ accepts α.

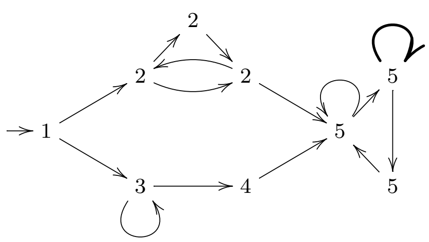
Note: Infinitely often the visited states form the set F_i iff from some point onwards the visited states form the set F_i. So for $\mathcal{A}'$ one may as well use the co-Büchi condition without changing the recognized ω-language.

For the converse we need some preparation:

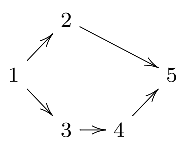
Recall: A *strongly connected component (SCC)* of (the transition graph of) $\mathcal{A}$ is a maximal strongly connected set, in other words a maximal loop of $\mathcal{A}$.

Remark 3.40. *The SCC's and the singletons which do not belong to a SCC form a partial order under the reachability relation.*

Example 3.41. The numbers indicate SCCs.



The partial order can be illustrated as follows:



⊠

Remark 3.42. *If $\mathcal{F}$ is closed under superloops and under subloops, then all loops of a SCC are accepting (in $\mathcal{F}$) or all rejecting (not in $\mathcal{F}$).*

Given a loop of $\mathcal{F}$ in the SCC S, S itself belongs to $\mathcal{F}$ (since $\mathcal{F}$ is closed under superloops) and hence all loops within S belong to $\mathcal{F}$ (since $\mathcal{F}$ is closed under subloops).

Call an SCC S accepting if all its loops are accepting.

From Büchi and co-Büchi to Staiger-Wagner Assume L is deterministically Büchi recognizable and deterministically co-Büchi recognizable.

Let $\mathcal{A}$ be a Muller automaton recognizing L, say with acceptance component $\mathcal{F}$. By Landweber's Theorem, $\mathcal{F}$ is closed under superloops and under subloops.

Any run ρ will finally remain within a certain SCC S.

For any SCC S, let S_+ be the set of states outside S and reachable from S by a single transition.

The run ρ will eventually stay in S if some state of S is visited in ρ but no state of S_+ is visited in ρ. So the Muller automaton $\mathcal{A}$ accepts α iff the run ρ of $\mathcal{A}$ on α satisfies the following:

> ρ reaches an accepting SCC S but does not visit one of the states in S_+.

So we may change the acceptance condition to the Staiger-Wagner condition with the following system $\mathcal{F}'$:

$$R \in \mathcal{F}' \quad :\Leftrightarrow \quad \begin{array}{l} \text{for some accepting SCC } S,\ R \cap S \neq \emptyset \\ \text{but } R \cap S_+ = \emptyset \end{array}$$

□

3.9 Parity Conditions

In a Muller automaton the accepting loops are enumerated (in an acceptance component $\mathcal{F}$). In a Rabin automaton the accepting loops are fixed by "bounds" (S is accepting iff S intersects some F_i but is disjoint from the corresponding E_i). Can one fix the accepting loops by a condition on their individual states? We use a "coloring" of states by numbers:

Definition 3.43. A *coloring* of Q is a function $c : Q \to \{0, \dots, k\}$. For a run ρ let $c(\rho)$ be the sequence of associated colors:

$$c(\rho) = c(\rho(0))c(\rho(1))\dots$$

Definition 3.44. (Weak and Strong Parity Automata) A (deterministic) *parity automaton* is an ω-automaton of the form $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$, where the acceptance component is a coloring $c : Q \to \{0, \dots, k\}$ for some natural number k.

A *weak parity automaton* is a parity automaton where a

> run ρ is successful if the maximal color occurring in ρ is even
> (formally: $\max(\mathrm{Occ}(c(\rho)))$ is even).

