Automata and Logic

Script for the lecture “Automata and Logic” by
University Professor Dr.–Ing. Franz Baader
at TU Dresden in term 02/03

March 3, 2005

Contents

Motivation and Context

In automata theory and formal languages one is interested in classes of languages und their properties. A *formal language* is a set $L \subseteq \Sigma^*$, i.e. a set of words over a given alphabet Σ. Σ is usually finite. A *class of formal languages* $\mathcal{K}$ assigns with each finite alphabet Σ a set $\mathcal{K}_\Sigma \subseteq 2^{\Sigma^*}$, i.e. a set of languages over Σ.

Given a class $\mathcal{K}$, one is interested in the following questions:

Characterization How can we characterize the languages belonging to the class?

We are looking for properties P such that:

$$L \in \mathcal{K}_\Sigma \quad \text{iff} \quad L \subseteq \Sigma^* \text{ and } L \text{ satisfies } P.$$

Usually one wants to have different equivalent characterizations (automata, grammars, ...). Some characterizations are better for certain purposes then other characterizations.

Closure properties Under which operations on languages (intersection, union, complement, homomorphic images, ...) is the class closed?

Decidability Which problems are decidable for this class?

e.g.: $w \in L$? $L \neq \emptyset$? $L_1 \subseteq L_2$?

One assumes that the (usually infinite) languages are given by a finite representation corresponding to one of the characterizations.

The most important classes are collected in the Chomsky hierarchy:

Class	Characterization
Type 0	• generated by general Chomsky grammars (transitions $u \to v$ where u contains a non–terminal) • accepted by Turing machines
Type 1 context sensitive	• generated by context sensitive grammars (transitions $u \to v$ where $1 \leq \|u\| \leq \|v\|$) • accepted by Turing machines with a linearly bounded tape
Type 2 context free	• generated by context free grammars (transitions $X \to v$ where X is non–terminal) • accepted by push–down automata
Type 3 regular	• generated by right linear grammars (transitions $X \to uY$, $X \to u$ where X, Y are non–terminal and u is a terminal word) • accepted by a finite automata

Examples of further characterization

Do deterministic machines (automata) yield the same class as non deterministic ones?

Type 0 Yes
Type 1 Open
Type 2 No
Type 3 Yes

Examples of closure properties

Is the class closed under complementation, i.e. $L \in \mathcal{K}_\Sigma \rightsquigarrow \Sigma^* \setminus L \in \mathcal{K}_\Sigma$?

Type 0	No (complements of recursively enumerable languages need not to be recursively enumerable)
Type 1	Yes (relatively new result, 1987).
Type 2	No (related to determinism)
Type 3	Yes

Examples of decision problems

Are the word problem and the equivalence problem decidable?

Class	$w \in L$?	$L_1 = L_2$?
Type 0	undecidable	undecidable
Type 1	decidable	undecidable
Type 2	decidable	undecidable
Type 3	decidable	decidable

The lecture will mostly concentrate on Type 3 languages. We will look at subclasses, different characterizations and generalizations to infinite words and trees. The script is organized in three parts:

1. Regular languages of finite words

As alternative characterizations we will consider:

- Algebraic characterizations
  - Every language can be associated with a monoid (syntactic monoid).
  - Regular languages are those whose syntactic monoid is finite.
- Logical characterizations
  - Logical formulae can define languages.
  - There is a logic (monadic second–order logic) that defines the regular languages.

These characterizations can be used to define subclasses of the class of regular languages. The most prominent one is the class of *star–free languages*:

- They are defined by formulae of first–order predicate logic.
- Their syntactic monoid is aperiodic.

2. Languages of infinite words

Instead of finite words (finite sequences of letters) one can consider infinite words (infinite sequences of letters) as input for finite automata. The only thing that must be changed is the acceptance condition.

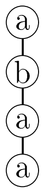
- Finite words: after reading the word a final state is reached
- Infinite words: conditions on the states that are reached infinitely often

We will show closure properties, decidability of the emptiness problem and we will look at the connection to logic. One can obtain interesting decidability results for logics.

3. Tree languages (or forrests)

Words can be viewed as labeled trees with branching factor ≤ 1.

$abaa \quad \widehat{=}$



The notion of a finite automaton can be extended to the one of a tree automaton by allowing for branching > 1. Many results for regular languages generalize to tree languages. There is again an interesting connection to logics.

The main emphasis will be on applications in logic (decidability results). Methods are as important as the results! There will be an emphasis on proofs!

Chapter 1

Regular languages, finite monoids and logical formulae

Goal of this chapter is to recapitulate some definitions and results for regular languages and establish a relationship to monoids and formulae.

1.1 Regular languages and finite automata

Definition 1.1 Let Σ be a finite alphabet. The class $\mathcal{R}eg_\Sigma$ of *regular languages* over Σ is the smallest class such that

- $\emptyset$, $\{\epsilon\}$ and $\{a\}$ for $a \in \Sigma$ are in Reg_Σ (where ϵ is the empty word),
- if $L, L_1, L_2 \in Reg_\Sigma$, then so are $L_1 \cup L_2$, $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \text{ and } v \in L_2\}$, $L^* = \{u_1 \cdots u_n \mid n \geq 0 \text{ and } u_i \in L\}$.

As usual, we will write *regular expressions* to describe regular languages. E.g.: $(ab)^*a$ describes $(\{a\} \cdot \{b\})^* \cdot \{a\}$, i.e. words over $\{a, b\}$ starting and ending with a, and where in between $a's$ and $b's$ alternate.

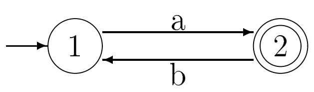
Because of the definition, regular languages are closed under union, concatenation, and Kleene star. To show closure under intersection and complement, an alternative characterization by finite automata is more appropriate.

Definition 1.2 A non–deterministic *finite automaton* $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ consists of

- a finite set of states Q,
- a finite alphabet Σ,
- a set of initial states $I \subseteq Q$,
- a transition relation $\Delta \subseteq Q \times \Sigma \times Q$,
- a set of final states $F \subseteq Q$.

As usual, we will draw graphs to represent automata.

e.g.:



$Q = \{1, 2\}$, $\Sigma = \{a, b\}$, $\quad I = \{1\}$ (shown by $\rightarrow$ 1)

$\Delta = \{(1, a, 2); (2, b, 1)\}$, $F = \{2\}$ (shown by 2)

A *path* in the automaton is a sequence $q_0 a_1 q_1 a_2 \ldots a_n q_n$ where $(q_{i-1}, a_i, q_i) \in \Delta$ for $1 \leq i \leq n$. We will often abbreviate such a path as $q_0 \xrightarrow{a_1 \ldots a_n}_{\mathcal{A}} q_n$. The path is *successful* if $q_0 \in I$ and $q_n \in F$.

The automaton $\mathcal{A}$ *accepts* the following language:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid q_o \xrightarrow{w}_{\mathcal{A}} q_n \text{ is a successful path in } \mathcal{A}\}.$$

A language $L \subseteq \Sigma^*$ is called *recognizable*, if there exists a finite automaton $\mathcal{A}$ that accepts L.

Kleene's theorem says that a language $L \subseteq \Sigma^*$ is recognizable iff it is regular. We will use this to show that the class of regular languages is closed under intersection.

Proposition 1.3 If $L_1, L_2 \in Reg_\Sigma$, then so is $L_1 \cap L_2$.

Proof: Let $\mathcal{A}_1 = (Q_1, \Sigma, I_1, \Delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, I_2, \Delta_2, F_2)$ be automata such that $L_1 = L(\mathcal{A}_1)$ and $L_2 = L(\mathcal{A}_2)$. We define $\mathcal{A} := (Q_1 \times Q_2, \Sigma, I_1 \times I_2, \Delta, F_1 \times F_2)$ where

$$\Delta := \{((q_1, q_2), a, (q_1', q_2')) \mid (q_1, a, q_1') \in \Delta_1 \text{ and } (q_2, a, q_2') \in \Delta_2\}.$$

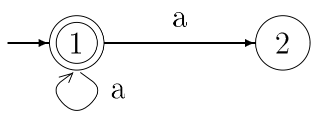
It is easy to see that

$$(q_1, q_2) \xrightarrow{w}_{\mathcal{A}} (q_1', q_2') \quad \text{iff} \quad q_1 \xrightarrow{w}_{\mathcal{A}_1} q_1' \text{ and } q_2 \xrightarrow{w}_{\mathcal{A}_2} q_2'.$$

Together with definition of initial and final states in $\mathcal{A}$ this implies $w \in L(\mathcal{A})$ iff $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. ■

To show the closure under complement, non deterministic automata are not appropriate.

Note: If $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ is *non–deterministic*, then the automaton $\overline{\mathcal{A}} := (Q, \Sigma, I, \Delta, Q \setminus F)$ need *not* satisfy $L(\overline{\mathcal{A}}) = \Sigma^* \setminus L(\mathcal{A})$.

E.g.: $\mathcal{A} :=$ 

$Q = \{1, 2\}$, $\Sigma = \{a\}$, $\quad I = \{1\}$

$\Delta = \{(1, a, 1); (1, a, 2)\}$, $F = \{1\}$

$L(\mathcal{A}) = a^*$, $L(\overline{\mathcal{A}}) = a^+$, but $a^* \setminus L(\mathcal{A}) = \emptyset$

This construction works if the automaton is deterministic.

Definition 1.4 The automaton $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ is called *deterministic* iff:

- $|I| = 1$, i.e. $I = \{q_0\}$,
- Δ is functional, i.e. for every $q \in Q$ and every $a \in \Sigma$ there is exactly one $q' \in Q$ such that $(q, a, q') \in \Delta$.

Instead of the transition relation Δ we will often use the *transition function*.

$$\begin{array}{rrcl} \delta: & Q \times \Sigma & \rightarrow & Q \\ & (q,a) & \mapsto & q' \text{ iff } (q,a,q') \in \Delta. \end{array}$$

The function δ can be extended to words by defining $\delta(q,w) := q'$, where q' is the unique state such that $q \xrightarrow{w}_{\mathcal{A}} q'$. We have $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q_0,w) \in F\}$.

The *power set construction* can be used to construct a deterministic automaton $P(\mathcal{A})$ from a given non–deterministic automaton $\mathcal{A} = (Q,\Sigma,I,\Delta,F)$. We define $P(\mathcal{A}) := (2^Q,\Sigma,q_o,\delta',F')$ where:

- $q_0 := I$,
- $\delta'(P,a) := \{q \in Q \mid \exists p \in P \text{ with } (p,a,q) \in \Delta\}$,
- $F' := \{P \subseteq Q \mid P \cap F \neq \emptyset\}$.

It is easy to see that $L(\mathcal{A}) = L(P(\mathcal{A}))$ and that $P(\mathcal{A})$ is deterministic.

Proposition 1.5 If $L \in Reg_\Sigma$, then $\overline{L} = \Sigma^* \setminus L \in Reg_\Sigma$.

Proof: For L there exists a *deterministic* automaton $\mathcal{A} = (Q,\Sigma,q_0,\delta,F)$ with $L = L(\mathcal{A})$. Thus $w \in L$ iff $\delta(q_0,w) \in F$. This is equivalent to saying that $w \in \overline{L}$ iff $\delta(q_0,w) \in Q \setminus F$. Consequently $\overline{\mathcal{A}} = (Q,\Sigma,q_0,\delta,Q \setminus F)$ accepts $\overline{L}$. ∎

Minimization of deterministic automata:
For every regular language there is a unique (up renaming of states) *minimal deterministic automaton* accepting this language.

