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:

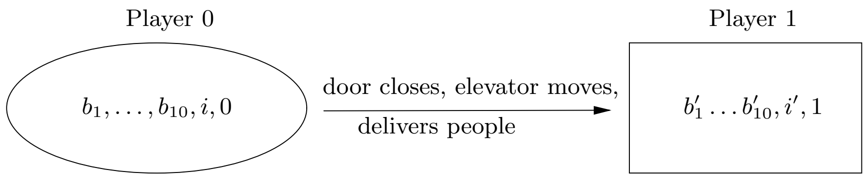
1. A set of level numbers that are requested by pushing a button (either on the respective floor or in the elevator). This set is represented by a bitvector $(b_1, \dots, b_{10})$ (with $b_i = 1 \Leftrightarrow$ level i is requested.)
2. A level number for the position of the elevator ($i \in \{1, \dots,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

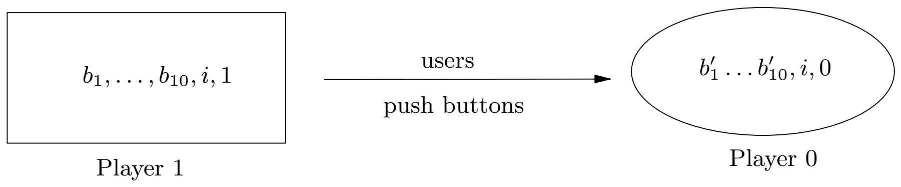
$$\mathbb{Z} = \mathbb{B}^{10} \times \{1, \dots, 10\} \times \{0, 1\}.$$

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.


Player 0
$b_1, \ldots, b_{10}, i, 0$
door closes, elevator moves,
delivers people
Player 1
$b'_1 \ldots b'_{10}, i', 1$

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. ...

⊠

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 $u\alpha$.

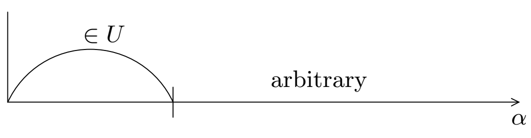
The concatenation of two languages is defined likewise:

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

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:



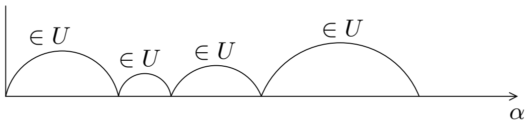
Example 1.1. Let $U_1 = 0110^* + (00)^+$. We obtain

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

⊠

2. $U^\omega := \{\alpha \in \Sigma^\omega \mid \alpha = u_0 u_1 u_2 \dots \text{ with } u_i \in U\}$

   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

$$\alpha = 0001100110000000\dots$$

Another word in U^ω

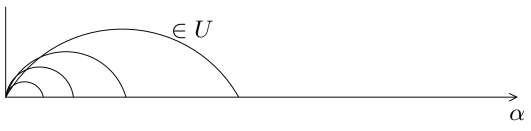
$$\alpha = 011001100011000 01\dots$$

⊠

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

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

   Visualization:



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

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

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.

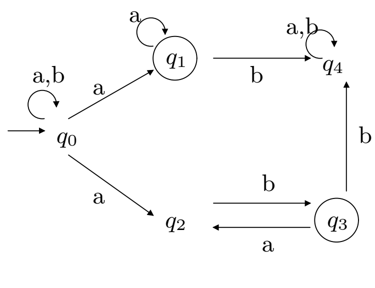
   A *run* of $\mathfrak{A}$ on α is a sequence of states $\rho = \rho(0)\rho(1)\cdots$ with $\rho(0) = q_0$ and $(\rho(i), \alpha(i), \rho(i+1)) \in \Delta$ for $i \geq 0$.

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 $\alpha = abbaababab a\ldots$, 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

$\mathfrak{A}$ *accepts* $\alpha \Leftrightarrow$ ex. a run ρ of $\mathfrak{A}$ on α 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

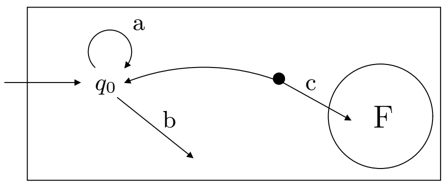
We will now, for the case of $U \subseteq \Sigma^*$ being regular, specify ω-automata for the ω-languages U^ω and $\lim U$.

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

Proof

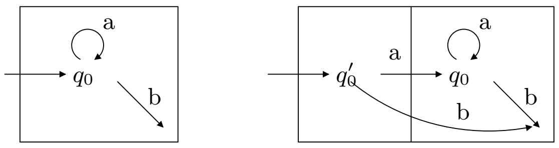
a) Consider an NFA $\mathfrak{A} = (Q, \Sigma, q_0, \Delta, F)$ that recognizes U.

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 $\mathfrak{B}$.

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

□

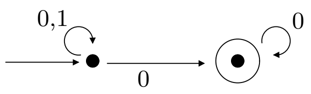
Note that the converse of Theorem 1.10(b) also holds. Every ω-language recognized by a deterministic Büchi automaton is of the form $\lim U$ for a regular language U.

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*

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

Proof

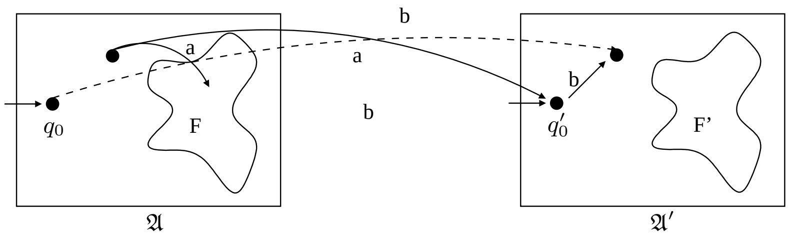
$\Leftarrow$ It suffices to show:

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 $\alpha = u_0 u_1 u_2 \dots$, with $u_0 \in U_{q_0 q}$, $u_1 \in U_{qq}$, $u_2 \in U_{qq}, \dots$, possible. Therefore the following holds.

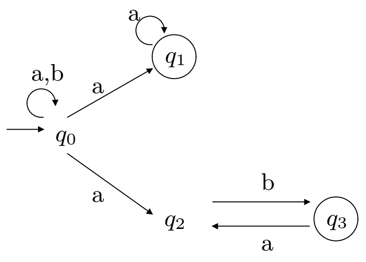
$$\mathfrak{A} \text{ accepts } \alpha \Leftrightarrow \text{ ex. } q \in F \text{ with } \alpha \in U_{q_0 q} \cdot U_{qq}^\omega \Leftrightarrow \alpha \in \bigcup_{q \in F} U_{q_0 q} \cdot U_{qq}^\omega$$

□

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.) $\square$

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).

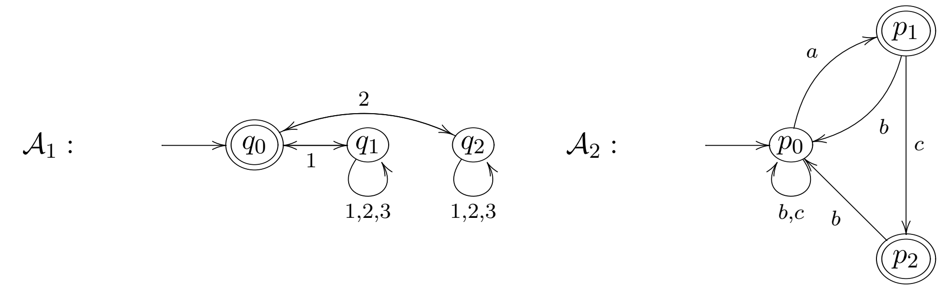
Hint: Assume that L is of the form $U \cdot V^\omega$ and consider the words in V.

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 $p_1, \dots, p_n$ are considered.

Definition 2.2. A *Kripke structure* over $p_1, \dots, p_n$ 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" $\lambda : S \to 2^{\{p_1,\dots,p_n\}}$, 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 $(b_1, \dots, b_n)$ 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.

$$\begin{tikzcd}
 & & \binom{1}{0} \arrow[r] \arrow[dd, bend left] & \binom{0}{0} \arrow[loop right] \\
{} \arrow[r] & \binom{1}{1} \arrow[ur] \arrow[dr] & & \\
 & & \binom{0}{1} \arrow[uu, bend left] &
\end{tikzcd}$$

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:

- Xp for "p is true next time",
- Fp for "eventually (sometime, including present) p is true",
- Gp for "always (from now onwards) p is true",
- p_1Up_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."

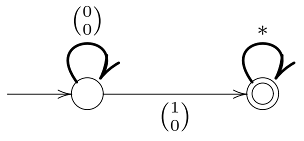
$$\mathrm{G}(p_1 \rightarrow \mathrm{X}(p_2\mathrm{U}p_1))$$

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

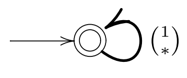
$$\mathrm{GF}p_1 \rightarrow \mathrm{GF}p_2$$

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.

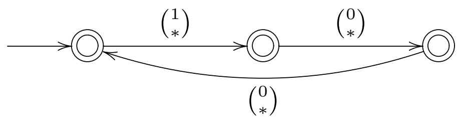
$\mathrm{F}p_1$:



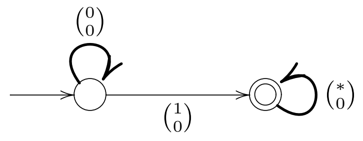
$\mathrm{G}P_1$:



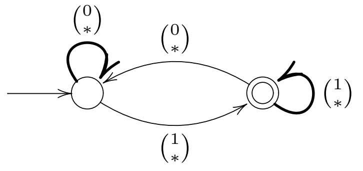
$p_1 \wedge \mathrm{X}\neg p_1 \wedge \mathrm{XX}\neg p_1 \wedge \mathrm{G}(p_1 \leftrightarrow \mathrm{XXX}p_1)$:



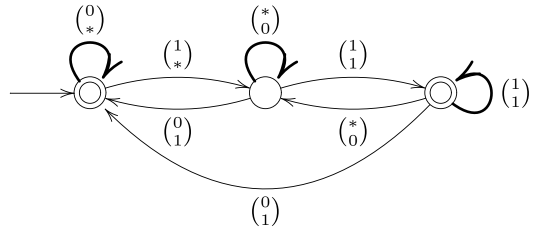
$\mathrm{F}p_1 \wedge \neg\mathrm{F}p_2$:



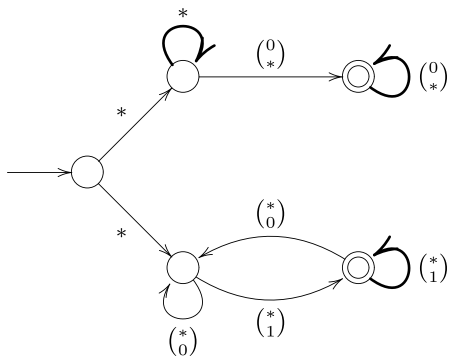
$\mathrm{GF}p_1$:



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



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



We leave $\mathrm{G}(p_1 \rightarrow \mathrm{X}(p_2 \mathrm{U} p_1))$ 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 $p_1, \dots, p_n$ 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 $p_1, \dots, p_n$ 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 $p_1, \dots, p_n$ 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 $\mathrm{GF}p_1$, $\mathrm{XX}(p_2 \rightarrow \mathrm{F}p_1)$, $\mathrm{F}(p_1 \wedge \mathrm{X}(\neg p_2 \mathrm{U} p_1))$, and the following Kripke structure:

$$\begin{tikzcd}
 & & \binom{1}{0} \arrow[r] \arrow[dd, bend left] & \binom{0}{0} \arrow[loop right] \\
{} \arrow[r] & \binom{1}{1} \arrow[ur] \arrow[dr] & & \\
 & & \binom{0}{1} \arrow[uu, bend left] &
\end{tikzcd}$$

We see that $\mathrm{GF}p_1$ fails, $\mathrm{XX}(p_2 \rightarrow \mathrm{F}p_1)$ 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".

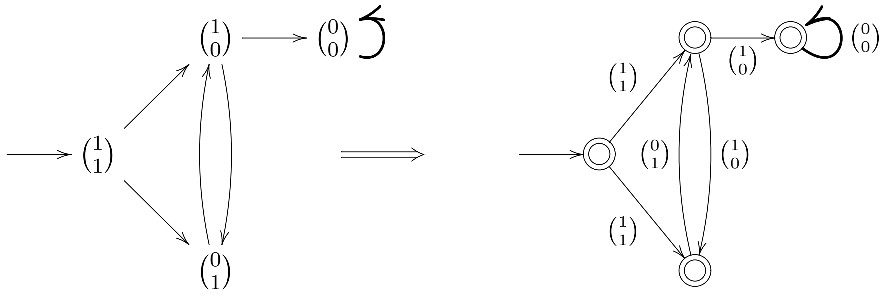
We already know algorithms for items 3. and 4. Items 1. and 2. still need to be taken care of.

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

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	$\ldots$

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:

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

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 $(\beta(l))_{j_2} = 0$. $\square$

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

$$\begin{tikzcd}
 & & \underset{\binom{1}{0}}{x_1} \arrow[dr] & & \underset{\binom{0}{0}}{x_2} \arrow[dr] & & \underset{\binom{1}{0}}{x_3} \arrow[dr] & \\
{} \arrow[r] & \underset{\binom{0}{0}}{y_0} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_1} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_2} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_3} \arrow[loop right] \\
 & & \overset{\binom{0}{1}}{\neg x_1} \arrow[ur] & & \overset{\binom{1}{1}}{\neg x_2} \arrow[ur] & & \overset{\binom{0}{1}}{\neg x_3} \arrow[ur] &
\end{tikzcd}$$

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

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 $\psi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$.

$$\begin{tikzcd}
 & & \underset{\binom{1}{0}}{x_1} \arrow[dr] & & \underset{\binom{0}{0}}{x_2} \arrow[dr] & & \underset{\binom{1}{0}}{x_3} \arrow[dr] & \\
{} \arrow[r] & \underset{\binom{0}{0}}{y_0} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_1} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_2} \arrow[ur] \arrow[dr] & & \underset{\binom{0}{0}}{y_3} \arrow[loop right] \\
 & & \overset{\binom{0}{1}}{\neg x_1} \arrow[ur] & & \overset{\binom{1}{1}}{\neg x_2} \arrow[ur] & & \overset{\binom{0}{1}}{\neg x_3} \arrow[ur] &
\end{tikzcd}$$

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 $(\mathcal{M}, s)_\psi \models \mathrm{G}\neg p_1 \vee \mathrm{G}\neg p_2$ $(= \varphi_\psi)$. □

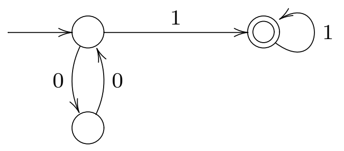
The translation from LTL to Büchi automata showed:

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 $n \geq n_0$)

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

$\varphi = \psi_1 \mathrm{U} \psi_2$ **:** 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 $n \geq n_0$:

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

More precisely:

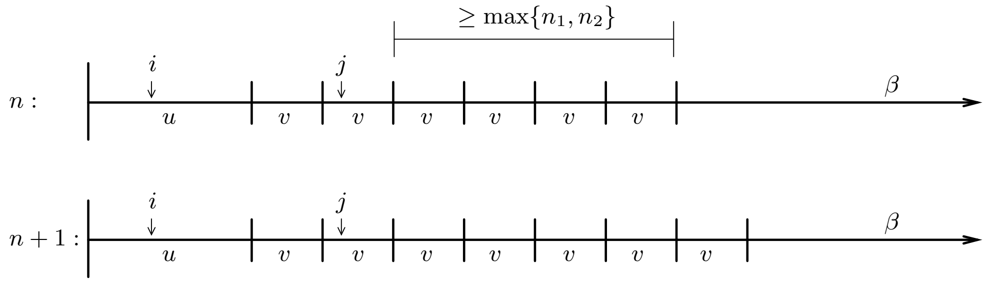
for some j: $(uv^n\beta)^j \models \psi_2$ and for every $i < j$: $(uv^n\beta)^i \models \psi_1$
iff for some j: $(uv^{n+1}\beta)^j \models \psi_2$ and for every $i < j$: $(uv^{n+1}\beta)^i \models \psi_1$.

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

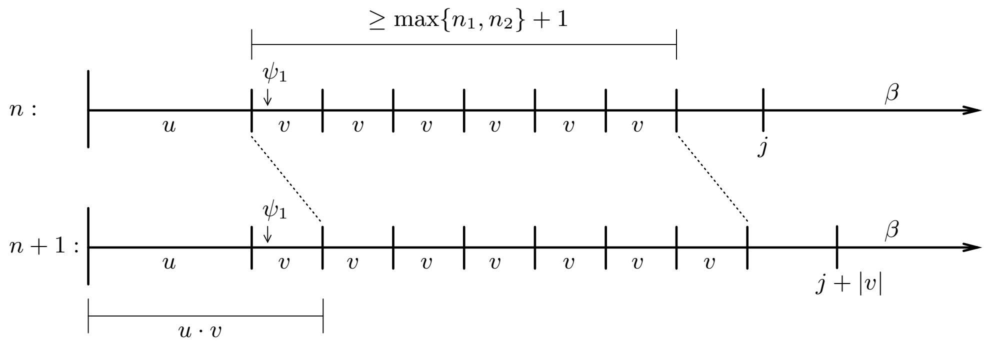
$$(uv^n\beta)^j \models \psi_2 \Leftrightarrow (uv^{n+1}\beta)^j \models \psi_2.$$

Therefore

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

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: $\varphi(X_1, \dots, X_n)$ indicates that at most the variables $X_1, \dots, X_n$ 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 $(\mathbb{N}, 0, +1, <, P_1, \dots, P_n) \models \varphi(X_1, \dots, X_n)$ if φ is true in this semantics, with $P_1 \subseteq \mathbb{N}, \dots, P_n \subseteq \mathbb{N}$ as interpretations of $X_1, \dots, X_n$. Therefore we need only specify $\overline{P} = P_1, \dots, P_n$. $\overline{P}$ 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: $\alpha(\overline{P}) \models \varphi(X_1, \dots, X_n)$.

Example 2.30. (Satisfaction of a S1S-formula)

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

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

	t	0	1	2	3	4	5	6	...
$\alpha(P_1, P_2)$:	P_1	1	0	1	0	1	0	1	...
	P_2	0	0	1	1	0	1	0	...
		t_1		t_2					

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 $\varphi(X_1, \dots, X_n)$ we have

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

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

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

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

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

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

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

⊠

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)

$$\begin{tikzcd} {} \arrow[r] & q_1 \arrow[r, bend left, "1"] & q_2 \arrow[l, bend left, "1"] \arrow[r, "0"] & q_3 \arrow[loop right, "1"] \end{tikzcd}$$

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

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

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

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). □

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:

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "1"] & q_1 \arrow[loop below, "0"] \arrow[r, bend right, "1"'] & \textcircled{q_2} \arrow[l, bend right, "0"']
\end{tikzcd}$$

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

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

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:

$$\begin{tikzcd}
\mathcal{A} : \arrow[r] & q_0 \arrow[r, bend left, "1"] & q_1 \arrow[l, bend left, "0"] \arrow[r, bend left, "1"] & q_2 \arrow[l, bend left, "0"]
\end{tikzcd}$$

(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

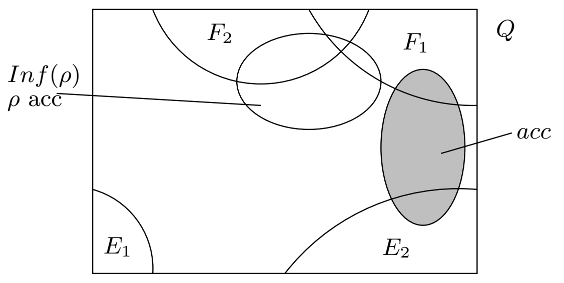
3.1 Deterministic Omega-Automata

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.

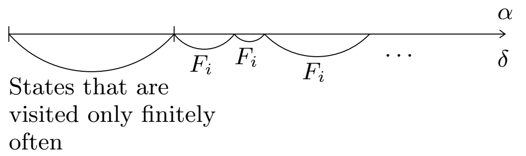
A part of the induction step: $L \subseteq \Sigma^\omega$ det. Muller recognizable $\Rightarrow \Sigma^\omega \setminus L$ det. Muller recognizable. If L is recognized by $(Q, \Sigma, q_0, \delta, \mathcal{F})$ then $\Sigma^\omega \setminus L$ will be recognized by $(Q, \Sigma, q_0, \delta, 2^Q \setminus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

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 $j \in \{1, \ldots, k\}$)

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

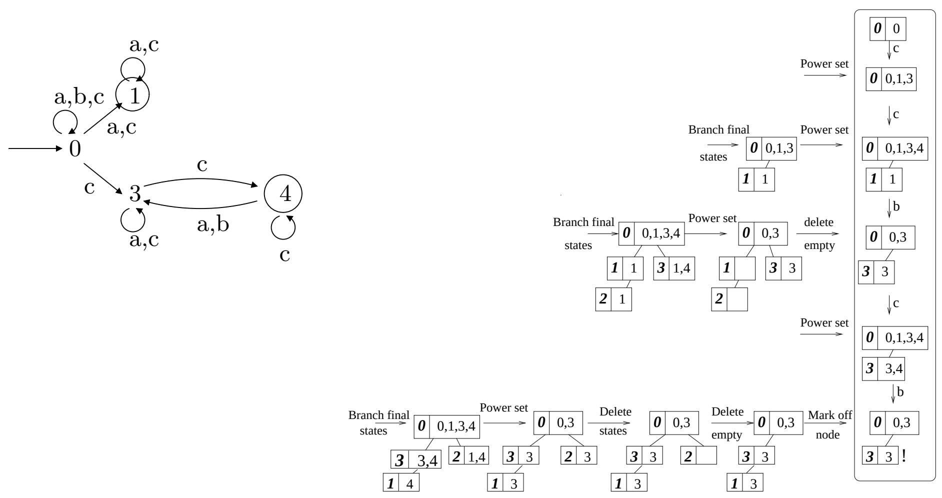
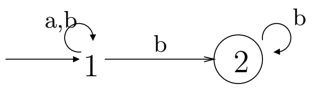


Figure 3.1: The column on the right, read top down, is the sequence of Safra trees for the given automaton on *ccbcb*. The intermediate steps are shown on the left. Within a node, the name of the node is on the left and the label on the right.

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 $\{1, \dots, 2|Q|\}$, 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*

$$\begin{tikzcd}
R_0 \arrow[r, rightsquigarrow, "u_1"] & P_1 \arrow[r, rightsquigarrow, "v_1"] \arrow[d, phantom, sloped, "\supseteq"] & R_1! \arrow[r, rightsquigarrow, "u_2"] \arrow[d, phantom, sloped, "="] & P_2 \arrow[r, rightsquigarrow, "v_2"] \arrow[d, phantom, sloped, "\supseteq"] & R_2! \arrow[d, phantom, sloped, "="] & \ldots & P_i \arrow[r, rightsquigarrow, "v_i"] \arrow[d, phantom, sloped, "\supseteq"] & R_i! \arrow[d, phantom, sloped, "="] \\
 & F_1 \arrow[r, rightsquigarrow, "v_1"] & Q_1 & F_2 \arrow[r, rightsquigarrow, "v_2"] & Q_2 & & F_i \arrow[r, rightsquigarrow, "v_i"] & Q_i
\end{tikzcd}$$

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 $u_1v_1u_2v_2 \ldots u_iv_i$ state r with $\geq i$ visits in final states.

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

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

Proof Consider the tree of states that is formed by runs from R_0 to r via $u_1v_1 \ldots u_iv_i$, 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 $u_1v_1 \ldots u_iv_i$. $\square$

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.

□

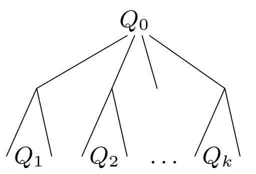
3.3 Complexity Analysis of the Safra Construction

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.

$$\begin{aligned}\text{The number of Safra trees is therefore } &\leq \text{number of quadrupels of those functions}\\ &\leq (2n+1)^n \cdot 2^{2n} \cdot (2n+1)^{2n} \cdot (2n+1)^{2n}\\ &\leq (2n+1)^{7n} \in 2^{O(n \log n)}.\end{aligned}$$

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

$$\begin{aligned}E_k &:= \text{Set of all Safra trees without the node } k,\\ F_k &:= \text{Set of all Safra trees with the node } k \text{ marked with "!"}.\end{aligned}$$

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:

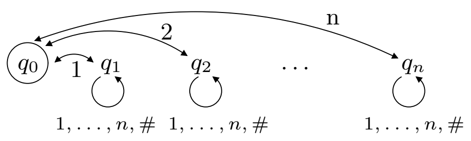
Theorem 3.12. (M. Michel 1988, C. Löding 1998) *There is no translation of nondeterministic Büchi automata with $O(n)$ states into deterministic Rabin automata with $2^{O(n)}$.*

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

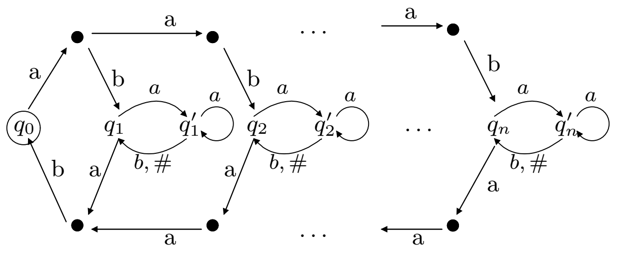
1. Specify a family $(L_n)_{n\geq 1}$ of ω-languages $L_n \subseteq \{1,\dots,n,\#\}^\omega$, which is recognized by a Büchi automaton with $O(n)$ states.

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 $\Sigma = \{1,\dots,n,\#\}$. 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 $1 \to ab$, $2 \to a^2b$, …, $n \to a^nb$, $\# \to \#$. The following Büchi automaton, where the states $q_0, q_1, q_2, \dots, q_n$ are initial, recognizes L_n.*



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

Proof

$\Leftarrow$ Let $(*)$ hold for $i_1,\dots,i_k$. 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 $i_2i_3, i_3i_4,\dots$ in the cycle $i_1,i_2,\dots,i_k,i_1$. 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. □

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.*

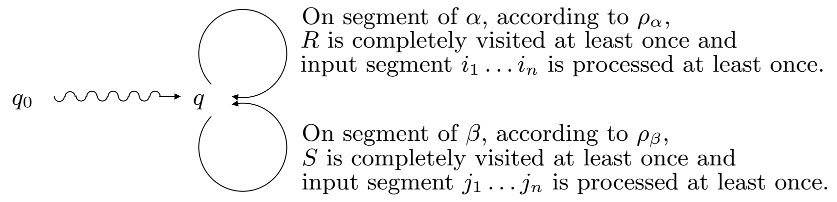
Proof Assume ρ_1, ρ_2 are not successful and ρ is successful. Then there exists an $i \in \{1, \dots, k\}$ with $\mathrm{Inf}(\rho) \cap E_i = \emptyset$ and $\mathrm{Inf}(\rho) \cap F_i \neq \emptyset$. Because of $\mathrm{Inf}(\rho_1) \cup \mathrm{Inf}(\rho_2) = \mathrm{Inf}(\rho)$, $\mathrm{Inf}(\rho_1) \cap E_i = \mathrm{Inf}(\rho_2) \cap E_i = \emptyset$ 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:



On segment of α, according to ρ_α,
R is completely visited at least once and
input segment $i_1 \dots i_n$ is processed at least once.

On segment of β, according to ρ_β,
S is completely visited at least once and
input segment $j_1 \dots j_n$ is processed at least once.

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:

$$\begin{array}{cccc} i_1 & \dots & i_{k-1} & i_k \\ \parallel & & \parallel & \nparallel \\ j_1 & & j_{k-1} & j_k \end{array}$$

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.

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

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).

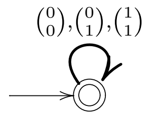
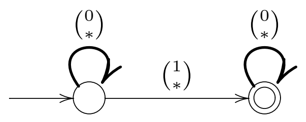
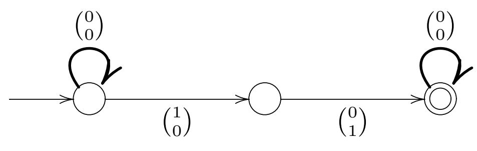
- The *successor function* only occurs in formulas of type $x' = y$: Instead of $X(s'')$ write
$$\exists y \exists z(s' = y \wedge y' = z \wedge X(z)).$$
- Eliminate the use of first-order variables by using different atomic formulas:
$$X \subseteq Y, \quad \mathrm{Sing}(X), \quad \mathrm{Succ}(X, Y),$$
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
$X_1 \subseteq X_2$, $\mathrm{Sing}(X_1)$, $\mathrm{Succ}(X_1, X_2)$,
- 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 $\exists X \varphi(X, X_1, \dots, X_n)$.
   Assume $\mathcal{A}$ is a Büchi automaton equivalent to $\varphi(X, X_1, \dots, X_n)$. In $\mathcal{A}$, change each transition label $(b, b_1, \dots, b_n)$ into $(b_1, \dots, b_n)$; 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 $\mathcal{A}$.

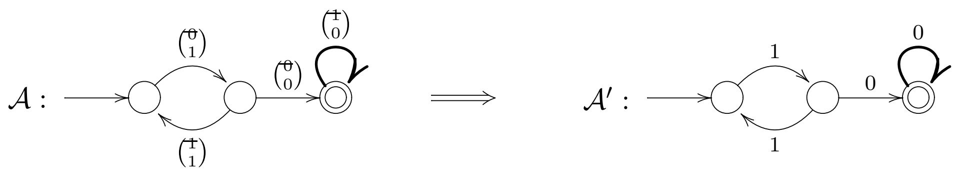
   $\mathcal{A}'$ accepts $\alpha \in (\mathbb{B}^n)^\omega$

   iff there exists a bit sequence $c_0c_1\dots$ such that $(c_0, \alpha(0)), (c_1, \alpha(1))\dots$ is accepted by $\mathcal{A}$

   iff $\exists c_0c_1\dots$ such that $\mathcal{A}$ accepts $(c_0, \alpha(0)), (c_1, \alpha(1))\dots$

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

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

$\forall X\,\exists Y(\forall t(X(t) \to Y(t)))$	true
$\forall X\,\exists t\,\forall s(X(s) \to s < t)$	false
$\forall X(X(0) \wedge \forall s(X(s) \to X(s')) \;\to\; \forall t X(t))$	true

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.

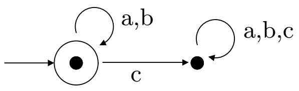
Definition 3.23. Given a deterministic automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$,

- $\mathfrak{A}$ *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 $\forall i : \rho(i) \in F$.

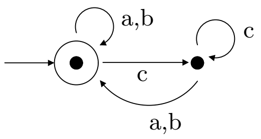
- $\mathfrak{A}$ *Büchi-accepts* $\alpha \Leftrightarrow$ exists a run, so that $\forall j \exists i \geq j : \rho(i) \in F$.
- $\mathfrak{A}$ *co-Büchi-accepts* $\alpha \Leftrightarrow$ exists a run ρ of $\mathfrak{A}$ on α, so that for almost all i (except of finitely many, written: $\forall^\omega i$) $\rho(i) \in F$ 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



⊠

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 $\mathfrak{B}$.

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

□

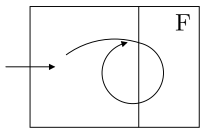
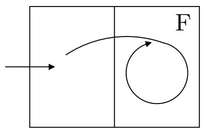
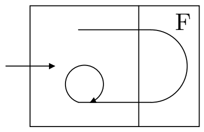
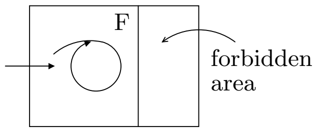
Büchi acceptance	**co-Büchi acceptance**
grasps “recurrence properties”	grasps “persistence properties”
Illustration: 	Illustration: 
System assumes desired states again and again	System finally assumes desired states only
E-acceptance	**A-acceptance**
grasps “guaranty properties”	grasps “safety properties”
Illustration: 	Illustration: 
System assumes desired state sometime	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.

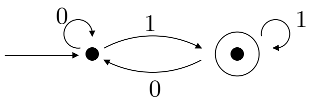
Proof

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.

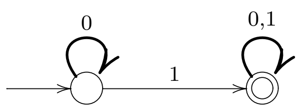
3. $(0^*1)^\omega$ is Büchi recognizable but not co-Büchi recognizable.

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

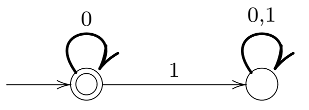
1. E-recognizability is clear.



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

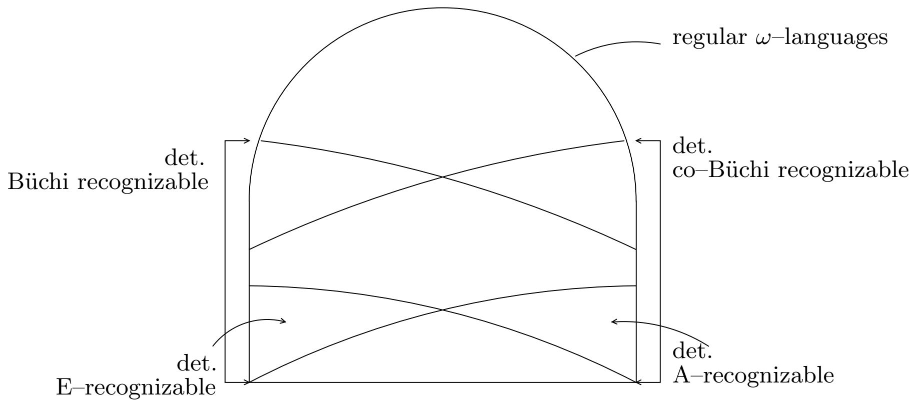


3. $(0^*1)^\omega$ being Büchi recognizable was shown for Theorem 3.27(b). It is easy to show that this language is not co-Büchi recognizable

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

□

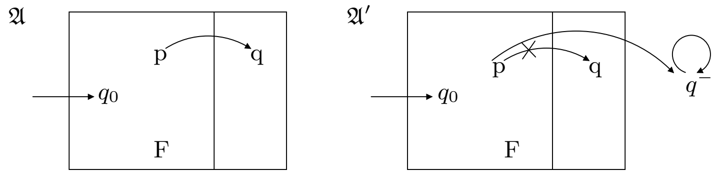
Theorem 3.29. (Hierarchy Theorem) *The following diagram of inclusions holds for the classes of ω-languages that can be recognized by deterministic automata with E-, A-, Büchi, and co-Büchi acceptance conditions:*



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

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

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

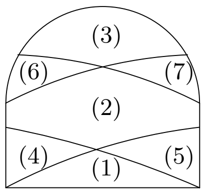
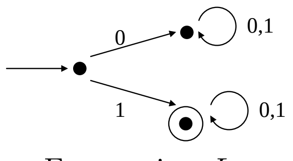


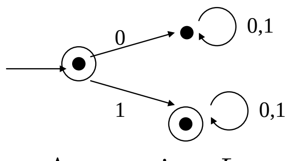
Figure 3.2: Seven inclusions

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:

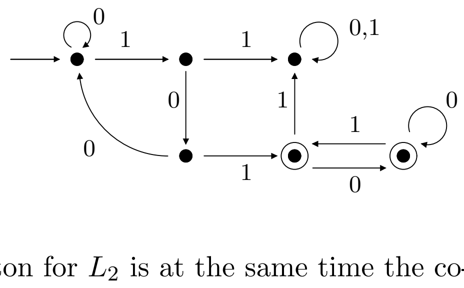


E-recognizes L_1



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 $\mathfrak{A}' = (Q, \Sigma, q_0, \delta, \bigcup \mathcal{F})$.

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

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 $\mathcal{F}$. $\square$

Proof of b)

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

$\mathfrak{A}$ accepts $\alpha \overset{\text{Def}\mathcal{F}_2}{\Longleftrightarrow} \mathfrak{A}$ 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. $\square$

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 $\exists i \rho(i) \in \{q\}$.