A *strong parity automaton* (sometimes just "parity automaton") is a parity automaton where a

> run ρ is successful if the maximal color occurring infinitely often in ρ is even
> (formally: $\max(\mathrm{Inf}(c(\rho)))$ is even).

Example 3.45. (Special cases)

An *E-automaton* with state set Q and final state set F amounts to a weak parity automaton with a coloring $c : Q \to \{1, 2\}$:

$$c(q) = \begin{cases} 1 & \text{for } q \notin F \\ 2 & \text{for } q \in F \end{cases}$$

A *Büchi automaton* can be presented similarly as a strong parity automaton with the same coloring.

An *A-automaton* (Q and F as before) amounts to a weak parity automaton with coloring $c : Q \to \{0, 1\}$:

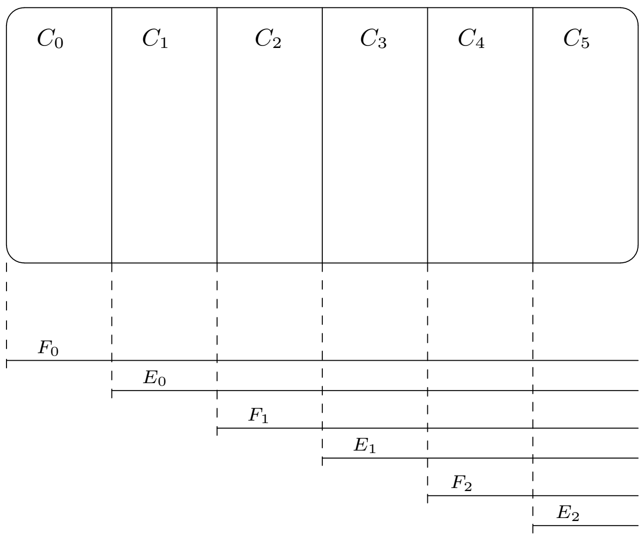
$$c(q) = \begin{cases} 0 & \text{for } q \in F \\ 1 & \text{for } q \notin F \end{cases}$$

⊠

Lemma 3.46. *Every deterministic parity automaton is equivalent to a deterministic Rabin automaton.*

Proof Given the parity automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, c)$ with $c : Q \to \{0, \ldots, k\}$, w.l.o.g. let k be odd. We write $C_i = \{q \mid c(q) = i\}$.

We define the sets $F_0, E_0, \ldots, F_k, E_k$ according to the following scheme:



thus $\left.\begin{array}{l} F_j = \{q \in Q \mid c(q) \geq 2j\} \\ E_j = \{q \in Q \mid c(q) \geq 2j+1\} \end{array}\right\} \quad j = 0 \ldots k$

The maximal infinitely often visited color is then $= 0$, if $\mathrm{Inf}(\rho) \cap F_0 \neq \emptyset, \mathrm{Inf}(\rho) \cap E_0 = \emptyset$,
$= 1$, if $\mathrm{Inf}(\rho) \cap F_1 = \emptyset, \mathrm{Inf}(\rho) \cap E_1 \neq \emptyset$,
...

Therefore $F_0 \supseteq E_0 \supseteq F_1 \supseteq E_1 \supseteq \cdots \supseteq F_k \supseteq E_k$ holds and

$$\max(\mathrm{Inf}(c(\rho))) \text{ even } \Leftrightarrow \bigvee_{j=0}^{r} (\mathrm{Inf}(\rho) \cap F_j \neq \emptyset \wedge \mathrm{Inf}(\rho) \cap E_j = \emptyset).$$

We obtain an equivalent Rabin automaton $\mathfrak{B} = (Q, \Sigma, q_0, \delta, \Omega)$ ($\Omega = \{(E_0, F_0), \ldots, (E_k, F_k)\}$). Because of the inclusion chain F_i, E_i also called *Rabin chain automaton* with accepting component $\Omega = ((E_0, F_0), \ldots, (E_r, F_r))$. $\square$

Aim:

- Weak parity automata have the same expressive power as Staiger-Wagner automata.
- Strong parity automata have the same expressive power as Muller automata.