Given a deterministic automaton $\mathcal{A} = (Q,\Sigma,q_0,\delta,F)$, one can minimize it as follows:

1. Remove *unreachable* states, i.e. states $q \in Q$ such that there is no $w \in \Sigma^*$ with $\delta(q_o,w) = q$.

2. Identify *equivalent* states.

   For $q \in Q$ let $\mathcal{A}_q := (Q, \Sigma, q, \delta, F)$.
   We define:

   $$q \sim_{\mathcal{A}} q' \text{ iff } L(\mathcal{A}_q) = L(\mathcal{A}_{q'}).$$

   The relation $\sim_{\mathcal{A}}$ is an equivalence relation. Now identify equivalent states. This yields the unique minimal automaton.

Alternatively, the minimal automaton can be obtained using the *Nerode right congruence*. For a language $L \subseteq \Sigma^*$ we define

$$u \rho_L v \text{ iff } \forall w \in \Sigma^* : (uw \in L \text{ iff } vw \in L).$$

This relation is an equivalence relation, which additionally satisfies:

$$u \rho_L v \leadsto uw \rho_L vw \qquad \text{(right congruence).}$$

Nerode's theorem says that a language L is regular iff ρ_L has a finite index, i.e. it has finitely many equivalence classes.
We can view these classes as states of an automaton:
for $u \in \Sigma^*$, $[u] := \{v \mid u \rho_L v\}$ denotes the ρ_L equivalence class of u.
We define $\mathcal{A}_L = (Q, \Sigma, q_0, \delta, F)$ where

- $Q := \{[u] \mid u \in \Sigma^*\}$ (finite if L is regular),
- $q_0 := [\epsilon]$,
- $\delta([u], a) := [ua]$ (independence of representatives!),
- $F := \{[u] \mid u \in L\}$ (independence of representatives!).

For a regular language L, $\mathcal{A}_L$ is the minimal deterministic automaton for L.

1.2 Regular languages and finite monoids

A monoid $(M, \bullet_M, 1_M)$ consists of a non empty set M, an associative binary operation $\bullet_M$ and a unit element $1_M \in M$, i.e. the following must be satisfied:

$$\begin{array}{lll} \forall x, y, z \in M: & (x \bullet_M y) \bullet_M z = x \bullet_M (y \bullet_M z) & \text{(associative)}, \\ \forall x \in M: & 1_M \bullet_M x = x \bullet_M 1_M = x & \text{(unit element)}. \end{array}$$

We will often write just M instead of $(M, \bullet_M, 1_M)$ and we will often omit the index M. A *monoid* $(M, \bullet_M, 1_M)$ is finite iff M is finite.
Let M, N be monoids. The mapping $\phi : M \to N$ is a *homomorphism* iff:

- $\phi(x \bullet_M y) = \phi(x) \bullet_N \phi(y)$,
- $\phi(1_M) = 1_N$.

Example 1.6 For an alphabet Σ, the set Σ^* together with concatenation as binary operation and the empty word ϵ as unit element is a monoid.

Σ^* is called the *free monoid* over Σ since it satisfies the following universal property:
For any monoid M and any mapping $f : \Sigma \to M$, this mapping can uniquely be extended to a homomorphism $\phi : \Sigma^* \to M$ with $\phi|_\Sigma = f$.
In fact: $\phi(\epsilon) := 1_M$ and $\phi(a_1 \dots a_n) := \phi(a_1) \bullet_M \dots \bullet_M \phi(a_n)$
This means: homomorphisms from $\Sigma^* \to M$ can be defined by mappings $f : \Sigma \to M$

Homomorphisms from Σ^* into a monoid M can be used to define languages (i.e. subsets of Σ^*).

Definition 1.7 Let M be a monoid, Σ an alphabet and $\phi : \Sigma^* \to M$ a homomorphism. Every subset N of M defines a subset of Σ^*:

$$\phi^{-1}(N) := \{w \in \Sigma^* \mid \phi(w) \in N\}$$

The language $L \subseteq \Sigma^*$ is *accepted* by M iff there is $N \subseteq M$ and a homomorphism $\phi : \Sigma^* \to M$ such that $L = \phi^{-1}(N)$.

Proposition 1.8 Let $L \subseteq \Sigma^*$. Then the following are equivalent:

1. L is accepted by a emphasize monoid.
2. L is regular.

Proof:

"1 $\rightsquigarrow$ 2" The monoid itself can be viewed as a finite automaton.
Let $\phi : \Sigma^* \to M$ be a homomorphism and $L = \phi^{-1}(N)$ for $N \subseteq M$ where M is a finite monoid. The *deterministic finite* automaton $\mathcal{A}_M := (M, \Sigma, 1, \delta, N)$ with transition function $\delta(m, \sigma) := m \bullet \phi(\sigma)$ accepts L.
First, one shows:

> For all $w \in \Sigma^*$ and all $m \in M$ we have $\delta(m, w) = m \bullet \phi(w)$ (induction over $|w|$).

Consequently, $\delta(1, w) = 1 \bullet \phi(w) = \phi(w)$.

$$\begin{aligned} w \in L = \phi^{-1}(N) \quad & \text{iff} \quad \phi(w) \in N \\ & \text{iff} \quad \delta(1, w) \in N \\ & \text{iff} \quad w \in L(\mathcal{A}_M). \end{aligned}$$

"2 $\rightsquigarrow$ 1": Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a deterministic automaton with $L = L(\mathcal{A})$. Every word $w \in \Sigma^*$ defines a function $\delta_w : Q \to Q : q \mapsto \delta(q, w)$.
Let $M = Q^Q$ be the set of function from Q to Q. Since Q is finite, M is also finite. With composition of functions as binary operation and the identity function as unit element, M is a monoid.
Notation: $(\delta_1 \circ \delta_2)(q) := \delta_2(\delta_1(q))$ (order!)
It is easy to see that $\phi : \Sigma^* \to M : w \mapsto \delta_w$ is a homomorphism.
How must $N \subseteq M$ be defined?
Condition: $L(\mathcal{A}) = \phi^{-1}(N)$

$$\begin{array}{ccc} w \in L(\mathcal{A}) & \text{iff} & \phi(w) \in N \\ \Downarrow & & \Downarrow \\ \delta(q_0, w) = \delta_w(q_0) \in F & \text{iff} & \delta_w \in N \end{array}$$

Thus, if we define $N := \{\delta_w \mid w \in \Sigma^* \text{ and } \delta_w(q_0) \in F\}$, then $\phi^{-1}(N) = L(\mathcal{A}) = L$ ∎

The image of Σ^* under ϕ is called the transition monoid of $\mathcal{A}$:

Definition 1.9 If $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ is a deterministic finite automaton, then its *transition monoid* is the submonoid $M_{\mathcal{A}} := \{\delta_w \mid w \in \Sigma^*\}$ of $M = Q^Q$.

The transition monoid of the minimal automaton is of particular interest.

Definition 1.10 For a regular language L, the transition monoid of the minimal automaton is called the *syntactic monoid* of L. We denote this monoid as M_L.

Since the minimal automaton is uniquely defined by L, M_L only depends on L. We can also define M_L directly from L (without the detour through automata).

Definition 1.11 For an arbitrary language $L \subseteq \Sigma^*$, its *syntactic congruence* $\sim_L$ on Σ^* is defined as follows:

$$\forall u, v \in \Sigma^* : u \sim_L v \text{ iff } \forall x, y \in \Sigma^* : xuy \in L \text{ iff } xvy \in L.$$

It is easy to show that $\sim_L$ is a congruence, i.e. it is an equivalence relation that also satisfies:

$$\forall u, v, x \in \Sigma^* : \left(u \sim_L v \leadsto (ux \sim_L vx \text{ and } xu \sim_L xv)\right).$$

Thus, we can construct the quotient monoid $\Sigma^*/_{\sim_L}$:

- domain $\{[w]_{\sim_L} \mid w \in \Sigma^*\}$ where $[w]_{\sim_L} := \{w' \mid w \sim_L w'\}$
- operation $[u]_{\sim_L} \bullet [v]_{\sim_L} := [uv]_{\sim_L}$ (independence of representatives since $\sim_L$ is a congruence)
- unit element $[\epsilon]_{\sim_L}$.

Proposition 1.12 Let $L \subseteq \Sigma^*$ be regular. Then $\Sigma^*/_{\sim_L}$ is isomorphic to the syntactic monoid M_L.

Proof: Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be the minimal automaton for L.
We define:

$$\begin{array}{rcl} \psi : \Sigma^*/_{\sim_L} & \to & M_L \\ [u]_{\sim_L} & \mapsto & \delta_u \end{array}$$

1. This definition is independent of the chosen representative,
   i.e. $u \sim_L v \to \delta_u = \delta_v$.

   Assume that $u \sim_L v$, but $\delta_u \neq \delta_v$. Thus there is a $q \in Q$ such that

   $$\delta_u(q) = \delta(q, u) = q_1 \neq q_2 = \delta(q, v) = \delta_v(q).$$

   Since $\mathcal{A}$ is minimal, q is reachable, i.e. there is an $x \in \Sigma^*$ such that $q = \delta(q_0, x)$.

   Since $u \sim_L v$, we know for all $y \in \Sigma^*$ that

   $$xuy \in L \text{ iff } xvy \in L.$$

   But then we know for all $y \in \Sigma^*$ that

   $$\delta(q_1, y) = \delta(q_0, xuy) \in F \text{ iff } \delta(q_0, xvy) = \delta(q_2, y) \in F$$

   This shows that $L(\mathcal{A}_{q_1}) = L(\mathcal{A}_{q_2})$.
   Since $\mathcal{A}$ is minimal, $q_1 = q_2$.

2. ψ is injective, i.e. $\delta_u = \delta_v \Rightarrow [u]_{\sim_L} = [v]_{\sim_L}$.

   Assume that $\delta_u = \delta_v$ and $xuy \in L$. We must show $xvy \in L$.
   $xuy \in L \Rightarrow \delta(q_0, xuy) \in F$. But this yields for $q_1 := \delta(q_0, x)$:

   $$\begin{aligned} & \delta(q_0, xuy) = \delta(q_1, uy) = \delta_y(\delta_u(q_1)) \\ \overset{\delta_u = \delta_v}{=} \; & \delta_y(\delta_v(q_1)) = \delta(q_1, vy) = \delta(q_0, xvy) \in F \Rightarrow xvy \in L \end{aligned}$$

3. ψ is surjective:

   δ_u is the image of $[u]_{\sim_L}$.

4. ψ is a homomorphism:

$$\psi([u]_{\sim_L} \bullet [v]_{\sim_L}) = \psi([uv]_{\sim_L}) = \delta_{uv} = \delta_u \circ \delta_v = \psi([u]_{\sim_L}) \circ \psi([v]_{\sim_L}).$$

$$\psi([\epsilon]_{\sim_L}) = \delta_\epsilon.$$

■

Corollary 1.13 L is regular iff $\sim_L$ has finite index.

Proof:

"$\Rightarrow$" If L is regular, then M_L is the transition monoid of the minimal automaton for L. This monoid is obviously finite.
We have just shown that $M_L \simeq \Sigma^*/_{\sim_L}$, and thus $\sim_L$ has only finitely many equivalence classes.