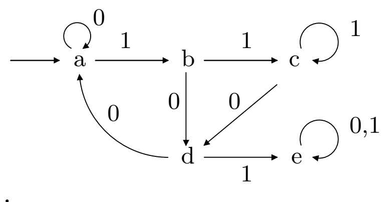
$\Rightarrow$ Consider a Boolean combination of guaranty conditions $\exists \rho(i) \in P_k$ (or $\mathrm{Occ}(\rho) \cap P_k \neq \emptyset$) for suitable sets $P_k \subseteq Q$. The DNF yields disjunction of conditions of the following kind (we denote the jth element of the disjunction with $(*)_j$):

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

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.



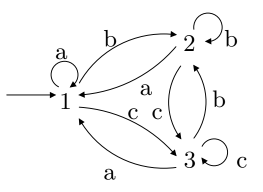
$\mathfrak{A}$ has got the following properties:

- 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 $\{a\}, \{a, b, d\}, \{a, b, d, e\}, \{a, b, c, d, e\}$. ⊠

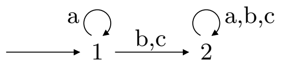
Remark 3.38. *There are SW-recognizable languages which cannot be recognized by a SW-automaton with only one set in its accepting component.*

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 $\alpha\}$.



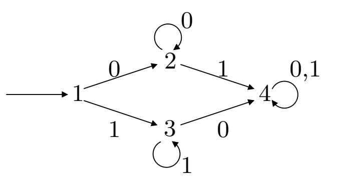
$\mathcal{F} = \{\{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$

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^n1^ω. 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 $\mathcal{F} = \{F_1, \ldots, F_k\}$, 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.

$$\begin{tikzcd}
 & & & 2 \arrow[dr] & & & \\
 & & 2 \arrow[ur] \arrow[rr, bend right] & & 2 \arrow[ll, bend right] \arrow[dr] & & 5 \arrow[loop above] \arrow[dd] \\
{} \arrow[r] & 1 \arrow[ur] \arrow[dr] & & & & 5 \arrow[loop above] \arrow[ur] & \\
 & & 3 \arrow[loop below] \arrow[rr] & & 4 \arrow[ur] & & 5 \arrow[ul]
\end{tikzcd}$$

The partial order can be illustrated as follows:

$$\begin{tikzcd}
 & 2 \arrow[drr] & & \\
1 \arrow[ur] \arrow[dr] & & & 5 \\
 & 3 \arrow[r] & 4 \arrow[ur] &
\end{tikzcd}$$

$\boxtimes$

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 $c : Q \to \{0, \ldots, k\}$. For a run ρ let $c(\rho)$ be the sequence of associated colors:

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

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 $c : Q \to \{0, \ldots, k\}$ 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 $c : Q \to \{0, 1\}$:

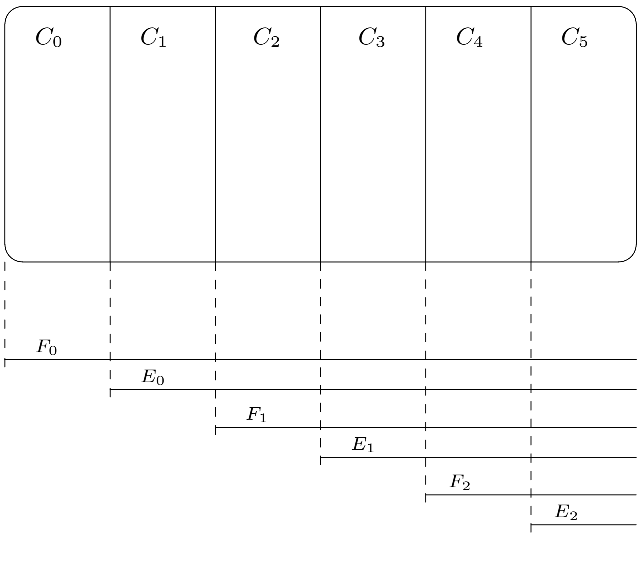
$$c(q) = \begin{cases} 0 & \text{for } q \in F \\ 1 & \text{for } q \notin F \end{cases}$$

⊠

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 $c : Q \to \{0, \dots, k\}$, w.l.o.g. let k be odd. We write $C_i = \{q \mid c(q) = i\}$.

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



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

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

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

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

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

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 $c : Q \to \{0, \dots, k\}$. 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 $Q' = Q \times 2^Q$, $q_0' = (q_0, \{q_0\})$.

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 $\mathcal{A}$. □

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.

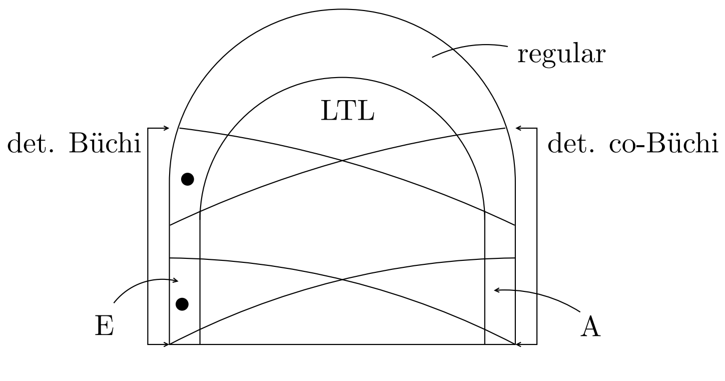
Exercise 3.9. Let $U \subseteq \Sigma^*$ be a finite language, and $L := U \cdot \Sigma^\omega$.

(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.

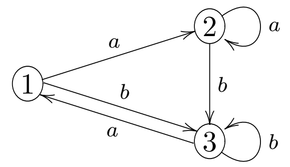
Exercise 3.13. Show that the language, which is defined by the ω-regular expression $(0^*1)^\omega$, is not Staiger-Wagner recognizable. (Consider, assuming that such an SW-automaton with n states exists, the ω-word $(0^n1)^\omega$ in order to derive a contradiction.)

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.

Chapter 4

Games and Winning Strategies

4.1 Basic Terminology

Definition 4.1. A *game graph* is a tuple $G = (Q, E)$ with vertex set $Q = Q_0 \dot{\cup} Q_1$, and edge relation $E \subseteq Q \times Q$, which is complete in the following sense: for all $q \in Q$ $\exists p : (q, p) \in E$ (i.e. every q has got an outgoing edge).

We define a *play* as a sequence $\rho = r_0 r_1 r_2 \ldots$ with $(r_i, r_{i+1}) \in E$.

A *game* is a tuple (G, φ) consisting of a game graph G and a winning condition φ for Player 0. A game $\rho \in Q^\omega$ is won by Player 0, if ρ fulfills φ. φ is usually specified by some requirement on $\mathrm{Occ}(\rho)$, $\mathrm{Inf}(\rho)$ or, after specifying a coloring $c : Q \to C$ (C a finite color set) of the vertices, some requirement on $\mathrm{Occ}(c(\rho))$ or $\mathrm{Inf}(c(\rho))$. The set $\mathrm{Win} \subseteq Q^\omega$ denotes the set of the plays won by Player 0.

We will consider the following problem: Given a game graph $G = (Q, E)$ and a set $\mathrm{Win} \subseteq Q^\omega$ of the plays won by Player 0, specify the set of vertices of G from which Player 0 can guaranty winning the game, and compute a winning strategy for Player 0 .

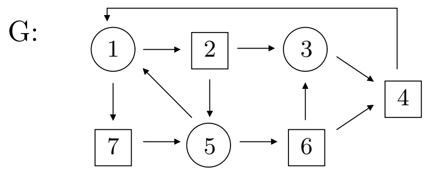
Definition 4.2. A *strategy* for Player 0 starting at vertex q_0 is a function $f : Q^+ \to Q$ that assigns to each play prefix $q_0 \ldots q_k$, with $q_k \in Q_0$, a vertex $r \in Q$ with $(q_k, r) \in E$.

A play $\rho = q_0 q_1 \ldots$ started in q_0 is *played according to* f if for every $q_i \in Q_0$, $q_{i+1} = f(q_0 \ldots q_i)$ holds.

A strategy f is a *winning strategy* from q_0 for Player 0 if every play played according to f is in Win. Winning strategies for Player 1 are defined analogously.

Strategies f for Player 0 and g for Player 1 define exactly one play from a vertex q. Both players cannot have a winning strategy from $q \in Q$ at the same time. Proof: Choose a winning strategy f for Player 0 and g for Player 1 respectively (starting from q). Consider the resulting play ρ. Since f is a winning strategy for Player 0 , $\rho \in \mathrm{Win}$ holds and, since g is a winning strategy for Player 1 , $\rho \notin \mathrm{Win}$ also holds. Contradiction.

Example 4.3. We will usually denote vertices of the game graph, which belong to Player 0, by circles and vertices, which belong to Player 1, by boxes. Consider the following game graph:



Winning condition 1 (for Player 0): $\mathrm{Occ}(\rho) \supseteq \{2,6\}$. This condition is fulfilled by the following strategy: Player 0 moves from 1 to 2 first, to 7 later, from 3 always to 4, and from 5 always to 6. If we want to describe this strategy as a function f_0 we obtain the following definition:

$$\begin{aligned} f_0(w1) &= \begin{cases} 2 & \text{if 1 does not occur in } w \\ 7 & \text{if 1 occurs in } w. \end{cases} \\ f_0(w3) &= 4 \\ f_0(w5) &= 6 \end{aligned}$$

As one can see, f_0 is a winning strategy starting from every vertex.
Winning condition 2: $\mathrm{Occ}(\rho) = \{1,\dots,7\}$.

Player 1 obtains a winning strategy f_1 starting from $1,2,4,5,6,7$, by choosing the following: from 2 move to 5, from 4 move to 1, 6 move to 4, and from 7 move to 5. Starting from vertex 3 Player 0 has got a winning strategy: f_0 defined as above. ⊠

Definition 4.4. For a game graph G and a winning set Win let

$$W_0 := \{q \in Q \mid \text{ Player 0 has got a winning strategy starting from } q\}$$

be the *winning region* of Player 0 . The winning region W_1 of Player 1 is defined accordingly.

Example 4.5. The winning regions of Example 4.3:
Winning condition 1: $W_0 = Q, W_1 = \emptyset$
Winning condition 2: $W_0 = \{3\}, W_1 = Q \setminus \{3\}$ ⊠

Remark 4.6. *For any game $W_0 \cap W_1 = \emptyset$ holds, since we already showed there can be no q such that both players have winning strategies from q.*

Question: Does $W_0 \cup W_1 = Q$ always hold?

Definition 4.7. A game is *determined* if every vertex of the game graph is either in W_0 or in W_1.

We will see later on that there are games that are not determined. So the answer to the above question is 'No'.

4.2 Special Strategies, Strategy Automata

We initially consider the case where no "memory of the past" is needed.

Definition 4.8. A strategy f for Player 0 is called *positional* (or *local, memoryless*) if the value of $f(q_0 \dots q_k)$ depends on $q_k \in Q_0$ only, i.e. the strategy can be described by a function $F : Q_0 \to Q$. Conversely, a strategy for Player 1 is called positional if the value of $f(q_0 \dots q_k)$ depends on $q_k \in Q_1$ only.

Remark 4.9. *A positional strategy for Player 0 can be described by a subset of the set of edges E, where exactly one vertex of the edges is in Q_0:*

$$\{(q,p) \in E \mid q \in Q_0, f(q) = p\} \subseteq E.$$

Analogously for Player 1.

Some strategies need a finite memory in order to be implemented. Such strategies can be specified by finite automata which generate output. The output corresponds to the choice of the next vertex by the player.

Definition 4.10. A *strategy automaton (Mealy automaton)* for Player 0 for the game graph $G = (Q, E)$ is a of the form $\mathfrak{A} = (S, Q, s_0, \sigma, \tau)$ with

S: finite memory
Q: input alphabet
$s_0 \in S$: initial state
$\sigma : S \times Q \to S$ memory update function
$\tau : S \times Q_0 \to Q$ transition choice function

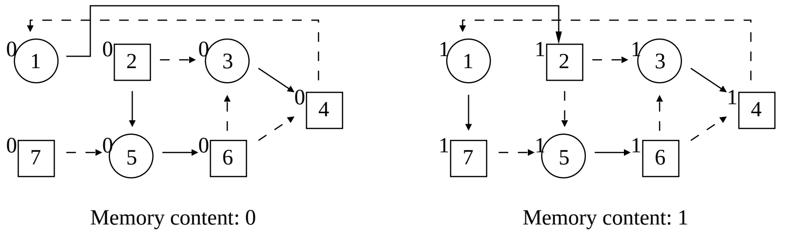
$\mathfrak{A}$ delivers a strategy $f_{\mathfrak{A}} : Q^+ \to Q$ for every vertex $q \in Q$ as follows:

Extend σ from $S \times Q \to S$ to $S \times Q^* \to S$ with $\sigma(s, \epsilon) = s$ and $\sigma(s, wq) = \sigma(\sigma(s, w), q)$. Therefore $\sigma(s, w)$ is the content of the memory obtained from s by by reading input w. The strategy computed by $\mathfrak{A}$ is: $f_{\mathfrak{A}}(q_0 \dots q_k) = \tau(\sigma(s_0, q_0 q_1 \dots q_{k-1}), q_k)$. A strategy f is called *automaton strategy* if $f = f_{\mathfrak{A}}$ holds for a suitable strategy automaton $\mathfrak{A}$.

An automaton strategy (for Player 0) can be displayed by a transition graph $S \cdot G$. The transition graph $S \cdot G$ consists of the set of vertices $S \times Q$ and the set of edges $(s, q) \to (s', q')$ which satisfy: if $q \in Q_0$, then $\sigma(s, q) = s'$, $\tau(s, q) = q'$ (in particular we have $(q, q') \in E$), and if $q \in Q_1$, then $\sigma(s, q) = s'$, $(q, q') \in E$.

Notice that in the transition graph for Player 0 over $S \times Q$ an automaton strategy is given by the set of edges. In this graph only one edge leads out of every $S \times Q_0$ vertex.

Example 4.11. The automaton strategy of f_0 in Example 4.3 can be displayed as follows. The edges leading from Q_1 vertices are dotted whilst the edges leading from Q_0 vertices (strategy edges) are solid.



Therefore $\mathfrak{A}$ is defined by $S = \{0, 1\}$, $Q = \{1, \dots, 7\}$, $s_0 = 0$. The output function is given by

$$\tau(0,1) = 2,\ \tau(1,1) = 7 \text{ and } \tau(0,q) = \tau(1,q) = \begin{cases} 4 \text{ if } q = 3 \\ 6 \text{ if } q = 6. \end{cases}$$

A memory update is performed only if Player 0 is at vertex 1 for the first time: then $\sigma(0, 1) = 1$. Otherwise $\sigma(s, q) = s$. ⊠

4.3 Guaranty and Safety Games

Theorem 4.12. *Let $G = (Q, E)$ be a game graph, $F \subseteq Q$ with the winning condition*

$$\rho \in \text{Win} :\Leftrightarrow \exists i : \rho(i) \in F \text{ (guaranty game).}$$

Then the winning regions W_0, W_1 of players 0 and 1 can be computed in polynomial time as well as corresponding positional winning strategies.

Proof For $i = 0, 1, 2, \dots$ construct the sets $\text{Attr}_0^i(F)$ where

$$\text{Attr}_0^i(F) = \{q \in Q \mid \text{ starting from } q \text{ Player 0 can force a visit to } F \text{ in } \leq i \text{ moves}\}.$$

Inductive construction over i:

$$\begin{aligned}
q \in \text{Attr}_0^0(F) &\Leftrightarrow q \in F \\
q \in \text{Attr}_0^{i+1}(F) &\Leftrightarrow q \in \text{Attr}_0^i(F) \\
&\quad \text{or } q \in Q_0 \text{, and there exist } (q, r) \in E \text{, such that } r \in \text{Attr}_0^i(F) \\
&\quad \text{or } q \in Q_1 \text{, and all } (q, r) \in E \text{ lead to } r \in \text{Attr}_0^i(F)
\end{aligned}$$

Obviously $\text{Attr}_0^0(F) \subseteq \text{Attr}_0^1(F) \subseteq \text{Attr}_0^2(F) \subseteq \dots$ holds. Hence for an $l \leq |Q|$ $\text{Attr}_0^l(F) = \text{Attr}_0^{l+1}(F)$ holds, i.e. the sequence of inclusions becomes static.

$$\text{Set } \text{Attr}_0(F) := \bigcup_{i=0}^{|Q|} \text{Attr}_0^i(F)$$

The set $\text{Attr}_0(F)$ is called *0-attractor* of F. Claim: $W_0 = \text{Attr}_0(F)$, $W_1 = Q \setminus \text{Attr}_0(F)$, and there are positional winning strategies for Player 0 on W_0, and Player 1 on W_1. The proof is derived from the following two remarks. Define

$$dist(q, F) := \begin{cases} \min\{i \mid q \in \text{Attr}_0^i(F)\} & \text{if } q \in \text{Attr}_0(F) \\ \infty & \text{if } q \notin \text{Attr}_0(F) \end{cases}$$

Reachability remark:

a) $q \in \text{Attr}_0(F) \setminus F, q \in Q_0 \Rightarrow \exists (q, r) \in E$ with $dist(r, F) < dist(q, F)$

b) $q \in \text{Attr}_0(F) \setminus F, q \in Q_1 \Rightarrow \forall (q, r) \in E : dist(r, F) < dist(q, F)$ holds

According to part a, Player 0 can choose edges from the set $\text{Attr}_0(F) \setminus F$ such that with every move the distance to F is reduced. According to part b, the distance to F is also reduced by every move of Player 1. Therefore F is eventually reached.

Avoidance remark:

a) $q \notin \text{Attr}_0(F), q \in Q_0 \Rightarrow \forall (q, r) \in E$: $r \notin \text{Attr}_0(F)$ holds

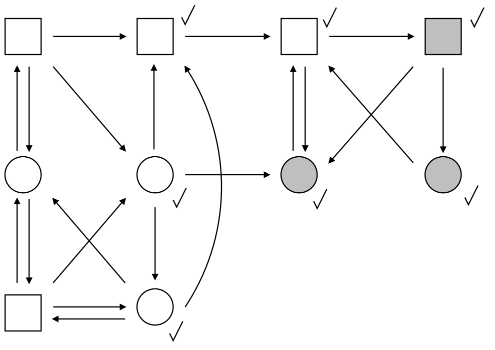
b) $q \notin \text{Attr}_0(F), q \in Q_1 \Rightarrow \exists (q, r) \in E$ with $r \notin \text{Attr}_0(F)$

According to part b, Player 1 can choose an edge that in turn leads back to $Q \setminus \text{Attr}_0(F)$. According to part a, every move of Player 0 also leads back to $Q \setminus \text{Attr}_0(F)$, so that the set F is avoided (distance to F remains ∞). The game is won by a player on his winning region following the attractor strategy defined below. □

Definition 4.13. The *attractor strategy* of Player 0 is defined as follows: from a vertex in $\mathrm{Attr}_0^{i+1} \setminus \mathrm{Attr}_0^i$ go to a vertex in Attr_0^i. Define the strategy function f such that for every $q \in Q_0 \cap (\mathrm{Attr}_0^{i+1} \setminus \mathrm{Attr}_0^i)$ $f(q) \in \mathrm{Attr}_0^i$ holds.

The attractor strategy for Player 1 is exactly the opposite: from a vertex in $Q \setminus \mathrm{Attr}_0$ return to a point in $Q \setminus \mathrm{Attr}_0$.

Example 4.14. The vertices in the following game graph marked with $\surd$ are in the 0-attractor of the gray vertices.



⊠

We give an algorithm for determining the sets W_0, W_1 and the sets of edges E_0, E_1 by backward breadth-first search.

Given $G = (Q, Q_0, E)$ and $F \subseteq Q$.

1. Preprocessing: Compute outdegree $\mathrm{out}(q)$ of each vertex $q \in Q_1$.

2. Set $n(q) := \mathrm{out}(q)$ for each $q \in Q_1$.

3. Perform breadth-first search backwards from F with the following conventions:

   - Mark all $q \in F$.
   - Mark $q \in Q_0$ if reached backwards from marked vertex.
   - At visit of $q \in Q_1$ from marked vertex: set $n(q) := n(q) - 1$.
   - Mark $q \in Q_1$ when $n(q) = 0$.

The marked vertices are the ones in $\mathrm{Attr}_0(F)$.

Lemma 4.15. *The given algorithm on (Q, Q_0, E) computes the winning regions and strategies in time $O(|Q| + |E|)$.*

For *safety games* we obtain the same result: Given the safety game over $G = (Q, E)$ with winning condition $\rho \in \mathrm{Win} \Leftrightarrow \forall i \rho(i) \in F$, consider the guaranty game over G with the new winning condition $\rho \in \mathrm{Win}' \Leftrightarrow \exists i \rho(i) \in Q \setminus F$. The winning region W_1' for Player 1 is the winning region W_0 for Player 0 in the given safety game.

4.4 Weak Parity and Staiger-Wagner Games

Definition 4.16. Let $G = (Q, E)$ be a game graph and c a function $c : Q \to \{0, \dots k\}$, also called *coloring*. A play $\rho = \rho(0)\rho(1)\rho(2)\dots$ delivers a sequence of colors $c(\rho) = c(\rho(0))c(\rho(1))c(\rho(2))\dots$. The *weak parity condition* defines the set of plays won by Player 0 by

$$\rho \in \text{Win} :\Leftrightarrow \max(\text{Occ}(c(\rho))) \text{ is even.}$$

Thus it is demanded that the maximal color occurring in ρ is even. These games are also called *weak parity games.*

First we classify guaranty and safety games within this framework.

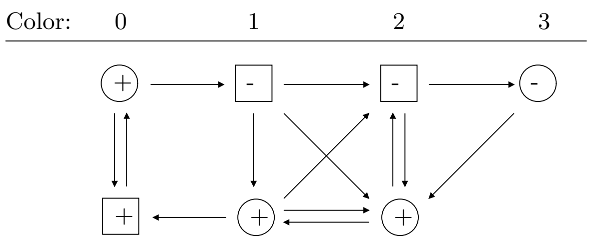
Remark 4.17. *A guaranty game over G with $\rho \in \text{Win} \Leftrightarrow \exists i \rho(i) \in F$ can be described as a special weak parity game over G with coloring*

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

A safety game with $\rho \in \text{Win} \Leftrightarrow \forall i \rho(i) \in F$ can be specified analogously by the coloring

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

Example 4.18. Player 0 has got a winning strategy starting from the vertices marked with +. Starting from the vertices marked with −, Player 1 has got a winning strategy.



$\boxtimes$

Definition 4.19. (Staiger-Wagner Game) A game is called *Staiger-Wagner game* or *obligation game* if for $\mathcal{F} = \{F_1, \dots, F_k\}$, with $F_i \subseteq Q$ its winning condition is of the form $\rho \in \text{Win} \Leftrightarrow \text{Occ}(\rho) \in \mathcal{F}$.

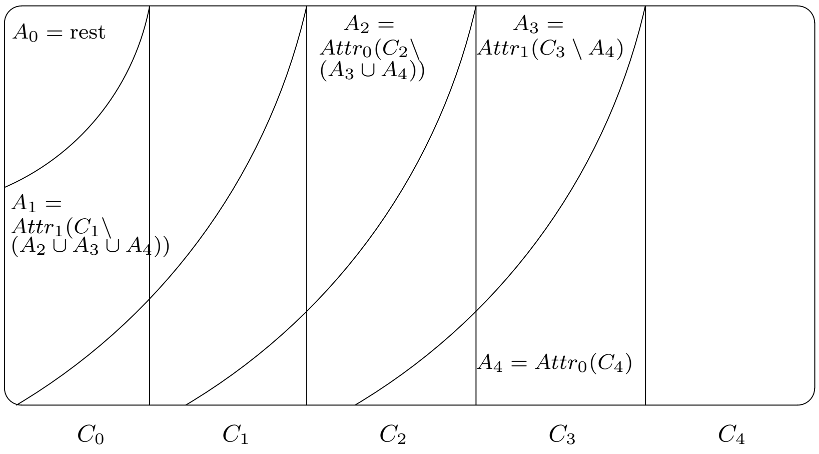
Remark 4.20. *Weak parity games are a special case of the SW-games. For $G = (Q, E)$ with $c : Q \to C$ (and $\rho \in \text{Win} \Leftrightarrow \max(\text{Occ}(c(\rho)))$ even) one can specify $\mathcal{F} \subseteq 2^Q$ (and $\rho \in \text{Win} \Leftrightarrow \text{Occ}(\rho) \in \mathcal{F}$) by setting $\mathcal{F} := \{P \subseteq Q \mid \max(c(P))$ is even$\}$.*

Theorem 4.21. *For weak parity games the winning regions W_0, W_1 can be computed and corresponding positional winning strategies can be specified.*

Proof Let $G = (Q, E), c : Q \to \{0, \dots, k\}$ with k w.l.o.g. even (else swap players). Set $C_i := \{q \in Q \mid c(q) = i\}$. We calculate the sets $A_k, A_{k-1}, \dots, A_0$:

$$A_k := \mathrm{Attr}_0(C_k)$$
$$A_i := \begin{cases} \mathrm{Attr}_0(C_i \setminus (A_{i+1} \cup \cdots \cup A_k)) & \text{if } i \text{ even} \\ \mathrm{Attr}_1(C_i \setminus (A_{i+1} \cup \cdots \cup A_k)) & \text{if } i \text{ odd} \end{cases} \quad \text{for } i \leq k-1$$

Visualization (for $k = 4$):



$A_0 =$ rest
$A_1 = Attr_1(C_1 \setminus (A_2 \cup A_3 \cup A_4))$
$A_2 = Attr_0(C_2 \setminus (A_3 \cup A_4))$
$A_3 = Attr_1(C_3 \setminus A_4)$
$A_4 = Attr_0(C_4)$
C_0
C_1
C_2
C_3
C_4

Every vertex is in one of the sets $A_0, \ldots, A_k$.

Claim:

$$W_0 = \bigcup_{i \text{ even}} A_i \qquad W_1 = \bigcup_{i \text{ odd}} A_i$$

and the union of the associated attractor strategies are positional winning strategies for Player 0 on W_0 and Player 1 on W_1 respectively.

We prove this by induction over j, i.e. we show for $j \leq k$

$$\bigcup_{\substack{i=k-j \\ i \text{ even}}}^{k} A_i \subseteq W_0 \quad \text{and} \quad \bigcup_{\substack{i=k-j \\ i \text{ odd}}}^{k} A_i \subseteq W_1$$

with the attractor strategies defined as above. We assume k to be even. The case "k odd" is dual.

Induction base: $j = 0$**:** $A_k = \mathrm{Attr}_0(C_k) \subseteq W_0$ is clear.

Induction step: Consider $A_{k-(j+1)}$, w.l.o.g. $k-(j+1)$ even. Let $i = k-(j+1)$. Show: From $q \in A_i$ Player 0 wins by attractor strategy (and hence we verify the claim $q \in W_0$).

Case 1: The attractor strategy for A_i either produces a visit to A_{i+2} or $A_{i+4} \ldots A_k$ (from a Q_1-vertex); then Player 0 wins by induction hypothesis.

Case 2: C_i (color i) is visited. Ensure that then only colors $\leq i$ are visited. Consider a vertex q after color i is visited. If $q \in Q_0$, then not all edges lead to higher colors; otherwise $q \in A_j$ for a higher j, contradiction.

If $q \in Q_1$ then all edges lead to vertices of color $\leq i$, because otherwise

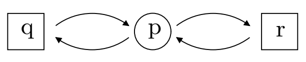
(a) Any edge to higher colors must lead to A_j with even j (this is Case 1).

(b) If there were edges to colors $> i$ in A_j with odd j, then q would already be in this A_j. □

Considering an example, we will see that in order to win Staiger-Wagner games we need more than just positional winning strategies.

Example 4.22.

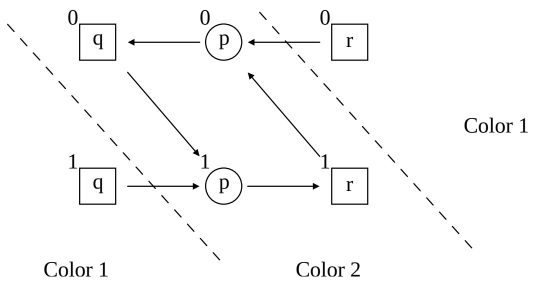
G_0 :



$$Q = \{q, p, r\}$$
$$\rho \in \text{Win} \Leftrightarrow \text{Occ}(\rho) = \{q, p, r\}$$

There are exactly two positional strategies for Player 0 (choose edge from p to r or q respectively). None of them is a winning strategy on $W_0 = \{q, p, r\}$. An automaton strategy with an additional bit as memory (which will guaranty alternating transitions to q and r) will give us a winning strategy.

The appropriate strategy automaton is $\mathfrak{A} = (\{0,1\}, Q, 0, \sigma, \tau)$ with $\sigma : (b,q) \to 1, (b,r) \to 0, (b,p) \to b$ and $\tau : (0,p) \to q$, $(1,p) \to r$. From that we can derive the following transition graph:



⊠

4.5 Game Reductions

In order to solve SW-games we introduce the notion of a game reduction.

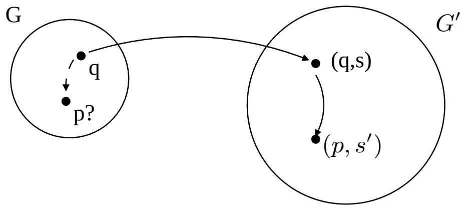
Definition 4.23. (Game Reduction) Let $G = (Q, E)$ and $G' = (Q', E')$ be game graphs with winning conditions φ and φ'. The game (G, φ) is *reducible* to (G', φ'), written $(G, \varphi) \leq (G', \varphi')$, iff the following conditions are satisfied.

1. $Q' = Q \times S$ for a finite set S.
2. Every play ρ over G is translated into a play ρ' over G' by
   (a) a function $f : Q \to Q \times S$ (the beginning of ρ')
   (b) $\forall (q,s) \in Q \times S \; \forall p \; : \; (q,p) \in E \Rightarrow$ exists exactly one s' with $((q,s),(p,s')) \in E'$
   (c) $\forall ((q,s),(p,s')) \in E' : (q,p) \in E$
3. For ρ and ρ', according to 2, $\rho \in \text{Win} \Leftrightarrow \rho' \in \text{Win}'$ holds.

Theorem 4.24. *Let $(G, \varphi) \leq (G', \varphi')$ as above. If Player 0 wins the game (G', φ') starting from $f(q)$ with a positional strategy then Player 0 wins the game (G, φ) from q using an automaton strategy.*

Proof For Player 0 construct an automaton winning strategy in (G, φ). This is accomplished by the strategy automaton $\mathfrak{A} = (S, Q, s_0, \sigma, \tau)$ where s_0 = second component of $f(q)$ and

$\sigma : S \times Q_0 \to Q$	$\sigma(s, q)$:= 1st component p of target vertex reached from (q, s) by applying the positional strategy.
$\tau : S \times Q \to S$	$\tau(s, q)$:= 2nd component s' (unique by part 1 of the definition) of the vertex (p, s') which is reached from (q, s) as above.



$\mathfrak{A}$ implements a winning strategy: The constructed play ρ (starting from q) is equivalent to a play ρ' starting from (q, s_0) in G'. Player 0 wins ρ' because of the positional winning strategy. That satisfies part 3 of the definition and we are done. □

Theorem 4.25. *Let $(G, \varphi) \leq (G', \varphi')$, and let Player 0 and Player 1 win on the winning regions W_0' and W_1' of G' by positional strategies. Then from W_0', W_1' one can determine the winning regions W_0, W_1 for (G, φ) and construct for $q \in W_0$ (or respectively W_1) corresponding automata strategies.*

Proof For $q \in Q$ determine $f(q) \in Q'$ and verify whether $f(q) \in W_0'$. If so then $q \in W_0$ holds and there is an automaton strategy according to Theorem 4.24. Analogously for Player 1 . □

This theorem allows us to migrate from positional to automaton winning strategies. We will now apply this theorem to solve SW-games.

Theorem 4.26. *Staiger-Wagner games can be reduced to weak parity games and thus the winning regions and corresponding automata strategies for SW-games can be computed.*

Proof Let (G, φ) be a SW-game, defined by the family $\mathcal{F} \subseteq 2^Q$ with $\rho \in \text{Win} \Leftrightarrow \text{Occ}(\rho) \in \mathcal{F}$. One can specify a weak parity game (G', φ') with $(G, \varphi) \leq (G', \varphi')$.

Idea: G'-vertices are of the form (p, R) with $R \subseteq Q$. This is to express that $R \cup \{p\}$ is the set of "vertices visited so far" in Q. Therefore begin with a vertex of the form $(p, \emptyset)$. For the weak parity condition fix an even color for (p, R) if $R \cup \{p\} \in \mathcal{F}$.

Thus define $G' = (Q', E')$ and the coloring $c : Q' \to \{0, \dots, k\}$ as follows:

- $Q' = Q \times 2^Q$
- $E' = \{((p, R), (q, R \cup \{p\})) \mid (p, q) \in E, R \subseteq Q\}$
- $c(p, R) = \begin{cases} 2 \cdot |R \cup \{p\}| & \text{if } R \cup \{p\} \in \mathcal{F} \\ 2 \cdot |R \cup \{p\}| - 1 & \text{if } R \cup \{p\} \notin \mathcal{F} \end{cases}$

Now check parts 1, 2, and 3 of Definition 4.23. For 1 and 2 use $S = 2^Q$ and $f : q \to (q, \emptyset)$. For 3 consider a play ρ over G and the corresponding induced play ρ' over G'.

At some time in ρ, a prefix is reached that contains all states in $\mathrm{Occ}(\rho)$. From that point onwards in ρ', the second component is constant ($= \mathrm{Occ}(\rho)$). Thus the maximal visited color is even iff $\mathrm{Occ}(\rho) \in \mathcal{F}$, i.e. $\rho \in \mathrm{Win} \Leftrightarrow \rho' \in \mathrm{Win}'$. □

Example 4.27. Consider the game with the first winning condition from Example 4.3 (page 65).

A positional winning strategy over $Q \times 2^Q$:	Example colors:	
from $(1, R)$: to 2 if $1 \notin R$		Color
to 7 if $1 \in R$	$(1, \emptyset)$	1
$(3, R)$: to 4	$(1, \{2, 3, 4\})$	7
$(5, R)$: to 6	$(1, \{1 \dots, 7\})$	14

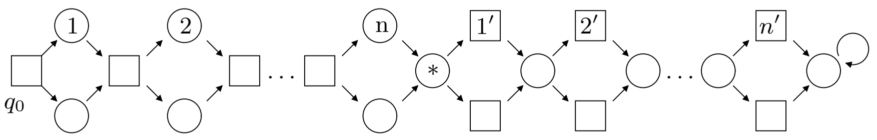
⊠

Result: In order to win SW-games over graphs with n vertices, automata with $\leq 2^n$ states suffice. We show that this upper bound cannot be improved.

Theorem 4.28. *There is a family $G_1, G_2, G_3, \dots$ of game graphs with SW-winning condition, such that the number of vertices of G_n in* $\mathrm{Occ}(\rho)$ *increases linearly in n, and there is an automaton winning strategy (from suitable vertices) but only for an automaton with at least 2^n states.*

Proof We consider the following game graph G_n with the winning condition

$$\rho \in \mathrm{Win} \Leftrightarrow \text{ (for } i = 1, \dots, n \ i \in \mathrm{Occ}(\rho) \Leftrightarrow i' \in \mathrm{Occ}(\rho) \text{ holds)}$$



There is an automaton winning strategy starting from q_0 which is specified by an automaton with 2^n states ("memorize the set of visited vertices of $\{1, \dots, n\}$").

Assume there is an automaton with $< 2^n$ states that implements a winning strategy from q_0. Then there are two play prefixes $w_1 \neq w_2$ which lead to the same state at vertex $*$. The rest of the play is unambiguous. Thus $w_1\rho$ and $w_2\rho$ are played according to the automaton strategy, but only one of them is winning. Contradiction. □

Problem 1 The state set 2^Q of the strategy automaton is too big in general. It would be more sensible to introduce states only when needed. One approach derives the attractor set decomposition for weak parity games directly from the specification of the SW-game. Minimization can be done in time $O(n \log(n))$.