From Parity to Staiger-Wagner and Muller Consider an automaton with coloring $c : Q \to \{0, \ldots, k\}$. Let $C_i = \{q \in Q \mid c(q) = i\}$.

The weak parity condition is a Boolean combination of E-acceptance conditions for a run ρ:

$$\bigvee_{j \text{ even}} \exists i \big(\rho(i) \in C_j \ \wedge \ \neg \exists i \rho(i) \in C_{j+1} \cup \ldots \cup C_k\big)$$

Similarly the strong parity condition is a Boolean combination of Büchi acceptance conditions.

Consequences:

- A weak parity automaton can be simulated by a Staiger-Wagner automaton.
- A strong parity automaton can be simulated by a Muller automaton.

Theorem 3.47. (From Staiger-Wagner to weak parity) *For a Staiger-Wagner automaton one can construct an equivalent weak parity automaton.*

Proof Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a Staiger-Wagner automaton. We define an equivalent weak parity automaton $\mathcal{A}' = (Q', \Sigma, q_0', \delta', c)$. Set $Q' = Q \times 2^Q, \quad q_0' = (q_0, \{q_0\})$.

Idea: Collect the visited states in the second component. Define $\delta'((p, R), a) = (\delta(p, a), R \cup \{\delta(p, a)\})$.

The coloring c is defined by

$$c(p, R) = \begin{cases} 2 \cdot |R| & \text{if } R \in \mathcal{F} \\ 2 \cdot |R| - 1 & \text{if } R \notin \mathcal{F} \end{cases}$$

Colors of a run increase monotonically, and from some point onwards stay constant (when all visited states have been seen at least once).

The maximal color is even iff the set of visited states belongs to $\mathcal{F}$. So $\mathcal{A}'$ is equivalent to $\mathcal{A}$. $\square$

Theorem 3.48. (From Muller automata to parity automata) *For a Muller automaton one can construct an equivalent strong parity automaton.*

Proof Idea: Extend the idea of "recording past states". Remember not only the *set* of visited states, but also the *order of their last occurrence.* The data structure for this information is called "Order vector" (McNaughton 1965), "Latest appearance record", short "LAR" (Gurevich, Harrington 1982).

The vector has the current state on position 1, the next previous state on position 2, etc. The position where the current state was taken from is marked as "hit position".

The complete proof will be given later on.

Example 3.49. $Q = \{1, 2, 3, 4\}$

run ρ:	LAR-run ρ':	underlined: hit
1	$\underline{1}234$	
3	$31\underline{2}4$	
4	$431\underline{2}$	
2	$243\underline{1}$	
3	$32\underline{4}1$	
1	$132\underline{4}$	
3	$3\underline{1}24$	
3	$\underline{3}124$	
1	$1\underline{3}24$	

⊠

3.10 Exercises

Exercise 3.1. Consider the Büchi automaton $\mathcal{A} = (\{0, 1, 2\}, \{a, b\}, 0, \Delta, \{1\})$ with Δ given by the following transition table:

	a	b
0	0,1	0
1		2
2	2	1

Construct, using the Safra construction, an equivalent deterministic Muller automaton.

Exercise 3.2. Let $L \subseteq \Sigma^\omega$ be an ω-language. We define the *right congruence* $\sim_L \subseteq \Sigma^* \times \Sigma^*$ by

$$u \sim_L v \text{ iff } \forall \alpha \in \Sigma^\omega : \ u\alpha \in L \Leftrightarrow v\alpha \in L.$$

(a) Show that every deterministic Muller automaton recognizing L needs at least as many states as there are $\sim_L$ equivalence classes.