"$\Leftarrow$" To show that L is regular, we show that L is accepted by the finite monoid $\Sigma^*/_{\sim_L}$.

We define: $\phi : \Sigma^* \to \Sigma^*/_{\sim_L} : u \mapsto [u]_{\sim_L}$ and $N := \{[u]_{\sim_L} \mid u \in L\}$

We must show the following: $\phi^{-1}(N) = L$

"$\supseteq$" $u \in L \Rightarrow \phi(u) = [u]_{\sim_L} \in N \Rightarrow u \in \phi^{-1}(N)$

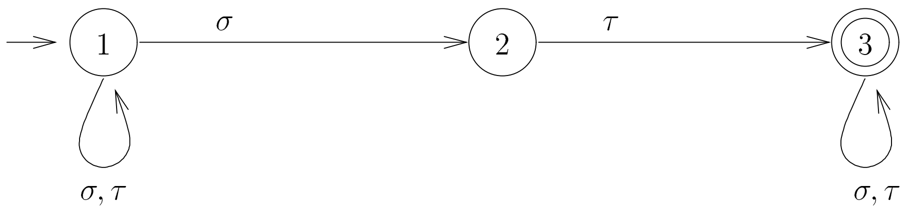
"$\subseteq$" $u \in \phi^{-1}(N) \Rightarrow \phi(u) = [u]_{\sim_L} \in N \Rightarrow u \in L$

■

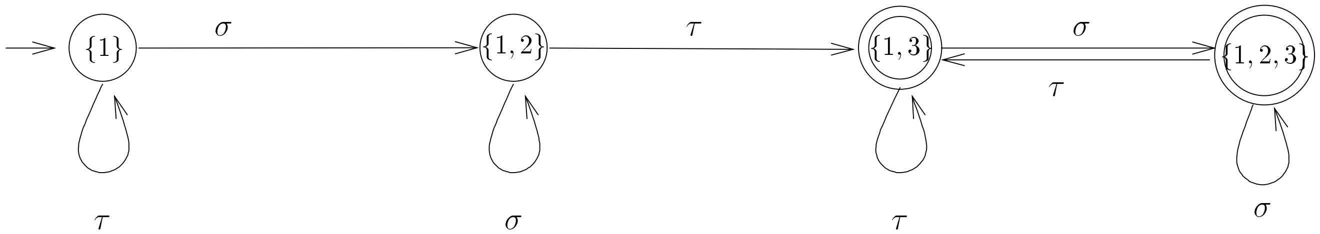
Note: The definition of N is again independent of the chosen representative since $(*)$ $u \in L$ and $u \sim_L v \Rightarrow v \in L$.

Example 1.14 Let $L = \{\sigma,\tau\}^*\sigma\tau\{\sigma,\tau\}^*$.

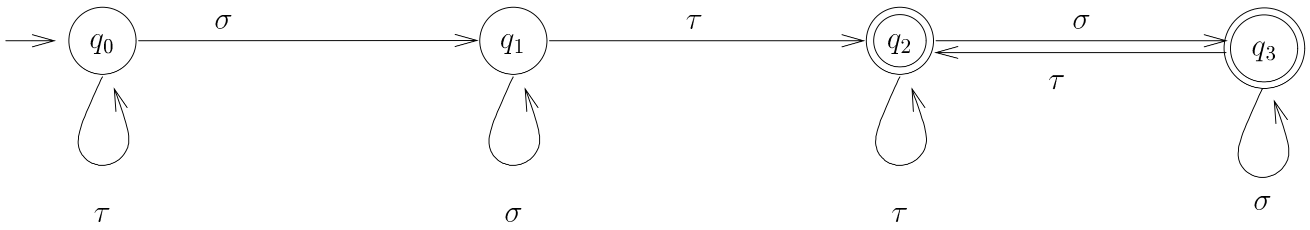
The following is a non–deterministic finite automaton for L:



Power set construction only generating reachable states:



Thus, we now want to minimize the automaton:



Identify equivalent states:

$$q \sim_{\mathcal{A}} q' \text{ iff } L(\mathcal{A}_q) = L(\mathcal{A}_{q'})$$

To compute $\sim_{\mathcal{A}}$ we compute the following relations $\sim_n$ $(n \geq 0)$.

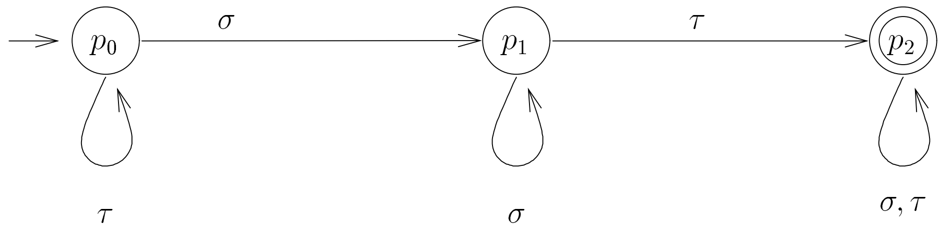
$$\begin{aligned} q \sim_0 q' \quad &\text{iff} \quad (q \in F \text{ and } q' \in F) \text{ or } (q \notin F \text{ and } q' \notin F), \\ q \sim_{n+1} q' \quad &\text{iff} \quad q \sim_n q' \text{ and } \forall a \in \Sigma : \delta(q,a) \sim_n \delta(q',a). \end{aligned}$$

$\sim_n$ are equivalence relations such that $Q \times Q \supseteq \sim_0 \supseteq \sim_1 \supseteq \ \ldots$
Since Q is finite, there is a k such that $\sim_k = \sim_{k+1}$. One can show that then $\sim_k = \sim_{\mathcal{A}}$.

In the example:

- $\sim_0$: has the classes $F = \{q_2, q_3\}$ and $\overline{F} = \{q_0, q_1\}$
- $\sim_1$: $\{q_0\}, \{q_1\}, \{q_2, q_3\}$
- $\sim_2 = \sim_1 = \sim_{\mathcal{A}}$

Thus, the minimal automaton looks as follows:



The syntactic monoid of L is the transition monoid of this automaton

$$\delta_\epsilon = \frac{p_0\ p_1\ p_2}{p_0\ p_1\ p_2}, \quad \delta_\sigma = \frac{p_0\ p_1\ p_2}{p_1\ p_1\ p_2} = \delta_{\sigma\sigma}, \quad \delta_\tau = \frac{p_0\ p_1\ p_2}{p_0\ p_2\ p_2} = \delta_{\tau\tau},$$

$$\delta_{\sigma\tau} = \frac{p_0\ p_1\ p_2}{p_2\ p_2\ p_2} = \delta_{\sigma\tau u} \text{ for all } u \in \Sigma^*$$

$$\delta_{\tau\sigma} = \frac{p_0\ p_1\ p_2}{p_1\ p_2\ p_2} = \delta_{\tau\sigma\sigma} \neq \delta_{\tau\sigma\tau} = \delta_{\sigma\tau}.$$

Consequently $M_{\mathcal{A}} = \{\delta_\epsilon, \delta_\sigma, \delta_\tau, \delta_{\sigma\tau}, \delta_{\tau\sigma}\}$. The multiplication table can be obtained from the observed identities.

Example 1.15 Not every finite monoid can be obtained as syntactic monoid of a regular language.

$M = \{1, a, b, c\}$

$\cdot$	1	a	b	c
1	1	a	b	c
a	a	a	b	c
b	b	a	b	c
c	c	a	b	c

a, b, c are right absorbing, i.e. $\forall x \in M$: $x \cdot a = a$, $x \cdot b = b$, $x \cdot c = c$.

Note: $\cdot$ is associative

We claim that M cannot be the syntactic monoid of some regular language:

Assume that L is a regular language and $\psi : M \to \Sigma^*/_{\sim_L}$ is a isomorphism. We know: $u \in L \Rightarrow [u]_{\sim_L} \subseteq L$ (see $(*)$ in the proof of Corollary 1.13)
For $m \in \{a, b, c\}$ we thus have:

$$\psi(m) \subseteq L \text{ or } \psi(m) \subseteq \overline{L}.$$

We have three elements and two possibilities for them to satisfy. Thus, two of these elements must behave the same. We consider the case $\psi(a) \subseteq L$ and $\psi(b) \subseteq L$ (all the other cases can be treated similarly).

Claim: $\psi(a) = \psi(b)$ (this is a contradiction to ψ being a isomorphism)

Proof of the Claim: Assume that $\psi(a) = [u]_{\sim_L}$ and $\psi(b) = [v]_{\sim_L}$. We must show that $u \sim_L v$. Consider $x, y \in \Sigma^*$. We must show that $xuy \in L$ iff $xvy \in L$.

Case 1: $[y]_{\sim_L} \neq [\epsilon]_{\sim_L}$, and thus $\psi^{-1}([y]_{\sim_L})$ is right–absorbing

$$\begin{aligned} \psi^{-1}([xuy]_{\sim_L}) &= \psi^{-1}([x]_{\sim_L})\psi^{-1}([u]_{\sim_L})\psi^{-1}([y]_{\sim_L}) \\ &= \psi^{-1}([y]_{\sim_L}) \\ &= \psi^{-1}([x]_{\sim_L})\psi^{-1}([v]_{\sim_L})\psi^{-1}([y]_{\sim_L}) = \psi^{-1}([xvy]_{\sim_L}) \end{aligned}$$

Thus $[xuy]_{\sim_L} = [xvy]_{\sim_L}$, i.e. $xuy \sim_L xvy \Rightarrow xuy \in L$ iff $xvy \in L$.

Case 2: $[y]_{\sim_L} = [\epsilon]_{\sim_L}$ and thus $\psi^{-1}([y]_{\sim_L}) = 1$.

$$\begin{aligned} \psi^{-1}([xuy]_{\sim_L}) &= \psi^{-1}([x]_{\sim_L})\psi^{-1}([u]_{\sim_L}) \\ &= \psi^{-1}([u]_{\sim_L}) = a \\ \psi^{-1}([xvy]_{\sim_L}) &= \psi^{-1}([x]_{\sim_L})\psi^{-1}([v]_{\sim_L}) \\ &= \psi^{-1}([v]_{\sim_L}) = b. \end{aligned}$$

We know $\psi(a) \subseteq L$ and $\phi(b) \subseteq L$. Thus $xuy \in L$ and $xvy \in L$.

■

We will use the connection between finite monoids and regular expressions to define subclasses of the class of regular languages.

Definition 1.16 Let V be a class of finite monoids. The corresponding class of languages $L(V)$ is defined as follows:

$$L(V)_\Sigma = \{L \subseteq \Sigma^* \mid M_L \in V\}.$$

Note: Since V contains only finite monoids, all languages in $L(V)_\Sigma$ are regular.

To obtain "reasonable" classes of languages, we must restrict the attention to "reasonable" classes of monoids. So called M–varieties have turned out to be reasonable in this context.

Definition 1.17 A non–empty class V of finite monoids is called M–variety iff it is closed under building submonoids, (binary) direct products and homomorphic images.

Submonoid: $N \subseteq M$ is a submonoid of $(M, \bullet, 1)$ iff

- $1 \in N$
- $n, n' \in N \Rightarrow n \bullet n' \in N$

Closure under building submonoids means: $M \in V$, N is a submonoid of $M \Rightarrow N \in V$.