Problem 2 The SW-condition is disjunctive: $\mathcal{F} = \{F_1, \dots, F_k\}$ "$\mathrm{Occ}(\rho) = F_1 \vee \dots \vee \mathrm{Occ}(\rho) = F_k$". A conjunctive winning condition requires preprocessing. Question: Is there a direct construction of a conjunctive winning condition from a strategy automaton?

The goal of the rest of this chapter is to find the winning regions W_0, W_1 of the players for a given game graph $G = (Q, E)$ with Büchi, parity, Muller, and Rabin winning conditions. Furthermore we want to construct winning strategies (positional, if possible) for both players on their respective winning regions.

4.6 Büchi Games

Let $G = (Q, E)$ be a graph with $Q = Q_0 \dot{\cup} Q_1$ and $F \subseteq Q$. In addition let the winning condition φ of Player 0 for the play ρ be

$$\varphi : \rho \in \text{Win} :\Leftrightarrow \text{Inf}(\rho) \cap F \neq \emptyset.$$

Such a game (G, φ) is called a *Büchi game*. The approach to determine the winning region of Player 0 is to determine, for $i \geq 1$, those $q \in F$ from which Player 0 can force at least i revisits to F. Thus

$$F \supseteq \underbrace{\text{Recur}_0^1(F)}_{\text{one revisit}} \supseteq \underbrace{\text{Recur}_0^2(F)}_{\text{two revisits}} \supseteq \dots$$

holds. Define $\text{Recur}_0(F) := \bigcap_{i \geq 1} \text{Recur}_0^i(F)$ and then $W_0 = \text{Attr}_0(\text{Recur}_0(F))$.

Preparation for the definition of $\text{Recur}_0^i(F)$: We define

$$\begin{aligned}
A_0^0 &= \emptyset, \\
A_0^{i+1} &= A_0^i \cup \{q \in Q_0 \mid \exists (q, r) \in E : r \in A_0^i \cup F\}, \\
&\quad \cup \{q \in Q_1 \mid \forall (q, r) \in E : r \in A_0^i \cup F\} \\
\text{Attr}_0^+(F) &= \bigcup_{i \geq 0} A_0^i.
\end{aligned}$$

With this definition we set

$$\begin{aligned}
\text{Recur}_0^0(F) &:= F, \\
\text{Recur}_0^{i+1}(F) &:= F \cap \text{Attr}_0^+(\text{Recur}_0^i(F)), \\
\text{Recur}_0(F) &:= \bigcap_{i \geq 0} \text{Recur}_0^i(F).
\end{aligned}$$

Remark 4.29. $\text{Recur}_0^{i+1}(F) \subseteq \text{Recur}_0^i(F)$.

Proof $i = 0$: OK

$$\begin{aligned}
i + 1 : \text{Recur}_0^{i+2}(F) &= F \cap \text{Attr}_0^+(\text{Recur}_0^{i+1}(F)) \\
&\overset{I.H.}{\subseteq} F \cap \text{Attr}_0^+(\text{Recur}_0^i(F)) \\
&= \text{Recur}_0^{i+1}(F)
\end{aligned}$$

□

Remark 4.30. *On* $\text{Recur}_0(F)$ *Player 0 has a positional winning strategy.*

Proof On $\text{Recur}_0(F) \cap Q_0$ it is possible to choose an edge back to $\text{Recur}_0(F)$ and on $\text{Recur}_0(F) \cap Q_1$ there are only edges back to $\text{Recur}_0(F)$. Thus $\text{Attr}_0(\text{Recur}_0(F)) \subseteq W_0$ holds.

Let i be such that $\text{Recur}_0^i(F) = \text{Recur}_0^{i+1}(F) = F \cap \text{Attr}_0^+(\text{Recur}_0^i(F))$ holds. Therefore choosing the edges according to the attractor strategy gives a positional winning strategy for Player 0 . □

Lemma 4.31. *Let $G = (Q, E)$ be a game graph with a Büchi winning condition F. Player 1 has got a positional winning strategy on $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0(F))$. Thus $W_0 = \mathrm{Attr}_0(\mathrm{Recur}_0(F))$ and $W_1 = Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0(F))$.*

Proof It suffices to show that Player 1, on $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^i(F))$, can (by following a positional strategy) force F to be visited $\leq i$ times. Prove that by induction on i:

$i = 0$ Consider $Q \setminus \mathrm{Attr}_0(F)$. Player 1 can avoid a visit to $\mathrm{Attr}_0(F)$ (and thus to F).

$i + 1$ Consider $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^{i+1}(F))$.

If the play is already within $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^i(F))$, then Player 1 can, by induction hypothesis, force F to be visited $\leq i < i + 1$ times.

Otherwise the play is within $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^{i+1}(F))$ and not in $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^i(F))$. Thus it is within $\mathrm{Attr}_0(\mathrm{Recur}_0^i(F)) \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^{i+1}(F))$. Consequently, Player 1 can avoid $\mathrm{Attr}_0(\mathrm{Recur}_0^{i+1}(F))$.

Case 1 This play is never in F. Then we are finished.

Case 2 If this play visits F, then it will visit $F \setminus \mathrm{Recur}_0^{i+1}(F)$. $F \setminus \mathrm{Recur}_0^{i+1}(F) = F \setminus \mathrm{Attr}_0^+(\mathrm{Recur}_0^i(F))$ holds. According to the induction hypothesis Player 1 can, while being in $Q \setminus \mathrm{Attr}_0(\mathrm{Recur}_0^i(F))$, force F to be visited at most i times. In total there are $\leq i + 1$ visits to F.

□

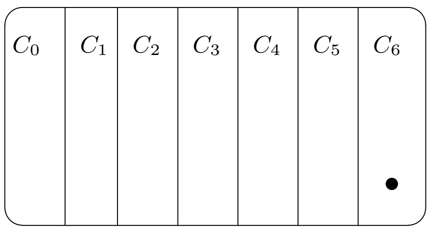
Corollary 4.32. *The winning regions W_0, W_1 are computable in Büchi games (and form a partition of the vertices of the game graph) and Player 0 as well as Player 1 have positional winning strategies.*

4.7 Parity Games

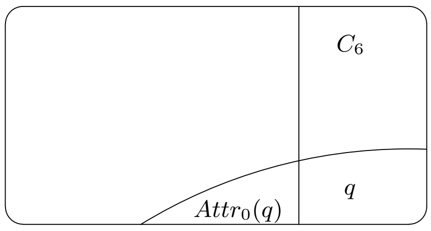
Given game graph $G = (Q, E)$, a coloring $c : Q \to \{1, \ldots, k\}$ with k even and $C_i := \{q \mid c(q) = i\}$ define the winning condition

$$\varphi : \rho \in \mathrm{Win} \Leftrightarrow \max(\mathrm{Inf}(c(\rho))) \text{ even}$$

Such a game is called a *parity game.*



The following approach to solve parity games exploits the fact that if the play ρ visits C_k, then Player 0 wins. Idea:



Theorem 4.33. *Let $G = (Q, E)$ be a game graph with coloring $c : Q \to \{0, \dots, k\}$ and parity winning condition. Then*

1. *every vertex in Q belongs to either W_0 or W_1, i.e. parity games are determined (see Definition 4.7).*

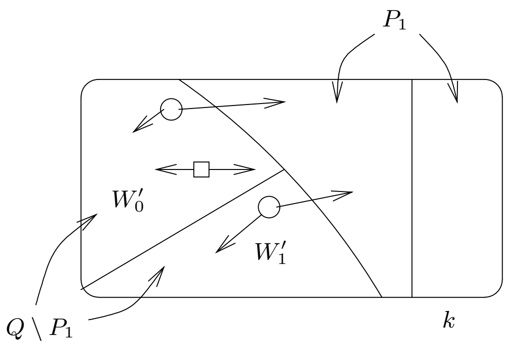
2. *If G is finite one can determine the winning regions W_0, W_1 (with $W_0 \cup W_1 = Q$) and corresponding positional winning strategies for Player 0 and Player 1.*

Proof (1. Determinacy of Parity Games) Given $G = (Q, Q_0, E)$ with coloring $c : \{0, \dots, k\}$, we prove the result by induction over the number of colors. For $k = 0$ clearly $W_0 = Q$ holds with any strategy.

In the induction step assume that the maximal color k is even (otherwise switch the roles of Players 0 and 1 below). Let P_1 be the set of vertices from which Player 1 has a positional winning strategy, and let f be a uniform positional winning strategy for Player 1 on P_1.

Show that from each vertex in $Q \setminus P_1$, Player 0 has a positional winning strategy. For that matter consider the subgame with vertex set $Q \setminus P_1$

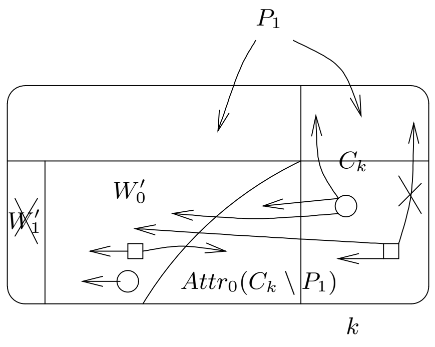
Case 1: $Q \setminus P_1$ does not reach the maximal color k.



$Q \setminus P_1$ defines a subgame. The induction hypothesis applies on this region, since it does not reach color k. $Q \setminus P_1$ can therefore be partitioned into winning regions W_0' and W_1' in which positional winning strategies exist for both Players by induction hypothesis.

Now the winning region of Player 1 is $W_1 = P_1 \cup W_1'$ and the winning region of Player 0 is $W_0 = W_0'$. Therefore the whole game graph is partitioned into winning regions with respective positional strategies.

Case 2: $Q \setminus P_1$ contains vertices of maximal (even) color. We claim that Player 0 can guaranty that, starting from a vertex in $Q \setminus P_1$, the play remains there.

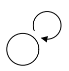
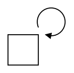


Either the play stays in $(Q \setminus P_1) \setminus \mathrm{Attr}_0(C_k \setminus P_1)$ or it visits $\mathrm{Attr}_0(C_k \setminus P_1)$ infinitely often.

In the first case Player 0 wins by induction hypothesis with a positional strategy. In the second case Player 0 wins by infinitely many visits to the highest (even) color, also with a positional strategy.

Altogether: Player 0 wins from each vertex in $Q \setminus P_1$ with a positional strategy. □

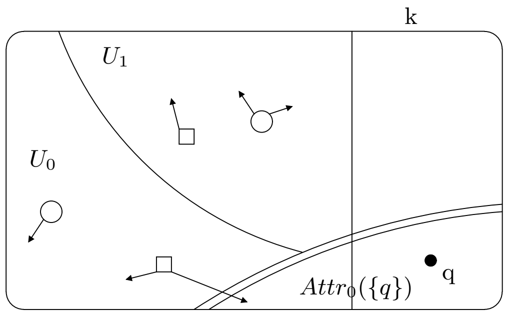
Proof (2. Algorithmic Solution) The algorithm is defined inductively over $n := |Q|$:

$n = 1$**:** The game graph is of the form  or  . Then the color of the vertex determines whether $Q = W_0$ or $Q = W_1$.

$n + 1$**:** We assume that the maximal color which occurs in the game graph is even. In the other case the induction step is analogous, if the Players are swapped.

Choose q with (even) maximal color. Notice that $Q \setminus \mathrm{Attr}_0(\{q\})$ gives a game graph with $\leq n$ vertices.

The induction hypothesis gives a partition of $Q \setminus \mathrm{Attr}_0(\{q\})$ into winning regions U_0, U_1 of Player 0 / Player 1 and corresponding positional winning strategies.



Case 1: Player 0 can guaranty transitions from q to $U_0 \cup \mathrm{Attr}_0(\{q\})$. That means
a) $q \in Q_0$ and ex. $(q, r) \in E$ with $r \in U_0 \cup \mathrm{Attr}_0(\{q\})$
or b) $q \in Q_1$ and all $(q, r) \in E$ fulfill $r \in U_0 \cup \mathrm{Attr}_0(\{q\})$.
Claim

(i) $U_0 \cup \mathrm{Attr}_0(\{q\}) \subseteq W_0$

(ii) $U_1 \subseteq W_1$

(Since $U_0 \cup U_1 \cup \mathrm{Attr}_0(\{q\}) = Q$, then $W_0 = U_0 \cup \mathrm{Attr}_0(\{q\})$ and $W_1 = U_1$ holds.)
Thus the positional strategy for Player 0 on $U_0 \cup \mathrm{Attr}_0(\{q\})$ is:

1. On U_0 play according to the positional strategy given by the induction hypothesis.
2. On $\text{Attr}_0(\{q\})$ play according to the attractor strategy. Then eventually reach q.
3. From q "move back" to $U_0 \cup \text{Attr}_0(\{q\})$.

For Player 1 use the positional strategy on U_1 given by the induction hypothesis.

Proof of the claim: (ii) is clear, since starting in U_1 Player 1 can guaranty that the play remains U_1. That is because from 1-vertices one edge leads to U_1, and from 0-vertices every edge leads to U_1.

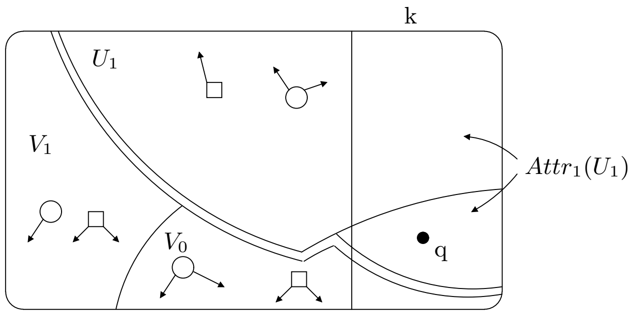
For (i): The play ρ remains in $U_0 \cup \text{Attr}_0(\{q\}) \subseteq W_0$, if the rule of vertex $p \in U_0 \cup \text{Attr}_0(\{q\})$ is followed. If ρ eventually remains in U_0, then Player 0 will win by induction hypothesis. If ρ passes through q again and again, then Player 0 will win, as q has the maximal even color.

Case 2: Player 1 can guaranty transitions to U_1 from q, i.e.

a') $q \in Q_0$ and all $(q, r) \in E$ fulfill $r \in U_1$

or b') $q \in Q_1$ and ex. $(q, r) \in E$ with $r \in U_1$.

Thus $q \in \text{Attr}_1(U_1)$. $Q \setminus \text{Attr}_1(U_1)$ forms a game graph with $< n$ vertices. The induction hypothesis gives winning regions V_0, V_1 and corresponding positional strategies in the game $(Q \setminus \text{Attr}_1(U_1), E_1)$, where E_1 is the restriction of the set of edges to $Q \setminus \text{Attr}_1(U_1)$.



Claim:

(i) $V_0 \subseteq W_0$ for the given positional winning strategy.

(ii) $V_1 \cup \text{Attr}_1(U_1) \subseteq W_1$

(i) is clear, since Player 0 can guaranty the play to stay within V_0.

For (ii): Player 1 can move to U_1 from $p \in \text{Attr}_1(U_1)$ and there he can guaranty the play to remain in U_1. From $p_1 \in V_1$ Player 0 can choose to move either to V_1 or $\text{Attr}(U_1)$. In both cases the play is won by Player 1 .

From (i) and (ii) we can derive $W_0 = V_0$ and $W_1 = V_1 \cup \text{Attr}_1(U_1)$. □

Remark 4.34. *The inductive proof leads to a recursive procedure which needs to be called 2^n times for a game graph with n vertices if case 1 persists. The running time is therefore estimated to be exponential in the number of vertices of the game graph.*

Result: The synthesis problem for winning strategies in parity games can be solved in exponential time.

Lemma 4.35. (Merging Lemma) *Given a parity game over $G = (Q, Q_0, E)$, there is a single positional strategy f such that from each $q \in W_0$ the strategy f is a winning strategy for Player 0 from q.*

Proof Number the states by natural numbers (if G is finite) or more generally by ordinal numbers (if G is infinite). Denote by q_i the state with number i. For $q_i \in W_0$ choose a corresponding positional winning strategy f_i.

Let F_i be the set of vertices reachable by plays from q_i according to f_i (Note: $F_i \subseteq W_0$ and $q_i \in F_i$). Define f on W_0 as follows:

$$f(q) = f_i(q) \text{ for the smallest } i \text{ such that } q \in F_i.$$

Show that f is a winning strategy from any $q \in W_0$.

Applying f during a play means to apply strategies f_i where i is weakly decreasing.

From some point k onwards, index i stays constant, i.e. the f-values coincide with the f_i-values. The highest color occurring infinitely often in the play is thus determined by the fixed strategy f_i. Since f_i is a winning strategy, Player 0 wins the play. $\square$

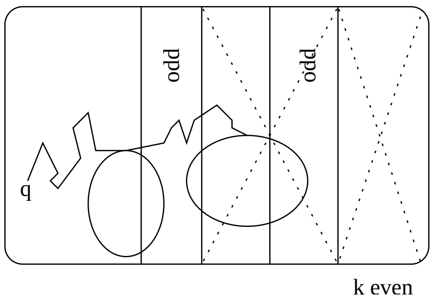
Definition 4.36. A strategy f of Player 0 is called a *uniform strategy* for Player 0 if f is a winning strategy for all plays started in vertices which belong to the winning region of Player 0.

Remark 4.37. *The problem*

> *Given: graph (Q, E), coloring $c : Q \to \{0, \dots, k\}$, vertex $q \in Q$.*
> *Question: Is $q \in W_0$?*

is in NP $\cap$ co-NP. Whether this problem is in P, is an open question.

Proof To show that the problem is in NP, nondeterministically guess a uniform strategy for Player 0 (which is a subset of vertices such that every 0-vertex has an unique outgoing edge). With the chosen strategy it can be verified in polynomial time whether q is in the winning region as follows: $q \in W_0 \Leftrightarrow$ in the strategy transition graph (one edge from 0-vertices and all edges from 1-vertices) Player 0 cannot enter a loop from q such that the maximal color is odd.



For that matter verify for the graphs over $(C_0 \cup \dots \cup C_{k-1})$, $(C_0 \cup \dots \cup C_{k-3})$, …, $(C_0 \cup C_1)$, whether they contain a SCC[1] reachable from q which intersects the colors $k-1$, $k-3$, …, 1 ($k-1$ is the maximal odd color).

[1]SCC = strongly connected component

The complementary problem "$q \in Q \setminus W_0$" can be formalized as "$q \in W_1$" and is in NP due to the same argument when the players are swapped. □

4.8 Muller Games and LAR-Construction

In this section we will consider Muller games:

Definition 4.38. A game consisting of a game graph $G = (Q, E)$ and $\mathcal{F} = \{F_1, \dots, F_k\}$, $F_i \subseteq Q$ with the winning condition

$$\rho \in \text{Win} \Leftrightarrow \text{Inf}(\rho) \in \mathcal{F}$$

is called a *Muller game.*

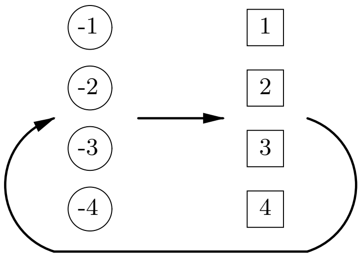
The solutions (winning regions for Player 0 / Player 1 and corresponding winning strategies) can be computed as follows:

To solve Muller games we will reduce them to parity games. We begin with an introductory example:

Example 4.39. (Dziembowski, Jurdziński, Walukiewicz)

Let the game graph $G^n = (Q_0^n \cup Q_1^n, E_n)$ be defined by

$$\begin{aligned} Q_0^n = &\ \{-1, \dots, -n\}, \\ Q_1^n = &\ \{+1, \dots, +n\}, \\ E_n = &\ \{(-i, j) \mid i, j \in \{1, \dots, n\}\} \cup \\ &\ \{(i, -j) \mid i, j \in \{1, \dots, n\}\}. \end{aligned}$$



with the winning condition $\varphi^n : \rho \in \text{Win}^n :\Leftrightarrow |\text{Inf}(\rho) \cap \{-1, \dots, -n\}| = \max(\text{Inf}(\rho) \cap \{1, \dots, n\})$, i.e. "The maximal vertex i that is visited infinitely often gives the number of minus-vertices that are visited infinitely often."

Examples for accepting runs:

- Exactly -1, -3, -4 are visited infinitely often: 3 infinitely often, 4 only finitely often.
- Eventually only -4 is visited: infinitely often through 1, only finitely often through 2,3,4.

The winning condition can be reformulated as a Muller condition:

$$F \in \mathcal{F} :\Leftrightarrow |F \cap \{-1, \dots, -n\}| = \max(F \cap \{1, \dots, n\}).$$

⊠

To reduce Muller games to parity game we introduce the following datastructure:

Definition 4.40. For $Q = \{1, \dots, n\}$ let $LAR(Q)$ be the set of pairs $((i_1 \dots i_k), h)$ with nonempty records of $i_1 \dots i_k$ pairwise different states, and $h \in \{0, \dots, n\}$ (=hitvalue or hit).

Notation: $(i_1 \dots \underline{i_h} \dots i_k)$. We call $\{i_1, \dots, i_h\}$ the *hit-segment* of of $(i_1 \dots i_k, h)$ and $(i_{h+1} \dots i_k)$ the *share of the past* of $(i_1 \dots i_k)$.

We consider the transformation of a play of Example 4.39 into an LAR-play. We apply this transformation to the sequence of visited minus-vertices.

	LAR
-1	$\underline{-1}$ -2 -3 -4
-3	-3 -1 $\underline{-2}$ -4
-3	$\underline{-3}$ -1 -2 -4
-4	-4 -3 -1 $\underline{-2}$
-2	-2 -4 -3 $\underline{-1}$
-4	-4 $\underline{-2}$ -3 -1
-3	-3 -4 $\underline{-2}$ -1
-4	-4 $\underline{-3}$ -2 -1

The underlining indicates the hitvalue. If exactly -3, -4 occur infinitely often then the maximal infinitely often visited hitvalue is 2. In this case Player 0 has to pass through 2 infinitely often and only finitely often through 3 and 4.

We can generally say that, if exactly k minus-vertices are visited infinitely often, then the maximal infinitely often assumed hitvalue is k.

We exploit this observation to find an automaton strategy of Player 0. Use the LAR of the minus-states as states of the strategy automaton and in each case go to the vertex whose name is equal to the current hit. We define the strategy automaton $\mathfrak{A}$ as follows. $\mathfrak{A} = (S, Q, s_0, \sigma, \tau)$, with:

$$\begin{aligned}
&S = LAR(-1, \dots, -n)\\
&s_0 = (-1, \dots, -n)\\
&Q = \{-1, \dots, -n, 1, \dots, n\}\\
&\sigma(s, -i) = \text{the LAR obtained by prepending } -i \text{ to } s.\\
&\sigma(s, i) = s\\
&\tau(s, i) = \text{hitvalue}(s)
\end{aligned}$$

Result: In this game over G^n, the winning region of Player 0 is the set Q and $\mathfrak{A}$ is the corresponding automaton strategy.

In order to solve Muller games in general, we need to know more about the transformation of Muller automata to parity automata. We first formally define the notion of an LAR-run.

Definition 4.41. For $\rho \in Q^\omega$ the *LAR-run* $\rho' \in (LAR(Q))^\omega$ is defined by: $\rho'(0) = ((1 \dots n, 1), 1)$. If $\rho'(i) = ((i_1 \dots i_k), h)$, $\rho(i+1) = l$, then $\rho'(i+1)$ is defined as follows: if $l = i_{h'}$ then $\rho'(i+1) := ((i_{h'} i_1 \dots i_{h'-1} i_{h'+1} \dots i_k), h')$.

Lemma 4.42. (LAR-Lemma) $\text{Inf}(\rho) = F$ *with* $|F| = m$ *iff*

1. *in* ρ' *the hit is* $> m$ *only finitely often,*
2. *in* ρ' *the hit-segment is* $= F$ *infinitely often.*

Proof

$\Rightarrow$ Let $\text{Inf}(\rho) = F$, $|F| = m$. Choose k and $k' > k$, such that $\rho(j) \in F$ holds for all $j \geq k$ and $\{\rho(k), \dots, \rho(k'-1)\} = F$ holds.

By the construction of ρ', the F-elements $\{i_1, \dots, i_m\} = F$ are at the beginning of $\rho'(k')$ (corresponding to the hit-segment) and for every $k \geq k_0$ (for some k_0 big enough) the hit is always $\leq m$ $(*)$. For the hit equal to m the hit-segment must be the set F. Thus part 1 is proven. For part 2 it suffices to show (because of $(*)$) that the hit is equal to m infinitely often. If the hit was equal to m only finitely often, then eventually the LAR-entries $i_m, i_{m+1}, \dots, i_k$ would not be changed anymore. Then $|\mathrm{Inf}(\rho)| < m$ would hold, which would be a contradiction to $\mathrm{Inf}(\rho) = F$ and $|F| = m$.

$\Leftarrow$ Let 1 and 2 hold.

Because of part 1, the LAR-entries $i_{m+1} \dots i_k$ in ρ' are fixed from some point j_0 onwards. The states $i_{m+1}, \dots, i_k$ are not visited again after j_0. Thus, because of part 2, the LAR-entries are not in F from position $m + 1$ onwards.

Now, show that $\mathrm{Inf}(\rho) = F$.

1. $\mathrm{Inf}(\rho) \subseteq F$ is clear, since states, which are not in F, are not visited again after j_0.
2. $F \subseteq \mathrm{Inf}(\rho)$: Assume $q \in F$, but $q \notin \mathrm{Inf}(\rho)$. Then from some point onwards q can only stay in the same position in the LAR or go to the right. The final position is a position $k < m$. That contradicts part 2, since then the hit-segment is $\{i_1, \dots, i_k\} \neq F$.

□

The datastructure LAR will now be used for the transformation of Muller automata into parity automata.

Theorem 4.43. *For every deterministic Muller automaton one can construct an equivalent deterministic parity automaton.*

Proof Given the Muller automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$. We set $\mathfrak{A}' := (LAR(Q), \Sigma, ((1 \dots n), 1), \delta', c)$ and still need to define δ' and c. The approach for c is to let the colors denote the length of the hit-segments. The length is even if the hit-segment is in $\mathcal{F}$ $(2 \cdot h)$, and it is odd, iff the hit-segment is not in $\mathcal{F}$ $(2 \cdot h - 1)$. Therefore the maximal color occurring in the run ρ is $2 \cdot |\mathrm{Inf}(\rho)|$ or $2 \cdot |\mathrm{Inf}(\rho)| - 1$ respectively.

Formal definition of δ': $\delta'(((i_1 \dots i_k), h), a) = (j_1 \dots j_{h'} \dots j_{k'}, h')$, where the right side of the equation is obtained as follows: $\delta(i_1, a) \in \{i_1, \dots, i_k\}$: Then h' is equal to the position of $\delta(i_1, a)$ in $i_1 \dots i_k$ and $j_1 \dots j'_k$ is created from $i_1 \dots i_k$ by bringing forward $\delta(i_1, a)$ as in Definition 4.41. We define $c : LAR(Q) \to \{1, \dots, 2n\}$ by

$$c((i_1 \dots i_k, h)) = \begin{cases} 2h - 1 & \text{hit-segment } \{i_1, \dots, i_h\} \notin \mathcal{F} \\ 2h & \text{hit-segment } \{i_1, \dots, i_h\} \in \mathcal{F} \end{cases}$$

□

Example 4.44. Let $\Sigma = \{a, b, c\}$, $L = \{\alpha \in \Sigma^\omega \mid$ if a infinitely often in α, then also $b\}$. The corresponding Muller automaton has got the states q_a, q_b, q_c, each of them having x-transitions to q_x. $\mathcal{F} = \{\{q_b\}, \{q_c\}, \{q_b, q_c\}, \{q_a, q_b\}, \{q_a, q_b, q_c\}\}$.

Input:		c	b	c	c	a	a	c	b	a	a	a	b	a	b
Muller run	q_a	q_c	q_b	q_c	q_c	q_a	q_a	q_c	q_b	q_a	q_a	q_a	q_b	q_a	q_b
	$\underline{q_a}$	q_c	q_b	q_c	$\underline{q_c}$	q_a	$\underline{q_a}$	q_c	q_b	q_a	$\underline{q_a}$	$\underline{q_a}$	q_b	q_a	q_b
LAR-run	q_b	q_a	q_c	$\underline{q_b}$	q_b	q_c	q_c	$\underline{q_a}$	q_c	q_b	q_b	q_b	$\underline{q_a}$	$\underline{q_b}$	$\underline{q_a}$
	q_c	$\underline{q_b}$	$\underline{q_a}$	q_a	q_a	$\underline{q_b}$	q_b	q_b	$\underline{q_a}$	$\underline{q_c}$	q_c	q_c	q_c	q_c	q_c
Hit:	0	0	0	2	1	3	1	2	3	3	1	1	2	2	2
Color:	1	1	1	4	2	6	1	3	6	6	1	1	4	4	4

$\boxtimes$

Theorem 4.45. (Büchi, Landweber 1969) *In a Muller game over a finite graph $G = (Q, E)$ one can efficiently determine the winning regions W_0, W_1 and for every $q \in W_i$ construct an automaton winning strategy for Player i from q.*

Proof We use the same approach as for Staiger-Wagner games: a game reduction to parity games.

- Specify a parity game (G', φ') with $(G, \varphi) \leq (G', \varphi')$ according to Definition 4.23.
- Use Theorem 4.33 for parity games (positional winning strategy).
- Theorem 4.25 for game reductions gives an automaton winning strategy.

Construction of $G' = (Q', E')$ and φ':

- $Q' = LAR(Q)$
- $f : q \to ((q, 1, \dots, q-1, q+1, \dots, |Q|), 1)$ the initiailization function.
- $((q, s), (r, s')) \in E' :\Leftrightarrow$ LAR s' is obtained from LAR s by shifting r to the front.
- Definition of the coloring $c : LAR(Q) \to \{0, \dots, 2|Q|\}$ by
  $$c(q,s) = \begin{cases} 2h & \text{hitsection } \in \mathcal{F} \\ 2h-1 & \text{hitsection } \notin \mathcal{F} \end{cases}$$

According to Theorem 4.43 on the transformation of Muller into parity automata, the following holds: $(*)$ $\text{Inf}(\rho) \in \mathcal{F} \Leftrightarrow \max(\text{Inf}(c(\rho')))$ is even.

For G' define the parity winning condition φ' by $\rho \in \text{Win} :\Leftrightarrow \max(\text{Inf}(c(\rho')))$ even. Therefore $\rho \in \text{Win} \Leftrightarrow \text{Inf}(\rho) \in \mathcal{F} \overset{(*)}{\Leftrightarrow} \max(\text{Inf}(c(\rho')))$ even $\Leftrightarrow \rho' \in \text{Win}'$ holds.

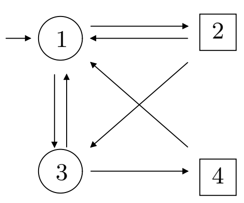
The positional winning strategy for Player 0 / Player 1 from $f(q)$ gives an automaton winning strategy for Player 0 / Player 1 from q. Therefore:

$$f(q) \in W'_0 \Rightarrow q \in W_0,$$
$$f(q) \in W'_1 \Rightarrow q \in W_1.$$

Thus $q \in W_0 \Leftrightarrow f(q) \in W'_0$.

According to the theorem on parity games, the regions W'_0, W'_1 (consisting of the vertices $f(q), q \in Q$) in G' can be determined. Set $W_0 = \{q \mid f(q) \in W'_0\}$, $W_1 = \{q \mid f(q) \in W'_1\}$. With the theorem on game reductions we can derive an automaton winning strategy (from q) for Player 0 / Player 1 from the positional winning strategy of Player 0 / Player 1 (from $f(q)$). $\square$

Example 4.46. G :



Winning condition:
$\varphi : \rho \in \text{Win} \Leftrightarrow \text{Inf}(\rho) \supseteq \{2,4\}$, thus $\mathcal{F} = \{\{2,4\},\{1,2,4\},\{3,2,4\},\{1,2,3,4\}\}$.

Example play		LAR-play	
$\rho :$	1	$\rho' :$	$\underline{1}\ 2\ 3\ 4$
	2		$2\ \underline{1}\ 3\ 4$
	1		$1\ \underline{2}\ 3\ 4$
	3		$3\ 1\ \underline{2}\ 4$
	4		$4\ 3\ 1\ \underline{2}$
	1		$1\ 4\ \underline{3}\ 2$
	2		$2\ 1\ 4\ \underline{3}$

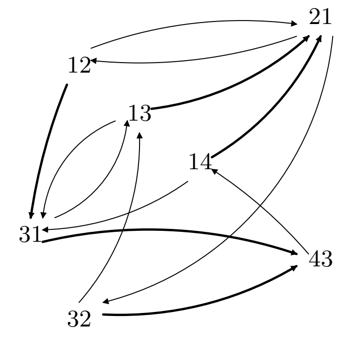
Approach for the automaton strategy:

from 1, alternately go to 2 and 3,
from 3, always go to 4.

Another strategy according to the proof using LAR gives:

from 1, coming from 4, go to 2,
coming from 2, go to 3,
from 3, always go to 4.

The graph G' where only the first two entries of the LARs, reachable from 12, are considered:



The automaton winning strategy starting from 12 uses $\mathfrak{A} = (S, \underbrace{Q_0}_{\{1,3\}}, s_0, \sigma, \tau)$. In our example $S = Q', s_0 = 12$ suffices, e.g. $\tau(\underbrace{14}_{\text{memory}}, \underbrace{12}_{\substack{\text{current}\\ Q_0-state}}) = 2$, $\sigma(14,12) = 21$. ⊠

4.9 Optimality of the LAR-Construction

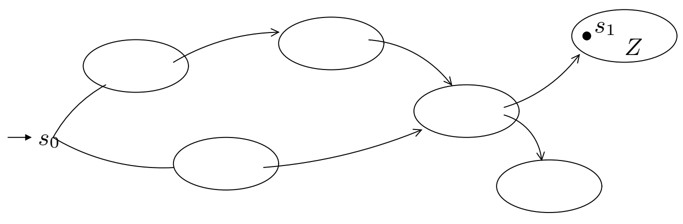
Theorem 4.47. (Dziembowski, Jurdziński, Walukiewicz) *In the game (G^n, φ^n) of Example 4.39, with the winning condition $\varphi^n : \rho \in \text{Win}^n \Leftrightarrow |\text{Inf}(\rho) \cap Q_0^n| = \max(\text{Inf}(\rho) \cap Q_1^n)$, $W_0 = Q$ holds and every automaton winning strategy for Player 0 has got $\geq n!$ memory states.*

Proof Show by induction over $n \geq 1$: Let $\mathfrak{A}^n = (S, Q_0, s_0, \sigma, \tau)$ be a strategy automaton which implements a winning strategy for Player 0 from -1 in the game (G^n, Win^n). Then there is a state s and an input word $w \in Q_0^*$ such that the following holds.

1. In $\mathfrak{A}^n$ the state s is reached again from s by w.
2. Every state $-i$ in Q_0 occurs in w.
3. The number of states in the cycle as in 1 is $\geq n!$.

$n = 1$: G^1: $$\begin{tikzcd} \circled{-1} \arrow[r] & \boxed{1} \arrow[l, bend right] \end{tikzcd}$$ Claim is clear.

$n > 1$: We assume that w.l.o.g. the transition graph of $\mathfrak{A}^n$ is strongly connected. If not partition the graph into strongly connected components, which constitutes an acyclic graph. Choose a last SCC Z and a state $s_1 \in Z$. The restriction of $\mathfrak{A}$ to a last SCC Z with initial state s_1 is in turn a winning strategy automaton, now strongly connected.