(b) Show that there is a non-regular ω-language L such that $\sim_L$ has finite index. (So the Nerode characterization of regular languages does not generalize to ω-languages.)
Hint: Let β be an ω-word which is not ultimately periodic and consider

$$L(\beta) := \{\alpha \in \Sigma^\omega \mid \alpha \text{ and } \beta \text{ have a common suffix}\}.$$

Exercise 3.3. Starting from Exercise 3.2 define a family of ω-languages $(L_n)_{n\geq 2}$ with the following properties.

1. L_n is recognized by a nondetermistic Büchi automaton with $\mathcal{O}(n)$ states.
2. Every deterministic Muller automaton that recognizes L_n has got at least 2^n states.

Exercise 3.4. Let UP be the set of all ω-words over $\{0, 1\}$ that are ultimately periodic. Show that UP is not regular.

Exercise 3.5. Show that there is a regular ω-language $L \subseteq \{a, b\}^\omega$, which cannot be recognized by a deterministic Muller automaton $\mathcal{A} = (Q, \{a, b\}, q_0, \delta, \mathcal{F})$ with $|\mathcal{F}| = 1$.

Exercise 3.6. Let $\mathcal{A}_1 = (Q_1, \Sigma, q_0^1, \delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0^2, \delta_2, F_2)$ be deterministic co-Büchi automata.

(a) Show that the product automaton $\mathcal{A}$ of $\mathcal{A}_1$ and $\mathcal{A}_2$ with final states $(F_1 \times Q_2) \cup (Q_1 \times F_2)$ does in general not recognize the language $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

(b) Correct the construction from (a) such that the new automaton $\mathcal{A}'$ recognizes $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Exercise 3.7. Let $L \subseteq \Sigma^\omega$ be an ω-language. Show:

(a) If L is deterministically A-recognizable, then L is deterministically co-Büchi recognizable.
(b) If L is deterministically E-recognizable, then L is deterministically co-Büchi recognizable.
(c) If L is deterministically co-B"uchi recognizable, then L is nondeterministically Büchi recognizable.

Exercise 3.8. Consider the ω-language

$$L_3 := \{\alpha \in \{0,1\}^\omega \mid \alpha \text{ contains 00 infinitely often, but 11 only finitely often}\}.$$

(a) Show that L_3 is Büchi recognizable.

(b) Show that L_3 is neither recognizable by a deterministic Büchi automaton nor by a deterministic co-Büchi automaton.

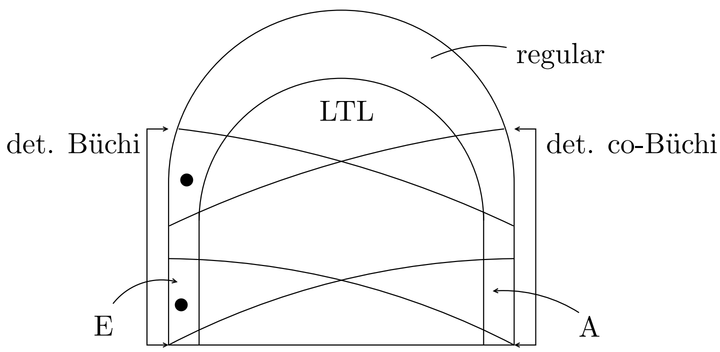
Exercise 3.9. Let $U \subseteq \Sigma^*$ be a finite language, and $L := U \cdot \Sigma^\omega$.

(a) Show that L is both E- and A-recognizable.

(b) Show the converse: If an ω-language $L \subseteq \Sigma^\omega$ is both E- and A-recognizable then there is finite language $U \subseteq \Sigma^*$ such that $L = U \cdot \Sigma^\omega$.

This shows that bounded specifications are captured by ω-languages which are both E- and A-recognizable.
Hint: For (b) it is useful to show that the complement of L is also E-recognizable. Then consider, for a proof by contradiction, Σ^* as a $|\Sigma|$-branching tree and apply König's Lemma.

Exercise 3.10. The inclusion diagram shows the LTL-definable languages inside the hierarchy of ω-languages.