Direct product: $M_1 \times M_2$ with

- $1_{M_1 \times M_2} = (1_{M_1}, 1_{M_2})$
- $(m_1, m_2) \circ (m_1', m_2') = (m_1 \bullet m_1', m_2 \bullet m_2')$

Homomorphic images: if $\phi : M_1 \to M_2$ is a surjective homomorphism, then M_2 is a homomorphic image of M_1.

Example 1.18 Let V_G be the class of all finite groups. It is not hard to show, that V_G is an M–variety.

Note: Closure under building submonoids only holds since we consider finite groups. E.g. $(\mathbb{Z}, +, 0)$ is a group, and $(\mathbb{N}, +, 0)$ is a submonoid, but it is not a group.

There is an alternative characterization of M–varieties using equations.

Let X be a countably infinite alphabet (of variables). An *equation* is of the form $u = v$ where $u, v \in X^*$ (instead of ϵ we usually write 1 in such equations).

The monoid M *satisfies* the equation $u = v$ iff $\phi(u) = \phi(v)$ for all homomorphisms $\phi : X^* \to M$.

Example: assume that $x, y \in X$
Then $xy = yx$ is an equation, which is satisfied by all commutative monoids: commutative: $\forall m, n \in M : m \bullet n = n \bullet m$

Take the homomorphism ϕ such that $\phi(x) = m$ and $\phi(y) = n$. If M satisfies $xy = yx$, then $m \bullet n = \phi(x) \bullet \phi(y) = \phi(xy) = \phi(yx) = \phi(y) \bullet \phi(x) = n \bullet m$.

Definition 1.19 Let $(u_n = v_n)_{n \geq 1}$ be a sequence of equations.

1. M *ultimately satisfies* $(u_n = v_n)_{n \geq 1}$ iff there is a $k \geq 1$ such that M satisfies $(u_n = v_n)$ for all $n \geq k$.
2. The class V of finite monoids *ultimately defined* by $(u_n = v_n)_{n \geq 1}$ consists of all the monoids that ultimately satisfy $(u_n = v_n)_{n \geq 1}$.

Theorem 1.20 [Eilenberg, Schützenberger] For a class V of finite monoids, the following are equivalent:

1. V is an M–variety.
2. V is ultimately defined by some sequence of equations.

Example 1.18 (continuation) The M–variety of all finite groups is ultimately defined by $(x^{\bar{n}} = 1)$ where $\bar{n} = \mathrm{lcm}(1, \ldots, n)$ (lcm = least common multiple).

Proof:

1. If $G \in V_G$ then G satisfies $x^{\bar{n}} = 1$ for all $n \geq |G|$.
   Let $k := |G|$
   Claim: For all $g \in G$ there is an $l \leq k$ such that $g^l = 1$.
   Consider $g^0 = 1, g^1 = g, g^2, g^3, \ldots, g^k$.
   Since $|G| = k$, there are $0 \leq i \leq k$ and $1 \leq l \leq k$ such that $g^i = g^{i+l}$.

   ∎

   Since $l \leq k$, we know that $l \mid \bar{n}$ for all $n \geq k \Rightarrow$ there is an r such that $l \cdot r = \bar{n}$, and thus $g^{\bar{n}} = (g^l)^r = 1^r = 1$.

2. If a monoid M satisfies the equation $x^{\bar{n}} = 1$, then M is a group: for $m \in M$, we know that $m^{\bar{n}-1}$ is an inverse since $m \bullet m^{\bar{n}-1} = m^{\bar{n}-1} \bullet m = m^{\bar{n}} = 1$.

The closure properties of M–varieties imply closure properties of the induced classes of languages. We will show closure under $\cap, \cup$ and $^-$. But first, we need one more lemma.

Lemma 1.21 Let V be an M–variety and $L \subseteq \Sigma^*$ a language that is accepted by some $M \in V$. Then $M_L \in V$.

Proof: Since M accepts L, there is a homomorphism $\phi : \Sigma^* \to M$ and a set $N \subseteq M$ such that $L = \phi^{-1}(N)$.

1. Without loss of generality, ϕ can be assumed to be surjective. Otherwise, consider $M' = \phi(\Sigma^*)$ and $N' = M \cap N$ instead of M, N. Since M' is a submonoid of M, we know $M' \in V$.

2. Define for $u, v \in \Sigma^*$: $u \sim_\phi v$ iff $\phi(u) = \phi(v)$.
   Then $\sim_\phi \subseteq \sim_L$.

Since: Assume $u \sim_\phi v$ and $xuy \in L$. We must show $xvy \in L$.

$$\begin{aligned}
\phi(xvy) &= \phi(x)\phi(v)\phi(y) \\
&= \phi(x)\phi(u)\phi(y) \text{ since } u \sim_\phi v \\
&= \phi(xuy) \in N \text{ since } xuy \in L = \phi^{-1}(N) \\
&\Rightarrow xvy \in \phi^{-1}(N) = L
\end{aligned}$$

3. We define:

$$\begin{aligned}
\psi : M &\to \Sigma^*/_{\sim_L} \\
m &\mapsto [u]_{\sim_L} \text{ if } \phi(u) = m.
\end{aligned}$$

- ψ is well–defined: if $\phi(u) = m = \phi(v)$, then $u \sim_\phi v$ and thus $u \sim_L v \Rightarrow [u] = [v]$
  In addition, for every m there is a u with $\phi(u) = m$ since ϕ was assumed to be surjective.
- ψ is surjective since for every $[u]$ we can take $m := \phi(u)$, and then $\psi(m) = [u]$.
- Obviously ψ is a homomorphism

■

Thus, $\Sigma^*/_{\sim_L}$ is a homomorphic image of $M \in V$, and thus $\Sigma^*/_{\sim_L} \in V$

Proposition 1.22 Let V be an M–variety. Then $L(V)$ is closed under intersection, union and complement.

Proof: It is enough to show closure under intersection and complement.

1. **complement:** $M_L = M_{\Sigma^* \setminus L}$
   Since: $M_L = \Sigma^*/_{\sim_L}$ and $M_{\Sigma^* \setminus L} = \Sigma^*/_{\sim_{\Sigma^* \setminus L}}$
   However: $\sim_L = \sim_{\Sigma^*/_L}$
   since $\quad xuy \in L$ iff $xvy \in L$
   is equivalent to: $\quad xuy \notin L$ iff $xvy \notin L$.

2. **intersection:** assume that $\Sigma^*/_{\sim_L} \in V$ and $\Sigma^*/_{\sim_{L'}} \in V$ (i.e. $L, L' \in L(V)_\Sigma$)
   Consider the homomorphism

$$\begin{array}{rcl} \phi : \Sigma^* & \to & \Sigma^*/_{\sim_L} : \; u \mapsto [u]_{\sim_L} \\ \phi' : \Sigma^* & \to & \Sigma^*/_{\sim_{L'}} : \; u \mapsto [u]_{\sim_{L'}} \end{array}$$

   We know (proof of Corollary 1.13) that, for $N = \phi(L)$, $N' = \phi'(L')$, we have $L = \phi^{-1}(N)$ and $L' = \phi'^{-1}(N')$.
   We define

$$\begin{array}{rcl} \psi : \Sigma^* & \to & \Sigma^*/_{\sim_L} \times \Sigma^*/_{\sim_{L'}} \in V! \text{ (closure under product)} \\ u & \mapsto & (\phi(u), \phi'(u)) \end{array}$$

   This is a homomorphism and we have

$$\begin{array}{rcl} \psi^{-1}(N \times N') & = & \{u \in \Sigma^* \mid \phi(u) \in N \text{ and } \phi'(u) \in N'\} \\ & = & \phi^{-1}(N) \cap \phi'^{-1}(N') \\ & = & L \cap L' \end{array}$$

   This shows that $L \cap L'$ is accepted by $\Sigma^*/_{\sim_L} \times \Sigma^*/_{\sim_{L'}}$. Since $\Sigma^*/_{\sim_L} \times \Sigma^*/_{\sim_{L'}} \in V$, Lemma 1.21 implies that $M_{L \cap L'} \in V$ and thus $L \cap L' \in L(V)_\Sigma$. ∎

Sometimes it is more appropriate to look at semigroups instead of monoids:

Semigroup: has a binary associative operation (no unit element is required)

Most of the notions and results can be transferred from monoids to semigroups.

Syntactic semigroup:
The syntactic congruence $\sim_L$ is also a congruence on the free semigroup $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. For a language $L \subseteq \Sigma^*$ the syntactic semigroup is $\Sigma^+/_{\sim_L}$.

Alternatively: The syntactic semigroup is the transition semigroup $\{\delta_w \mid w \in \Sigma^+\}$ of the minimal automaton for L.

A *S–Variety* S is a class of finite semigroups that is closed under direct product, homomorphic images and building subsemigroups.

Note: Even if $1 \in S$, it need not to be in its subsemigroups.

S–varieties can also be ultimately defined by equations (Prop. 1.20 holds in a semigroup variant 1.20s). The equations may not contain 1.

Note: S–varieties sometimes yield a more fine–grained division into classes.

For example: the equation $xy = y$
Only trivial monoids (i.e. monoids of cardinality 1) satisfy this equation.
In fact, let $m \in M$: $m = 1 \bullet m = 1$

Nontrivial semigroup satisfying $xy = x$:

$\cdot$	a	b
a	a	a
b	b	b

$\cdot$ is associative

Corresponding class of languages
If V is an S–variety, then $L(V)_\Sigma = \{L \subseteq \Sigma^* \mid S_L \in V\}$

Lemma 1.21 and Prop. 1.22 also hold in semigroup variants 1.21s and 1.22s.

1.3 Regular languages and logical formulae

Which logic? For the moment, first order predicate logic.

Syntax: extra logical symbols are

$$\underset{\text{binary}}{=} \quad , \quad \underset{\text{binary}}{<} \quad , \quad \underset{\text{unary symbols}}{P_1, \ldots, P_k}$$

Semantics: we consider finite interpretations only (for the moment)

$=$ is interpreted as equality,
$<$ is a total ordering (linear ordering) on the domain
$P_1, \ldots, P_k$ are interpreted as subsets of the domain.

Such interpretations can be viewed as words over $\Sigma = \{0, 1\}^k$

Let I be an interpretation.

- $\mathrm{dom}(I) = \{d_1, \ldots, d_n\}$ for some $n \geq 1$ where $d_1 < d_2 < \cdots < d_n$.
- P_j is interpreted as a set $P_j^I \subseteq \mathrm{dom}(I)$
- Let $\sigma_i = (b_{i1}, \ldots, b_{ik})$ where

$$b_{i_j} = \begin{cases} 1 & d_i \in P_j^I \\ 0 & d_i \notin P_j^I \end{cases}$$

Then I corresponds to the word $\sigma_1\sigma_2\cdots\sigma_n$. Conversely, every such word yields an interpretation.

Example 1.23 $k = 2$, i.e. $\Sigma = \{0,1\}^2$
The word $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ over Σ corresponds to the interpretation

- $\mathrm{dom}(I) = \{d_1, d_2, d_3\}$
- $d_1 < d_2 < d_3$
- $P_1^I = \{d_1, d_3\}$
- $P_2^I = \{d_2\}$

Instead of interpretations, we will use words. Thus it makes sense to say that a word $w \in \Sigma^+$ makes a formula φ true ($w \models \varphi$).