Let now H_i^n be the subgame obtained from G^n by deleting $-i$. Let $\mathfrak{A}_i^n$ have the input alphabet $Q_0^n \setminus \{-i\}$. $\mathfrak{A}_i^n$ implements a winning strategy in H_i^n. $\mathfrak{A}_i^n$ does not output n. Hence we can delete the vertex n and obtain a graph H_i^n which is isomorphic to G^{n-1}. By induction hypothesis 1, 2, and 3 hold for $\mathfrak{A}_i^n$, $i = 1, \ldots, n$.

Choose $s_i \in \mathfrak{A}_i^n$ and w_i according to the induction hypothesis. Let S_i be the set of states that are visited by $\mathfrak{A}_i^n$ during the cycle $s_i \xrightarrow{w_i} s_i$. We show: $i \neq j \Rightarrow S_i \cap S_j = \emptyset$.

Assume $\overline{s} \in S_i \cap S_j$. Then

$$\begin{array}{ll}\mathfrak{A}_i^n: & s_0 \xrightarrow{u_1} s_i \overbrace{\xrightarrow{u_1} \overline{s} \xrightarrow{v_1} s_i}^{w_i} \\ \mathfrak{A}_j^n: & \underbrace{s_j \xrightarrow{u_2} \overline{s} \xrightarrow{v_2} s_j}_{w_j}\end{array}$$

for suitable u_1, v_1, u_2, v_2.

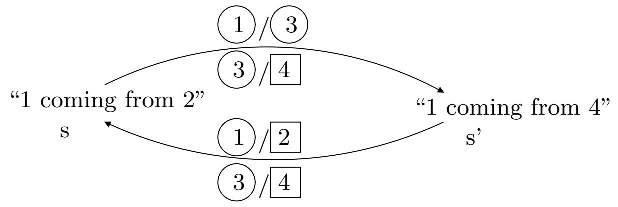
Then form a path in $\mathfrak{A}^n$ with the input $u_0u_1(v_2u_2v_1u_1)^\omega$. All states $-i$ $(i = 1, \ldots, n)$ are occurring infinitely often in this play. $\mathfrak{A}_i^n, \mathfrak{A}_j^n$ do not output n, thus $n-1$ is the maximal output value given infinitely often by $\mathfrak{A}^n$ on $u_0u_1(v_2u_2v_1u_1)^\omega$. This shows $i \neq j \Rightarrow S_i \cap S_j = \emptyset$.

Now use the fact that $\mathfrak{A}^n$ is strongly connected for the induction hypothesis. Choose s as the initial state and an input word w that leads $\mathfrak{A}^n$ back to the initial state, uses every input letter, and passes through every state.

Then obtain that the number of states of $\mathfrak{A}^n$ is $|\mathfrak{A}^n| \overset{I.H.(3)}{\geq} \underbrace{|S_1|}_{\geq (n-1)!} + \cdots + \underbrace{|S_n|}_{\geq (n-1)!} \geq n!$ □

4.10 Minimization of Strategy Automata

Example 4.48. Consider the game in Example 4.46. The improved LAR-construction gives seven states. But two states already suffice:



⊠

Theorem 4.49. *A strategy automaton of the form $\mathfrak{A} = (S, Q, s_0, \sigma, \tau)$ with*

$$\tau(s,q) \begin{cases} \in Q & \text{if } q \in Q_0 \\ = * & \text{if } q \in Q_1 \end{cases}$$

can be transformed (in polynomial time) into a minimal automaton, which is unique up to isomorphism.

Proof For a given strategy automaton $\mathfrak{A}$ define the function $f_{\mathfrak{A}} : Q^* \to (Q \cup \{*\})^*$ inductively by

$$\begin{aligned} &f_{\mathfrak{A}}(\epsilon) = \epsilon, \\ &f_{\mathfrak{A}}(uq) = f_{\mathfrak{A}}(u)\,\underbrace{\tau(\sigma(s_0,u),q)}_{\in Q \cup \{*\}}. \end{aligned}$$

1. Minimization step: Elimination of the unreachable states.
2. Minimization step: Merging of equivalent states.

$$s \sim_{\mathfrak{A}} s' :\Leftrightarrow \text{for } \mathfrak{A}_s, \mathfrak{A}_{s'} \text{ ($\mathfrak{A}$ with s or s' as initial states) } f_{\mathfrak{A}_s} = f_{\mathfrak{A}_{s'}} \text{ holds}$$

Obviously, $\sim_{\mathfrak{A}}$ is an equivalence relation. For $s \in S$ let $[s]$ denote the equivalence class of s and let $[S]$ denote the set of all these equivalence classes, thus $[S] = \{[s] \mid s \in S\}$. The reduced automaton is built with state set $[S]$ and initial state $[s_0]$. The memory update function $[\sigma] : [S] \times Q \to [S]$ is defined by

$$[\sigma]([s], q) := [\sigma(s,q)], \text{ and}$$

the output function $[\tau] : [S] \times Q \to Q \cup \{*\}$ is defined by

$$[\tau]([s], q) := [\tau(s,q)] \text{ (independent of the representative).}$$

We define $\mathfrak{A}_{red} = ([S], Q, [s_0], [\sigma], [\tau])$ which obviously computes $f_{\mathfrak{A}}$.

Let now f be computable by an automaton $\mathfrak{A}$). We can derive a canonical automaton $\mathfrak{A}_f$ that computes f. We show that $\mathfrak{A}_f$ has got the minimal number of states among those automata that compute f, and also show that every automaton $\mathfrak{A}_{red}$ that computes f is isomorphic to $\mathfrak{A}_f$.

Let $f : Q^* \to (Q \cup \{*\})^*$ be computable by an automaton. For $u, v \in Q^*$ define

$$\begin{aligned} u \sim_f v :\Leftrightarrow \forall w \in Q^* \quad : \quad & f(uw) \text{ and } f(vw) \text{ share the same suffix} \\ & \text{after the prefixes } f(u) \text{ and } f(v) \text{ respectively.} \end{aligned}$$

Obviously $\sim_f$ is an equivalence relation. The equivalence class of $u \in Q^*$ is denoted by $<u>$. Thus $\mathfrak{A}$ computes $f \Rightarrow \sim_{\mathfrak{A}}$ has got at least as many states as $\sim_f$.

Define the automaton $\mathfrak{A}_f$ with

States	$:= \sim_f$-classes
Initial state	$:= <\epsilon>$
Memory update function	$\sigma_f(<u>, q) :=<uq>$
Output function	$\tau_f(<u>, q) :=$ suffix (i.e. letter) of $f(uq)$ after prefix $f(u)$.

It is easy to show that $\mathfrak{A}_f$ computes the function f and that it has the minimal number of states.

Define an isomorphism from the reduced automaton $\mathfrak{A}_{red}$, which computes f, to $\mathfrak{A}_f$. Since all states of $\mathfrak{A}_{red}$ are reachable (say $q_0, \ldots, q_k$) there is in each case a u_i with $q_i = \sigma(q_0, u_i)$.

Consider the states $<u_0>, \ldots, <u_k>$ of $\mathfrak{A}_f$. The mapping $I : q_i \to <u_i>$ is an isomorphism.

□

Thus we can always find an optimal implementation of a *given* automaton strategy. A transformation of such a strategy into *another* strategy with an even smaller automaton is outlined in the following section.

4.11 Strategy Improvement

Preparation If Players 0 and 1 fix positional strategies f and g, then from each vertex q a play $\rho_q(f, g)$ is fixed, ending in a certain loop whose maximal color decides the winner. From $\rho_q(f, g)$ a value $v(q)$ is determined. Here v is a *valuation function* $v : Q \to D$ into some value domain D, which is ordered by a preference order $\prec$.

Algorithm for Strategy Improvement Given: Colored game graph G, valuation function v.

1. Pick two strategies f, g for Players 0 and 1.
2. Determine the values $v(q)$ for all $q \in Q$, referring to the plays $\rho_q(f, g)$.
3. Change strategy f of Player 0 by local improvement: For each Q_0-vertex, choose the out-edge leading to the neighbor vertex with highest value (by preference order).
4. Given the new f find the optimal response strategy of Player 1 and use it as new strategy g.
5. If the new strategies coincide with the previous strategies, then stop; otherwise go back to step 2.

Assume: The vertices are numbered, and the numbers are the colors. For our example we define a *preference order* $\prec$ for vertices $1, \ldots, 8$ (from the viewpoint of Player 0):

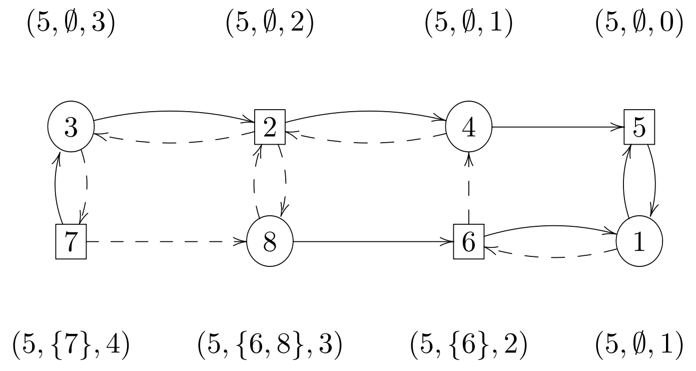
$$7 \prec 5 \prec 3 \prec 1 \prec 2 \prec 4 \prec 6 \prec 8$$

Naturally, the odd numbers, especially the high ones, are unfavorable for Player 0, whereas the high even numbers are the most favorable.

Definition 4.50. Let f, g be strategies for Player 0 and 1 for the game graph $G = (Q, E)$, and $q \in Q$. The *most relevant vertex* of $\rho_q(f, g)$ is the vertex with the highest color in the loop of $\rho_q(f, g)$. The *play profile* of $\rho_q(f, g)$ is the triple (r, P, d), where

- r is the most relevant vertex of $\rho_q(f, g)$,
- P is the set of higher valued vertices on the path from q to (and excluding) r,
- d is the distance between q and r on this path.

Example 4.51. Consider the following game graph, where the solid edges form the initial strategies of the Players. Notice the play profiles for the states.



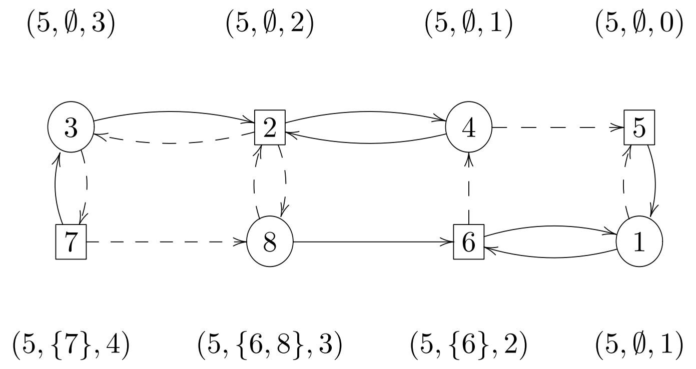
⊠

Definition 4.52. (Comparison of Play Profiles) The preference order, defined above, is extended from vertices to play profiles:

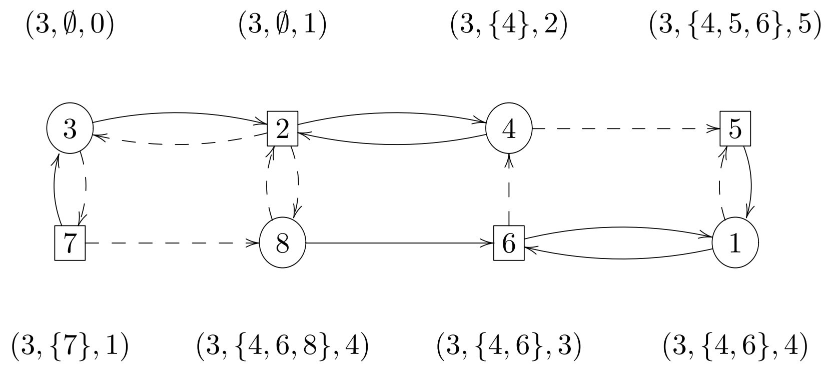
$$(r, P, d) \prec (r', P', d') \text{ iff}$$

1. $r \prec r'$, or
2. $r = r'$ and the highest vertex in the symmetric difference of P, P' is even and belongs to P', or is odd and belongs to P, or
3. $r = r'$ and $P = P'$ and $d < d'$ if r is odd, or $d' < d$ if r is even.

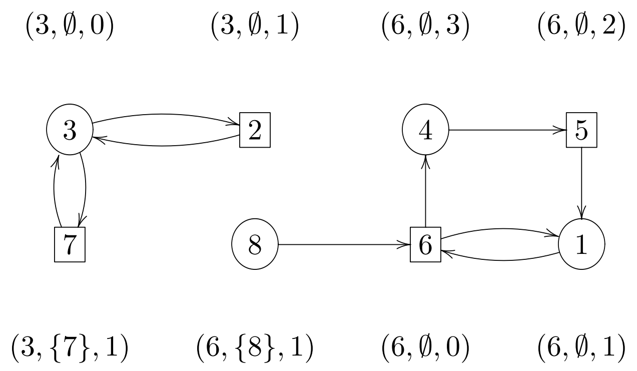
Example 4.53. (Example 4.51 continued) Local improvement by Player 0: The strategy of Player 0 can be improved at two states. In state 4 Player 0 should choose the edge leading to state 2 instead of state 5, since $(5, \emptyset, 0) \prec (5, \emptyset, 2)$. The same applies for state 1; Player 0's new choice is the edge leading to state 6 $(((5, \emptyset, 0) \prec (5, \{6\}, 2))$.



The optimal response by Player 1 and its corresponding valuation:



Player 0 can in turn improve his strategy by redirecting the outgoing edge at state 4 to state 5 ($(3, \emptyset, 1) \prec (3, \{4, 5, 6\}, 5)$). A response by Player 1 will not change the strategy. No further local improvement by Player 0 is possible. We obtain a game graph, which is partitioned into winning regions $W_0 = \{2, 3, 7\}$ and $W_1 = \{1, 4, 5, 6, 8\}$:



⊠

Theorem 4.54. (Vöge, Jurdzinski) *With the valuation by play profiles, the strategy algorithm terminates producing strategies f and g for Players 0 and 1 such that*

- *$q \in W_0$ ($q \in W_1$) iff the play $\rho_q(f, g)$ ends in a loop with even (respectively, odd) highest vertex,*
- *f and g are winning strategies for Player 0, respectively 1, from the vertices in W_0, respectively W_1.*

Thus the algorithm produces valid output. But at what cost? The time complexity of this algorithm has got an exponential upper bound. Further complexity properties:

- Each improvement round costs polynomial time.
- The number of improvement steps is bounded by the number of possible strategies.
- In experiments, only linearly many improvement steps have been observed.

4.12 Rabin and Streett Games

Goal: We want to show that Rabin games allow Player 0 to construct positional winning strategies on his winning region.

Definition 4.55. (Rabin Game) Let $\Omega = ((E_1, F_1), (E_2, F_2), \dots, (E_k, F_k))$ with $E_i, F_i \subseteq Q$. Then a game (G, Ω) with the winning condition

$$\rho \in \mathrm{Win} \Leftrightarrow \bigvee_{i=1}^{k} (\mathrm{Inf}(\rho) \cap E_i = \emptyset \wedge \mathrm{Inf}(\rho) \cap F_i \neq \emptyset)$$

for Player 0 is called *Rabin game.*

Theorem 4.56. *Player 0 wins in a Rabin game in his winning region by using a positional winning strategy.*

Proof Let $G = (Q, E)$ be the game graph and let $q \in Q$. Show: if Player 0 has got no positional winning strategy from q, then he has got none at all, i.e. $q \in W_1$.

Proof by induction over the number $n = |E \cap (Q_0 \times Q)|$ of edges available to Player 0.

Then $n \geq |Q_0|$, i.e. we choose the induction basis at $n = |Q_0|$.

$n = |Q_0|$**:** Every strategy of Player 0 is positional. Thus the claim holds.

$n > |Q_0|$**:** Choose a vertex p with two exiting edges e_1, e_2. Instead of the original graph $G = (Q, E)$ consider the graphs $G_1 = (Q, E \setminus \{e_1\})$ and $G_2 = (Q, E \setminus \{e_2\})$.

Consider $q \in Q$, such that Player 0 does not have a positional winning strategy from q. Then in G_1 and G_2 Player 0 does not have a positional winning strategy from q, either (fewer edges).

The induction hypothesis for G_1, G_2 therefore gives $q \in W_1$ for G_1, G_2. Let f_1 be a winning strategy for Player 1 from q in G_1 and let f_2 be a winning strategy for Player 1 from q in G_2.

The remaining task is to construct, from f_1, f_2, a winning strategy f for Player 1 from q in G (then $q \in W_1$, and we are finished).

Case 1: No play played according to f_1 (or f_2 resp.) starting from q reaches vertex p. Then choose $f = f_1$ (or $f = f_2$ resp.) and obtain a winning strategy for Player 1 from q in G.

Else: Player 1 wins, starting from p, with f_1 in G_1 and with f_2 in G_2. Fix f: Play according to f_1 till the first visit to p.

From p, after choosing an edge $\neq e_1$ (in G_1), play according to f_1 till the next visit to p. From p, after choosing an edge e_1 (in G_2), play according to f_2 till the next visit to p.

Show that every play played according to f is won by Player 1 .

Case A: The play ρ visits p only finitely often. Then the play will eventually be in G_1 (or G_2 resp.) according to f_1 (or f_2 resp.) and will be won by Player 1 according to the choice of f_1 (or f_2 resp.).

Case B: The play ρ visits p infinitely often.

- Player 0 eventually chooses only the edge e_1 in p. Then, as above, he will eventually be in G_2 and Player 1 wins the play.
- Player 0 eventually chooses only edges $\neq e_1$ in p. Analogously, we are finished.
- Player 0 chooses the edge e_1 and edges $\neq e_1$ again and again in p.

$$q \longrightarrow p_{e_1} \longrightarrow p_{e_1} \longrightarrow p_{\neq e_1} \longrightarrow p \dots$$

Extract two plays ρ_1, ρ_2 in which the $p \longrightarrow p$-segments are gathered according to the edge with which they are entered is equal to e_1 or not equal to e_1.

ρ_1 is played in G_1 according to f_1 hence Player 1 wins.
ρ_2 is played in G_2 according to f_2 hence Player 1 wins.

According to Lemma 3.16 ρ will not be successful either. Thus Player 1 wins the play ρ.

□

We will now consider *Streett* games. This is mainly motivated by a weakness of Staiger-Wagner conditions and of Muller conditions; both of them are disjunctive. For $\mathcal{F} = \{F_1, \dots, F_k\}$, the Muller condition reads

$$\text{Inf}(\rho) = F_1 \text{ or } \text{Inf}(\rho) = F_2 \text{ or } \dots \text{ or } \text{Inf}(\rho) = F_k.$$

In practical situations, conjunctions of conditions arise more naturally. An important example is the "Streett condition". It is called a *strong fairness condition* and is derived from the negation of the Rabin condition.

Definition 4.57. (Rabin and Streett Condition) Given a state set Q and a family Ω of pairs $(E_1, F_1), \dots, (E_k, F_k)$ with $E_i, F_i \subseteq Q$, a sequence $\rho \in Q^\omega$ satisfies the *Rabin condition* for Ω if

$$\bigvee_{i=1}^{k} (\text{Inf}(\rho) \cap E_i = \emptyset \wedge \text{Inf}(\rho) \cap F_i \neq \emptyset)$$

and a sequence $\rho \in Q^\omega$ satisfies the *Streett condition* for Ω if it satisfies the negation of the Rabin condition:

$$\bigwedge_{i=1}^{k} (\text{Inf}(\rho) \cap E_i \neq \emptyset \vee \text{Inf}(\rho) \cap F_i = \emptyset).$$

This condition is equivalent ($B \vee A \equiv \neg A \to B$) to

$$\bigwedge_{i=1}^{k} (\text{Inf}(\rho) \cap F_i \neq \emptyset \to \text{Inf}(\rho) \cap E_i \neq \emptyset).$$

In other words: For all $i = 1, \dots, k$: if F_i is visited infinitely often then E_i is visited infinitely often.

The necessary datastructures for solving Street games by a reduction to parity games are "index appearance records" (IAR).

Definition 4.58. An *index appearance record IAR* over r is a triple (π, e, f) where π is a permutation of $(1 \dots r)$, $e, f \in \{1, \dots, r\}$. $IAR(r)$:= set of IARs over r.

Lemma 4.59. *For a Streett game (G, φ) over $G = (Q, E)$ with pairs $(E_1, F_1), \dots, (E_r, F_r)$, one can construct a parity game (G', φ') with $(G, \varphi) \leq (G', \varphi')$.*

Corollary 4.60. *For Streett games with $\Omega = ((E_1, F_1), \dots, (E_r, F_r))$ over finite graphs one can compute the winning regions and determine automata winning strategies for both Players with memory size $r! \cdot r^2$.*

Proof of Lemma 4.59 Given: the game graph $G = (Q, E)$ and the Streett pairs $(E_1, F_1), \dots, (E_r, F_r)$ (with $E_r = F_r = Q$). Define G' over the set of vertices $Q \times IAR(r)$. The transformation ρ (over G) $\to \rho'$ (over G') according to Definition 4.23 is given by $f : q \to (q, ((1 \dots r), r, r))$ and the following edge relation: For an IAR $(q, (\pi, e, f))$ and $(q, q') \in E$ we introduce the edge $((q, (\pi, e, f)), (q', (\pi', e', f'))) \in E'$. π' is obtained from π by shifting all j with $q' \in E_j$ to the left (thus the j with $q' \notin E_j$ stand on the right), and for $\pi = (j_1 \dots j_r)$ and $\pi' = (j'_1 \dots j'_r)$ setting e' := the last k with $q' \in E_{j_k}$ (Notice that the position k refers to the previous permutation π) and f' := the last k with $q' \in F_{j'_k}$ (in this instance k refers to the current permutation π').

The coloring $c : Q \times IAR(r) \to \{1, \dots, 2r\}$ is defined by

$$c((q, ((j_1 \dots j_r), e, f))) = \begin{cases} 2e & \text{if } e \geq f \\ 2f - 1 & \text{if } e < f \end{cases}$$

The winning condition for ρ' is: $\rho' \in \mathrm{Win}'_0 :\Leftrightarrow \max(\mathrm{Inf}(c(\rho')))$ even.

Claim: $\underbrace{\rho \in \mathrm{Win}_0}_{\text{Streett}} \Leftrightarrow \underbrace{\rho' \in \mathrm{Win}'_0}_{\text{parity}}$.

This claim will be shown by the following observations. Subsequent to the proof we will consider an example for the transformation of a play into the corresponding IAR-play.

(i) In the play ρ' a suffix $i_{m+1} \dots i_r$ of the IAR-component eventually remains fixed. It consists of the j, such that E_j-vertices are visited only finitely often ("suffix part of the *IAR*'s"). Let m be the maximal position in front of this suffix part. Therefore from some point onwards: e-value $\leq m$.

(ii) Player 1 wins the play ρ in G (Streett condition violated).

$\Longleftrightarrow$ exists j with E_j visited only finitely often, F_j visited infinitely often (in ρ).

$\overset{(i)}{\Longleftrightarrow}$ exists j, which eventually stands in the suffix part of the IAR's of ρ', such that F_j is visited infinitely often.

$\Longleftrightarrow$ Infinitely often in ρ' the f-value is $> m$ (= maximal position k with the current state $\in F_{j_k}$).

$\Longleftrightarrow$ Infinitely often $f > m$ holds.

In Example 4.61: $m = 3$. At vertex 5 $f = 4 > 3 = m$ holds.

(iii) Player 1 wins the play ρ.

$\overset{(ii)}{\Longleftrightarrow}$ Infinitely often in ρ': $f > m$.

$\Longleftrightarrow$ Infinitely often in ρ': $f > m \geq e$-value (Choice of m in (i)).

$\Longleftrightarrow$ in ρ' an odd color $2f - 1 > 2m$ is reached infinitely often.

(iv) Player 0 wins the play ρ.

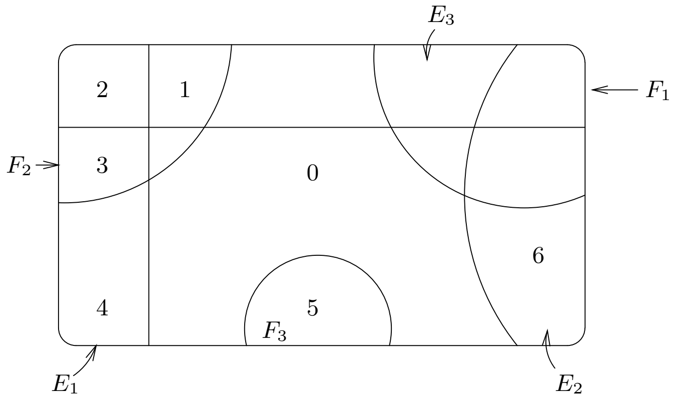
$$\begin{aligned}
&\overset{\text{(ii)}}{\iff} && f \le m \text{ eventually holds in } \rho' \text{ and infinitely often } e = m.\\
&\iff && \text{In } \rho' \text{ the maximal odd color eventually is } \le 2m-1 \text{ and maximal even}\\
& && \text{color is } = 2m.
\end{aligned}$$

Altogether the following holds

$$\text{Player 0 wins } \rho \Leftrightarrow \max(\mathrm{Inf}(c(\rho'))) \text{ even} \Leftrightarrow \text{Player 0 wins } \rho'.$$

$\square$

Example 4.61. Consider the following schematically represented game graph, with edges $(i, (i+1) \bmod 7)$.



Let $E_4 = F_4 = Q$. Consider the play $\rho = (0 \dots 6)^\omega$. In the IAR's, always those j with $q \notin E_j$ are underlined. In this example $m = 3$ (from (i) of the proof) and at the vertex 5 $f = 4 > 3 = m$ holds. The values for the *IAR*, e, and f in the first line are initial values and do not have a special meaning.

ρ'	q	IAR	e	f
	0	$123\underline{\overline{4}}$	4	4
	1	$41\overline{2}\underline{3}$	4	3
	2	$4\underline{1}\overline{2}3$	2	3
	3	$4\underline{1}\overline{2}3$	2	3
	4	$\overline{4}\underline{1}23$	2	1
	5	$\underline{4}12\overline{3}$	1	4
	6	$\overline{4}2\underline{1}3$	3	1
	0	$\underline{\overline{4}}213$	1	1
	1	$\underline{4}2\overline{1}3$	1	3

$\boxtimes$

4.13 Solving Games with Logical Winning Conditions

We now extend the results of the previous sections to games where we have more freedom in defining the acceptance condition.

Theorem 4.62. *Given a finite game graph G and a winning condition φ expressed in the logic S1S or the logic LTL, the corresponding game can be solved effectively, i.e. one can compute the winning regions of the two players, as well as corresponding finite-state winning strategies.*

The formulas may either refer to the names of the vertices or to a coloring of a game graph.

Example 4.63. We use vertex names $1, \dots, n$. Consider the winning condition:

"Between any two visits of vertex 2, at least one visit of state 1 occurs."

An equivalent S1S-formula:

$$\forall t_1 \forall t_2 (t_1 < t_2 \wedge X_2(t_1) \wedge X_2(t_2) \;\rightarrow\; \exists s (t_1 < s \wedge s < t_2 \wedge X_1(s))$$

An equivalent LTL-formula:

$$\mathrm{G}(p_2 \wedge \mathrm{XF} p_2 \;\rightarrow\; \mathrm{X}(\neg p_2 \mathrm{U} (p_1 \wedge \neg p_2))$$

⊠

The game can be solved by reduction. Recall that LTL-formulas can be written as S1S-formulas. Now consider a game graph G and a S1S-winning condition φ. Transform φ into a Büchi automaton and after that into a Muller automaton.

This yields a game reduction $(G, \varphi) < (G', \varphi')$ where φ' is a Muller condition. Another game reduction $(G', \varphi') < (G'', \varphi'')$ yields a parity game.

The solution of the parity game gives a solution of (G, φ) by finite-state strategies. But there is one essential problem: The blow-up of the size of the game graph.

Let us consider the algorithm in more detail:

Given: Game graph $G = (Q, Q_0, E)$ with $Q = \{1, \dots, n\}$ and an S1S formula $\varphi(X_1, \dots X_n)$ expressing the winning condition for Player 0. (Here the X_i are assumed to define a partition: Each position in an ω-word belongs precisely to one X_i.) Consider the Muller automaton $\mathcal{A} = (S, \{1, \dots, n\}, s_0, \delta, \mathcal{F})$ equivalent to φ. We define a game reduction $(G, \varphi) \leq (G', \varphi')$, with transformation of plays ρ over G into a play ρ' over G', such that ρ satisfies φ iff ρ' satisfies φ':

Given $\mathcal{A} = (S, \{1, \dots, n\}, s_0, \delta, \mathcal{F})$ equivalent to φ, establish a game reduction $(G, \varphi) \leq (G', \varphi')$ as follows:

- $G' = (S \times Q, S \times Q_0, E')$
- E' with edges $((s, p), (\delta(s, p), q))$ for $(p, q) \in E$
- the initialization function $f : q \mapsto (s_0, q)$
- φ' as the Muller condition on sequences $\rho' \in (S \times Q)^\omega$ using $\mathcal{F}$ and the first components in S

Then $\rho \models \varphi(X_1, \ldots, X_n)$ iff ρ' defined by 2 and 3 satisfies the Muller condition φ'.

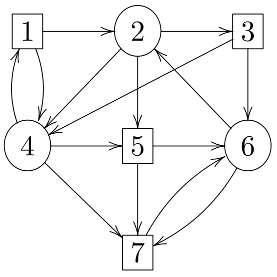
S1S formulas can be transformed into nondeterministic Büchi automata. Why not use them instead of Muller automata? Player 0 could be asked

- to make his moves,
- to simultaneously build up a successful run of the Büchi automaton equivalent to the S1S-formula,

and thus play a Büchi game. Later on we shall see that (and why) this cannot work.

4.14 Exercises

Consider the following game graph for Exercise 4.1 and 4.2:



Exercise 4.1. Let the winning conditions for Player 0 be

(a) $|\mathrm{Occ}(\rho))| \leq 2$, and

(b) $|\mathrm{Occ}(\rho))| \leq 3$.

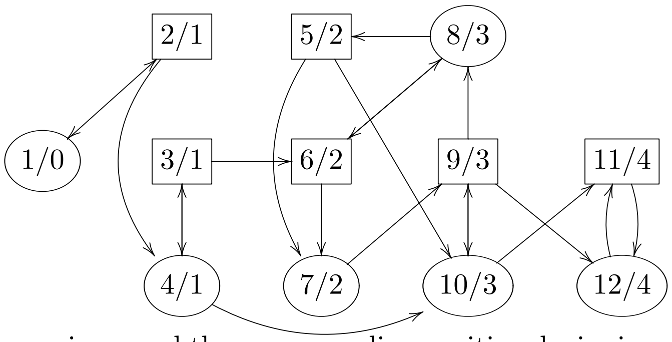
Find for each winning condition the winning region for Player 0 and describe (informally) a winning strategy. Hint: positional strategies suffice.

Exercise 4.2. Let the winning condition for Player 0 be $\mathrm{Occ}(\rho) = \{1, 2, 3, 4, 5, 6, 7\}$.

(a) Find the winning region for Player 0 and describe a winning strategy.

(b) Show that there is no positional winning strategy for Player 0 in this game.

Exercise 4.3. Consider the following weak parity game, where vertex i with color j is denoted by i/j.



Compute the winning regions, and the corresponding positional winning strategies, for Player 0 and 1.

Exercise 4.4. Let G be a game graph for a weak parity game and let W_0 be the winning region of Player 0 . For all $q \in W_0$ let Player 0 have a positional winning strategy f_q starting from q. From these winning strategies construct a "uniform" positional strategy f which is a winning strategy from q for every $q \in W_0$ (up to now only such uniform strategies have been used). The strategy f, constructed from the strategies f_q, means that for every $p \in Q$ there is a $q \in Q$ with $f(p) = f_q(p)$.

Exercise 4.5. (a) Specify an ω-language L, such that L can be recognized by a weak parity automaton with 3 colors but not by a weak parity automaton with 2 colors.

(b) Generalize the result of (a) in the following sense. Specify a family $(L_n)_{n\geq 1}$ of ω-languages, such that $L_n \subseteq \Sigma_n^\omega$ can be recognized by a weak parity automaton with n colors but not with $n-1$ colors. The alphabets Σ_n may grow with n.

Exercise 4.6. Give a SW-game in which Player 0 has got a positional winning strategy f_q from q for every vertex q but no uniform positional winning strategy f.

Exercise 4.7. For $n \geq 1$ specify game graphs G_n with $\mathcal{O}(n)$ vertices, a marked vertex q_0, and a SW-winning condition, so that the following holds:

(1) Player 0 wins over G_n starting from q_0 using a strategy automaton $\mathcal{A}_n$ that has got n states. (Specify this automaton.)

(2) No strategy automaton with $< n$ states implements a winning strategy over G_n for Player 0 starting from vertex q_0.

Exercise 4.8. A *conjunctive guaranty condition* is given by sets $G_1, \dots, G_k \subseteq Q$, and the winning condition for Player 0 in a conjunctive guaranty game is

$$\mathrm{Occ}(\rho) \cap G_i \neq \emptyset \text{ for every } i \in \{1, \dots k\}.$$

(a) Construct a game graph and a conjunctive guaranty condition such that the winning region for Player 0 in the conjunctive guaranty game is not $\bigcap_{i=1}^k \mathrm{Attr}_0(G_i)$.

(b) Show that if a winning strategy for Player 0 exists then a memory of size $2^{O(k)}$ suffices to implement an automaton winning strategy for Player 0.

Exercise 4.9. Find a family of game graphs $(G_n)_{n\geq 1}$ with designated sets $(F_n)_{n\geq 1}$ (and Büchi winning condition) such that $\mathrm{Recur}_0^{i+1}(F_n) \subsetneqq \mathrm{Recur}_0^i(F_n)$ for $i = 1, \dots, n$.

Exercise 4.10. Show: For a game over the graph $G = (Q, E)$ with a subset $F \subseteq Q$ and co-Büchi winning condition ($\rho \in \mathrm{Win} \Leftrightarrow \mathrm{Inf}(\rho) \cap F = \emptyset$) one can compute the winning regions W_0, W_1 and specify positional winning strategies for Player 0 / Player 1 .

Exercise 4.11. Let $G = (Q = Q_0 \,\dot\cup\, Q_1, E)$ be a game graph with $F \subseteq Q$. Let W_0 and W_1 be the winning regions of Player 0 and Player 1 in the Büchi game for G and F. Prove or disprove:

(a) The winning set of Player 0 in the safety game for G and W_0 is W_0.

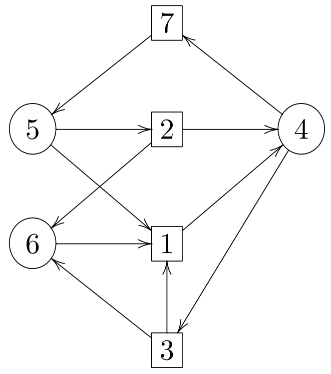
(b) If f_0 is a winning strategy for Player 0 in the safety game for G and W_0, then f_0 is also a winning strategy for Player 0 in the Büchi game for G and F.

(c) The winning set of Player 1 in the guaranty game for G and W_0 is W_1.

(d) If f_1 is a winning strategy for Player 1 in the guaranty game for G and W_0, then f_1 is also a winning strategy for Player 1 in the Büchi game for G and F.

Exercise 4.12.

Consider the game graph G on the right, and the Muller condition $\mathcal{F} = \{\{2,4,5,7\},\{1,2,3,4,5,6,7\}\}$.
Find an automaton winning strategy for Player 0 in the Muller game with as few states as possible, and show that any other automaton strategy with less states is not winning for Player 0.



Exercise 4.13.

(a) Find a family $(G_n, \mathcal{F}_n)_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ Player 0 has a winning strategy in the Muller game $(G_n, \mathcal{F}_n)$ which can be realized by a strategy automaton with n states, but there is no winning strategy which can be realized by a strategy automaton with less than n states.

(b) Modify your family from (a) such that Player 0 has two winning strategies f_1, f_2 and the minimal automaton implementing f_2 needs n more states than the minimal automaton implementing the strategy f_1.