(a) Show that the languages $L_4 := \mathbb{B}^*1\mathbb{B}^\omega$, $L_5 := \{0^\omega\}$, $L_6 := (0^*1)^\omega$, and $L_7 := \mathbb{B}^*0^\omega$ are LTL-definable, i.e. these languages are located in the inner part of the diagram.

(b) Partly verify the inclusion diagram by providing ω-languages for the two language classes marked by dots.
Hint: Find counting versions of the appropiate LTL-definable ω-languages mentioned in (a).

Exercise 3.11.

(a) Construct Staiger-Wagner automata accepting the ω-languages

$$L_1 := \{\alpha \in \{a,b,c\}^\omega \mid \text{ if } a \text{ occurs in } \alpha \text{ then } b \text{ occurs later on}\}$$

and

$$L_2 := \{\alpha \in \{a,b,c\}^\omega \mid \alpha \text{ contains } aa \text{ and before that } b \text{ only occurs in blocks of length } \leq 2\}.$$

(b) Let $\mathcal{A}_1 = (Q_1, \Sigma, q_0^1, \delta_1, \mathcal{F}_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0^2, \delta_2, \mathcal{F}_2)$ be Staiger-Wagner automata. Construct the Staiger-Wagner product automaton $\mathcal{A}_3$ recognizing $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. Verify your construction.

Exercise 3.12. Show that for every $n \geq 1$ there is an ω-language L_n which can be recognized by a Staiger-Wagner automaton with n state sets as its accepting component. Also show that the language cannot be recognized by a Staiger-Wagner automaton with less than n state sets in its accepting component.

(a) For that matter use the alphabet $\Sigma_n = \{a_1, \ldots, a_n\}$ and the language $L_n = a_1^\omega + \cdots + a_n^\omega$.

(b) Extend the result of (a) to languages over an alphabet with two elements.

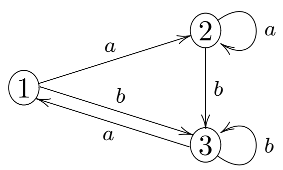
Exercise 3.13. Show that the language, which is defined by the ω-regular expression $(0^*1)^\omega$, is not Staiger-Wagner recognizable. (Consider, assuming that such an SW-automaton with n states exists, the ω-word $(0^n1)^\omega$ in order to derive a contradiction.)

Exercise 3.14. Directly construct an equivalent deterministic Büchi automaton $\mathcal{B}$ for a Staiger-Wagner automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$. Hint: In order to simulate $\mathcal{A}$, $\mathcal{B}$ needs to memorize the visited states.

Exercise 3.15. A set $\mathcal{F} \subseteq 2^Q$ is *closed under subloops* if every subloop $S' \subseteq S$ of a loop $S \in \mathcal{F}$ also belongs to $\mathcal{F}$. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a Muller automaton. Show that

$$L(\mathcal{A}) \text{ is co-Büchi recognizable} \iff \mathcal{F} \text{ is closed under subloops.}$$

Exercise 3.16. Decide whether the language recognized by the following Muller automaton is E-recognizable or Büchi recognizable. Let $\mathcal{F} = \{\{2\}, \{1,2,3\}\}$.



If your answer is positive specify suitable automata with E- and Büchi acceptance conditions.

Exercise 3.17.

(a) Find an ω-language that is recognized by a parity automaton with colorset $\{1, 2, 3\}$ but not by a parity automaton with a colorset $\{1, 2\}$. (Hint: Landweber's Theorem 3.32 for (deterministic) Büchi automata).

(b) Propose a family L_n of ω-languages, such that L_n is recognized by a parity automaton with colorset $\{1, \ldots, n\}$ but not by parity a automaton with color set $\{1, \ldots, n-1\}$.

Exercise 3.18.

(a) Present (by direct construction) weak parity automata recognizing the ω-languages L_1, L_2 from Exercise 3.11.

(b) Show that L_1 cannot be recognized by a weak parity automaton with only two colors.