Definition 1.24 Let φ be a closed formula (no free variables) of first–order predicate logic using the extra–logical symbols $=, <, P_1, \ldots, P_k$. Let $\Sigma = \{0,1\}^k$. Then φ *defines* the language

$$L(\varphi) = \{w \in \Sigma^+ \mid w \models \varphi\}.$$

Note: Since interpretations must have non–empty domains, the empty word does not describe an interpretation, and $L(\varphi) \subseteq \Sigma^+$. This is not a real restriction. For example, $L \subseteq \Sigma^*$ is regular iff $L \setminus \{\epsilon\}$ is regular.

Example 1.25 $k = 1$, i.e. $\Sigma = \{0, 1\}$. The language 1^*10^* is defined by the following formula:

$$\exists x. (P_1(x) \wedge \forall y. (y < x \Rightarrow P_1(y)) \wedge \forall z. (x < z \Rightarrow \neg P_1(z)))$$

Proposition 1.26 Boolean operations in formulae correspond to Boolean operations on languages:

$$\begin{array}{rcl} L(\neg\varphi) & = & \Sigma^+ \setminus L(\varphi), \\ L(\varphi \wedge \psi) & = & L(\varphi) \cap L(\psi), \\ L(\varphi \vee \psi) & = & L(\varphi) \cup L(\psi). \end{array}$$

In order to define languages, we will introduce some useful abbreviations:

$\mathbf{Q}_\sigma(x)$ For every $\sigma \in \Sigma$ we can construct a formula $Q_\sigma(x)$ with one free variable that says that σ occurs at position x.

e.g. $k = 2$, $\Sigma = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$

$Q_{(1,1)}(x) := P_1(x) \wedge P_2(x)$, $Q_{(1,0)}(x) := P_1(x) \wedge \neg P_2(x)$

Min(x) We can construct a formula $\mathrm{Min}(x)$ that says that x is the first position of the word.

$\mathrm{Min}(x)$: $\neg\exists y. (y < x)$.

Max(x) Correspondingly we can express the last position by a formula $\mathrm{Max}(x)$.

$\mathrm{Max}(x)$: $\forall y. (y \leq x)$

Succ(x, y) y is the successor position of x:

$\mathrm{Succ}(x, y)$: $x < y \wedge \neg\exists z. (x < z \wedge z < y)$.

s(x) Sometimes it is more convenient to use a function symbol to express the successor:

"$s(x) = y$" corresponds to $\mathrm{Succ}(x, y) \vee (\mathrm{Max}(x) \wedge x = y)$

min, max Correspondingly we will sometimes use constants min, max for the first and the last position

Pred(x, y) Just like successor we can introduce predecessor as formula $\text{Pred}(x, y)$ or as function p.

Example 1.27 The regular language $a(ba)^*$ can be defined through

$$\begin{array}{l} Q_a(\min)\wedge \\ \forall x, y\,(Q_a(x) \wedge \text{Succ}(x, y) \Rightarrow Q_b(y))\wedge \\ \forall x, y\,(Q_b(x) \wedge \text{Succ}(x, y) \Rightarrow Q_a(y)\wedge \\ Q_a(\text{Max}). \end{array}$$

What kind of languages can be defined with formulae from first order predicate logic (PL1)? We will see later on that only regular languages can be defined this way. Can all regular languages be defined with PL1–formulae? No!

Example 1.28 $L = a(aa)^*$ is regular, but it cannot be defined using a PL1–formula. We will see later on how this can be proved. (The connection to monoids becomes important.)

How can we extend PL1 to get all regular languages?

One has to introduce:
Quantification over unary predicates

- Variables for unary predicates: X, Y capital letters

- Variables for objects: x, y lower case letters

The language $a(aa)^*$ is defined by the formula

$$\begin{array}{l} \exists X \exists Y\ (X(\min) \wedge \\ \qquad \forall x, y\ (X(x) \wedge \text{Succ}(x, y) \Rightarrow Y(y)) \wedge \\ \qquad \forall x, y\ (Y(x) \wedge \text{Succ}(x, y) \Rightarrow X(y)) \wedge \\ \qquad X(\max) \wedge \forall x (X(x) \Leftrightarrow \neg Y(x)) \wedge \\ \qquad \forall x\ Q_a(x)). \end{array}$$

We will show that adding quantification over unary predicates gives us exactly the regular languages.

Chapter 3 will be concerned with the class of languages defined by PL1–formulae.

Chapter 2 is a warm–up exercise where we consider a smaller class of languages, which can be defined using quantifier–free PL1–formuale.

Chapter 2

Generalized–definite languages and quantifier–free formulae

First we introduce the class of languages directly. Then we characterize it using semigroups and formulae.

2.1 Generalized–definite languages

Informally, these are languages such that there is a k such that only the first and last k letters of each word are relevant.

Definition 2.1 The class B_0 of all *generalized–definite languages* is defined as follows: $L \subseteq \Sigma^*$ belongs to $(B_0)_\Sigma$ iff there is a $k \geq 0$ such that we have for all $w \in L$:

$$\text{if } w = uv = v'u' \text{ for } |u| = |u'| = k, \text{ then } u\Sigma^* \cap \Sigma^* u' \subseteq L.$$

Note that $u\Sigma^* \cap \Sigma^* u'$ consists of the words starting with u and ending with u'

Whether a word belongs to L or not depends only on the last and first k letters.

Lemma 2.2 $(B_0)_\Sigma$ is the Boolean closure of the languages:

$$\{u\Sigma^* \mid u \in \Sigma^*\} \cup \{\Sigma^* u' \mid u' \in \Sigma^*\}$$

Proof:

1. Let $L \in (B_0)_\Sigma$ and let k be the corresponding number from Def. 2.1.
   (a) $k = 0$:
   Thus $L = \emptyset$ or $L = \Sigma^*$.
   In the second case $\Sigma^* = \epsilon\Sigma^*$ and in the first $\emptyset = u\Sigma^* \cap \overline{u\Sigma^*}$ for an arbitrary u
   (b) $k > 0$:
   $$L = \bigcup_{\substack{|u|=k=|u'| \\ u\Sigma^* \cap \Sigma^* u' \subseteq L}} (u\Sigma^* \cap \Sigma^* u') \cup \{v \in L \mid |v| < k\}.$$
   "$\supseteq$" is trivial.
   "$\subseteq$" Let $w \in L$. If $|w| < k$, then $w \in \{v \in L \mid |v| < k\}$. Assume that $|w| \geq k$. Thus there exist words u, u', v, v' such that $|u| = |u'| = k$ and $w = uv = v'u'$. By the definition of $(B_0)_\Sigma$, $w \in L$ implies $u\Sigma^* \cap \Sigma^* u' \subseteq L$. Thus, $u\Sigma^* \cap \Sigma^* u'$ is in the union and obviously $w \in u\Sigma^* \cap \ \Sigma^* u'$
   It remains to show that $\{v \in L \mid |v| < k\}$ is in the Boolean closure. It is enough to show that $\{v\}$ is in the Boolean closure
   $$\{v\} = v\Sigma^* \setminus v\Sigma\Sigma^* = v\Sigma^* \setminus (\bigcup_{\sigma \in \Sigma} v\sigma\Sigma^*)$$
2. (a) We show $L = w\Sigma^* \in (B_0)_\Sigma$. Take $k = |w|$.
   Assume that $w' \in L$ and that $w' = uv = v'u'$ for $|u| = |u'| = k$. But then $u = w$ and thus $u\Sigma^* \cap \Sigma^* u' \subseteq u\Sigma^* = w\Sigma^* = L$
   (b) The case $L = \Sigma^* w$ can be treated analogously.
   (c) closure under union:
   $L_1 \in (B_0)_\Sigma$ with number k_1 and $L_2 \in (B_0)_\Sigma$ with number k_2. We choose $k = \max\{k_1, k_2\}$. Note: $u = u_1 u_2 \Rightarrow u_1\Sigma^* \supseteq u\Sigma^*$
   Using this fact, it is easy to show that k is the right number for $L_1 \cup L_2$.

(d) closure under complement:
Let $L \in (B_0)_\Sigma$ and let k be the corresponding number. But then k also works for $\overline{L}$. Let $w = uv = v'u' \in \overline{L}$ where $|u| = |u'| = k$. We show $u\Sigma^* \cap \Sigma^* u' \subseteq \overline{L}$.
Assume, that there is a $w' \in u\Sigma^* \cap \Sigma^* u'$ such that $w' \in L$. But this implies $u\Sigma^* \cap \Sigma^* u' \subseteq L$. But then $w \in L$, which is a contradiction.

∎

2.2 The corresponding S–variety

To define this S–variety we need the notion of an idempotent element.

Definition 2.3 Let S be a semigroup. The element $e \in S$ is *idempotent* iff $e \bullet e = e$.

Unit elements are obviously idempotent, but there may be other idempotents as well.

Proposition 2.4 Let S be a finite semigroup and $m \in S$. Then the set $\{m, m^2, m^3, \ldots\}$ contains an idempotent element.

Proof: Since S is finite, there are $i, k \geq 1$ such that $m^i = m^{i+k}$:

$$\begin{tikzcd}
 & & & & m^{i+k-1} \arrow[dl] & \cdots \arrow[l] \\
m \arrow[r] & m^2 \arrow[r] & \cdots \arrow[r] & m^i \arrow[dr] & & \\
 & & & & m^{i+1} \arrow[r] & m^{i+2} \arrow[uu]
\end{tikzcd}$$

Obviously there is an ℓ such that

- $\ell \equiv 0(k)$, i.e. $\exists \ell'.(\ell = k\ell')$.
- $i \leq \ell < i + k$, i.e. $\ell = i + p$ for some p, $0 \leq p < k$

Claim: m^ℓ is idempotent

$$m^\ell \bullet m^\ell \overset{\substack{k\bullet\ell'=l,\\ i+p=\ell}}{=} m^i \bullet m^{k\bullet\ell'} \bullet m^p \overset{m^{i+k}=m^i}{=} m^i \bullet m^p = m^\ell.$$

■

Note: if $m^i, m^{i+1}, \ldots, m^{i+k-1}$ are different from each other (i.e. k was chosen minimal), then $\{m^i, m^{i+1}, \ldots, m^{i+k-1}\}$ is a cyclic group with unit element m^ℓ.

For a given semigroup S we want a number $\bar{n}$ such that $m^{\bar{n}}$ is idempotent for all $m \in S$.

Corollary 2.5 Let S be a finite semigroup and $|S| \leq n$. Then we have for all $m \in S$ and $\overline{n} = \text{lcm}(1, \ldots, n)$: $m^{\overline{n}}$ is idempotent.