Exercise 4.14. Consider the DJW game over the graph with vertices $\{A,B,C,D,1,2,3,4\}$ as given in Example 4.39. We propose the following *latest appearance queue (LAQ)* strategy for Player 0, initializing the queue with ABCD.

- Add the current vertex at the front of the LAQ, delete the last vertex.
- Move to the vertex whose number is the number of different vertices in the current LAQ.

The following table shows the evolution of the LAQ for an example sequence of visited letter states:

	A	C	C	D	B	D	C	D	D	...
ABCD	AABC	CAAB	CCAA	DCCA	BDCC	DBDC	CDBD	DCDB	DDCD	...

Decide whether the new LAQ strategy is a winning strategy for Player 0. Prove this or give a counter-example.

Exercise 4.15. Let $G = (Q, E)$ with $Q = Q_0 \dot\cup Q_1$ a be game graph. A *nondeterministic positional strategy (NPS)* for Player 0 in G is a relation $R \subseteq (Q_0 \times Q) \cap E$. A play $q_0, q_1, \ldots$ is played according to R if for all $q_i \in Q_0$ we have $(q_i, q_{i+1}) \in R$. Let ϕ be a winning condition. An NPS R is a nondeterministic winning strategy for player 0 from q if all plays from q played according to R satisfy ϕ.

Decide whether the union of two winning NPS from q is again a winning NPS from q for

(a) Büchi conditions ($\phi = \forall i \exists j > i \ \rho(i) \in F$).

(b) Reachability conditions ($\phi = \exists i \ \rho(i) \in F$).

Exercise 4.16. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a deterministic Muller automaton. A set $W \subseteq Q$ is called a *set of relevant states* if for every loop $F \in \mathcal{F}$ and every loop $F' \in 2^Q \setminus \mathcal{F}$ we have that $W \cap (F \bigtriangleup F') \neq \emptyset$, i.e. whether a loop belongs to $\mathcal{F}$ can be decided by just looking at the states of W. (Here $F \bigtriangleup F' := (F \setminus F') \cup (F' \setminus F)$ denotes the *symmetric difference* of the sets F and F'.)

(a) Show that a run ρ of $\mathcal{A}$ is accepting iff $\mathrm{Inf}(\rho) \cap W = F \cap W$ for an $F \in \mathcal{F}$.

(b) Show that one can construct a parity automaton $\mathcal{B}$ equivalent to $\mathcal{A}$ that uses as set of states $Q \times LAR(W)$ (instead of the set of LARs over Q).

Exercise 4.17. Let $G = (Q, Q_0, E)$ be a game graph and $\mathcal{F}$ a Muller condition such that

$$\forall P_1, P_2 \subseteq Q \quad P_1, P_2 \notin \mathcal{F} \Rightarrow P_1 \cup P_2 \notin \mathcal{F}.$$

Find a Rabin condition equivalent to $\mathcal{F}$.
Hint: Use for every accepting loop P in $(G, \mathcal{F})$ a Rabin pair. To find this pair consider the union of all non-accepting loops contained in P.
Comment: In the lecture the Union Lemma was shown for the Rabin condition. Here we show a converse statement.

Exercise 4.18. Let $G = (Q, E)$ be a game graph. A *liveliness condition* is given by a set $\Omega = \{(E_1, F_1), \ldots, (E_r, F_r)\}$ of pairs of state sets (like a Rabin condition). Player 0 wins a play ρ in the liveliness game (G, Ω) iff

$$\bigwedge_{i=1}^{r} \forall x \ \left(\rho(x) \in F_i \rightarrow \exists y > x \ (\rho(y) \in E_i)\right).$$

Specify a reduction of liveliness games to parity or Büchi games.

Exercise 4.19. Let $\mathcal{A}$ be a strategy automaton. Solve either (a) or (b).

(a) Specify an algorithm, which gives the divides the states of $\mathcal{A}$ into $\sim_{\mathcal{A}}$-classes.

(b) Show (using the properties of $\mathcal{A}_{\mathrm{red}}$ and $\mathcal{A}_f$ as in Section 4.10) that the function $I : Q(\mathcal{A}_{\mathrm{red}}) \rightarrow Q(\mathcal{A}_f)$, defined in Section 4.10, is a bijection.

Exercise 4.20. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, c)$ with $c : Q \rightarrow \{1, \ldots, k\}$ be a parity automaton. Construct a Streett automaton $\mathcal{B} = (Q, \Sigma, q_0, \delta, \Omega)$ equivalent to $\mathcal{A}$ which has the same transition structure as $\mathcal{A}$.

Chapter 5

Tree Automata and the Logic S2S

5.1 Trees and Tree Automata

In Chapter 2 we specified sequence properties in a linear way. We developed a *linear-time specification* of infinite behavior by setting the same conditions on every infinite path through a system which was modeled by a pointed Kripke structure.

Now we want to be able to set global conditions on the structure of *all* possible paths of a system. Namely, we are going to specify properties of the *computation trees* of a system. This will entail the *branching-time specifications*, which are conditions on the structure of the tree formed by all paths through a Kripke structure.

Definition 5.1. If $(\mathcal{M}, s_0)$ is a pointed Kripke structure, $t_{(\mathcal{M},s_0)}$ is the *tree of runs* from s_0.

This tree will also be called *unraveling* of $\mathcal{M}$ from s_0

Example 5.2. Consider the Kripke structure $\mathcal{M}$ and its *labeled* tree of runs $t_{(\mathcal{M},s_0)}$.

$\mathcal{M}$:

$$\begin{tikzcd}
 & \overset{\binom{1}{0}}{s_1} \arrow[r] \arrow[dd, leftrightarrow] & \underset{\binom{1}{1}}{s_3} \arrow[loop right] \\
\binom{0}{0}\; s_0 \arrow[ur] \arrow[dr] & & \\
 & \underset{\binom{0}{1}}{s_2} &
\end{tikzcd}$$

$t_{(\mathcal{M},s_0)}$:

$$\begin{tikzcd}
 & & \binom{0}{0} \arrow[dl] \arrow[dr] & \\
 & \binom{1}{0} \arrow[dl] \arrow[dr] & & \binom{0}{1} \arrow[d] \\
\cdots & & \cdots & \cdots
\end{tikzcd}$$

⊠

In order to achieve the above mentioned goals, we are going to do the following: We will specify properties of infinite trees instead of properties of infinite words. For these specifications we will use automata accepting trees, and logic formulas which will be evaluated over trees.

We will introduce

- trees and tree languages
- automata accepting tree languages

and study the expressive power of tree automata.

The application of these new concepts will yield very strong decidability results in logic and verification.

Definition 5.3. Let the words $u \in \{0,1\}^*$ denote the nodes of the unlabeled infinite binary tree, where ϵ is the root of the tree and $v0$ denotes the left child of the node $v \in \{0,1\}^*$ and vice versa. A Σ-*valued binary tree* is a function $t : \{0,1\}^* \to \Sigma$, i.e. at node $u \in \{0,1\}^*$ the value of t is $t(u)$.

Some additional tree terminology will be needed later on:

Definition 5.4. (Tree Terminology)

- T^ω_Σ is the set of all Σ-valued binary trees.
- A set $T \subseteq T^\omega_\Sigma$ is called *tree language* (of Σ-valued binary infinite trees).
- A *path* through a tree t is a sequence $\pi = u_0u_1u_2\ldots$ of tree nodes with $u_0 = \epsilon$ (root of the tree) and $u_{i+1} = u_i0$ or $u_{i+1} = u_i1$ for $i \geq 0$.
- The ω-word determined by π is $t|_\pi = t(u_0)\ t(u_1)\ t(u_2)\ldots$.

Now we will define automata that can accept binary trees.

Definition 5.5. (Tree Automaton) A *tree automaton* (over Σ-valued binary trees) is of the form $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathrm{Acc})$, where

- Q is the finite set of states, q_0 the initial state,
- $\Delta \subseteq Q \times \Sigma \times Q \times Q$ is the transition relation,
- Acc is the acceptance component.

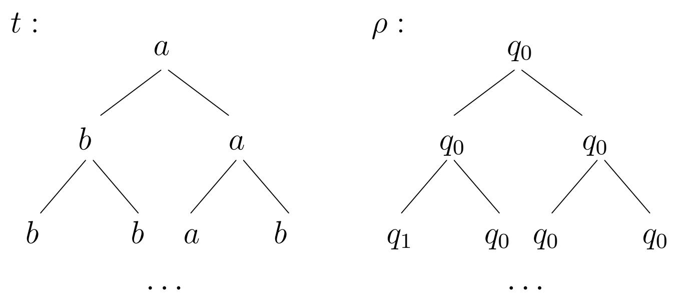
A transition (q, a, q_1, q_2) allows the automaton in state q at an a-labeled node u to proceed to states q_1, q_2 at the successor nodes $u0, u1$.

The automaton is *deterministic* if for any $(q, a) \in Q \times \Sigma$ at most one transition exists in Δ whose first two components are q and a.

Definition 5.6. A *run* of $\mathcal{A}$ on t is an assignment of states to tree nodes, i.e. a tree $\rho : \{0,1\}^* \to Q$ with

- $\rho(\epsilon) = q_0$
- $(\rho(u), t(u), \rho(u0), \rho(u1)) \in \Delta$ for all $u \in \{0,1\}^*$

Example 5.7. We assume that the following transitions are allowed in a tree automaton: (q_0, a, q_0, q_0), (q_1, a, q_1, q_1), (q_0, b, q_1, q_0), (q_1, b, q_1, q_1). The following picture shows a run ρ (on the right) on the tree t on the left.



⊠

Definition 5.8. (Acceptance of Trees) Let A be an acceptance condition for ω-automata. The tree automaton run ρ is successful for condition A if *each path* of ρ is successful with respect to A.

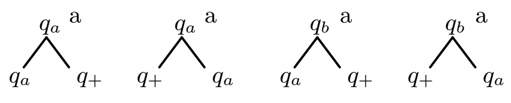
This is applied to any acceptance condition A known from ω-automata theory. Thus we get *Büchi tree automata*, *Muller tree automata*, *Rabin tree automata* etc. For instance a Büchi tree automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ accepts the tree t if there exists a run ρ of $\mathcal{A}$ on t such that on each path of ρ a state from F occurs infinitely often.

Definition 5.9. The *tree language* recognized by the tree automaton $\mathcal{A}$, denoted $T(\mathcal{A})$, is the set of all trees accepted by $\mathcal{A}$ with the acceptance condition A under consideration.

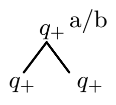
Example 5.10. Let $T_1 = \{t \in T^\omega_{\{a,b\}} \mid \text{exists path through } t \text{ with infinitely many } b\}$, in short: $T_1 = \{t \in T^\omega_{\{a,b\}} \mid \exists\pi\exists^\omega i\ t|_\pi(i) = b\}$.

We define the following Büchi tree automaton $\mathcal{A}_1$:

- State set $Q = \{q_a, q_b, q_+\}$, initial state q_a,
- Final state set $F = \{q_b, q_+\}$,
- Transition relation Δ :



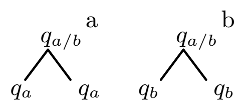
analogous for input b
with state q_b at the bottom



A run of $\mathcal{A}_1$ will necessarily "guess" a path by assuming states q_a, q_b there. All other paths will finally have state q_+ only (accepting). State q_a signals input label a, q_b signals input label b.

Acceptance on the path with states q_a, q_b requires infinitely many visits to q_b, i.e infinitely many b on the path have to be assumed. So the Büchi tree automaton $\mathcal{A}_1$ recognizes T_1 ⊠

Example 5.11. Let $T_2 = \{t \in T^\omega_{\{a,b\}} |\text{ each path through } t \text{ has only finitely many } b\}$. Define a Muller tree automaton $\mathcal{A}_2 = (Q, \{a, b\}, q_a, \Delta, \mathcal{F})$ with state set $Q = \{q_a, q_b\}$, $\mathcal{F} = \{\{q_a\}\}$ and transition relation Δ :



Then for the unique run ρ on t and for all paths π: $t|_\pi$ has infinitely many b iff $\rho|_\pi$ has infinitely many q_b. The acceptance condition requires $\text{Inf}(\rho|_\pi) = \{q_a\}$ for all π. So $\mathcal{A}_2$ accepts precisely the trees in T_2. ⊠

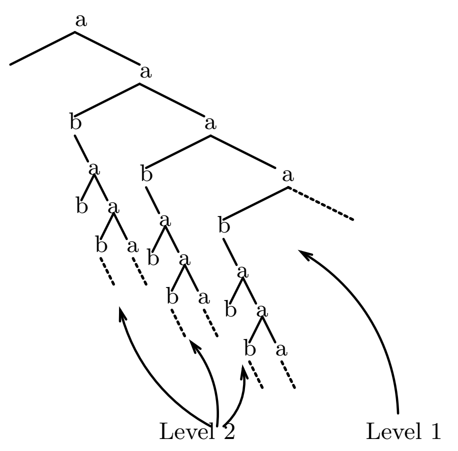
In contrast to the ordinary ω-automata, deterministic Muller tree automata and nondeterministic Büchi tree automata are not equivalent.

Theorem 5.12. *The tree language $T_2 = \{t \in T^\omega_{\{a,b\}} |$ each path through t has only finitely many $b\}$ is recognized by a Muller tree automaton, but not by a (nondeterministic) Büchi tree automaton.*

Proof We verified that T_2 is recognized by an (even deterministic) Muller tree automaton. Assume: the Büchi automaton $\mathcal{A} = (Q, \{a, b\}, q_0, \Delta, F)$ recognizes T_2

Let $n = |F| + 1$. Consider the following input tree t:

$$t(w) = \begin{cases} b & w \in (1^+0)^i \text{ for } i \in \{1, \dots, n\} \\ a & \text{else} \end{cases}$$



We have $t \in T_2$, so there is a successful run ρ of $\mathcal{A}$ on t. On the path 1^ω a final state is visited, say at $v_0 = 1^{m_0}$.

On the path $1^{m_0}01^\omega$, final states are visited infinitely often. After 0 say for the first time at $v_1 = 1^{m_0}01^{m_1}$. We therefore obtain visits to final states at the nodes $v_0 = 1^{m_0}$, $v_1 = 1^{m_0}01^{m_1}$, $\dots$, $v_n = 1^{m_0}0\dots01^{m_n}$. For certain $i < j$ the same final state appears at v_i and v_j. Between v_i, v_j at least one label b occurs (at v_i0).

Construct a new tree t' by iterated copying of the part between v_i and v_j and a corresponding run ρ'. Then $\mathcal{A}$ also accepts t'. Thus $t' \in T_2$, but in t' infinitely many b occur on the new path π. Contradiction. □

We have shown: nondeterministic Büchi tree automata are strictly weaker than Muller tree automata. On the other hand, we will show:

- Nondeterministic Muller tree automata and parity tree automata have the same expressive power.

- Parity tree automata have good logical closure properties.
- The emptiness problem is decidable for parity tree automata.

5.2 Parity tree automata

Definition 5.13. A *parity tree automaton* is of the form $\mathcal{A} = (Q, \Sigma, q_0, \Delta, c)$ with coloring $c : Q \to \{0, \dots, k\}$. It accepts a tree t if there is a run ρ of $\mathcal{A}$ on t such that on each path of ρ, the maximal color assumed infinitely often is even.

Example 5.14. A parity automaton that recognizes

$$T_2 = \{t \in T^\omega_{\{a,b\}} \mid \text{each path through } t \text{ has only finitely many } b\}.$$

Use q_a, q_b to signal input letters a, b respectively. Define $c(q_a) = 0,\ \ c(q_b) = 1$.

The maximal color occurring infinitely often on a path of the run is even (i.e., equal to 0) iff the letter b occurs only finitely often on the path. ⊠

Theorem 5.15. (Muller versus Parity Tree Automata)

1. *For any parity tree automaton one can construct an equivalent Muller tree automaton.*
2. *For any Muller tree automaton one can construct an equivalent parity tree automaton.*

Proof 1 $\Rightarrow$ 2: Given a parity tree automaton with coloring c, keep states and transitions and define the system $\mathcal{F}$ as follows:

$$R \in \mathcal{F} \ :\Leftrightarrow\ \max\{c(q) \mid q \in R\} \text{ is even}$$

Proof 2. $\Rightarrow$ 1: Copy the simulation of Muller games by parity games. Given a Muller tree automaton with state set Q, use for the parity tree automaton the state set LAR(Q) and define the transitions according to the LAR update rule.

Allow the transition

$$(((p_1 \dots p_n), i), a, ((q_1 \dots q_n), j), ((r_1 \dots r_n), k))$$

for a transition (p_1, a, q_1, r_1) of the Muller automaton, where $((q_1 \dots q_n), j)$ is the LAR update for a visit of q_1, and $((r_1 \dots r_n), k)$ is the LAR update for a visit of r_1.

Define coloring as in the simulation of Muller games by parity games. □

We will now prove some closure properties of parity tree automata.

Lemma 5.16. (Closure under Union) *Given parity tree automata $\mathcal{A}_1$, $\mathcal{A}_2$, one can construct a parity tree automaton recognizing $T(\mathcal{A}_1) \cup T(\mathcal{A}_2)$.*

Proof Assume the state sets Q_1, Q_2 of $\mathcal{A}_1$, $\mathcal{A}_2$ are disjoint, with initial states q_1, q_2. Define the new automaton over $\{q_0\} \cup Q_1 \cup Q_2$ with new initial state q_0 (say of color 0).

Take all transitions of $\mathcal{A}_1$, $\mathcal{A}_2$. Add, for any transition (q_1, a, r_1, r_2) or (q_2, a, r_1, r_2) the new transition (q_0, a, r_1, r_2). □

Definition 5.17. Given trees $s \in T_\Gamma^\omega, t \in T_\Sigma^\omega$, define the tree $s^\wedge t \in T_{\Gamma\times\Sigma}^\omega$ by $s^\wedge t(u) = (s(u), t(u))$. The Σ-*projection* of $s^\wedge t$ is the tree t.

Given a tree language T over $\Gamma \times \Sigma$ define

$$\text{proj}_\Sigma(T) := \{t \in T_\Sigma^\omega \mid \exists s : s^\wedge t \in T\}.$$

Lemma 5.18. (Closure under Projection) *Given a parity tree automaton recognizing T over the alphabet $\Gamma \times \Sigma$ one can construct a parity tree automaton recognizing* $\text{proj}_\Sigma(T)$.

Proof Given a parity tree automaton $\mathcal{A}$ over $\Gamma \times \Sigma$, define a parity tree automaton $\mathcal{B}$ over Σ, which does in any step the following: For input letter $b \in \Sigma$ guess the Γ-component a and proceed by an $\mathcal{A}$-transition for the input letter (a, b). The state set is not changed. □

The complementation of parity tree automata remains to be investigated. This is the most difficult logical closure property. M. O. Rabin succeeded 1969 in showing it for Muller tree automata, which was a breakthrough for the analysis of logical theories.

Today a simpler proof is available that uses the parity acceptance condition. One useful idea for the complementation of parity tree automata is to formulate acceptance in the terminology of games.

5.3 Tree Automata and Games, Complementation

With any parity tree automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, c)$ and any input tree t associate a parity game over a graph $G_{\mathcal{A},t}$ played by two players "Automaton" and "Pathfinder" on the tree t.

Described intuitively, the game proceeds as follows:

- First Automaton picks a transition from Δ, which can serve to start a run at the root of the input tree.
- Then Pathfinder decides on a direction (left or right) to proceed to a son of the root.
- Then Automaton chooses again a transition for this node (compatible with the first transition and the input tree).
- Then Pathfinder reacts again by branching left or right from the momentary node, ...

Such a play gives a sequence of transitions (and hence a state sequence from Q), built up along a path chosen by Pathfinder. The Automaton wins the play iff the constructed state sequence satisfies the parity condition.

The game graph $G_{\mathcal{A},t}$ for a parity tree automaton $\mathcal{A}$ and an input tree t is defined as follows:

The vertices of Automaton are the triples

$$(\text{tree node } w, \text{tree label } t(w), \text{state } q \text{ at } w).$$

By choice of a transition τ of the form $(q, t(w), q', q'')$, a vertex of Pathfinder is reached.

The vertices of Pathfinder are the triples

$$(\text{tree node } w, \text{tree label } t(w), \text{transition } \tau \text{ at } w).$$

Notice that the game graph is infinite and depends on t. A small part of a game graph is shown in the following figure:

$$\begin{tikzcd}
(w,t(w),q) \arrow[rr] & & \boxed{(w,t(w),\underbrace{(q,t(w)),q',q'')}_{\tau})} \arrow[dl] \arrow[dr] & \\
 & (w0,t(w0),q') & & (w1,t(w1),q'')
\end{tikzcd}$$

To give a formal definition:

Definition 5.19. Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, c)$ be a parity tree automaton and t be an input tree. Then the *associated game graph* $G_{\mathcal{A},t} = (Q_0 \cup Q_1, E)$ is defined by

$$Q_0 = \text{ set of triples } (w, t(w), q) \in \{0,1\}^* \times \Sigma \times Q,$$

$$Q_1 = \text{ set of triples } (w, t(w), \tau) \in \{0,1\}^* \times \Sigma \times \Delta,$$

and the edge relation E such that succeeding game positions match and are also compatible with t.

The color of a triple $(w, t(w), q)$, resp. $(w, t(w), (q, a, q', q''))$, is the color $c(q)$. The standard initial position of the play is Automaton's position $(\epsilon, t(\epsilon), q_0)$.

A successful run of $\mathcal{A}$ on t yields a winning strategy for Automaton in the parity game over $G_{\mathcal{A},t}$: Along each path the suitable choice of transitions is fixed by the run.

Conversely, a winning strategy for Automaton over $G_{\mathcal{A},t}$ clearly provides a method to build up a successful run of $\mathcal{A}$ on t. Just apply the winning strategy along arbitrary paths.

Lemma 5.20. (Run Lemma) *The tree automaton $\mathcal{A}$ accepts the input tree t iff in the parity game over $G_{\mathcal{A},t}$ there is a winning strategy for player Automaton from the initial position $(\epsilon, t(\epsilon), q_0)$.*

We will also consider an easier case of the Run Lemma, which is concerned with parity automata with no input. Applying the Run Lemma this way, we can determine whether there is a successful run of the tree automaton at all.

Definition 5.21. Given an input-free parity tree automaton $\mathcal{A} = (Q, q_0, \Delta, c)$ with $\Delta \subseteq Q \times Q \times Q$, define a simpler game graph $G_{\mathcal{A}}$, in which the tree t and the parameter w in the game positions are suppressed: Let $Q_0 = Q$, $Q_1 = \Delta$. Let E contain the edges $(q, (q, q', q''))$, $((q, q', q''), q')$ and $((q, q', q''), q'')$ for $(q, q', q'') \in \Delta$. The coloring coincides with c on Q_0 $(= Q)$ and maps a transition $(q, q', q'') \in Q_1$ to $c(q)$.

Lemma 5.22. (Run Lemma, input-free case) *Player* 0 *(Automaton) has a winning strategy in $G_{\mathcal{A}}$ from position q_0 iff the automaton $\mathcal{A}$ admits at least one successful run.*

We will now briefly outline the path towards a proof of the complementation of tree automata. To complement a tree automaton means to express that a given automaton $\mathcal{A}$ does not accept t by the acceptance of t of another automaton.

In view of the Run Lemma this means to conclude from *non-existence* of a winning strategy for Automaton over $G_{\mathcal{A},t}$ the *existence* of a winning strategy for Automaton in a different game $G_{\mathcal{B},t}$ (such that $\mathcal{B}$ depends only on $\mathcal{A}$ but not on t).

- First step: Use determinacy of parity games to show: If Automaton has no winning strategy over $G_{\mathcal{A},t}$, then Pathfinder has a winning strategy (from the standard initial position) over $G_{\mathcal{A},t}$.
- Second step: Pathfinder's strategy is converted to an Automaton strategy.

Theorem 5.23. (Determinacy of Parity Tree Automata Games) *Let $\mathcal{A}$ be a parity tree automaton and t be an input tree for $\mathcal{A}$. Then in the parity game over $G_{\mathcal{A},t}$, from any game position either Automaton or Pathfinder has a positional winning strategy.*

This is just a special case of the determinacy of parity games.

Theorem 5.24. (Complementation Theorem) *For any parity tree automaton $\mathcal{A}$ over the alphabet Σ one can effectively construct a Muller tree automaton (and hence also a parity tree automaton) $\mathcal{B}$, which recognizes $T^{\omega}_{\Sigma} \setminus T(\mathcal{A})$.*

Proof Let $\mathcal{A} = (Q, \Sigma, q_0, \Delta, c)$ be a parity tree automaton. We find a Muller tree automaton $\mathcal{B}$ accepting precisely the trees $t \in T^{\omega}_{\Sigma}$, which are not accepted by $\mathcal{A}$.

Step 1 (Applying determinacy) : Start with the following equivalences: For any tree t,

$\mathcal{A}$ does not accept t

iff (by Run Lemma)
Automaton has no winning strategy from the initial position $(\epsilon, t(\epsilon), q_0)$ in the parity game over $G_{\mathcal{A},t}$

iff (by Determinacy Theorem)
$(*)$ over $G_{\mathcal{A},t}$, Pathfinder has a positional winning strategy from $(\epsilon, t(\epsilon), q_0)$.

Step 2 (Conversion to automaton strategy): Reformulate $(*)$ in the form "$\mathcal{B}$ accepts t" for some tree automaton $\mathcal{B}$. Pathfinder's strategy is a function f from the set $\{0,1\}^* \times \Sigma \times \Delta$ of his vertices into the set $\{0,1\}$ of directions.

Decompose this function into a family

$$f_w : \Sigma \times \Delta \to \{0,1\}$$

of "local instructions", parameterized by $w \in \{0,1\}^*$.

The set I of possible local instructions $i : \Sigma \times \Delta \to \{0,1\}$ is finite. Thus Pathfinder's winning strategy can be coded by the I-labeled tree s with $s(w) = f_w$.

Let $s^\wedge t$ be the corresponding $(I \times \Sigma)$-labeled tree with

$$s^\wedge t(w) = (s(w), t(w)) \text{ for } w \in \{0,1\}^*.$$

Now $(*)$ is equivalent to the following:

> There is an I-labeled tree s such that for all sequences $\tau_0\tau_1\ldots$ of transitions chosen by Automaton and for all (in fact for the unique) $\pi \in \{0,1\}^{\omega}$ determined by $\tau_0\tau_1\ldots$ via the strategy coded by s, the generated state sequence violates the parity condition.

A reformulation of this yields:

> (1) There is an I-labeled tree s such that $s^\wedge t$ satisfies:
>
> > (2) for all $\pi \in \{0,1\}^\omega$
> >
> > > (3) for all $\tau_0\tau_1 \ldots \in \Delta^\omega$
> > >
> > > > (4) if the sequence $s|\pi$ of local instructions applied to the sequence of tree labels $t|\pi$ and to the transition sequence $\tau_0\tau_1 \ldots$ indeed produces the path π, then the state sequence determined by $\tau_0\tau_1 \ldots$ violates the parity condition.

- Condition 4 describes a property of ω-words over $I \times \Sigma \times \Delta \times \{0,1\}$, which obviously can be checked by a sequential Muller automaton $\mathcal{M}_4$, independently of t.

- Condition 3 describes a property of ω-words over $I \times \Sigma \times \{0,1\}$, which results from 4 by a universal quantification (equivalently, by a negation, a projection, and another negation). Condition 3 is checked by a sequential and deterministic Muller automaton $\mathcal{M}_3$.

- Condition 2 defines a property of $(I \times \Sigma)$-labeled trees, which can be checked by a deterministic Muller *tree* automaton $\mathcal{M}_2$, simulating $\mathcal{M}_3$ along each path. (Note that, by determinism of $\mathcal{M}_3$, the $\mathcal{M}_3$-runs on different paths of an $(I \times \Sigma)$-labeled tree agree on the respective common prefix and hence can be merged into one run of $\mathcal{M}_2$)

- Applying nondeterminism, a Muller tree automaton $\mathcal{B}$ can be built, which checks Condition 1, by guessing a tree s on the input tree t and working on $s^\wedge t$ like $\mathcal{M}_2$.

$\mathcal{B}$ does not depend on the tree t under consideration. Thus $\mathcal{B}$ accepts precisely those trees, which $\mathcal{A}$ does not accept, as was to be shown. □

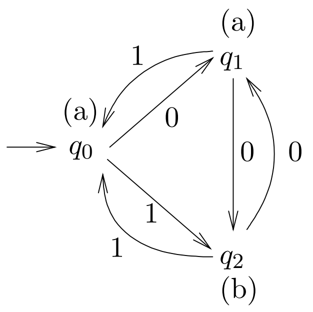
5.4 Towards the Nonemptiness Problem

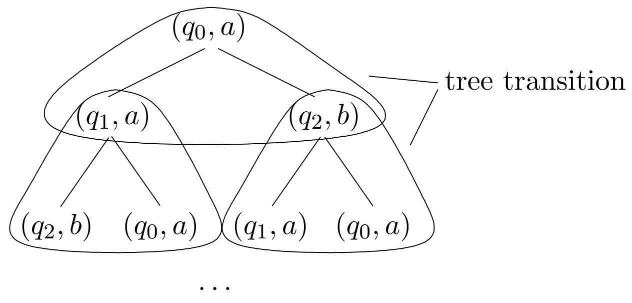
Recall: A nonempty regular ω-language contains an ultimately periodic ω-word. We show a corresponding result for nonempty tree languages, which are recognized by parity tree automata.

Definition 5.25. A tree $t \in T^\omega_\Sigma$ is called *regular* if it is "finitely generated" in the following sense: There is a deterministic finite automaton equipped with output, which tells for any given input $w \in \{0,1\}^*$ the label at node w (i.e. the value $t(w)$).

The automaton is of the form $\mathcal{B} = (Q_\mathcal{B}, \{0,1\}, q_{0\mathcal{B}}, \delta_\mathcal{B}, f_\mathcal{B})$ with $f_\mathcal{B} : Q_\mathcal{B} \to \Sigma$ (output function).

Example 5.26. Consider the deterministic output automaton $\mathcal{B}$. The computation tree of its runs forms an input-free deterministic tree automaton.





Therefore we can characterize regular trees by tree automata. ⊠

Theorem 5.27. *A tree $t \in T^\omega_\Sigma$ is regular iff there is a deterministic input-free tree automaton $\mathcal{C}$ with state set $Q \times \Sigma$ such that the Σ-projection of the unique run of $\mathcal{C}$ is the tree t.*

Proof "⇒": Given a DFA $\mathcal{B} = (Q_\mathcal{B}, \mathbb{B}, q_{0\mathcal{B}}, \delta_\mathcal{B}, f_\mathcal{B})$ with output function as above. As seen in Example 5.26, the construction of the corresponding tree automaton is straightforward. Define $\mathcal{C} = (Q_\mathcal{B} \times \Sigma, (q_{0\mathcal{B}}, a_0), \Delta)$ by

$$a_0 := f_\mathcal{B}(\delta_\mathcal{B}(q_0, \epsilon)).$$

Introduce the transition $((q_1, a_1), (q_2, a_2), (q_3, a_3))$ to Δ iff $f_\mathcal{B}(q_i) = a_i$ for $i = 1, 2, 3$, $\delta_\mathcal{B}(q_1, 0) = q_2$, and $\delta_\mathcal{B}(q_1, 1) = q_3$.

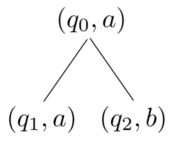
"⇐": For the converse we start with $\mathcal{C} = (Q \times \Sigma, (q_0, a_0), \Delta)$ and define $\mathcal{B} = (Q \times \Sigma, \mathbb{B}, (q_0, a_0), \delta, f)$. In order to construct $\delta((q, a), 0/1)$, inspect the unique $\mathcal{C}$-transition $((q, a), (q', a'), (q'', a''))$ with first state (q, a) and set

$$\delta((q, a), 0) := (q', a'), \quad \delta((q, a), 1) := (q'', a'').$$

Finally define: $f(q, a) := a$. □

The last construction is illustrated in the following example:

Example 5.28. The tree transition



is transformed into a part of the DFA's transition function: $\delta((q_0, a), 0) \mapsto (q_1, a)$, $\delta((q_0, a), 1) \mapsto (q_2, b)$. The output function uses the node label of the parent state: $f(q_0, a) \mapsto a$. ⊠

Theorem 5.29. (Rabin Basis Theorem) *For every parity tree automaton $\mathcal{A}$, the emptiness problem "$T_\omega(\mathcal{A}) = \emptyset$?" is decidable, and any nonempty set $T_\omega(\mathcal{A})$ contains a regular tree (whose generating automaton $\mathcal{B}$ is obtained from $\mathcal{A}$).*

Proof Assume $\mathcal{A} = (Q, \Sigma, q_0, \Delta, c)$ is a parity tree automaton. Proceed to an "input-guessing" (and input-free) tree automaton $\mathcal{A}'$, which nondeterministically generates an input tree t and on t works like $\mathcal{A}$. Note that $\mathcal{A}' = (Q \times \Sigma, \{q_0\} \times \Sigma, \Delta', c')$ has possibly several initial states. We know that $T(\mathcal{A}) \neq \emptyset$ iff $\mathcal{A}'$ has some successful run.

We now consider the *finite* game graph $G_{\mathcal{A}'}$ associated to $\mathcal{A}'$. An example of such a game graph is shown in Figure 5.4. By the Run Lemma (Lemma 5.22, input-free case) we have:

$\mathcal{A}'$ has some successful run iff in the parity game over $G_{\mathcal{A}'}$ the player Automaton wins from some initial position (q_0, a).

Whether this holds can be checked effectively.

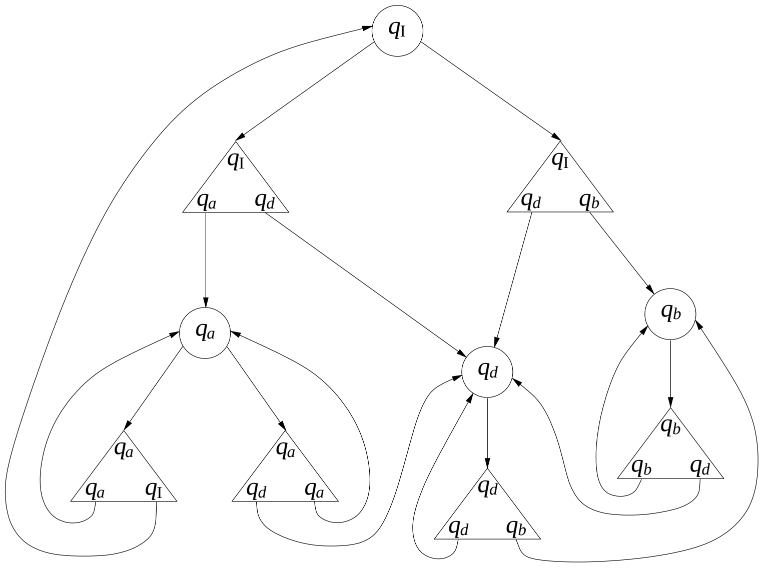


Figure 5.1: The finite game graph $G_{\mathcal{A}'}$ induces a regular tree when a positional strategy of Automaton is fixed. Such a positional strategy is sufficient, since this game is a parity game.

We now show the second statement of the theorem: Assume $T_\omega(\mathcal{A}) \neq \emptyset$, i.e., that $\mathcal{A}'$ admits a successful run. So in the parity game over $G_{\mathcal{A}'}$ the player Automaton wins from some initial position (q_0, a), by means of a positional strategy.

This strategy induces a deterministic tree automaton as "subautomaton" of $\mathcal{A}'$, where for each state (q, a) (as game position for Automaton) only one transition exists (as a move of Automaton) for the continuation of a run. This deterministic tree automaton generates a regular tree. By construction of $\mathcal{A}'$, this regular tree belongs to the tree language recognized by $\mathcal{A}$. □

5.5 S2S and Rabin's Tree Theorem

In this section we will apply the automata theoretic results of the previous section to problems in logic. First, we will define a logic in which we can specify exactly all the tree languages which can be recognized by parity tree automata.

Recall the logical system *S1S* (Section 2.6): It has *variables* $x, y, \ldots$ for natural numbers $X, Y, \ldots$ for sets of natural numbers. *Terms* are built from variables $x, y, \ldots$, and 0 by application of $'$. *Atomic formulas* are $X(\tau)$, $\sigma < \tau$, $\sigma = \tau$ for terms σ, τ from which all S1S-formulas are built using Boolean connectives and quantifiers $\exists, \forall$ over both first- and second-order variables. Recall that $<$ and $=$, as well as first order variables can be eliminated, using the logic $\mathsf{S1S}_0$ with atomic formulas $X \subseteq Y$, $\mathrm{Sing}(X)$, $\mathrm{Succ}(X, Y)$, which is equivalent to S1S.

Definition 5.30. (S2S) The logical system *S2S* (the "second-order system of 2 successors") is defined over *variables* $x, y, \dots$ of words over $\mathbb{B}$ (single nodes in the binary tree) and over second order variables $X, Y, \dots$ of sets of words over $\mathbb{B}$ (sets of nodes of the binary tree). *Terms* are built from variables $x, y, \dots$, and ϵ by concatenating with 0 or 1. Examples: ϵ, $x0$, $x001$, 001. Let σ, τ be terms. *Atomic formulas* are:

$$X(\tau) \text{ (“}\tau\text{ is in }X\text{”)}, \quad \sigma \preceq \tau \text{ (“}\sigma\text{ is a prefix of }\tau\text{”)}, \quad \sigma = \tau \text{ (“}\sigma\text{ is equal to }\tau\text{”)}$$

We use Boolean connectives and quantifiers $\exists, \forall$ over both first- and second-order variables to obtain all formulas in S2S.

Definition 5.31. (Model $\underline{T_2}$) The structure of the infinite binary tree is $\underline{T_2} = (\mathbb{B}^*, \epsilon, S_0, S_1)$, where S_i is the i-th successor function: $S_0(u) = u0$, $S_1(u) = u1$.

The *theory S2S* is the set of S2S-sentences that are true in $\underline{T_2}$. It is also called the *monadic second-order theory (short: the monadic theory) of the infinite binary tree*, denoted by $\mathrm{MTh}_2(\underline{T_2})$.

Example 5.32. We give some examples of the expressiveness of S2S-formulas:
$x \preceq y$ ("node x is prefix of node y"): This can be restated as "all sets X that contain y and are closed under predecessor, contain x":

$$\forall X\big((X(y) \wedge \forall z(X(z0) \to X(z)) \wedge \forall z(X(z1) \to X(z))) \to X(x)\big).$$

Chain(X) ("X is linearly ordered by $\preceq$"):

$$\forall x \forall y((X(x) \wedge X(y)) \to (x \preceq y \vee y \preceq x)).$$

Path(X) ("X is a path, i.e. a maximal chain"):

$$\mathrm{Chain}(X) \wedge \neg\exists Y(X \subseteq Y \wedge X \neq Y \wedge \mathrm{Chain}(Y)).$$

$X \subseteq Y$:

$$\forall z(X(z) \to Y(z)).$$

$X = Y$:

$$\forall z(X(z) \leftrightarrow Y(z)).$$

$x < y$ ("x lexicographically preceeds y"):

$$\exists z(z0 \preceq x \wedge z1 \preceq y) \ \vee (\ x \preceq y \wedge x \neq y).$$

Finite(X) ("each subset Y of X has a minimal and a maximal element w.r.t. $<$"):

$$\forall Y((Y \subseteq X \wedge Y \neq \emptyset) \to (\exists y \text{ “}y\text{ is }<\text{-minimal in }Y\text{” } \wedge \exists y \text{ “}y\text{ is }<\text{-maximal in }Y\text{” })).$$

⊠

Definition 5.33. S2S-formulas $\varphi(X_1, \dots, X_n)$ with free set variables $X_1, \dots X_n$ are interpreted in expanded structures $\underline{t} = (\underline{T_2}, P_1, \dots, P_n)$. We write

$$\underline{t} \models \varphi(X_1, \dots, X_n)$$

if $\underline{t}$ satisfies φ. $\underline{t}$ is identified with the corresponding infinite tree $t \in T^{\omega}_{\mathbb{B}^n}$: for each node $w \in \mathbb{B}^*$ we have

$$t(w) = (c_1, \ldots, c_n) \text{ where } c_i = 1 \text{ iff } w \in P_i.$$

Given a S2S-formula $\varphi(X_1, \ldots, X_n)$ the tree language defined by φ is the set

$$T(\varphi) = \{t \in T^{\omega}_{\mathbb{B}^n} \mid \underline{t} \models \varphi\}.$$

A tree language $T \subseteq T^{\omega}_{\mathbb{B}^n}$ is called *S2S-definable* if $T = T(\varphi)$ for some S2S-formula $\varphi(X_1, \ldots, X_n)$.

Lemma 5.34. *A tree language is S2S-definable if it is recognizable by a parity tree automaton.*

Proof (From Tree Automata to S2S-Formulas) We copy the proof, which shows that Büchi recognizable ω-languages are S1S-definable. Preparations:

- Use the formula Partition$(Y_1, \ldots, Y_m)$ for expressing that $Y_1, \ldots, Y_m$ define a partition of the set of all tree nodes.
- For $a \in \mathbb{B}^n$, say $a = (b_1, \ldots, b_n)$ we write $X_a(t)$ as an abbreviation for

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

  where $(b_i) = \neg$ for $b_i = 0$, and b_i is empty for $b_i = 1$.

How can one describe the existence of a successful run in S2S? Given the parity tree automaton $\mathcal{A} = (Q, \mathbb{B}^n, 1, \Delta, c)$, with $Q = \{1, \ldots, m\}$ and coloring $c : Q \to \{0, \ldots, k\}$, define

$$\begin{aligned}\varphi(X_1, \ldots, X_n) \quad = \quad & \exists Y_1 \ldots Y_m \big(\text{Partition}(Y_1, \ldots, Y_m) \wedge Y_1(\epsilon) \\ & \wedge \forall z \big(\bigvee_{(i,a,j,j') \in \Delta} (Y_i(z) \wedge X_a(z) \wedge Y_j(z0) \wedge Y_{j'}(z1))\big) \\ & \wedge \forall Z(\text{Path}(Z) \to \bigvee_{m \in \text{EC}} (m \text{ is the maximal color occurring} \\ & \text{infinitely often on } Z))\big)\end{aligned}$$

(EC denotes the set of even colors contained in $\{0, \ldots, k\}$.)

The parity acceptance condition "m is maximal color occurring infinitely often on Z" can be formalized as follows:

$$\forall x[Z(x) \to \exists y(x \neq y \wedge x \preceq y \wedge Z(y) \wedge \bigvee_{c(j)=m} Y_j(y)]$$

$$\wedge \exists x[Z(x) \wedge \forall y(x \preceq y \wedge Z(y) \to \bigvee_{m' \leq m} \bigvee_{c(j)=m'} Y_j(y))]$$

We conclude: A tree language recognizable by a parity tree automaton is S2S-definable. □

Now we prove the converse:

Lemma 5.35. *For any S2S-formula $\varphi(X_1, \ldots, X_n)$ one can effectively construct a parity tree automaton $\mathcal{A}$ which is equivalent to φ, that is:*

$$\mathcal{A} \textit{ accepts a tree } t \textit{ iff } \underline{t} \models \varphi.$$

Before proving the above lemma we note an important consequence (for the case $n = 0$).

Theorem 5.36. (Rabin's Tree Theorem) *The theory S2S is decidable.*

Proof Consider an S2S-sentence φ. By Lemma 5.35 it can be transformed into an input-free parity tree automaton $\mathcal{A}$ such that the unlabelled infinite binary tree $\underline{T_2}$ satisfies φ iff $\mathcal{A}$ has some successful run. Whether there exists a successful run can effectively be checked (see Rabin's Basis Theorem (5.29)). □

For the proof of Lemma 5.35, we will use the same idea as in the second half of Section 2.6; we proceed from S2S to $\mathsf{S2S}_0$ by eliminating first order variables, and then to parity tree automata.

Definition 5.37. The logic $\mathsf{S2S}_0$ is built up from the atomic formulas $X \subseteq Y$, $\mathrm{Sing}(X)$ (X is a singleton set), $\mathrm{Succ}_0(X, Y)$ (both sets are singleton sets and the the element in Y is the 0-successor of the element in X), $\mathrm{Succ}_1(X, Y)$ (1-successor). $\mathsf{S2S}_0$ uses the same Boolean connectives as S2S, but only second-order quantifiers.

Remark 5.38. (as for S1S): $\mathsf{S2S}_0$ *is expressively equivalent to S2S.*

Proof Lemma 5.35 Show the claim by induction on $\mathsf{S2S}_0$-formulas: For any $\mathsf{S2S}_0$-formula $\varphi(X_1, \dots, X_n)$ one can effectively construct a parity tree automaton $\mathcal{A}$, which is equivalent to φ.

Induction basis: Atomic formulas $X \subseteq Y$, $\mathrm{Sing}(X)$, $\mathrm{Succ}_0(X, Y)$, $\mathrm{Succ}_1(X, Y)$.

Induction step: Disjunction, negation, existential quantification suffice.

Atomic Formulas: We use tree labels with two components. The first component indicates an element of X, while the second component indicates an element of Y.

For the formula $X \subseteq Y$, consider the tree automaton with the following transitions:

$$\begin{tikzcd}[column sep=small]
 & q_0 \arrow[dl, no head] \arrow[dr, no head] & \binom{0}{1}/\binom{0}{1}/\binom{1}{1} \\
q_0 & & q_0
\end{tikzcd}$$

Each of the three transitions ensures that a member of X is also a member of Y.

A tree automaton equivalent to $\mathrm{Sing}(X)$ has already been developed in the exercises. That automaton ensures that there is only one 1-labeled node in the whole tree.

$\mathrm{Succ}_0(X, Y)$ and $\mathrm{Succ}_1(X, Y)$ are handled in a similar way; as above, look for the singleton node with the X-element. Then ensure that the 0/1-successor of this node contains the only Y-element.

Induction Step: Given parity tree automata $\mathcal{A}_1$ and $\mathcal{A}_2$, which are equivalent (by induction hypothesis) to $\varphi_1(X_1, \dots, X_n)$ and $\varphi_2(X_1, \dots, X_n)$, respectively, one can construct a parity tree automaton equivalent to

1. $\varphi_1(X_1, \dots, X_n) \vee \varphi_2(X_1, \dots, X_n)$
2. $\neg\varphi_1(X_1, \dots, X_n)$
3. $\exists X_1 \varphi_1(X_1, \dots, X_n)$

Item 1 is clear by the Union Lemma, item 2 by the Complementation Theorem, item 3 by the Projection Lemma. □

The concept of equivalence between binary trees and S2S can be expanded to n-branching trees and SnS.

Definition 5.39. The structure

$$\underline{T_n} = (\{0, \dots, n-1\}^*, \epsilon, S_0, \dots, S_{n-1})$$

is called the *infinite n-branching tree.* The logical system SnS is defined accordingly.

Theorem 5.40. *The monadic second-order theory of $\underline{T_n}$ is decidable.*

Proof method: Introduce parity automata over n-branching trees and copy the proofs from case $n = 2$.

Sometimes one may wish to consider quantifiers over finite sets only. For this case there is another modification of the S2S theory.

Definition 5.41. The system *WS2S* (*weak S2S*) has the same syntax as S2S, but set quantifications range over finite sets only.

The set of sentences true in $\underline{T_2}$ under this weak interpretation is called the weak monadic second-order theory of $\underline{T_2}$, denoted $\mathrm{WMTh}_2(\underline{T_2})$.

Theorem 5.42. *The weak monadic second-order theory of $\underline{T_2}$ is decidable.*

Proof Rewrite a WS2S-sentence φ as a S2S-sentence φ' by replacing

$$\begin{array}{lcl} \text{“}\exists X \dots\text{”} & \text{with} & \text{“}\exists X(\mathrm{Finite}(X) \wedge \dots\text{”} \\ \text{“}\forall X \dots\text{”} & \text{with} & \text{“}\forall X(\mathrm{Finite}(X) \rightarrow \dots\text{”}. \end{array}$$

Then φ is in $\mathrm{WMTh}_2(\underline{T_2})$ iff φ' is in $\mathrm{MTh}_2(\underline{T_2})$. □

So far, the results of our treatment have been:

1. S2S and parity tree automata have the same expressive power.
2. The emptiness problem for parity automata is decidable.
3. The monadic second-order theory of the binary tree is decidable.

5.6 Exercises

Exercise 5.1. Let $G = (Q, Q_0, E)$ be a game graph and $\mathcal{F}$ a Muller condition such that

$$\forall P_1, P_2 \subseteq Q \quad P_1, P_2 \notin \mathcal{F} \Rightarrow P_1 \cup P_2 \notin \mathcal{F}.$$

Find a Rabin condition equivalent to $\mathcal{F}$.
Hint: Use for every accepting loop P in $(G, \mathcal{F})$ a Rabin pair. To find this pair consider the union of all non-accepting loops contained in P.
Comment: In the lecture the Union Lemma was shown for the Rabin condition. Here we show a converse statement.

Exercise 5.2. Consider the following tree languages:

(a) $T_1 := \{t \in T^{\omega}_{\{a,b\}} \mid t \text{ contains exactly one node labeled } b\}$

(b) $T_2 := \{t \in T^{\omega}_{\{a,b\}} \mid \text{on every path in } t \text{ there is at most one node labeled } b\}$.

Construct Büchi and Muller tree automata recognizing T_1, and deterministic Büchi and Muller tree automata recognizing T_2.

Exercise 5.3. (a) Prove Lemma 16.3 of the lecture: Given a parity tree automaton recognizing a tree language T over $\Gamma \times \Sigma$ one can construct a parity tree automaton recognizing $proj_\Sigma(T)$.

(b) Given a parity tree automaton $\mathcal{A}$ construct an input-free tree automaton $\mathcal{A}'$ such that

$$T(\mathcal{A}) \neq \emptyset \iff \mathcal{A}' \text{ has some successful run.}$$

Exercise 5.4. Show in detail that a tree is regular iff it contains only finitely many non isomorphic subtrees.
Notation: The subtree of t rooted a node w is the tree $t\restriction_w (u) = t(wu)$.

Exercise 5.5. Let T be the set of trees over $\Sigma = \{a, b\}$ such that for every b-labeled node u there is another b-labeled node $v \neq u$ with prefix u (so v is located in the subtree rooted at u).

(a) Show that T is recognized by a parity tree automaton (perhaps using the facts on parity tree automata shown in the lecture).

(b) Show that T is recognized by a Büchi tree automaton.

Exercise 5.6. Consider the lexicographic order $<$ over the nodes of $\underline{T_2}$, i.e. over words from $\{0,1\}^*$.

(a) Show that the order $<$ contains a dense subset X without first and last element, i.e.

$$\begin{aligned}
&\forall x \in X\ \exists y \in X\ x < y \\
&\forall x \in X\ \exists y \in X\ y < x \\
&\forall x \in X\ \forall y \in X\big(x < y \rightarrow \exists z \in X(x < z < y)\big)
\end{aligned}$$

(b) Present a set of nodes ordered by $<$ in the following way:

$$\begin{tikzcd}
\bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \cdots & \cdots & \bullet \arrow[l] & \bullet \arrow[l] & \bullet \arrow[l] & \bullet \arrow[l]
\end{tikzcd}$$

Exercise 5.7. (a) Construct a parity tree automaton which checks for input trees over $\{0,1\}^2$ whether the S2S$_0$-formula $Succ_0(x_1, x_2)$ is satisfied.

(b) Find a sentence ϕ which is true in $\underline{T_2}$ under the S2S-interpretation, but false under the WS2S-interpretation. Find also a sentence ψ for which the converse is true. (It is not allowed to choose ψ equivalent to $\neg\phi$.)

Chapter 6

Decidability of Monadic Theories

In this section we will consider decidability properties of theories over structures other than the binary tree.

Plan:

1. Preparation: An undecidable monadic theory.
2. The monadic theory of a pushdown transition graph.
3. Prefix-recognizable and automatic graphs (survey).
4. Branching-time logics over Kripke structures.

Definition 6.1. The *infinite grid* is the structure

$$G_2 = (\mathbb{N} \times \mathbb{N}, (0,0), s_1, s_2),$$

where $s_1(i,j) = (i+1,j), \quad s_2(i,j) = (i, j+1)$.

$$\begin{tikzcd}
00 \arrow[r] \arrow[d] & 01 \arrow[r] \arrow[d] & 02 \arrow[r] \arrow[d] & \cdots \\
10 \arrow[r] \arrow[d] & 11 \arrow[r] \arrow[d] & 12 \arrow[r] \arrow[d] & \cdots \\
20 \arrow[r] \arrow[d] & 21 \arrow[r] \arrow[d] & 22 \arrow[r] \arrow[d] & \cdots \\
{} & {} & {} &
\end{tikzcd}$$

Theorem 6.2. *The monadic second-order theory of the infinite grid is undecidable.*

Preparation for the proof: We can express

$$S_2^+(x,y) :\Leftrightarrow s_2^{(i)}(x) = y \text{ for some } i > 0$$

by saying "each set containing $s_2(x)$ and closed under right successors contains y".

Proof of Theorem 6.2 The proof is a reduction of the halting problem for Turing machines to the decidability problem for the infinite grid: For any Turing machine M construct a sentence φ_M of the monadic second-order language of G_2, such that

$$M \text{ halts when started on the empty tape iff } G_2 \models \varphi_M.$$

We first need to describe Turing machine states in terms of the infinite grid. Without loss of generality we assume that M works on a left-bounded tape.

A halting computation of M can be coded by a finite sequence of configuration words $C_0, C_1, \ldots, C_m$.

We can arrange the configurations row by row in a right-infinite rectangular array:

$$\begin{array}{cccccccc}
q_0 & a_0 & a_0 & a_0 & a_0 & a_0 & a_0 & \ldots \\
a_1 & q_1 & a_0 & a_0 & a_0 & a_0 & a_0 & \ldots \\
q_0 & a_1 & a_2 & a_0 & a_0 & a_0 & a_0 & \ldots \\
a_3 & q_2 & a_2 & a_0 & a_0 & a_0 & a_0 & \ldots
\end{array}$$

where $a_0, \ldots, a_n$ are the tape symbols (a_0 is the blank) $q_0, \ldots, q_k$ are the states of M, and q_s is a special halting state. Now we need to describe the properties of an M-run in G_2-formulas.

The sentence φ_M will express over G_2 the existence of an array of configurations that corresponds to a terminating M-run.

We use set variables $X_0, \ldots, X_n$, $Y_0, \ldots, Y_k$. X_i collects the grid positions where a_i occurs, Y_i collects the grid positions where state q_i occurs.

$$\begin{aligned}
\varphi_M \quad : \quad & \exists X_0, \ldots, X_n, Y_0, \ldots, Y_k(\text{Partition}(X_0, \ldots Y_k)\\
& \wedge \text{“the first row is the initial } M\text{-configuration”}\\
& \wedge \text{“a successor row is the successor configuration of the preceding one”}\\
& \wedge \text{“at some position the halting state is reached”})
\end{aligned}$$

In detail, the subformulas are expressed as follows:

- “the first row is the initial M-configuration”:

$$Y_0((0,0)) \wedge \forall x(S_2^+((0,0), x) \rightarrow X_0(x)).$$

- “at some position the halting state is reached”:

$$\exists x Y_s(x).$$

- “a successor row is the successor configuration of the preceding one”: Use a formula $\psi(x)$ that states “the letters on positions x and three next to the right and on the four positions below are compatible with M”. Write a disjunction over all 8-tuples of letters, which are compatible with M. □

6.1 Towards More General Graphs

Are there interesting infinite graphs other than $\underline{T_2}$ which have a decidable monadic second-order theory? We present such a class of graphs: the transition graphs of pushdown automata.

Definition 6.3. A *pushdown system* is an automaton $\mathcal{P} = (P, \Sigma, \Gamma, p_0, \gamma_0, \Delta)$ where

- P is a finite set of states, p_0 the initial state,
- Σ the input alphabet, Γ the stack alphabet, γ_0 the initial stack symbol,
- Δ a finite set of transition rules $p\gamma \rightarrow_a p'u$
  with $\gamma \in \Gamma$, $u \in \Gamma^*$, $a \in \Sigma$ and $p, p' \in P$

A *configuration* is a word $pw \in P\Gamma^+$. These are transformed by transition rules as usual.
We define the *pushdown graph* generated by $\mathcal{P}$ as the structure $G = (V, v_0, (E_a)_{a\in\Sigma})$ where

- v_0 is the configuration $p_0\gamma_0$,
- $(pw, p'w') \in E_a$ iff $w = \gamma w_0, w' = uw_0$ for a transition rule $p\gamma \rightarrow_a p'u \in \Delta$,
- V is the set of configurations reachable from v_0 by applying E_a-transitions for $a \in \Sigma$.

Example 6.4. Consider the context free language $L = \{a^n bc^n \mid n \geq 0\}$. The stack alphabet for the corresponding pushdown automaton is $\Gamma = \{\gamma_0, \gamma\}$. We obtain the pushdown graph:

$$\begin{tikzcd}
q_0\gamma_0 \arrow[r, "a"] \arrow[d, "b"] & q_0\gamma\gamma_0 \arrow[r, "a"] \arrow[d, "b"] & q_0\gamma\gamma\gamma_0 \arrow[r, "a"] \arrow[d, "b"] & \cdots \\
q_1\gamma_0 & q_1\gamma\gamma_0 \arrow[l, "c"] & q_1\gamma\gamma\gamma_0 \arrow[l, "c"] & \cdots \arrow[l, "c"]
\end{tikzcd}$$

⊠

Example 6.5. Consider the language generated by the productions $S \rightarrow () \mid [] \mid (S) \mid [S] \mid SS$. In this case, the pushdown graph has the structure of a tree:

$$\begin{tikzcd}
 & & q((\gamma_0 \arrow[dl, bend left, ")"] & \cdots \\
 & q(\gamma_0 \arrow[ur, bend left, "("] \arrow[dr, bend left, "["] \arrow[ddl, bend left, ")"] & & \\
 & & q[(\gamma_0 \arrow[ul, bend left, "]"] & \cdots \\
q\gamma_0 \arrow[uur, bend left, "("] \arrow[ddr, bend left, "["] & & & \\
 & & q([\gamma_0 \arrow[dl, bend left, ")"] & \cdots \\
 & q[\gamma_0 \arrow[ur, bend left, "("] \arrow[dr, bend left, "["] \arrow[uul, bend left, "]"] & & \\
 & & q[[\gamma_0 \arrow[ul, bend left, "]"] & \cdots
\end{tikzcd}$$

⊠

Definition 6.6. The monadic second-order language over pushdown graphs has variables $x, y, \dots$ for vertices, $X, Y, \dots$ for sets of vertices.

Atomic formulas are: $E_a(x,y)$, $x = y$, $V_p(x)$ ($V_p(x)$ means that the configuration x has state p). Write $E(x,y)$ for $\bigvee_{a\in\Sigma} E_a(x,y)$.

Example 6.7. Example sentence: "there is a cycle of three vertices (configurations), where state p occurs with an outgoing a-labelled edge":

$$\exists x \exists y \exists z \big(x \neq y \wedge y \neq z \wedge z \neq x \wedge V_p(x) \wedge E_a(x,y) \wedge E(y,z) \wedge E(z,x)\big)$$

⊠

Theorem 6.8. (Theorem of Muller and Schupp) *The monadic second-order theory of any pushdown graph G is decidable.*

Proof We rewrite a given sentence φ as an SnS-sentence φ' such that

$$G \models \varphi \text{ iff } \underline{T_n} \models \varphi'$$

(We "interpret G in $\underline{T_n}$" by monadic formulas). Then the right-hand side can be checked.

Important note: We write the tree nodes $w \in \{0, \dots, n-1\}^*$ in reverse order: The first letter of a word is the last from the root.

Translation to SnS-formula:

$$\begin{array}{rcl}
(x = y)' & := & x = y \\
E'_a(x,y) & := & \displaystyle\bigvee_{(p\gamma \to_a p'u)\in\Delta} \exists z (x = p\gamma z \wedge y = p'uz) \\
V'_p(x) & := & V(x) \wedge \exists z x = pz \\
E'(x,y) & := & \displaystyle\bigvee_{a\in\Sigma} (E_a(x,y))' \\
V(x) & := & \forall X((X(q_0\gamma_0) \wedge \forall y \forall z (X(y) \wedge E'(y,z) \to X(z))) \to X(x)) \\
(\neg\varphi)' & := & \neg\varphi' \\
(\varphi \wedge \psi)' & := & \varphi' \wedge \psi' \\
(\exists x \varphi)' & := & \exists x (V(x) \wedge \varphi')(\forall x \varphi)' := \forall x (V(x) \to \varphi') \\
(\exists X \varphi)' & := & \exists X (\text{“}X \subseteq V\text{''} \wedge \varphi')(\forall X \varphi)' := \forall X (\text{“}X \subseteq V\text{''} \to \varphi')
\end{array}$$

□

Example 6.9. φ: "there is a cycle of three vertices (configurations) containing state p with an outgoing a-labelled edge". Then φ': $\exists x \exists y \exists z (V(x) \wedge V(y) \wedge V(z) \wedge x \neq y \wedge y \neq z \wedge z \neq x$ $\wedge V'_p(x) \wedge E'_a(x,y) \wedge E'(y,z) \wedge E'(z,x))$. ⊠

The result of Muller and Schupp's Theorem can be generalized to *prefix rewrite rules*: The pushdown graphs have bounded outdegree and bounded indegree. We present a class of graphs where the degrees of vertices may be infinite.

Liberalize the rewrite rules $p\gamma \to_a p'u$ by

- dropping the distinction between states and stack letters,

- allowing several left-hand sides and several right-hand sides in one rule.

Use a single stack alphabet Γ (and cancel P).

Definition 6.10. A *prefix rewrite rule* for a is of the form

$$U_1 \to_a U_2,$$

where $U_1, U_2 \subseteq \Gamma^*$ are regular sets of words.

A *prefix rewrite system* is of the form $\mathcal{P} = (\Sigma, \Gamma, \Delta)$, where Δ is a finite set of prefix rewrite rules for letters $a \in \Sigma$.

Notice that each prefix rewrite rule $U_1 \to_a U_2$ is finitely representable by two finite automata recognizing U_1, U_2.

We obtain an induced rewrite relation E_a over Γ^*:

$$E_a(w, w') \;:\Leftrightarrow\; w = u_1 w_0 \wedge w' = u_2 w_0 \text{ for some } u_1 \in U_1, u_2 \in U_2.$$

Definition 6.11. A graph $G = (V, (E_a)_{a \in \Sigma})$ is *prefix recognizable* if there is a prefix rewrite system $\mathcal{P} = (\Sigma, \Gamma, \Delta)$ such that

- $V \subseteq \Gamma^*$ is a regular set of words,
- each edge relation E_a is induced by the rules for a of the system Δ.

Example 6.12. $\Gamma = \{0\}$, Rewrite rules: $\epsilon \to_a 0, \quad 0^+ \to_b \epsilon$, $V = 0^*$.

$$\begin{tikzcd}
\epsilon \arrow[r, bend left, "a"] & 0 \arrow[l, bend left, "b"] \arrow[r, bend left, "a"] & 00 \arrow[l, bend left, "b"] \arrow[r, bend left, "a"] \arrow[ll, bend left=50, "b"] & 000 \arrow[l, bend left, "b"] \arrow[r, "a"] \arrow[ll, bend left=50, "b"] \arrow[lll, bend right=40, "b"'] & \cdots
\end{tikzcd}$$

⊠

For prefix recognizable graphs we obtain the following result:

Theorem 6.13. (Caucal's Theorem) *The monadic second-order theory of a prefix-recognizable graph is decidable.*

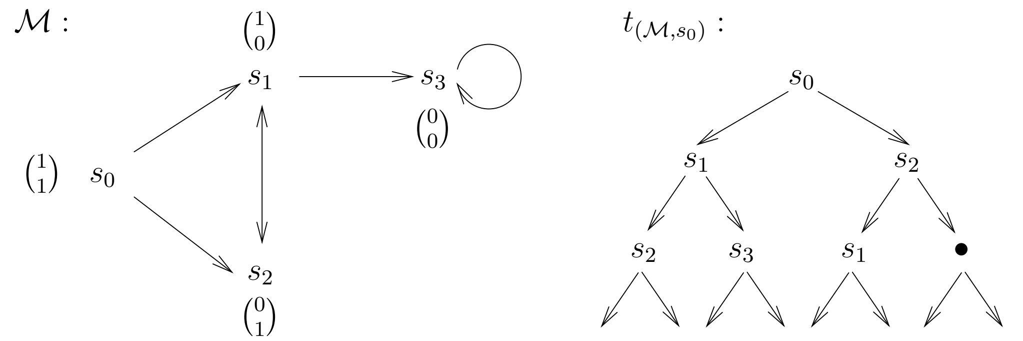
Therefore: Given such a graph G one can decide by an algorithm whether a given monadic second-order sentence φ is true in G or not.

6.2 Unravelling Structures

Idea: Given a transition system with designated initial state, unravel it to obtain its computation tree.

Temporal logics of branching time and certain program logics will serve to express conditions on this computation tree.

Example 6.14. (for a pointed Kripke structure $(\mathcal{M}, s)$):



⊠

We will follow up on this idea and do the following:

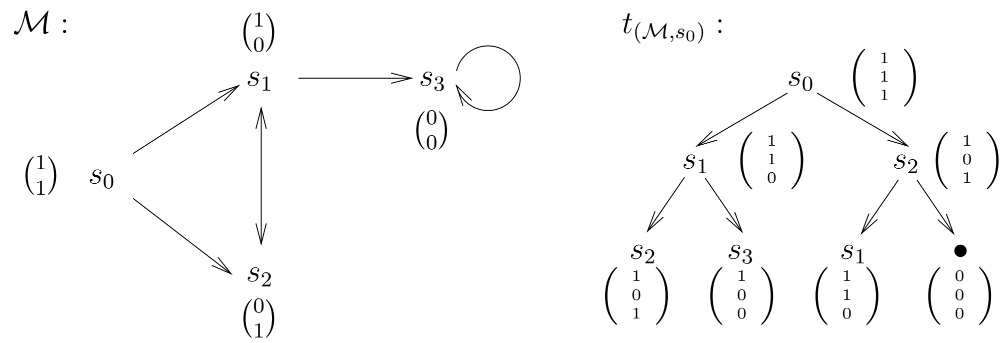
1. Define the computation trees from finite pointed transition graphs.
2. Introduce the Branching-time logic CTL* (Emerson, Sistla 1984), and show the decidability of the CTL*-model-checking problem via Rabin's Tree Theorem.
3. Compare CTL* with a restricted version CTL and analyze the complexity of CTL-model-checking.
4. Introduce the propositional dynamic logic PDL and solve its satisfiability problem.
5. Unravel arbitrary relational structures and obtain the Muchnik-Walukiewicz Theorem.

Recall the notion of a Kripke structure: Given *state properties* $p_1, \dots, p_n$. A *Kripke structure* over $p_1, \dots, p_n$ is of the form $\mathcal{M} = (S, R, \lambda)$ with

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

Write $\lambda(s)$ as a bit vector $(b_1, \dots, b_n)$ with $b_i = 1$ iff $p_i \in \lambda(s)$. A *pointed Kripke structure* $(\mathcal{M}, s)$ has $s \in S$ as initial state.

Example 6.15. (Computation tree)



The first component of the tree-labels indicate whether the respective node is reachable. The other components represent the atomic values of the vertex of the Kripke structure represented by the tree node. ⊠

Preparation for the general definition of computation trees: Let $(\mathcal{M}, s_0)$ be a pointed Kripke structure,

$$\mathcal{M} = (S, R, \lambda), \quad |S| = m, \quad \lambda : S \to 2^{\{p_1,\dots,p_n\}}.$$

We represent a finite path $s_0 \dots s_r$ by a finite path through T_m.

For each $s \in S$ we assume the targets of the out-edges $(s, s') \in R$ to be ordered as $s'_1, \dots s'_k$ with $k \le m$; we speak of the first, second, ..., k-th successor of s.

Each initial path segment $(s_0, s_1, \dots, s_r)$ induces a sequence $\epsilon, (i_1), (i_1 i_2), \dots, (i_1 \dots i_r)$ of nodes of T_m such that $i_j = k$ iff s_j is the k-th successor of s_{j-1}.

The computation tree of $(\mathcal{M}, s_0)$ will be a $\{0,1\}^{n+1}$-valued m-branching tree $t = t(\mathcal{M}, s_0)$.

Definition 6.16. For a pointed Kripke structure $(\mathcal{M}, s_0)$,

$$\mathcal{M}(S, R, \lambda), \quad |S| = m, \quad \lambda : S \to 2^{\{p_1,\dots,p_n\}}$$

the *computation tree* of $(\mathcal{M}, s_0)$ is a $\{0,1\}^{n+1}$-valued m-branching tree t with

- $t(\epsilon) = (1, \lambda(s_0))$,
- $t(i_1 \dots i_r) = (0, \dots, 0)$ if $i_1 \dots i_r$ is *not* induced by an initial path segment starting in s_0,
- $t(i_1 \dots i_r) = (1, b_1, \dots, b_n)$ if $i_1 \dots i_r$ is induced by an initial path segment $s_0, \dots, s_r$ starting in s_0 and ending in s_r with $\lambda(s_r) = (b_1, \dots, b_n)$.

Note that we can limit the branching the in the definition of a computation tree above to the maximal out-degree of a vertex of $\mathcal{M}$. This was done in Example 6.15.

Preparation for the definition of CTL*: CTL*-formulas over $p_1, \dots, p_n$ are introduced in two versions: as *state formulas* and as *path formulas*. Path formulas are interpreted in paths starting in tree nodes u, whereas state formulas are interpreted in subtrees at nodes u.

A state formula is also a path formula (then speaking about the first node u of a path).

Definition 6.17. (Syntax of CTL*)

- Each atomic formula p_i is a state formula.
- If φ and ψ are state formulas, so are
$$\neg\varphi, \quad \varphi \wedge \psi, \quad \varphi \vee \psi, \quad \varphi \to \psi.$$
- If φ is a path formula, then $\mathrm{E}\varphi$ and $\mathrm{A}\varphi$ are state formulas.
- A state formula is also a path formula.
- If φ and ψ are path formulas, then so are
$$\neg\varphi, \quad \varphi \wedge \psi, \quad \varphi \vee \psi, \quad \mathrm{X}\varphi, \quad \mathrm{G}\varphi, \quad \mathrm{F}\varphi, \quad \varphi\mathrm{U}\psi.$$

Example 6.18. E G$(p_1 \rightarrow$ E XA(GF$p_2 \vee$ GFp_3)) says: there is a path such that at any point u of that path, where p_1 holds, a path starts, which, at the second node v, has the following property: all paths starting in v have infinitely many points where p_2 holds or have infinitely many points where p_3 holds. ⊠

Definition 6.19. (Semantics of CTL*) We define the semantics of state formulas in subtrees $t|u$, the semantics of path formulas in paths $t|\pi$ for paths π starting in some node u:

$t\|u \models p_i$	iff component i of $t(u) = 1$
$t\|u \models \neg\varphi$	iff not $t\|u \models \varphi$ (similarly for $\wedge$, $\vee$, $\rightarrow$)
$t\|u \models \mathrm{E}\varphi$	iff exists path π starting in u with $t\|\pi \models \varphi$
$t\|u \models \mathrm{A}\varphi$	iff for all paths π starting in u: $t\|\pi \models \varphi$
$t\|\pi \models p_i$	iff for the first node u of $t\|\pi$: $t\|u \models p_i$
$t\|\pi \models \neg\varphi$	iff not $t\|\pi \models \phi$ (similarly for $\wedge$, $\vee$, $\rightarrow$)
$t\|\pi \models \mathrm{X}\varphi$	iff for the path π^1: $t\|\pi^1 \models \varphi$ (similarly for $\mathrm{G}, \mathrm{F}, \mathrm{U}$)

Here π^i is π from the i-th node onwards (counting from 0).