Proof: Obviously one obtains in the proof of Prop. 2.4 a k such that $i+k-1 \leq n$. Thus, $\ell \leq n$ and hence ℓ is a divisor of $\overline{n}$, i.e. there is an ℓ' such that $\overline{n} = \ell \cdot \ell'$. It follows that $m^{\overline{n}} = (m^\ell)^{\ell'} = m^\ell$. ■

Definition 2.6 The class $\widehat{\mathbb{D}}$ consists of all finite semigroups S that satisfy the following: for all idempotent $e \in S$ we have $eSe = e$.
(i.e. $\{eme \mid m \in S\} = \{e\}$)
(i.e. $\forall m \in S: \; eme = e$)

Proposition 2.7 $\widehat{\mathbb{D}}$ is an S–variety, which is ultimately defined by

$$(*)(x^{\overline{n}} y x^{\overline{n}} = x^{\overline{n}})_{n\geq 1}.$$

Proof: it is enough to show that $\mathbb{D}$ is ultimaltely defined by $(*)$

1. Let $S \in \widehat{\mathbb{D}}$. By Corollar 2.5 we know that for $n \geq |S|$ and $m \in S$ we have $m^{\overline{n}}$ is idempotent. By the definition of $\widehat{\mathbb{D}}$ it follows that $m^{\overline{n}} m' m^{\overline{n}} = m^{\overline{n}}$ for all $m' \in S$. Thus S satisfies $(*)$ for all $n \geq |S|$.
2. Assume that S satisfies $(*)$ for some n. Then we have for all idempotents e and all $m \in S$:

$$eme = e^{\overline{n}} m e^{\overline{n}} = e^{\overline{n}} = e, \text{ and thus } eSe = e$$

■

Lemma 2.8 Let $S \in \widehat{\mathbb{D}}$.

1. If $n > |S|$ and $m_1, \ldots, m_n \in S$, then $m_1 \bullet \cdots \bullet m_n$ is idempotent.
2. if $e, f \in S$ are idempotent and $m \in S$, then $emf = ef$.

Proof:

1. Consider $m_1, m_1m_2, m_1m_2m_3, \ldots, m_1 \cdots m_n$. Since $n > |S|$ there are $i < j$ such that $m_1 \cdots m_i = m_1 \cdots m_i m_{i+1} \cdots m_j$. By Prop. 2.4 there is an ℓ such that $(m_{i+1} \cdots m_j)^\ell$ is idempotent. But then

$$\begin{aligned}
m_1 \cdots m_n &= (m_1 \cdots m_i)(m_{i+1} \cdots m_j)(m_{j+1} \cdots m_n) \\
&= (m_1 \cdots m_i)(m_{i+1} \cdots m_j)^\ell (m_{j+1} \cdots m_n) \\
&= (m_1 \cdots m_i) \underbrace{(m_{i+1} \cdots m_j)^\ell}_{e} \underbrace{(m_{j+1} \cdots m_n)(m_1 \cdots m_i)}_{\in S} \bullet \\
& \underbrace{(m_{i+1} \cdots m_j)^\ell}_{e} (m_{j+1} \cdots m_n) \\
&= (m_1 \cdots m_i)(m_{i+1} \cdots m_j)(m_{j+1} \cdots m_n) \\
& (m_1 \cdots m_i)(m_{i+1} \cdots m_j)(m_{j+1} \cdots m_n) \\
&= (m_1 \cdots m_n)(m_1 \cdots m_n).
\end{aligned}$$

   The second identity holds since $m_1 \ldots m_i = m_1 \ldots m_i m_{i+1} \ldots m_j$.

2. $e(mf) \overset{eSe=e}{=} \underline{efe}(mf) = e(\underline{femf}) \overset{fSf=f}{=} ef$. ∎

Proposition 2.9

$$L(\widehat{\mathbb{D}}) = B_0,$$

i.e. for every finite alphabet Σ and every $L \subseteq \Sigma^*$ we have
$S_L \in \widehat{\mathbb{D}} \Leftrightarrow L \in (B_0)_\Sigma$.

Proof:

"$\Leftarrow$" Let $L \in (B_0)_\Sigma$, i.e. L is in the Boolean closure of the languages $\{u\Sigma^* \mid u \in \Sigma^*\} \cup \{\Sigma^* u' \mid u' \in \Sigma^*\}$ (Lemma 2.2). By Prop. 1.22s, $L(\widehat{\mathbb{D}})$ is closed under Boolean operations and thus it is sufficient to show that $u\Sigma^*$ und $\Sigma^* u'$ belong to $L(\widehat{\mathbb{D}})$. We consider $L = u\Sigma^*$ ($\Sigma^* u'$ can be treated symmetrically).

Let $|u| = n$.

Case 1: $n = 0$

Thus $u\Sigma^* = \Sigma^*$: in this case $\sim_L = \Sigma^* \times \Sigma^*$, and thus $S_L = \Sigma^+/_{\sim_L}$ consist of a single class, which is obviously idempotent. In addition, $S_L \in \widehat{\mathbb{D}}$ since $S_L = \{e\}$ and $e \bullet e \bullet e = e$.

Case 2: $n > 0$

Claim: If $|w| \geq n$, then $wv \sim_L w$, for all $v \in \Sigma^*$.

Proof of the claim: for all $x, y \in \Sigma^*$ we have

$$\begin{array}{lll} xwvy \in L = u\Sigma^* & \text{iff} & xwvy \text{ starts with } u \\ & \text{iff} & xw \text{ starts with } u \\ & & \quad (\text{since } |w| \geq n) \\ & \text{iff} & xwy \text{ starts with } u \\ & \text{iff} & xwy \in L = u\Sigma^*. \end{array}$$

■ Claim

Let $e = [x]_{\sim_L} \in S_L = \Sigma^+/_{\sim_L}$ be idempotent.

We must show $eS_L e = e$.

Since $|x| \geq 1$, we know that $|x^n| \geq n$ and thus $w := x^n$ satisfies the precondition of the claim. Let $m = [y]_{\sim_L} \in S_L$. Then we have

$$em = e^n m = [x^n]_{\sim_L} \bullet [y]_{\sim_L} = [x^n y]_{\sim_L} \overset{\text{by the claim}}{=} [x^n]_{\sim_L} = e^n = e.$$

In particular, this implies $eme = ee = e$, which shows $S_L \in \widehat{\mathbb{D}}$.

"$\Rightarrow$" Let $S_L \in \widehat{\mathbb{D}}$, $n = |S_L| + 1$. By Lemma 2.8 (1) we have for all words u of length n that $[u]_{\sim_L}$ is idempotent:

$$[u]_{\sim_L} = [\sigma_1 \ldots \sigma_n]_{\sim_L} = [\sigma_1]_{\sim_L} \bullet \cdots \bullet [\sigma_n]_{\sim_L}.$$

Let $x \in L$ with $|x| \geq 2n$. Then $x = uvu'$ for words u, v, u' with $|u| = |u'| = n$. We know that $[u]_{\sim_L}, [u']_{\sim_L}$ are idempotent. By Lemma 2.8

(2) we have for all words w:

$$\begin{aligned} [uwu']_{\sim_L} &= [u]_{\sim_L} \bullet [w]_{\sim_L} \bullet [u']_{\sim_L} \\ &= [u]_{\sim_L} \bullet [u']_{\sim_L} \\ &= [u]_{\sim_L} \bullet [v]_{\sim_L} \bullet [u']_{\sim_L} \\ &= [uvu']_{\sim_L} = [x]_{\sim_L} \end{aligned}$$

Thus, $x \in L$ implies that $u\Sigma^* u' \subseteq L$.
To sum up, this shows the following: if $|u| = |u'| = n$, then $u\Sigma^* u' \subseteq L$ or $u\Sigma^* u' \subseteq \overline{L}$. Consequently,

$$L = \bigcup_{\substack{u\Sigma^* u' \subseteq L \\ |u|=|u'|=n}} u\Sigma^* u' \cup \underbrace{\{w \in L \mid |w| < 2n\}}_{\text{in}_{(B_0)_\Sigma} \text{ since singleton } \{w\} \in (B_0)_\Sigma}$$

It remains to be shown that $u\Sigma^* u' \in (B_0)_\Sigma$:

$$u\Sigma^* u' = \underbrace{(u\Sigma^* \cap \Sigma^* u')}_{\in (B_0)_\Sigma} \setminus \underbrace{\{w \mid |w| < 2n\}}_{\in (B_0)_\Sigma} \in (B_0)_\Sigma$$

■

Corollary 2.10 Let $L \subseteq \Sigma^*$ be a regular language (given by a regular expression or finite automaton). Then we can effectively decide whether $L \in (B_0)_\Sigma$ or not.

Proof: Given $L \subseteq \Sigma^*$, we compute its syntactic semigroup S_L (by computing the minimal automaton and then its transition semigroup). For this finite semigroup we can obviously decide whether $eme = e$ holds for all idempotents $e \in S_L$ and all elements $m \in S_L$. ■

2.3 Quantifier–free formulae

As extra–logical symbols we will use binary relations $=$, $<$, unary relations $P_1, \dots, P_k$, additionally constants min, max, unary functions s, p. The interpretation of these symbols is fixed as described in Chapter 1.3.

Note: In Chapter 1.3 we have seen that min, max, s, p can be expressed by formulae using only the other symbols. However these formulae need quantifiers. Since we consider quantifier-free formulae here, we need these symbols explicitly.

Proposition 2.11 Let $\Sigma = \{0,1\}^k$. For $L \subseteq \Sigma^*$ the following are equivalent:

1. $L \in (B_0)_\Sigma$.
2. $L \setminus \{\epsilon\} = L(\varphi)$ for a quantifier–free closed formula φ over the extra logical symbols $=$, $<$, $P_1, \dots, P_k, \min, \max, s$, and p.

Note: Since $\{\epsilon\} \in (B_0)_\Sigma$, we know that $L \in (B_0)_\Sigma$ iff $L \setminus \{\epsilon\} \in (B_0)_\Sigma$.

Proof:
"1 → 2" Since the Boolean operations on languages can be expressed using the connections $\wedge$, $\vee$, $\neg$ it is enough to consider the languages $u\Sigma^*$ and Σ^*u'.

We consider only the case $L = u\Sigma^*$.

Formula expressing $u\Sigma^*$:

Let $u = \sigma_1 \cdots \sigma_l$ for some $l \geq 1$.

$$\begin{aligned} Q_{\sigma_1}(\min) \wedge Q_{\sigma_2}(s(\min)) \wedge Q_{\sigma_3}(s(s(\mathrm{Min}))) \wedge \dots \\ \dots \wedge Q_{\sigma_l}(s^{l-1}(\min)) \\ \wedge s^{l-2}(\min) < \max \text{ (is left out if } l = 1) \end{aligned}$$

If $u = \epsilon$, then $L = u\Sigma^* = \Sigma^*$, and thus $L \setminus \{\epsilon\} = \Sigma^+$. Thus, we can take any formula that is trivially true; for example, $\min = \min$.

"2 → 1" Let φ be such a closed quantifier–free formula. Thus, φ does not contain variables. Terms are built using $\min, \max, s, p$ (e.g.: $s(p(s(min))))$.

These terms can be normalized into terms of the following form:

$$s^n(\min), p^n(\max)\ n \geq 0$$

To this purpose, we use the fact, that $s(\max) = \max$, $p(\min) = \min$, $s(p(d)) = d = p(s(d))$ unless d is one of the extremal points $\min, \max$.