The Fragment CTL The *Computation Tree Logic CTL* allows path quantifiers E, A only in connection with a single LTL-operator. All CTL-formulas are state formulas.

Definition 6.20. (Syntax and Semantics of CTL)

- $p_1, \dots, p_n$ are CTL-formulas.
- if φ, ψ are CTL-formulas, then so are

$$\neg\varphi, \quad \varphi \wedge \psi, \quad \varphi \vee \psi, \quad \varphi \rightarrow \psi.$$

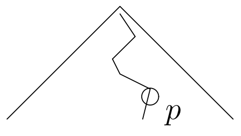
- if φ, ψ are CTL-formulas, then so are

$$\mathrm{EX}\varphi, \quad \mathrm{EF}\varphi, \quad \mathrm{EG}\varphi, \quad \mathrm{E}(\varphi \mathrm{U} \psi), \quad \mathrm{AX}\varphi, \quad \mathrm{AF}\varphi, \quad \mathrm{AG}\varphi, \quad \mathrm{A}(\varphi \mathrm{U} \psi).$$

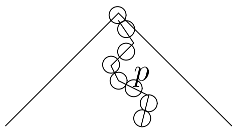
Semantics are derived from CTL*.

Example 6.21. (CTL examples)

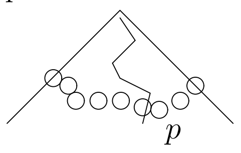
EFp: exists a path, where at some point p is true.



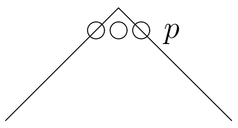
EGp: exists a path, where p is always true.



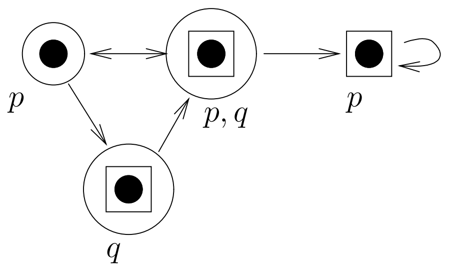
AFp: in all paths, p is true at some point.



AXp: at all successors of the current state, p is true.



ApUq: in all paths p is true until q is true. The boxed states fulfill AXp, whereas the circled states fulfill ApUq:



⊠

When comparing the expressive power of the logics, which we have introduced so far, we see that CTL* is superior to both LTL and CTL, and CTL and LTL are not comparable. For instance the CTL-formula AG EFp is not expressible in LTL. The LTL-formula FGp (i.e. in CTL*: AFGp) is not expressible in CTL.

We will now treat the model-checking-problem for CTL and CTL*.

CTL*-Model-Checking Problem The CTL*-model-checking problem is the following question:

> Given a pointed finite Kripke structure $(\mathcal{M}, s_0)$ and a CTL*-formula φ, does $t(\mathcal{M}, s_0)$ satisfy φ? (Formally: $t(\mathcal{M}, s_0) \models \varphi$?)

The LTL-model-checking problem is a special case:

> Given a pointed finite Kripke structure $(\mathcal{M}, s_0)$ and an LTL-formula ψ, consider the CTL*-formula $\varphi := \mathrm{A}\psi$ and check whether $t(\mathcal{M}, s_0) \models \varphi$.

Theorem 6.22. *The CTL* model-checking problem is decidable*

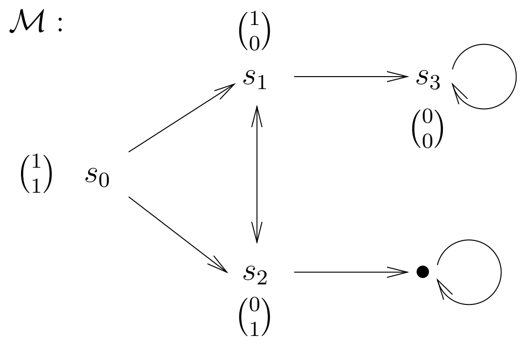
We use a reduction to SmS if $\mathcal{M}$ has m states. A more direct proof will show a much better complexity bound (namely singly exponential).

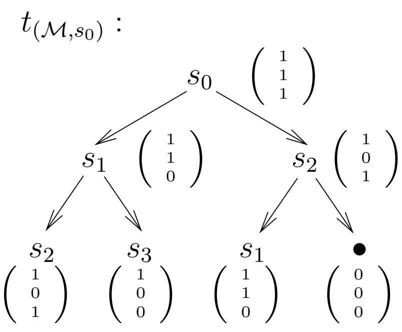
Proof idea: From $(\mathcal{M}, s_0)$ find an SmS-formula $\psi_{(\mathcal{M},s_0)}(X_0, \dots, X_n)$ which describes the $\{0,1\}^{n+1}$-valued tree $t(\mathcal{M}, s_0)$. Translate the CTL*-formula φ to a corresponding SmS-formula $\varphi'(X_0, X_1, \dots X_n)$.

Then check whether

$$\underline{T_m} \models \exists X_0 \dots X_n \; (\psi_{(\mathcal{M},s_0)}(X_0, \dots, X_n) \wedge \varphi'(X_0, X_1, \dots X_n)).$$

Example 6.23. Reconsider the Kripke structure and its computation tree of Example 6.15. We will use predicates Y_0, Y_1, Y_2, Y_3 for the states s_0, s_1, s_2, s_3 and the predicate Z for the newly introduced dummy state $\bullet$.





Describing $t(\mathcal{M}, s_0)$: We use the Kripke structure $(\mathcal{M}, s_0)$ as an automaton that generates the regular tree $t(\mathcal{M}, s_0)$ (in this example over the unlabelled binary tree T_2).

In our example $t(\mathcal{M}, s_0)$ is described by

$$\begin{aligned}\psi_{(\mathcal{M},s_0)}(X_0,X_1,X_2) \quad := \quad & \exists Y_0,\dots,Y_3,Z(\text{Partition}(Y_0,\dots,Y_3,Z) \wedge Y_0(\epsilon)\\ & \wedge \forall x[(Y_0(x) \rightarrow\ X_0(x) \wedge X_1(x) \wedge X_2(x) \wedge Y_1(s_0(x)) \wedge Y_2(s_1(x)))\\ & \wedge (Y_1(x) \rightarrow\ X_0(x) \wedge X_1(x) \wedge \neg X_2(x) \wedge Y_2(s_0(x)) \wedge Y_3(s_1(x)))\\ & \wedge (Y_2(x) \rightarrow\ X_0 \wedge \neg X_1(x) \wedge X_2(x) \wedge Y_1(s_0(x)) \wedge Z(s_1(x)))\\ & \wedge (Y_3(x) \rightarrow\ X_0(x) \wedge \neg X_1(x) \wedge \neg X_2(x) \wedge Y_3(s_0(x)) \wedge Z(s_1(x)))\\ & \wedge (Z(x) \rightarrow\ \neg X_0(x) \wedge \neg X_1(x) \wedge \neg X_2(x) \wedge Z(s_0(x)) \wedge Z(s_1(x)))]\end{aligned}$$

Translating CTL* to SmS: We have to express a CTL*-formula as an SmS-formula. We need an auxiliary SmS-formula Path$'(X, y)$ saying "X is a path starting in node y". Formally this can be written as

$$\begin{aligned}\text{Path}'(X,y) :=& \text{Chain}(X) \wedge X(y) \wedge \forall x(x \preceq y \wedge x \neq y \rightarrow \neg X(y))\\ & \wedge \text{ "X is maximal with these properties".}\end{aligned}$$

Translation Example: $\varphi = \text{E GF}(p_1 \wedge \text{E G}p_2)$ is translated into

$$\begin{aligned}\varphi'(X_0,X_1,X_2) \quad = \quad & \exists X(\text{Path}(X) \wedge X \subseteq X_0 \wedge \forall x(X(x) \rightarrow \exists y(X(y) \wedge x \preceq y \wedge X_1(y)\\ & \wedge \exists Y(\text{Path}'(Y,y) \wedge Y \subseteq X_0 \wedge \forall z(Y(z) \rightarrow X_2(z)))))).\end{aligned}$$

⊠

Example 6.24. Example formula φ of CTL*: E G$(p_1 \rightarrow \text{E XA}(\text{GF}p_2 \vee \text{GF}p_3))$.

Then

$$\begin{aligned}\varphi'(X_0,X_1,X_2) \quad := \quad & \exists X[\text{Path}(X) \wedge X \subseteq X_0 \wedge \forall x(X(x) \wedge X_1(x) \rightarrow\\ & (\exists Y(\text{Path}'(Y,x) \wedge Y \subseteq X_0 \wedge \exists y(Y(y) \wedge (y = s_0(x) \vee y = s_1(x)) \wedge\\ & \forall Z(\text{Path}'(Z,y) \wedge Z \subseteq X_0 \rightarrow\\ & \forall z(Z(z) \rightarrow \exists z'(Z(z') \wedge z \preceq z' \wedge X_2(z')) \vee\\ & \forall z(Z(z) \rightarrow \exists z'(Z(z') \wedge z \preceq z' \wedge X_3(z')))))))))]\end{aligned}$$

is the equivalent formula in S2S. ⊠

Comments on CTL-Model-Checking CTL-Model-Checking for a given Kripke structure $(\mathcal{M}, s_0)$ can be done by a simple labeling procedure on $\mathcal{M}$, avoiding the reference to $t(\mathcal{M}, s_0)$.

Model-checking φ in the structure $\mathcal{M} = (S, R, \lambda)$: For the subformulas ψ of φ in increasing complexity: Label those vertices $s \in S$ with ψ where $(\mathcal{M}, s) \models \psi$ holds. Finally check whether s_0 has label φ.

Example 6.25. (Examples of CTL Labeling Rules)

- If $\psi = \psi_1 \wedge \psi_2$ then label s with ψ if
  s has already labels ψ_1 and ψ_2 (similarly for other Boolean connectives).
- If $\psi = \mathrm{EX}\psi_1$ then label s with ψ if
  some edge $(s, s') \in R$ exists where s' has already label ψ_1.
- If $\psi = \mathrm{E}\psi_1\mathrm{U}\psi_2$ then label all s with ψ
  from which there is a finite path labeled ψ_1 up to a vertex labeled ψ_2.
- If $\psi = \mathrm{AF}\psi_1$ then label all s
  from which all paths meet a vertex labeled with ψ_1.

⊠

Summary

- CTL* and CTL are defined with reference to the computation trees of transition systems.
- Decidability of model-checking CTL* follows from a reduction to SmS.
- CTL-model-checking is possible by a (polynomial-time) labelling procedure over the given Kripke structure.

6.3 Propositional Dynamic Logic

Aim: Solve the satisfiability problem for a basic program logic. We will use the *Propositional Dynamic Logic PDL* (Harel 1979, Fischer-Ladner 1979).

Idea: Merge Boolean logic with (nondeterministic) regular programs, interpreted over Kripke structures (finite or countable). Boolean logic can express properties of states. A *nondeterministic program* defines a relation R between states, consisting of the pairs (s, t) such that some program computation leads from s to t.

Definition 6.26. (Regular Programs) *Regular programs* are constructed from atomic programs $a_1, \dots, a_n$ using

- nondeterministic branching (union),
- composition (concatenation),
- iteration (Kleene star),
- tests.

PDL-statements may involve two kinds of objects:

- programs α
- formulas ψ

Example 6.27. Example of a formula with both components: $\langle\alpha\rangle\psi$. $(\mathcal{M}, s_0) \models \langle\alpha\rangle\psi$ means: "some α-computation from state s_0 reaches a state where ψ holds". ⊠

Definition 6.28. (PDL Syntax) Let $\Phi_0 = \{p_1, \dots, p_m\}$ be a set of atomic formulas, and $\Psi_0 = \{a_1, \dots, a_n\}$ be a set of atomic programs. Then the following formula formation rules apply:

- If φ, ψ are formulas, so are $\neg\varphi,\ \varphi \wedge \psi,\ \varphi \vee \psi,\ \varphi \rightarrow \psi$.
- If α, β are programs, so are $\alpha \cup \beta,\ \alpha;\beta,\ \alpha^*$.
- If φ is a formula and α a program then $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ are formulas.
- If φ is a formula then $\varphi?$ is a program.

Example 6.29. Idea for using tests: The program $\varphi?$ leads from state s to state s if φ is satisfied in s, otherwise terminates. Simulation of "if-then-else" and "while"

- if φ then α else β: $(\varphi?;\alpha) \cup (\neg\varphi?;\beta)$
- while φ do α: $(\varphi?;\alpha)^*;\neg\varphi?$

Notation: Often one skips the ";"

⊠

Definition 6.30. (Semantics of PDL) Given a Kripke structure $(\mathcal{M}, s)$, the meaning of a formula φ is a set of states s (those where the formula is true). Notation: $(\mathcal{M}, s) \models \varphi$.

The meaning of a program α is a set of pairs of states (the pairs (s,t) such that a computation of α leads from s to t). Notation: $s \rightarrow_\alpha t$.

We consider Kripke structures $\mathcal{M} = (S, R_1, \dots R_m, \lambda)$ with $R_i \subseteq S \times S$ and $\lambda : S \rightarrow 2^{\{p_1, \dots p_n\}}$, where $(s,t) \in R_i$ iff some a_i-computation leads from s to t.

$(\mathcal{M}, s) \models p_i$	iff $p_i \in \lambda(s)$
$(\mathcal{M}, s) \models \neg\varphi$	iff not $(\mathcal{M}, s) \models \varphi$
	similarly for $\wedge, \vee, \rightarrow$
$(\mathcal{M}, s) \models \langle\alpha\rangle\varphi$	iff exists t with $s \rightarrow_\alpha t$ and $(\mathcal{M}, t) \models \varphi$
$(\mathcal{M}, s) \models [\alpha]\varphi$	iff for all t with $s \rightarrow_\alpha t$: $(\mathcal{M}, t) \models \varphi$
$s \rightarrow_{a_i} t$	iff $(s,t) \in R_i$
$s \rightarrow_{\alpha\cup\beta} t$	iff $s \rightarrow_\alpha t$ or $s \rightarrow_\beta t$
$s \rightarrow_{\alpha;\beta} t$	iff exists s' with $s \rightarrow_\alpha s'$ and $s' \rightarrow_\beta t$
$s \rightarrow_{\alpha^*} t$	iff exists $s_0, \dots, s_k$ with $s_0 = s$, $s_k = t$
	and $s_i \rightarrow_\alpha s_{i+1}$ for $0 \leq i < k$
$s \rightarrow_{\varphi?} t$	iff $(\mathcal{M}, s) \models \varphi$ and $s = t$

Example 6.31. Define a Kripke structure $\mathcal{M}$: State set S: $\mathbb{N}$ (value domain of program variable x). p is defined true only in state 0, a leads from state $i+1$ to i (for $i > 0$).

Consider $\alpha =$ while $\neg p$ do a, and $\varphi := \langle\alpha\rangle p$ (which can also be written as $(\neg p?; a)^*; p)$. Then for each state i: $i \rightarrow_\alpha 0$

$$(\mathcal{M}, i) \models \varphi.$$

⊠

Example 6.32. Use atomic propositions p, q and atomic programs a, b.

Formulas:
$p \rightarrow [a; b]\neg p$
$p \rightarrow \langle a\rangle\neg p$
$p \rightarrow [(ab^*a)^*]p$

Kripke structure:

$$\begin{tikzcd}
\overset{pq}{1} \arrow[r, leftrightarrow, "a"'] \arrow[d, leftrightarrow, "b"'] & \overset{\neg pq}{2} \arrow[d, leftrightarrow, "b"] \\
\underset{p\neg q}{3} \arrow[r, leftrightarrow, "a"] & \underset{\neg p\neg q}{4}
\end{tikzcd}$$

In this Kripke structure all given formulas are true. Starting on the left side (state 1 or 3) we can follow the computations given by $[a; b]$, $\langle a\rangle$, and $[(ab^*a)^*]$ and verify the respective formulas. ⊠

Definition 6.33. (Satisfiability Problem) A PDL-formula φ is *satisfiable* if some model $(\mathcal{M}, s)$ exists with $(\mathcal{M}, s) \models \varphi$. The *satisfiability problem* for PDL-formulas is then: Given a PDL-formula φ, decide whether it is satisfiable.

Approach for the solution:

1. Note that tree models suffice
2. Describe models by $\{0,1\}^k$-labelled binary trees for appropriate k
3. Translate PDL-formulas to S2S-formulas $\varphi(X_1, \ldots, X_k)$
4. Decide truth of $\exists X_1 \ldots X_k \varphi(X_1, \ldots, X_k)$

Lemma 6.34. (Tree Model Lemma) *For $\mathcal{M} = (S, R_1, \ldots, R_n, \lambda)$ define the tree model $\mathcal{M}^+ = (S^*, R_1^*, \ldots, R_n^*, \lambda^*)$, where*

- *S^* is the set of finite sequences of S-states,*
- *$R_i^* = \{((s_1 \ldots s_r), (s_1 \ldots s_r, s_{r+1})) \mid (s_r, s_{r+1}) \in R_i\}$,*
- *$\lambda^*(s_1 \ldots s_r) := \lambda(s_r)$.*

Then:

$$(\mathcal{M}, s) \models \varphi \quad \textit{iff} \quad (\mathcal{M}^+, s) \models \varphi.$$

Coding Tree Models in T_2: The Kripke structures under consideration may be infinite. Therefore identify states with natural numbers. The tree models have states $(i_1 \ldots i_r)$ for $r \geq 0$. We code the tree model state $(i_1 \ldots i_r)$ by the T_2-node $1^{i_1}0 \ldots 1^{i_r}0$ and define

$$K_j := \{1^{i_1}0 \ldots 1^{i_r}0 \mid p_j \in \lambda(i_r)\},$$

$$L_j := \{1^{i_1}0 \ldots 1^{i_r}0 \mid (i_{r-1}, i_r) \in R_j\}.$$

Definition 6.35. (Tree Model Description) An arbitrary tree model in T_2 (for the signature $\{p_1, \ldots, p_m, a_1, \ldots a_n\}$) is given by

- a set K of nodes of the form $1^{i_1}0 \ldots 1^{i_r}0$ which is closed under taking prefixes,
- for $j = 1, \ldots, m$ subsets K_j of K (for the states satisfying p_j),
- for $j = 1, \ldots, n$ subsets L_j of K (for the states $(i_1 \ldots i_r)$ where $((i_1 \ldots i_{r-1}), (i_1 \ldots i_r))$ is in R_j^*.

Lemma 6.36. *There is a S2S-formula $\mu(X, X_1, \ldots, X_m, Y_1, \ldots, Y_n)$, which describes the possible tree models over T_2.*

Example 6.37. $\varphi = [(a_1; a_1)^*; p_1?]p_2$ is translated to "for each finite path in X which, except the start vertex, consists of an even number of nodes in Y_1 and ends in a node of X_1, the last node also belongs to X_2". ⊠

Lemma 6.38. (Translation Lemma) *For any PDL-formula φ there is a S2S-formula*

$$\varphi'(x, X, X_1, \ldots, X_m, Y_1, \ldots, Y_n)$$

such that for any tree model $\mathcal{M}$ in T_2 and any state u of it we have

$$(\mathcal{M}, u) \models \varphi \text{ iff } (\underline{T_2}, u, K, K_1, \ldots, K_m, L_1, \ldots, L_n) \models \varphi'(x, X, X_1, \ldots, X_m, Y_1, \ldots, Y_n).$$

Here $K, K_1, \ldots, K_m, L_1, \ldots, L_n$ are the sets coding $\mathcal{M}$.

Corollary 6.39. *The PDL-sentence φ is satisfiable iff the S2S-sentence*

$$\exists X, X_1, \ldots, X_m, Y_1, \ldots, Y_n$$
$$\big(\mu(\epsilon, X, X_1, \ldots, X_m, Y_1, \ldots, Y_n) \wedge \varphi'(\epsilon, X, X_1, \ldots, X_m, Y_1, \ldots, Y_n)\big)$$

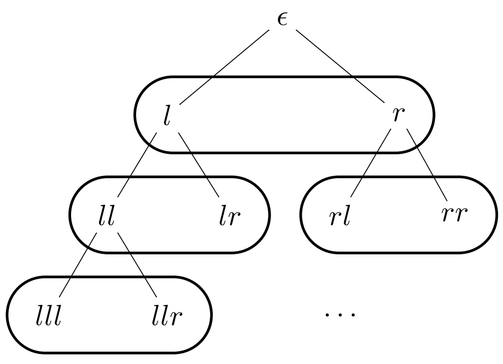
is true in $\underline{T_2}$.

Thus the satisfiability of PDL-formulas is decidable.

6.4 Tree Iteration

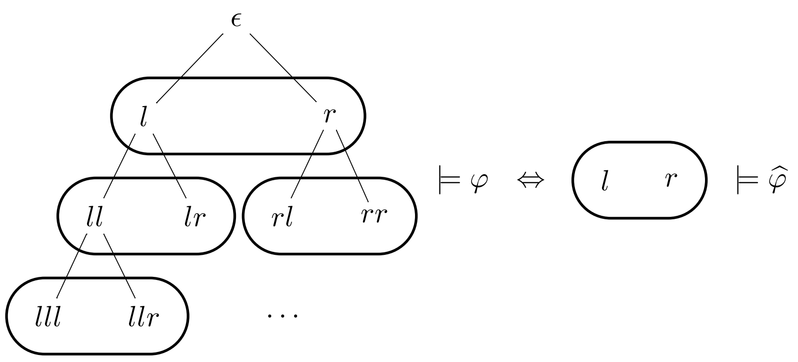
This section is concerned with a more general result on the decideability of monadic formulas. The idea of unravelling Kripke structures is extended to arbitrary relational structures.

Example 6.40. We can describe a binary tree by means of the structure $M = (\{l, r\}, P_l, P_r)$. The predicates P_l and P_r are true for the left / right successor of a node represented by a word over $\{l, r\}$.



The binary tree is obtained by *iterating* (i.e. copying) parent nodes and adding letters l and r. ⊠

Idea for the translation of monadic formulas If there is a computable function that transforms a given formula φ into a formula $\widehat{\varphi}$ such that φ is fulfilled by the iterated structure (in this case the binary tree) iff $\widehat{\varphi}$ is fulfilled by the original structure, then we could obtain Rabin's Theorem very easily. Checking a formula in the binary tree would amount to checking a formula in a two-element-structure.


ϵ
l
r
ll
lr
rl
rr
lll
llr
$\cdots$
$\models \varphi \;\Leftrightarrow$
l
r
$\models \widehat{\varphi}$

Definition 6.41. (Tree Iteration of Arbitrary Structures) Consider any relational structure

$$\mathcal{M} = (M, P_1, \dots P_m,\ R_1, \dots, R_n),$$

where the P_i are subsets of M and the R_i are binary relations over M. The *tree iteration of* $\mathcal{M}$ is the structure

$$\mathcal{M}^{\#} = (M^*, S^M, P_1^{\#}, \dots P_m^{\#}, R_1^{\#}, \dots, R_n^{\#}),$$

where M^* is the set of finite sequences over M and for $x, y \in M^*$

- $S^M(x, y)$ iff $x^\wedge m = y$ for some $m \in M$,
- $P_i^{\#}(x)$ iff there are $z \in M^*, m \in M$ with $x = z^\wedge m$ and $P_i(m)$,
- $R_i^{\#}(x, y)$ iff there are $z \in M^*, m, m' \in M$ such that $x = z^\wedge m$, $y = z^\wedge m'$, $R_i(m, m')$.

Theorem 6.42. (Shelah-Stupp-Theorem (1975)) *Let ϕ be a monadic second-order formula. There is a computable function $\varphi \mapsto \widehat{\varphi}$ such that for every structure $\mathcal{M}$:*

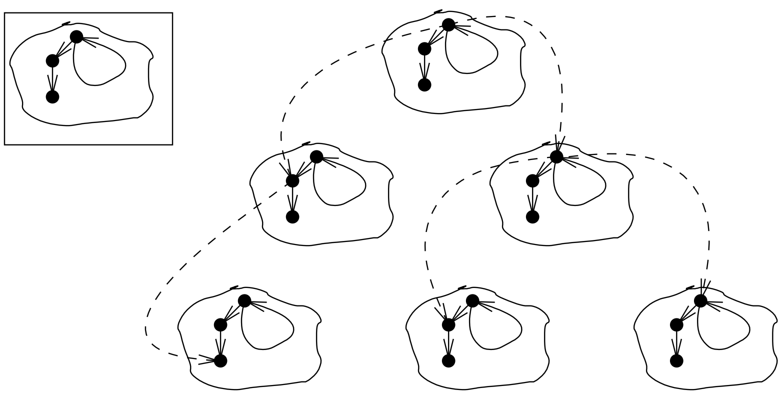
$$\mathcal{M}^{\#} \models \varphi \text{ iff } \mathcal{M} \models \widehat{\varphi}.$$

Corollary 6.43. *If the monadic second-order theory of $\mathcal{M}$ is decidable, so is the monadic second-order theory of $\mathcal{M}^{\#}$*

Corollary 6.44. (Rabin's Tree Theorem) *The monadic second-order theory of the infinite binary tree is decidable.*

The idea of tree iteration can be extended to the unravelling of arbitrary relational structures. We will define this within the framework of tree iteration after an introductory example.

Example 6.45. (Extended Tree Iteration)



⊠

The central idea of the extended tree iteration is to keep the information from which ancestor the current copy of a structure originated. This is neccessary to be able to define the unravelling of a graph as defined before inside the structure one gets when applying the tree iteration to the graph. For this purpose the predicate C (clone) is introduced.

Definition 6.46. Let the *extended tree iteration* of $\mathcal{M}$ be the structure

$$\mathcal{M}^+ = (M^*, S^M, C^M, P_1^\#, \dots P_m^\#, R_1^\#, \dots, R_n^\#),$$

where $C^M = \{x^\wedge m^\wedge m \mid x \in M^+, m \in M\}$ is the *clone predicate.*

Theorem 6.47. (Muchnik-Walukiewicz-Theorem) *Let ϕ be a monadic second-order formula. There is a computable function $\varphi \mapsto \widehat{\varphi}$ such that for every structure $\mathcal{M}$*

$$\mathcal{M}^+ \models \varphi \quad \textit{iff} \quad \mathcal{M} \models \widehat{\varphi}.$$

This theorem entails important corollaries.

Corollary 6.48. *If the monadic second-order theory of $\mathcal{M}$ is decidable, so is the monadic second-order theory of $\mathcal{M}^+$*

Corollary 6.49. (Case of Kripke structures) *For every monadic second-order sentence φ one can construct a monadic second-order sentence $\widehat{\varphi}$ such that for every Kripke structure $\mathcal{M}$*

$$t(\mathcal{M}) \models \varphi \;\; \textit{iff} \;\; \mathcal{M} \models \widehat{\varphi}.$$

6.5 Exercises

Exercise 6.1. Consider the first example of a pushdown graph G of the lecture, generated by the pushdown rules

$$q_0\gamma_0 \rightarrow_a q_0\gamma\gamma_0 \qquad q_0\gamma \rightarrow_a q_0\gamma\gamma$$
$$q_0\gamma_0 \rightarrow_b q_1\gamma_0 \qquad q_0\gamma \rightarrow_b q_1\gamma$$
$$q_1\gamma \rightarrow_c q_1$$

(a) Indicate the nodes of G consisting of words up to length 4 in the tree T_4 where we identify $0, \dots, 3$ with $\gamma_0, \gamma_1, q_0, q_1$ and write the configuration words in reverse order (as in the lecture).

(b) Give an S4S formula $V(x)$ defining the regular set $(q_0 + q_1)\gamma^*\gamma_0$ of nodes of the configuration graph. The successors of a node x may be written as $q_0x, q_1x, \gamma x$ and $\gamma_0 x$, as in $P(x) = \exists z\ x = \gamma\gamma_0 z$ defining the language $P = \gamma\gamma_0\{\gamma_0, \gamma, q_0, q_1\}^*$.

(c) Consider the sentence

$$\phi := \exists X\Big(X(q_0\gamma\gamma_0) \wedge \forall x\big(X(q_0\gamma x) \rightarrow X(q_0\gamma\gamma x)\big)\Big)$$

which states that there is a set which contains $q_0\gamma\gamma_0$ and is closed under the second a-rule. Using $V(x)$ translate the sentence ϕ into an S4S-sentence ϕ' following the translation rules from the lecture.

Chapter 7

Infinite Games on Infinite Graphs

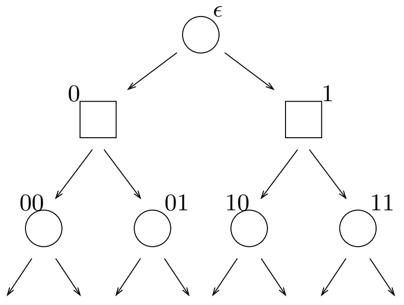
7.1 Gale-Stewart Games

As Chapter 4, this chapter is concerned with infinite games, but in this instance we are going to extend two aspects of infinite games:

- First aim: Generalization of the winning conditions (beyond LTL, S1S, and automata theoretic winning conditions).
- Second aim: Study of games over infinite graphs. We use a standardized infinite game graph: the infinite tree, using the root as the start vertex of plays.
- Side aspect: Studying the history of the theory of infinite games will direct us in developing algorithmic approaches. The roots of this theory can be traced back to

  Cantor (set theory) around 1880,
  Banach, Mazur, Mycielski and others in the 1920's,
  Gale and Stewart 1953, Martin 1975.

We will first introduce games over infinite graphs. Consider the infinite binary tree as the game graph.



Player 0 starts plays at the root node. As every vertex is uniquely determined by a word over $\{0,1\}$, every play is an ω-word over $\{0,1\}$.

Definition 7.1. (Gale-Stewart Games) For any ω-language $L \subseteq \{0,1\}^\omega$ we introduce a *Gale-Stewart game* $\Gamma(L)$. The game graph $G = (Q, Q_0, E)$ is the infinite binary tree.

$$\begin{aligned}
G &= (Q,E),\ \ Q = Q_0 \mathbin{\dot\cup} Q_1\\
Q_0 &= \{w \in \{0,1\}^* \mid |w| \text{ even}\}\\
Q_1 &= \{w \in \{0,1\}^* \mid |w| \text{ odd}\}\\
E &= \{(w,wi) \mid w \in \{0,1\}^*,\ i \in \{0,1\}\}
\end{aligned}$$

The root ϵ is fixed as standard initial vertex of plays. Player 0 wins the play $\alpha \in \{0,1\}^\omega$ if $\alpha \in L$.

We additionally consider two variants of this game:

$\Gamma^*(L)$: The two players move in alternation Player 0 chooses a bit word in each move, player 1 responds by single bit.

$\Gamma^{**}(L)$: Player 0 and player 1 choose finite bit sequences in alternation.

Definition 7.2. (Strategy) A *strategy* for Player 0 in a Gale-Stewart game is a function $f : Q_0 \to \{0,1\}$ (defined on words w of even length). Analogously a function $g : Q_1 \to \{0,1\}$ is a strategy for Player 1.

Example 7.3. Consider the following strategy f of Player 0.

dom(f)	ϵ	00	01	10	11	0000	0001	$\cdots$
f	0	1	1	1	0	1	0	$\cdots$

⊠

Note that all strategies over the game tree can be assumed positional. Given two strategies f and g by Players 0 and 1, a unique play will result, denoted by $\langle f,g\rangle$.

Definition 7.4. (Winning Strategy) For a strategy f for Player 0 and a choice of bits $\beta = b_0, b_1, b_2, \ldots$ by Player 1 fix the play

$$\epsilon,\ f(\epsilon),\ f(\epsilon)b_0,\ f(\epsilon)b_0 f(f(\epsilon)b_0),\ f(\epsilon)b_0 f(f(\epsilon)b_0)b_1, \ldots.$$

We write $\overline{f}(\beta)^\wedge\beta$; here $\overline{f}(\beta)$ is the bit sequence chosen by Player 0 via his strategy f.

For a strategy g for Player 1, $\beta^\wedge\overline{g}(\beta)$ is defined analogously.

- f is a *winning strategy for Player 0* iff $\forall\beta :\ \overline{f}(\beta)^\wedge\beta \in L$.
- g is a *winning strategy for Player 1* iff $\forall\beta :\ \beta^\wedge\overline{g}(\beta) \notin L$.

As before a game $\Gamma(L)$ is called *determined* iff Player 0 or Player 1 has a winning strategy.

Remark 7.5. $\Gamma(L)$ *is determined iff*

$$\neg\exists f\ \forall\beta :\ \overline{f}(\beta)^\wedge\beta \in L \textit{ iff } \exists g\ \forall\beta :\ \beta^\wedge\overline{g}(\beta) \notin L.$$

We showed determinacy for parity games over countable graphs. This does not hold for arbitrary Gale-Stewart games: There is a set L such that $\Gamma(L)$ is not determined.

The proof idea is the following: Find a winning set L such that whatever strategy f Player 0 applies, Player 1 can respond by a strategy g such that Player 0 loses ($\langle f,g\rangle \notin L$), and whatever strategy g Player 1 applies, Player 0 can respond by a strategy f such that Player 1 loses ($\langle f,g\rangle \in L$).

To do this we need a systematic way to go through all possible strategies and in this way build up the desired set L. The functions $f : \{0,1\}^* \to \{0,1\}$ and thus the strategies can be identified with languages of bit words. Moreover the set of all possible strategies can be put in bijection to real numbers.

A Short Course on Ordinals and Well-Orderings

Ordinal numbers are a concept, which allows us to "enumerate" infinite sets of arbitrary size.

Definition 7.6. Definition of the ordinal numbers and their ordering $\prec$:

1. 0 is an ordinal.
2. For any set S of ordinals, there is a smallest ordinal with respect to $\prec$ which is greater than all ordinals in S.

If S has a maximal element, say ξ, the next greater ordinal is called the *successor* of ξ and denoted $\xi + 1$. Otherwise we speak of a *limit ordinal.* The first limit ordinal is ω, which is the first ordinal after $0, 1, 2, \ldots$.

Ordinal numbers can be represented as sets.

- The ordinal 0 is introduced as the set $\emptyset$ $(0 : \ \emptyset)$.
- The successor of ξ is the set $\xi \cup \{\xi\}$ $(1 : \ \emptyset \cup \{\emptyset\} = \{\emptyset\}, \, 2 : \{\emptyset\} \cup \{\{\emptyset\}\} = \{\emptyset, \{\emptyset\}\})$.
- The limit ordinal above S is $\bigcup S$.

Observe that each ordinal is obtained as the set of all smaller ones, and that the ordering $\prec$ between ordinals coincides with the membership relation $\in$ between them.

The first ordinals are:

$$0, 1, 2, \ldots, \omega, \omega + 1, \ldots, \omega + \omega, \ldots, \omega \cdot \omega, \ldots, \omega^{\omega}, \ldots$$

where ω is the first ordinal after the natural numbers.

Ordinals are standard representations of certain orderings: the well-orderings.

Definition 7.7. (Well-Ordering) A linear ordering $(M, <)$ is called *well-ordering* if each nonempty subset of it contains a $<$-smallest element.

Example 7.8.

- $(\mathbb{N}, <)$ is a well-ordering.
- $(\mathbb{Z}, <)$ is not a well-ordering.
- For each ordinal ξ, $(\{\eta \mid \eta \prec \xi\}, \prec)$ is a well-ordering.

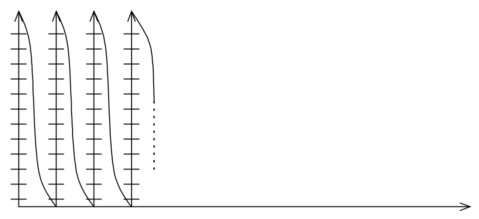
$\boxtimes$

Theorem 7.9. *For each well-ordering $(M, <)$ there is an ordinal ξ such that $(M, <)$ is isomorphic to $(\{\eta \mid \eta \prec \xi\}, \prec)$.*

Example 7.10. The set $\mathbb{N} \times \mathbb{N}$ can be ordered as follows:

$$(0, 0), (0, 1), (0, 2), \ldots, (1, 0), (1, 1), (1, 2), \ldots, (2, 0), (2, 1), (2, 2), \ldots.$$

This order has the "order-type" of ω^2.