The formula φ is a Boolean combination of atomic formulae

$$P_i(t),\ t = t',\ t < t',\ \text{where } t, t' \text{ are normalized}$$

Since $(B_0)_\Sigma$ is closed under Boolean operations, it is sufficient to show that the atomic formulae define languages in $(B_0)_\Sigma$

- $P_i(s^n(\min))$ is satisfied by
  - words of lenghth $< n+1$ satisfying some conditions
    Finite sets of words belong to $(B_0)_\Sigma$
  - words of length $\geq n+1$ whose $(n+1)$th symbol $\sigma \in \{0,1\}^k$ has as i–th component a 1:

$$\bigcup_{\substack{|u| = n \\ \sigma \in \Sigma \text{ with } i\text{–th component } 1}} u\sigma\Sigma^* \ \in (B_0)_\Sigma$$

- $P_i(p^n(\max))$ can be treated similarly.
- formulae $t = t'$, $t < t'$ are either true for all words (Σ^+ is in $(B_0)_\Sigma$) or they restrict the length of the words:
  - $s^n(\min) = s^m(\min)$ for $n < m$ says that the word has length $\leq n+1$. Finite sets of words belong to $(B_0)_\Sigma$.
  - $s^n(\min) < s^m(\min)$ for $n < m$ says that the word is of length $> n+1$. The complement is a finite set of words.

All other cases can be treated similarly! ∎

Chapter 3

Star–free languages

These are the languages definable by PL1–formulae.

3.1 The class of languages

The class of regular languages is the smallest class that contains all finite languages and that is closed under

- union ($L_1 \cup L_2$),
- concatenation ($L_1 \cdot L_2$) und
- star (L^*)

Disallowing star here would leave us only with finite languages. This ist a small class strictly contained in $(B_0)_\Sigma$. We know that regular languages are also closed under $\cap$ and $^-$. We now disallow *, but explicitly allow $\cap$ and $^-$.

Definition 3.1 For a finite alphabet Σ, the class SF_Σ of all star–free languages over Σ is the smallest class that satisfies:

- all finite languages over Σ belong to SF_Σ

- if $L, L_1, L_2 \in SF_\Sigma$, then so are $L_1 \cdot L_2$, $L_1 \cup L_2$ $L_1 \cap L_2$, $\overline{L} = \Sigma^* \setminus L$

Example 3.2

1. $\Sigma^* \in SF_\Sigma$ since $\Sigma^* = \overline{\emptyset}$ and $\emptyset$ is finite.
2. If $\Delta \subseteq \Sigma$, then $\Delta^* \in SF_\Sigma$ since $\Delta^* = \Sigma^* \setminus (\Sigma^* \cdot (\Sigma \setminus \Delta) \cdot \Sigma^*)$
3. $\Sigma = \{a, b\}$ Then $a(ba)^* \in SF_\Sigma$ since

$$a \cdot (ba)^* = a \cdot (\Sigma^* \setminus (a\Sigma^* \cup \Sigma^* b \cup \Sigma^* aa\Sigma^* \cup \Sigma^* bb\Sigma^*))$$

The example shows that languages whose straight–forward representation uses star may well be star–free.

How can we decide, for a given regular language, whether it is star–free or not? More concretely: how can we show that $a(aa)^*$ is not star–free?

These questions can be answered by looking at a characterization of star–free languages using finite monoids.

3.2 Aperiodic monoids

In chapter 2 we have seen that for an element m of a finite semigroup the set $\{m, m^2, m^3, \ldots\}$ always contains an idempotent:

$$\begin{tikzcd}
 & & & & m^{i+k-1} \arrow[dl] & \cdots \arrow[l] \\
m \arrow[r] & m^2 \arrow[r] & \cdots \arrow[r] & m^i \arrow[dr] & & \\
 & & & & m^{i+1} \arrow[r] & m^{i+2} \arrow[uu]
\end{tikzcd}$$

For aperiodic monoids, we can choose $k = 1$.

Definition 3.3 The finite monoid M is called *aperiodic* iff there is an $n \geq 1$ such that $m^{n+1} = m^n$ holds for all $m \in M$.

Obviously the *class Ap of aperiodic monoids* is ultimately defined by

$$(x^{n+1} = x^n)_{n \geq 1}.$$

Note: If $m^{n+1} = m^n$, then $m^{n'+1} = m^{n'}$ for all $n' \geq n$. This shows that Ap is an M–variety.

Recall: if $m^i = m^{i+k}$, then $m^i, m^{i+1}, \ldots, m^{i+k-1}$ is a group. If k is minimal, this group contains k elements.

Definition 3.4 Let $(M, \bullet, 1)$ be a monoid. The subset $G \subseteq M$ is a *group in M* iff G is a subsemigroup of M $(m, m' \in G \to m \bullet m' \in G)$ that is a group w.r.t. the operation $\bullet$ of M restricted to G.

Note: The unit of the group G is an idempotent element of M, but it need not to be the unit 1 of M.

For aperiodic monoids, the cyclic groups $\{m^i, \ldots, m^{i+k-1}\}$ have cardinality 1. This is true for all groups in M.

Proposition 3.5 The finite monoid M is *aperiodic* iff it contains only trivial groups (i.e. groups of cardinality 1).

Proof: $(x^{n+1} = x^n)_{n \geq 1}$

"⇒" Let M be aperiodic and let n be such that $m^{n+1} = m^n$ for all $m \in M$. Assume that $G \subseteq M$ is a group in M with $|G| > 1$. Thus G contains in addition to its unit element e another element $g \neq e$. We know that $g^{n+1} = g^n$. By multiplying this equation with $(g^n)^{-1}$, we obtain $g = e$.

"⇐" Let $m \in M$. We consider $\{m, m^2, m^3 \ldots\}$. Let k be minimal such that there is an i with $m^{i+k} = m^i$. Then we know that $\{m^i, m^{i+1}, \ldots, m^{i+k-1}\}$ is a group in M, and thus $k = 1$ since M contains only trivial groups. Thus we have for all $m \in M$ an $i_m \geq 1$ such that $m^{i_m+1} = m^{i_m}$. Obviously, $m^{j+1} = m^j$ for all $j \geq i_m$. Thus, if $n \geq \max\{i_m \mid m \in M\}$ then $m^{n+1} = m^n$ holds for all $m \in M$. ∎

We are interested in aperiodic monoids since the correspond to star–free languages.

Proposition 3.6 (Schützenberger) $L(Ap) = SF$, i.e. for all $L \subseteq \Sigma^+$:

$$M_L \in Ap \quad \text{iff} \quad L \in SF_\Sigma.$$

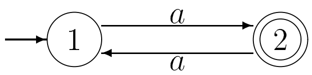
The proof (in particular of "$\Rightarrow$") is rather involved (see *Automata, Languages, and Machines*).

Corollary 3.7 Let $L \subseteq \Sigma^*$ a regular language (given by regular expression, finite automaton, ...). Then it is decidable whether $L \in SF_\Sigma$ or not.

Proof: Construct the syntactic monoid M_L, and then test whether it is aperiodic. ∎

Example 3.8 Let $\Sigma = \{a\}$. Then $a(aa)^* \notin SF_\Sigma$.

Proof: The minimal automaton for $L = a(aa)^*$ is



Transition monoid

$$\delta_\epsilon = \frac{1\ 2}{1\ 2} \quad \delta_a = \frac{1\ 2}{2\ 1}$$

$$\delta_{aa} = \delta_\epsilon$$

Thus $M_L = \{\delta_\epsilon, \delta_a\}$ with unit element δ_ϵ and $\delta_a \circ \delta_a = \delta_\epsilon$. Consequently M_L itself is a group of cardinality > 1. Thus M_L is not aperiodic. ∎

3.3 Formula of first–order–logic

We will show that the star–free languages are exactly the ones definable by first order formulae.

Proposition 3.9 For a language $L \subseteq \Sigma^+$ the following are equivalent

1. $L \in SF_\Sigma$.
2. $L = L(\varphi)$ for a closed formula φ of first–order predicate logic using the extra logical symbols $=$, $<$, Q_a $(a \in \Sigma)$.

Note: Instead of $P_1, \ldots, P_k$ we use Q_a for $a \in \Sigma$ directly. This is just for convenience.

The proof of "$1 \to 2$" is rather simple, whereas "$2 \to 1$" is more involved.

3.3.1 Proof of "$1 \to 2$" of Theorem 3.9

Star–free languages are obtained from the finite languages using Boolean operations and concatenation.

Finite languages

Obviously, it is sufficient to consider singleton sets $\{w\}$ for $w \in \Sigma^+$. Let $w = a_1 \cdots a_n \in \Sigma^+$.

$$\begin{aligned} \varphi_w \quad : \exists x_1, \ldots, x_n \quad & Q_{a_1}(x_1) \wedge \ldots \wedge Q_{a_n}(x_n) \wedge \\ & \bigwedge_{j=1}^{n-1} \big(x_j < x_{j+1} \wedge \neg\exists z.(x_j < z \wedge z < x_{j+1})\big) \wedge \\ & \neg\exists z.(z < x_1 \vee x_n < z) \end{aligned}$$

Obviously, $L(\varphi_w) = \{w\}$.

Boolean operations

Boolean operations correspond to the logical connectives $\wedge$, $\vee$, $\neg$ (by Prop. 1.26).

Concatenation

Concatenation corresponds to the existential quantifier (in principle).

First, let us consider an example: $L_1 := a^+$ and $L_2 := b^+$

$$\varphi_1 : \forall x. Q_a(x) \text{ is a formula defining } L_1 \text{ and}$$
$$\varphi_2 : \forall x. Q_b(x) \text{ is a formula defining } L_2$$

A formula for $L_1 \cdot L_2$:

$$\exists z.\ (\forall x. (x \leq z \Rightarrow Q_a(x)) \wedge \forall x. (x > z \Rightarrow Q_b(x)) \wedge \exists z'. z' > z)$$

The quantifier in φ_1 is relativized to the position $\leq z$ and the one in φ_2 to the position $> z$. In general, the *relativization* $\varphi^{\leq z}$ of φ to the position $\leq z$ is defined as follows:

- $(\psi_1 \wedge \psi_2)^{\leq z} = \psi_1^{\leq z} \wedge \psi_2^{\leq z}$
- $(\psi_1 \vee \psi_2)^{\leq z} = \psi_1^{\leq z} \vee \psi_2^{\leq z}$
- $(\neg\psi)^{\leq z} = \neg(\psi^{\leq z})$
- $(\exists x. \psi(x))^{\leq z} = \exists x. (x \leq z \wedge \psi(x)^{\leq z})$
- $(\forall x. \psi(x))^{\leq z} = \forall x. (x \leq z \Rightarrow \psi(x)^{\leq z})$

(where we assume that $\geq$ does not occur in the formula.)

The relativization $\varphi^{\geq z}$ is defined analogously.

Assume that $L = L_1 \cdot L_2 \subseteq \Sigma^+$ and that L_1 and L_2 are star–free. If $L_1, L_2 \in \Sigma^+$ then we know by induction that there are formulae φ_1 and φ_2 with $L_1 = L(\varphi_1), L_2 = L(\varphi_2)$. Then $L_1 \cdot L_2 = L\ (\exists z. ({\varphi_1}^{\leq z} \wedge {\varphi_2}^{\leq z} \wedge \exists z'. z < z'))$
If say L_2 contains ε, then $L_1 \cdot L_2 = L_1 \cdot (L_2 \setminus \{\varepsilon\}) \cup L_1$. ∎

3.3.2 Proof of "$2 \rightarrow 1$" of Theorem 3.9

To show that every closed formula of PL1 defines a star–free language we want to use induction over the *quantifier–depth* of the formula, i.e. the maximal nesting of quantifiers in the formula (every closed formula contains at least one quantifier.).
Since we have negation, we may assume that the formula contains only existential quantifiers.
Since the connectives $\wedge, \vee, \neg$ correspond to $\cap, \cup, ^-$ of languages, it is sufficient to consider formulae of the form $\exists x. \varphi(x)$

Induction base: quantifier–depth 1

Thus φ does not contain any quantifiers and since $\exists x. \varphi(x)$ is closed, φ does not contain variables different from x. We can assume without loss of generality that φ is a positive Boolean combination (only $\wedge, \vee$) of formulae

- $Q_a(x)$ or $\neg Q_a(x)$,
- $x < x$ or $\neg(x < x)$,
- $x = x$ oder $\neg(x = x)$.