There is also an ω-type order of $\mathbb{N} \times \mathbb{N}$:

$$(0,0),(0,1),(1,0),(2,0),(1,1),(0,2),\ldots.$$

$\boxtimes$

Theorem 7.11. (Zermelo's Well-Ordering Theorem) *Assuming the axiom of choice, any set M can be well-ordered, i.e. there is a relation $<$ on M such that $(M,<)$ is well-ordered.*

Consequence: The elements of any set M can be enumerated by the ordinals η with $\eta \prec \xi$ for some ordinal ξ.

Let $\mathfrak{c}$ be the smallest ordinal which can be used to enumerate the set $\mathbb{R}$ of real numbers (and can as well be used for the set $\mathbb{S}$ of all strategies of Player 0). In the list of elements $(r_\eta)_{\eta\prec\mathfrak{c}}$, at any stage $\eta \prec \mathfrak{c}$ only a set of elements has been listed of smaller cardinality than $\mathfrak{c}$.

The main advantage of ordinals is their use in proofs and definitions by transfinite induction. A typical definition of a set M by transfinite induction up to ξ_0 proceeds as follows:

- Fix M_0.
- For $\eta \prec \xi_0$: Given M_η fix $M_{\eta+1}$.
- For limit ordinals $\eta \prec \xi_0$ fix $M_\eta := \bigcup_{\alpha\prec\eta} M_\alpha$.

We use an induction over the ordinal $\xi_0 = \mathfrak{c}$.

In order to enumerate strategies, apply the Well-Ordering Theorem to the strategy sets: $\{f \mid f : Q_0 \to \{0,1\}\}$, $\{g \mid g : Q_1 \to \{0,1\}\}$ So $\{f \mid f : Q_0 \to \{0,1\}\} = \{f_\xi \mid \xi < \mathfrak{c}\}$ and $\{g \mid g : Q_1 \to \{0,1\}\} = \{g_\xi \mid \xi < \mathfrak{c}\}$

Towards a Non-Determined Game Define for $\xi < \mathfrak{c}$ sets L_ξ and M_ξ with the following properties:

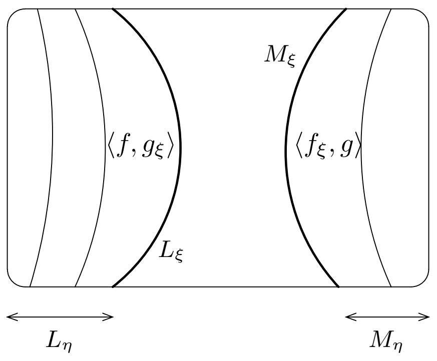
- $L_\xi \cap M_\xi = \emptyset$,
- $|L_\xi|, \ |M_\xi| < \mathfrak{c}$,
- $\forall \eta < \xi \ [\exists f\ (\langle f, g_\eta\rangle \in L_\xi)$ and $\exists g\ (\langle f_\eta, g\rangle \in M_\xi)]$.

Let $M_0 = L_0 = \emptyset$. For limit ordinals ξ set $L_\xi = \bigcup_{\eta<\xi} L_\eta$ and $M_\xi = \bigcup_{\eta<\xi} M_\eta$.

For successor ordinals ξ consider f_ξ. Choose g such that the play $\langle f_\xi, g\rangle$ differs from all plays in the previously defined sets L_η, M_η. This is possible since $|\bigcup_{\eta<\xi}(L_\eta \cup M_\eta)| < \mathfrak{c}$ and $|\{\langle f_\xi, g\rangle \mid g \text{ strategy for } 1\}| = \mathfrak{c}$.

Then add the play $\langle f_\xi, g\rangle$ to the M_η-sets and thus obtain M_ξ.

For g_ξ choose f analogously with $\langle f, g_\xi\rangle \notin \bigcup_{\eta<\xi}(L_\eta \cup M_\eta) \cup M_\xi$, and obtain L_ξ.



Theorem 7.12. (Nondeterminacy of Gale-Stewart Games) *Given sets L_ξ and M_ξ as above, let $L := \bigcup_{\xi<\mathfrak{c}} L_\xi$. Then the game $\Gamma(L)$ is not determined.*

Proof Assume Player 0 has a winning strategy f for $\Gamma(L)$. It has to occur in the listing of strategies as some f_ξ.

By construction, Player 1 can respond with g such that $\langle f_\xi, g\rangle \in M_\xi$ and hence $\langle f_\xi, g\rangle \notin L$. Thus f is not a winning strategy. Contradiction.

Dually, one shows that Player 1 has no winning strategy: For given g, $g = g_\xi$, Player 0 can respond with f such that $\langle f, g_\xi\rangle \in L$. □

A natural approach to guarantee determinacy is to consider sets $L \subseteq \{0,1\}^\omega$ by "increasing complexity". Cantor invented a set-theoretic topology for such a classification of sets $L \subseteq \mathbb{B}^\omega$.

Definition 7.13. The *Cantor topology* is defined on the set $\mathbb{B}^\omega$ $(= \{0,1\}^\omega)$ of all 0-1-sequences via the distance function

$$d : \mathbb{B}^\omega \times \mathbb{B}^\omega \to \mathbb{R}_{\geq 0}$$

with $d(\alpha,\beta) = \begin{cases} 0 & \text{, if } \alpha = \beta \\ \frac{1}{2^n} \text{ for smallest } n \text{ with } \alpha(n) \neq \beta(n) & \text{, if } \alpha \neq \beta \end{cases}$

Remark 7.14. *The distance function d is a metric, i.e.*

1. *$d(\alpha,\beta) \geq 0$ and $d(\alpha,\beta) = d(\beta,\alpha)$*
2. *$d(\alpha,\gamma) \leq d(\alpha,\beta) + d(\beta,\gamma)$ (triangle inequality)*

Proof Item 1. is clear.

Item 2.: Suppose $\alpha,\beta,\gamma \in \mathbb{B}^\omega$ with $\alpha \neq \gamma$. Pick smallest n with $\alpha(n) \neq \gamma(n)$, thus $d(\alpha,\gamma) = \frac{1}{2^n}$. Then either $\alpha(n) \neq \beta(n)$ or $\gamma(n) \neq \beta(n)$, so $d(\alpha,\beta) + d(\beta,\gamma) \geq \frac{1}{2^n}$. □

The topological space $(\mathbb{B}^\omega, d)$ is called *Cantor space.*

Having defined a distance function for paths, we can consider the neighborhoods of paths. Two sequences α, β have a distance $d(\alpha,\beta) \geq \frac{1}{2^n} \Leftrightarrow$ for some $i \leq n : \alpha(i) \neq \beta(i)$. Thus

$$d(\alpha,\beta) < \frac{1}{2^n} \Leftrightarrow \underbrace{\alpha(0)\ldots\alpha(n)}_{\alpha[0,n]} = \underbrace{\beta(0)\ldots\beta(n)}_{\beta[0,n]}.$$

Therefore the $\frac{1}{2^n}$*(= ϵ)-neighborhood* of $\alpha \in \mathbb{B}^\omega$ is the set

$$\{\beta \in \mathbb{B}^\omega \mid \beta[0,n] = \alpha[0,n]\} = \alpha[0,n] \cdot \mathbb{B}^\omega.$$

We can use these neighborhoods to define open sets in this topology.

Definition 7.15. (Cantor Topology) The *Cantor topology* is induced by the metric d. The *open sets* are countable unions of $\frac{1}{2^n}$-neighborhoods of sequences $\alpha \in \mathbb{B}^\omega$.

Remark 7.16. *Assuming $w := \alpha[0,n]$, the $\frac{1}{2^n}$-neighborhood of α is the set $\{w\} \cdot \mathbb{B}^\omega$. Unions of such sets $\bigcup_{i \in I}(\{w_i\} \cdot \mathbb{B}^\omega)$ are representable as*

$$\underbrace{\left(\bigcup_{i \in I}\{w_i\}\right)}_{W \subseteq \mathbb{B}^*} \cdot \mathbb{B}^\omega.$$

Consequence: The open sets of the Cantor topology are the ω-languages of the form $W \cdot \mathbb{B}^\omega$ (for $W \subseteq \mathbb{B}^*$). This reminds us that each ω-language that is recognized by a deterministic E-automaton is open; such a language L has the form $L = W \cdot \mathbb{B}^\omega$ for a *regular* language $W \subseteq \mathbb{B}^*$. But not every open set is E-recognizable, e.g. recall the non-regular set $\{0^i 1 \mid i \text{ prime}\}$.

Thus we can say:

> The open sets L (of the form $L = W \cdot \mathbb{B}^\omega$) represent the "abstract guaranty winning conditions" and define general reachability games.

Remark 7.17. (Suffix-closure) *Given W, let $W' = \{wv \in \mathbb{B}^* \mid w \in W, v \in \mathbb{B}^*\}$ (the "suffix closure of W"). Then α has a prefix in W iff α has a prefix in W'. So $W \cdot \mathbb{B}^\omega = W' \cdot \mathbb{B}^\omega$.*

Definition 7.18. (Closed Sets) An $L \subseteq \mathbb{B}^\omega$ is *closed* iff $\mathbb{B}^\omega \setminus L$ is open.

Remark 7.19. *L is closed iff for some $V \subseteq \mathbb{B}^*$ we have*

$$\alpha \in L \Leftrightarrow \textit{ all prefixes of } \alpha \textit{ are in } V.$$

Proof Given L closed, $\mathbb{B}^\omega \setminus L$ is open so $\mathbb{B}^\omega \setminus L = W \cdot \mathbb{B}^\omega$ for some $W \subseteq \mathbb{B}^*$. Set $V := \mathbb{B}^* \setminus W$. Then

$\alpha \in L$

iff no prefix of α exists in W

iff all prefixes of α are in V.

□

Lemma 7.20. (Path Lemma) *Let L be closed. Assume α has infinitely many prefixes w which can be extended to an ω-word $w\beta$ in L. Then $\alpha \in L$.*

Proof Take V such that $\alpha \in L$ iff all prefixes of α are in V. V may be assumed to be closed under prefixes (restrict to those $v \in V$ which are prefix of some $\alpha \in L$).

Consider α as in the Lemma. It has infinitely many prefixes in V. Since V is closed under prefixes, all prefixes of α are in V. So $\alpha \in L$. □

Summary

- For an open set L a set W of finite words exists with $\alpha \in L$ iff some prefix of α is in W.
- For a closed set L a set W of finite words exists with $\alpha \in L$ iff all prefixes of α are in W.

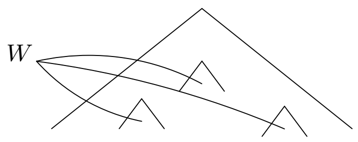
The closed sets capture the "abstract safety conditions".

7.2 Determinacy of Open and Closed Games

First we look at the games $\Gamma(L)$, then at $\Gamma^*(L)$.

Theorem 7.21. (Gale-Stewart) *If $L \subseteq \mathbb{B}^\omega$ is open then $\Gamma(L)$ determined.*

Proof Since L is open, there is a set $W \subseteq \mathbb{B}^*$ with $L = W \cdot \mathbb{B}^\omega$.



As shown in Remark 7.17 on suffix-closed sets, we may assume that W is closed under suffixes.

Apply the definition of the attractor: For $i \geq 0$ define $\mathrm{Attr}_0^i(W) =$ set of nodes from which Player 0 can force a visit in W in $\leq i$ steps. Then Player 0 wins $\Gamma(L)$ iff $\epsilon \in \bigcup_{i\geq 0} \mathrm{Attr}_0^i(W)$. Proof:

$\Leftarrow$: Apply the strategy "decrease distance to W".

$\Rightarrow$: If $\epsilon \notin \bigcup_{i\geq 0} \mathrm{Attr}_0^i(W)$, then Player 1 can ensure to stay outside $\bigcup_{i\geq 0} \mathrm{Attr}_0^i(W)$.

□

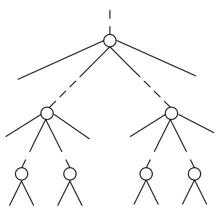
We add a discussion on the games $\Gamma^*(L)$ for closed sets L. This will clarify the tight connection between determinacy and Cantor's Continuum Hypothesis (CH). The CH is the following claim:

> Every set of real numbers is either at most countable ("very small") or in bijection to the set $\mathbb{R}$ itself ("very large").

Cantor and his student Bendixson proved this for closed sets.

Theorem 7.22. (Cantor-Bendixson Theorem) *A closed set L is at most countable or it contains a closed set $L_0 \neq \emptyset$ without isolated paths.*

We skip the proof here but discuss the consequences. There is a copy of the binary tree in L_0:



Definition 7.23. L *contains a binary tree* iff there is a set $B \subseteq \mathbb{B}^*$ of tree nodes such that

- each node of B is on a path from L,
- the B-nodes define a structure isomorphic to the binary tree $(\{0,1\}^*, s_0, s_1)$, with the two successor relations

  $S_0(x,y) \Leftrightarrow x, y \in B \wedge s_0(x) \preceq y$

  $S_1(x,y) \Leftrightarrow x, y \in B \wedge s_1(x) \preceq y$

Remark 7.24. *If L is closed and contains a binary tree B, then each path through infinitely many B-nodes belongs to L.*

Corollary 7.25. *If L is closed and contains a binary tree B, then L has at least as many paths as the binary tree.*

Proof of Remark 7.24 L is closed, so pick $W \subseteq \mathbb{B}^*$ with

$$(*) \quad \alpha \in L \Leftrightarrow \forall i: \ \alpha[0,i] \in W.$$

Consider $\alpha \in \mathbb{B}^\omega$ with infinitely many $\alpha[0,i] \in B$. Then α has infinitely many prefixes w which can be extended to an ω-word $w\beta$ in L. Since L is closed, we can apply the Path Lemma 7.20. Thus $\alpha \in L$. □

Applying the Cantor-Bendixson Theorem for closed sets we obtain:

Corollary 7.26. *If L is closed then L is at most countable or contains a subset which is in bijection to $\mathbb{B}^\omega$.*

So the CH is true for closed sets. Intuitively this means:

Each closed set is either "very small" or "very large".

Use this for the games $\Gamma^*(L)$. Intuition dictates:

- If L is very large, Player 0 has an advantage in winning plays.
- If L is very small, Player 1 has an advantage in winning plays.

We will show this in a precise form.

Theorem 7.27. *If L is closed then $\Gamma^*(L)$ is determined.*

Proof Apply the corollary of the Cantor-Bendixson Theorem: L is at most countable or contains a binary tree. We show:

- In the first case Player 1 has a winning strategy in $\Gamma^*(L)$.
- In the second case Player 0 has a winning strategy in $\Gamma^*(L)$.

Case 2 Assume L contains a binary tree. Then a winning strategy of Player 0 is: "move to the next B-vertex".

This ensures that each play passes infinitely often through B. Since L is closed, the Path Lemma applies, so any such path belongs to L.

Case 1 Claim: If L is at most countable then Player 1 has a winning strategy in $\Gamma^*(L)$. Proof: Let $L = \{\gamma_i \mid i \in \mathbb{N}\}$. Player 1 has to ensure that the play is different from all γ_i.

The structure of a play γ is the following:

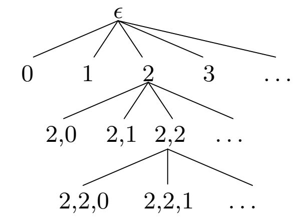
$$\gamma :\ b_0^0 \dots b_{n_0-1}^0\, \gamma(n_0)\, b_0^1 \dots b_{n_1-1}^1\, \gamma(n_0+n_1+1) \dots\ b_0^i \dots b_{n_i-1}^i\, \gamma(n_0+n_1+\cdots n_i+i)\ \dots.$$

Player 1 chooses

- the bit $\gamma(n_0) = 1 - \gamma_0(n_0)$
- the bit $\gamma(n_0+n_1+1) = 1 - \gamma_1(n_0+n_1+1)$
- ...
- the bit $\gamma(n_0+n_1+\cdots n_i+i) = 1 - \gamma_i(n_0+n_1+\cdots n_i+i)$ etc.

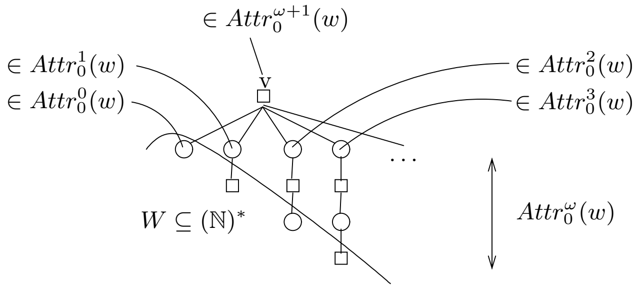
□

Omega-Branching Trees In ω-branching trees, we code the nodes by finite words over the infinite alphabet $\mathbb{N}$.



We can copy the metric from the Cantor space. One obtains the so-called *Baire space* ω^ω.

The ω-branching can cause problems when computing the attractor for a set $W \subseteq \mathbb{N}^*$.



In the Baire space, the previous attractor definition $\bigcup_{i\in\mathbb{N}} \mathrm{Attr}_0^i(W)$ is not sufficient for solving reachability games. We have to take the union over all countable ordinals instead of over all natural numbers ($\bigcup_{\alpha \text{ count. ordinal}} \mathrm{Attr}_0^\alpha(w)$).

In general the countable ordinal numbers arise in the study of systems with unbounded nondeterminism.

7.3 The Borel Hierarchy

The Borel hierarchy over $\mathbb{B}^\omega$ is built up from the open and closed sets by alternating applications of countable intersections and countable unions.

Define for $n \geq 1$ the classes Σ_n, Π_n of ω-languages:

$$\begin{array}{lcl} \Sigma_1 & := & \text{class of open sets } L \subseteq \mathbb{B}^\omega \\ \Pi_1 & := & \text{class of closed sets } L \subseteq \mathbb{B}^\omega \\ \Sigma_{n+1} & := & \text{class of countable unions } L = \bigcup_{i\geq 0} L_i \text{ with } L_i \in \Pi_n \\ \Pi_{n+1} & := & \text{class of countable intersections } L = \bigcap_{i\geq 0} L_i \text{ with } L_i \in \Sigma_n \end{array}$$

We have introduced the first levels with indices by natural numbers (the "finite Borel hierarchy"). This classification extends to transfinite (countable) ordinals. At limit stages one takes the union over all previous stages.

Hausdorff used a different notation for the levels.

G	for Σ_1	("Gebiet")
F	for Π_1	("fermé")
G_δ	for Π_2,	$G_{\delta\sigma}$ for Σ_3
F_σ	for Σ_2,	$F_{\sigma\delta}$ for Π_3, etc.

We also define a Level 0: the sets which are both closed and open, also called *clopen sets* Recall: For any open set $L = W \cdot \mathbb{B}^\omega$ there is a prefix-free set W_0 with $L = W_0 \cdot \mathbb{B}^\omega$.

Theorem 7.28. *A set $L \subseteq \mathbb{B}^\omega$ is open and closed ("clopen") iff $L = W \cdot \mathbb{B}^\omega$ for a finite set W.*

Proof $\Rightarrow$: Assume $L = W \cdot \mathbb{B}^\omega$ and $\mathbb{B}^\omega \setminus L = V \cdot \mathbb{B}^\omega$ for prefix-free V, W. Show that $V \cup W$ is finite.

It suffices to verify that the set T of words from $\mathbb{B}^*$ without a prefix in $V \cup W$ is finite. Clearly T forms a tree and is finitely branching. An infinite path in T would yield an ω-word $\alpha \notin (W \cdot \mathbb{B}^\omega \cup V \cdot \mathbb{B}^\omega)$. By König's Lemma, T is finite.
$\Leftarrow$: Conversely, let $L = W \cdot \mathbb{B}^\omega$ for finite W. Let l be the maximal length of words in W. Set $V = \{v \in \mathbb{B}^* \mid |v| = l,\ v \text{ has no prefix in } W\}$. Then $\mathbb{B}^\omega \setminus L = V \cdot \mathbb{B}^\omega$. □

The clopen sets correspond to "bounded specifications". A specification is bounded if membership of a sequence α in the corresponding L can be decided by an α-prefix of predefined length.

Level 2 Π_2 = class of ω-languages $L = \bigcap_{i\geq 0}(W_i \cdot \mathbb{B}^\omega)$.

Theorem 7.29. $L \in \Pi_2 \Leftrightarrow L = \lim W$ *for some* $W \subseteq \mathbb{B}^*$.

Proof $\Leftarrow$: Suppose $L = \lim W$. Then

$$\begin{aligned} & \alpha \in L \\ \Leftrightarrow\ & \forall i \exists j \geq i : \alpha[0 \ldots j] \in W \\ \Leftrightarrow\ & \forall i : \alpha[0 \ldots i] \text{ extendable to a word in } W \\ \Leftrightarrow\ & \forall i : \alpha \text{ has prefix in } \mathbb{B}^{\geq i+1} \cap W \qquad \big(\mathbb{B}^{\geq i} := \{w \in \mathbb{B}^* \| |w| \geq i\}\big) \end{aligned}$$

$\Leftrightarrow\ \forall i : \alpha \in (\mathbb{B}^{\geq i+1} \cap W) \cdot \mathbb{B}^\omega$

$\Leftrightarrow\ \alpha \in \bigcap_{i\geq 0} \underbrace{(\mathbb{B}^{\geq i+1} \cap W) \cdot \mathbb{B}^\omega}_{\text{open for each } i}.$

So $L \in \Pi_2$.
$\Rightarrow$: Let $L \in \Pi_2$, so

$$L = \bigcap_{i\geq 0} W_i \cdot \mathbb{B}^\omega$$

where $W_i \subseteq \mathbb{B}^*$ (w.l.o.g. closed under extensions). It is easy to show that

$$W \cdot \mathbb{B}^\omega = \min(W) \cdot \mathbb{B}^\omega$$

where $\min(W) = \{w \in W|$ each proper prefix of w is not in $W\}$.

Let $\overline{W_i} := W_0 \cap \dots \cap W_i$. Then $L = \bigcap_{i\geq 0} \overline{W_i} \cdot \mathbb{B}^\omega$.

So $\overline{W_0} \supseteq \overline{W_1} \supseteq \overline{W_2} \supseteq \dots$. Proceed to $\min(\overline{W_0})$, $\min(\overline{W_1}), \dots$ and make them disjoint by proceeding to longer and longer words.

We obtain sets $V_0, V_1, \dots$ with $L = \bigcap_{i\geq 0} V_i \cdot \mathbb{B}^\omega$. Then $\alpha \in L \Leftrightarrow \forall i \exists j \geq i : \alpha$ has prefix in V_j. Thus $L = \lim(\bigcup_{i\geq 0} V_i)$. □

Theorem 7.30. *Each regular ω-language is in the Boolean closure of* Π_2.

Proof Each regular ω-language is Muller recognizable. The Muller recognizable ω-languages over $\mathbb{B}$ are the Boolean combinations of sets $\lim(W)$ for regular languages $W \subseteq \mathbb{B}^*$. As shown, the sets $\lim(W)$ are the Π_2-sets. □

Lemma 7.31.

1. *$L \in \Sigma_n$ iff $\overline{L} \in \Pi_n$*
2. *The classes Σ_n and Π_n are closed under finite union and intersection.*

Proof by induction

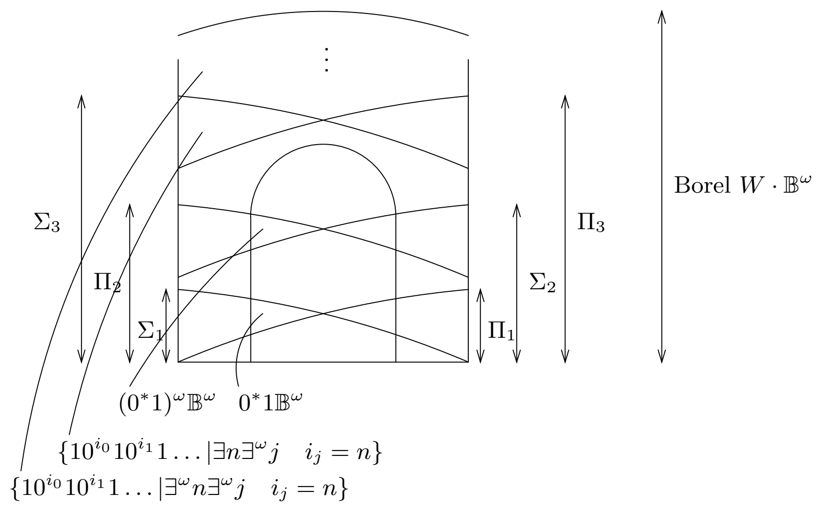
1. Induction step: Let $L \in \Sigma_{n+1}$, i.e. $L = \bigcup_{i\geq 0} L_i$ with $L_i \in \Pi_n$.

   Then $\mathbb{B}^\omega \setminus L = \bigcap_{i\geq 0}(\mathbb{B}^\omega \setminus L_i)$, where $\mathbb{B}^\omega \setminus L_i \in \Sigma_n$ by induction hypothesis. So $\mathbb{B}^\omega \setminus L \in \Pi_{n+1}$.

2. Induction step: Consider two Σ_{n+1}-sets L, M with $L = \bigcup_{i\geq 0} L_i$ and $M = \bigcup_{i\geq 0} M_i$, where the L_i, M_i are Π_n-sets. Define $R_{ij} = L_i \cap M_j$.

   By induction hypothesis: all R_{ij} are Π_n-sets. Thus $\alpha \in L \cap M$ iff $\exists i : \alpha \in L_i \wedge \exists j : \alpha \in M_j$ iff $\exists (i,j) : \alpha \in R_{ij}$. Therefore $L \cap M$ is a countable union of Π_n-sets: namely $L \cap M = \bigcup_{i,j\geq 0} R_{ij}$. □

The following inclusion diagram holds:



Σ_3
Π_2
Σ_1
Π_1
Σ_2
Π_3
Borel $W \cdot \mathbb{B}^\omega$
$(0^*1)^\omega\mathbb{B}^\omega$
$0^*1\mathbb{B}^\omega$
$\{10^{i_0}10^{i_1}1\ldots \mid \exists n \exists^\omega j \quad i_j = n\}$
$\{10^{i_0}10^{i_1}1\ldots \mid \exists^\omega n \exists^\omega j \quad i_j = n\}$

The inner circle denotes the class of ω-regular languages.

We will prove some of the proper inclusions of the diagram.

Lemma 7.32. *For each $n \geq 1$:*

- *there is a Π_n-set which is not Σ_n.*
- *there is a Σ_n-set which is not in Π_n.*

For $n = 1, 2$ this extends the known results.

- There is an A-recognizable ω-language which is not E-recognizable.
- There is a co-Büchi recognizable ω-language which is not Büchi recognizable.

For the cases $n = 1, 2$ we take the old examples and copy the proofs.

For the non-inclusion proofs we have:

Claim: 0^ω is not open.
Proof Assume 0^ω is open, e.g. $0^\omega = W \cdot \mathbb{B}^\omega$. Then for some i we have $0^i \in W$, and hence $0^i1^\omega \in 0^\omega$, contradiction. □

Claim: $(0+1)^* \cdot 0^\omega$ is not Π_2.
Proof Assume $L = (0+1)^* \cdot 0^\omega$ is in Π_2, i.e. $L = \lim(W)$ for some $W \subseteq \mathbb{B}^*$. Then $0^\omega \in L$ has a prefix in W, say 0^{n_0}.

Consider $0^{n_0}10^\omega$ in $L = \lim(W)$, with prefix $0^{n_0}10^{n_1}$ in W. In this way obtain a sequence 0^{n_0}, $0^{n_0}10^{n_1}$, $0^{n_0}1\ldots10^{n_i}$ in W.

So the ω-word $0^{n_0}1\ldots10^{n_i}1\ldots$ is in $\lim(W)(= L)$, contradiction. □

Theorem 7.33. (Hierarchy Theorem) *For $n \geq 1$ there exists an ω-language U_n with the following properties:*

1. *$U_n \in \Sigma_n$,*

2. $U_n \notin \Pi_n$ *(and hence $U_n \notin \Sigma_{n-1}$).*

Moreover there is a reducibility relation $\leq$ between ω-languages such that

$$L \leq U_n \quad \Leftrightarrow \quad L \in \Sigma_n.$$

In short form: U_n is universal (or: complete) *for Σ_n with respect to $\leq$.*

First we will introduce sets which can serve as U_n, then we will discuss the reducibility relation $\leq$.

Remark 7.34. *(Universal Sets)*

- $0^*1 \cdot \mathbb{B}^\omega$ *is universal for* Σ_1,
- $\{0^\omega\}$ *is universal for* Π_1,
- $0^* \cdot \{1^\omega\}$ *is universal for* Σ_2,
- $(0^*1)^\omega$ *is universal for* Π_2,

and, in general, for $n > 1$,

- $\{\alpha \in \{0,1\}^\omega \mid \forall^\omega k_1 \ldots \forall^\omega k_{n-1} :$ *α has only finitely many segments* $001^{k_1}01^{k_2}\ldots01^{k_{n-1}}00\}$ *is universal for* Σ_{2n},
- $\{\alpha \in \{0,1\}^\omega \mid \exists^\omega k_1 \ldots \exists^\omega k_{n-1} :$ *α has infinitely many segments* $001^{k_1}01^{k_2}\ldots01^{k_{n-1}}00\}$ *is universal for* Π_{2n}.

Definition 7.35. A function $f : \mathbb{B}^\omega \to \mathbb{B}^\omega$ is *continuous* iff $f^{-1}(L)$ is open for every open set L.

Write $L \leq M$ if there exists a continuous function $f : \mathbb{B}^\omega \to \mathbb{B}^\omega$ with $f^{-1}(M) = L$. In other words:

$$\alpha \in L \Leftrightarrow f(\alpha) \in M \text{ for } \alpha \in \mathbb{B}^\omega.$$

Remark 7.36. *A function f is continuous iff any finite prefix of an f-image $f(\alpha)$ is determined by a finite prefix of α.*

Remark 7.37. *If L, $M \subseteq \mathbb{B}^\omega$ are ω-languages with $L \leq M$ then also $\mathbb{B}^\omega \setminus L \leq \mathbb{B}^\omega \setminus M$.*

These reductions preserve the level of the languages.

Remark 7.38. *If $L \leq M$ and $M \in \Sigma_n$ (resp. $M \in \Pi_n$) then also $L \in \Sigma_n$ (resp. $L \in \Pi_n$).*

Proof Clear for $M \in \Sigma_1$ (and hence for Π_1). Induction step: Let $M \in \Sigma_{n+1}$, i.e. $M = \bigcup_{i\geq 0} M_i$ with $M_i \in \Pi_n$. Set $L_i = f^{-1}(M_i)$. Then $L_i \leq M_i$ and $L_i \in \Pi_n$ (ind. hyp.).

So $\alpha \in L_i$ iff $f(\alpha) \in M_i$.

$$\alpha \in L \Leftrightarrow f(\alpha) \in M \Leftrightarrow \exists i : f(\alpha) \in M_i \Leftrightarrow \exists i : \alpha \in L_i \Leftrightarrow \alpha \in \bigcup_{i\geq 0} L_i.$$

Therefore $L \in \Sigma_{n+1}$. □

For checking $K \leq L$ we introduce the *Wadge Game*.

Definition 7.39. Given $K \subseteq \mathbb{B}^\omega$, $L \subseteq \mathbb{B}^\omega$, the *Wadge Game* $W(K, L)$ is the infinite game between Players 0 and 1.

Player 0 and 1 move in alternation, building up ω-words α, β. Player 0 chooses bits from $\mathbb{B}$, Player 1 chooses finite words from $\mathbb{B}^*$.

At the end of an infinite play (in ω moves),

- Player 0 has produced an ω-sequence $\alpha \in \Sigma_A^\omega$ of letters,
- Player 1 has produced an ω-sequence of finite words which concatenated give rise to a finite or ω-word β.

Player 1 wins the play (α, β) iff β is infinite and $\alpha \in K \Leftrightarrow \beta \in L$.

Lemma 7.40. (Wadge) *$K \leq L$ iff Player 1 has a winning strategy in $W(K, L)$.*

Example 7.41. $K = 0^*1(0+1)^\omega$, $L = (0^*1)^\omega$. To show $K \leq L$ we have to formulate a strategy which ensures the following for a play (α, β):

1 occurs in α iff infinitely 1 occur in β

Winning strategy for Player 1:

- As long as Player 0 chooses 0 reply with 0.
- If Player 0 picks 1, then from that point onwards pick 1.

⊠

Example 7.42. Let $L = (0^*1)^\omega$. We verify that L is complete for Π_2.

Consider $K \in \Pi_2$, say $K = \bigcap_{n \geq 0} W_n \cdot \mathbb{B}^\omega$ with $W_n \subseteq \mathbb{B}^*$. Show $K \leq L$.

Winning strategy for Player 1 in $W(K, L)$: Build up infinite β, with infinitely many 1 iff α belongs to each set $W_i \cdot \mathbb{B}^\omega$. Idea: Produce the i-th letter 1 in β to signal that in the construction of α a prefix in W_i has been seen. Formally: Set $i := 0$ and repeat the following: If Player 0's current position u does not have any prefix in W_i,

- then play letter 0 and i is not increased,
- else play letter 0 and increase $i := i + 1$.

⊠

Theorem 7.43. (Martin) *For each Borel set L, the Gale-Stewart Game $\Gamma(L)$ is determined.*

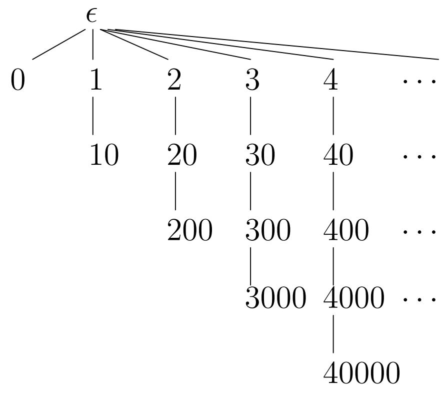
Two extensions are conceivable:

1. Prove determinacy for non-Borel sets.
2. Sharpen the determinacy result by deciding algorithmically who wins and by establishing computable winning strategies.

The set-theorists have turned to question *1.*, the computer scientists to question *2.* The treatment of regular L is just a first step. For example: Generalize strategies to be computable by stronger machines than finite automata.

A Non-Borel Set Consider the ω-branching tree T_ω whose nodes are addressed by finite sequences $(i_1, \dots, i_r)$ of natural numbers. A subtree of T_ω is given by a set T of sequences $(i_1, \dots, i_r)$ which is closed under prefixing. A *finite-path tree* is a subtree of T_ω which has only finite paths.

Example 7.44.



$\boxtimes$

We code subtrees of T_ω by ω-words. Use the alphabet $\{0, 1, \#\}$, and write an element of ω^* as $u_1\#u_2\#\dots\#u_k\#\#$ where each u_i is a binary number.

To code a tree T concatenate the codes of its nodes by increasing length of these words over $\{0, 1, \#\}$, and for fixed length arrange the codes in lexicographical order, thus obtaining an ω-word α_T representing T.

Theorem 7.45. *The ω-language*

$$\mathrm{FPT} := \{\alpha_T \mid T \textit{ is a finite-path tree}\}$$

is not a Borel set.

FPT is in Π^1_1, the first level of the "*projective hierarchy*".

7.4 Exercises

Exercise 7.1. Give an open set $L = W \cdot \mathbb{B}^\omega$, such that $Attr_0^i(W) \subsetneq Attr_0^{i+1}(W)$ holds for all $i \in \mathbb{N}$. (This means that for every i a node of the tree exists with distance i to W.)

Exercise 7.2. We consider the ω-languages $L_1 = \{\alpha \in \mathbb{B}^\omega \mid \alpha \text{ contains infinitely many 1s}\}$ and $L_2 = \{\alpha \in \mathbb{B}^\omega \mid \alpha \text{ contains finitely many 1s}\}$. Represent L_1 as a countable intersection of open (or even E-recognizable) ω-languages and L_2 as a countable union of closed (or even A-recognizable) ω-languages.

Index