We assume that φ is in disjunctive normal form $D_1 \vee \ldots \vee D_n$. Disjunct D_i is of the form $C_1 \wedge \ldots \wedge C_m$. This can be further normalized:

- replace $x < x$, $\neg(x = x)$ by false, i.e. remove any disjunct D_j containing such an expression.
- replace $\neg(x < x), x = x$ by true, i.e. remove it from the disjunct.
- if a disjunct contains $Q_a(x)$, then
  - remove the whole disjunct if it contains $\neg Q_a(x)$ or $Q_b(x)$ for $a \neq b$
  - otherwise remove from the disjunct all $\neg Q_b(x)$ for $b \neq a$

Thus, we end up with disjuncts of the form

- $Q_a(x)$,
- $\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_n}(x)$.

Since $(\exists x. D_1 \wedge \ldots \wedge D_n) \equiv \exists x. D_1 \ldots \exists x. D_n$, we can restrict the attention to

- $\exists x. Q_a(x)$
- $\exists x. (\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_n}(x))$

$$L(\exists x. Q_a(x)) = \Sigma^* a \Sigma^* \in SF_\Sigma$$
$$L(\exists x. (\neg Q_{a_1}(x) \wedge \ldots \wedge \neg Q_{a_n}(x))) = \Sigma^* \cdot (\Sigma \setminus \{a_1, \ldots, a_n\}) \cdot \Sigma^* \in SF_\Sigma$$

This completes quantifier–depth 1.

Induction step: $n \to n+1$

Let $\exists x. \varphi(x)$ be of quantifier–depth $n+1$. Before we can show that $L(\exists(.x)\varphi x)$ is star–free, we need some notation and two propositions.

Definition 3.10 With $L_{k,n}$ we denote the set of formulae of PL1 (over the extra logical symbols $=$, $<$, Q_a for $a \in \Sigma$) having k *free variables* and quantifier–depth $\leq n$

Example: $\varphi = \exists x. (y < x \wedge x < z \wedge Q_a(x))$ belongs to $L_{2,1}$. To interpret this formula, a word (e.g. baa) is not enough. We must also say how y and z are interpreted (e.g. y by 1 (first position in baa) and z by 3 (third position in baa)). We say that $(baa,\ 1, 3)$ satisfies φ.

In general, formulae from $L_{k,n}$ are interpreted by tuples $(w, \vec{s})$ where $w \in \Sigma^+$ and $\vec{s} = s_1 \cdots s_k$ with $s_i \in \{1, \ldots, |w|\}$. "$(w, \vec{s})$ satisfies $\varphi \in L_{k,n}$" $(w, \vec{s}) \models \varphi$ is defined in the obvious way. For $k = 0$, we dispense with the empty sequence $\vec{s}$.

Definition 3.11 For $n \geq 0$ and $k \geq 0$ we define

$$(w, \vec{s}) \equiv_{k,n} (v, \vec{t}) \quad \text{iff} \quad \text{for all } \varphi \in L_{k,n} \text{ we have } (w, \vec{s}) \models \varphi \text{ iff } (v, \vec{t}) \models \varphi.$$

Obviously $\equiv_{k,n}$ is an equivalence relation. For $k = 0$ we write $\equiv_n$ in place of $\equiv_{0,n}$. In this case, we extend $\equiv_n$ to Σ^* by defining $\varepsilon \equiv_n \varepsilon$, i.e. $\{\varepsilon\}$ is a $\equiv_n$–equivalence class.

To proof "$2 \rightarrow 1$" of Theorem 3.9, we need two propositions, which we will prove in separate subsections.

Proposition 3.12 For all $n \geq 0$ and $k \geq 0$ there is a *finite* set $\Gamma_{k,n} \subseteq L_{k,n}$ such that every element of $L_{k,n}$ is equivalent to some element of $\Gamma_{k,n}$.
Equivalent means that the formulae are satisfied by the same tuples $(w, \vec{s})$.
As a simple consequence of this proposition we obtain $\equiv_{k,n}$ has finite index.

Corollary 3.13 For all $k \geq 0$ and $n \geq 0$, $\equiv_{k,n}$ has only finitely many equivalence classes.

Proof: The class of $(w, \vec{s})$ is uniquely determined by the following subset of $\Gamma_{k,n}$:

$$\Gamma = \{\varphi \in \Gamma_{k,n} \mid (w, \vec{s}) \models \varphi\}$$

Since there are only finitely many such subsets of $\Gamma_{k,n}$, there are only finitely many equivalence classes. ∎

We can extend the notion of the language defined by a formula also to formulae with free variables: for $\varphi \in L_{k,n}$ we define $L(\varphi) := \{(w, \vec{s}) \mid (w, \vec{s}) \models \varphi\}$.

Corollary 3.14 For all $n \geq 0$ and $k \geq 0$ and all $\equiv_{k,n}$–classes W there is a formula $\varphi_W \in L_{k,n}$ such that $W = L(\varphi_W)$.

Proof:

$$W = [(w, \vec{s})]_{\equiv_{k,n}} = L\Big(\bigwedge_{\substack{\varphi \in \Gamma_{k,n} \\ (w,\vec{s}) \models \varphi}} \varphi \wedge \bigwedge_{\substack{\psi \in \Gamma_{k,n} \\ (w,\vec{s}) \not\models \psi}} \neg\psi \Big)$$

∎

Corollary 3.15 For all $n \geq 0$ and $k \geq 0$ and all $\varphi \in L_{k,n}$ the following holds: φ is equivalent to a finite disjunction of formulae φ_W for $\equiv_{k,n}$–classes W.

Proof:

$$\varphi \text{ is equivalent to } \bigvee_{\substack{W=[(w,\vec{s})]_{\equiv_{k,n}} \\ (w,\vec{s})\models\varphi}} \varphi_W.$$

This disjunction is finite since $\equiv_{k,n}$ has finite index. ∎

The second proposition shows a connection between $\equiv_{0,n}$ and $\equiv_{1,n}$.

Proposition 3.16 Let $n \geq 0$, $u, v, u', v' \in \Sigma^*$ and $a \in \Sigma$.

$$u \equiv_n u' \wedge v \equiv_n v' \Rightarrow (uav, |u|+1) \equiv_{1,n} (u'av', |u'|+1)$$

We are now ready to finish the induction step. Thus, let $\exists x. \varphi(x)$ be a closed formula of quantifier–depth $n+1$. Consequently, $\varphi(x) \in L_{1,n}$ and thus φ is a finite disjunction $\varphi(x) = \bigvee \varphi_W(x)$. Consequently, $\exists x. \varphi(x)$ is equivalent to $\bigvee \exists x. \varphi_W(x)$. Thus is enough to show $L(\exists x. \varphi_W(x))$ are star–free.

Lemma 3.17

$$L(\exists x. \varphi_W(x)) = \bigcup_{\substack{U=[u_0]_{\equiv_{0,n}},\ V=[v_0]_{\equiv_{0,n}} \\ a\in\Sigma \ \text{mit}(u_0av_0,|u_0|+1)\in W}} UaV$$

Proof:

"$\subseteq$"

$$\begin{aligned} w \in L(\exists x. \varphi_W(x)) \quad & \text{iff} \quad \text{there exist } u, v \in \Sigma^* \text{ such that } w = uav \\ & \qquad\ \ \text{and } (uav, |u|+1) \models \varphi_W(x) \\ & \text{iff} \quad \text{there exist } u, v \in \Sigma^*, a \in \Sigma \text{ such that } w = uav \\ & \qquad\ \ \text{and } (uav, |u|+1) \in W \end{aligned}$$

Consequently $w \in [u]a[v]$ and $[u]a[v]$ occurs in the union on the right–hand side.

"$\supseteq$" From $(u_0av_0, |u_0|+1) \in W$ it follows that $(u_0av_0,\ |u_0|+1) \models \varphi_W(x)$, and thus $u_0av_0 \in L(\exists x.\varphi_W(x))$.

It remains to show that for all $u \equiv_{1,n} u_0$ and $v \equiv_{1,n} v_0$ we also have $uav \in L(\exists x.\varphi_W(x))$.

With Prop. 3.16 we have $(u_0av_0, |u_0|+1) \equiv_{1,n} (uav, |u|+1)$ and thus $(uav, |u|+1) \in W$. As above, thus implies $uav \models \exists x.\varphi_W(x)$. ∎

By Corollary 3.14, the $\equiv_{0,n}$–classes U, V of the lemma are of the form $U = L(\varphi_U)$, $V = L(\varphi_V)$ for $\varphi_U, \varphi_V \in L_{0,n}$ or $U = \{\varepsilon\}$ or $V = \{\varepsilon\}$. So the induction hypothesis yields that $L(\varphi_V)$ and $L(\varphi_U)$ are also star–free: Since star–free languages are closed under union and concatenation, $L(\exists x.\varphi_W(x))$ is star–free. ∎

3.3.3 Proof of Propositions 3.12 and 3.16

Proposition 3.12 For all $n \geq 0$ and $k \geq 0$ there is a *finite* set $\Gamma_{k,n} \subseteq L_{k,n}$ such that every element of $L_{k,n}$ is equivalent to some element of $\Gamma_{k,n}$.

Proof: We prove this proposition by induction on n.
Let $\varphi \in L_{k,n}$. Such a formula is a Boolean combination of

- elements of $L_{k,n-1}$
- formulae of the form $\exists x.\psi(x, y_1, \ldots, y_k)$ where $\psi(x, y_1, \ldots, y_k) \in L_{k+1,n-1}$

Induction base $n = 0$**:** Let $k \geq 0$ be arbitrary. A formula $\varphi(y_1, \ldots, y_k) \in L_{k,0}$ is a Boolean combination of atomic formulae.

$$(*)\ Q_a(x) \text{ for } a \in \Sigma,\ x < y,\ x = y \text{ where } x, y \in \{y_1, \ldots, y_k\}.$$

Without loss of generality, we consider only elements of $L_{k,0}$ containing free varibales from $\{y_1, \ldots, y_k\}$. But then there are only finitely many formulae of the form $(*)$. Thus, there are only finitely many Boolean combinations of these formulae (up to equivalence).

Induction step $(n-1 \to n)$**:** Assume that we already have finite sets $\Gamma_{k,n-1}$ and $\Gamma_{k+1,n-1}$ with the desired properties. Thus φ is equivalent to a Boolean combination of elements of $\Gamma_{k,n-1}$ and formula $\exists x.\psi$ with $\psi \in \Gamma_{k+1,n-1}$. Thus, up to equivalence there are finitely many such Boolean combinations. ∎

Our proof of Prop. 3.16 will use a game theoretic characterization of the relations $\equiv_{k,n}$.

Definition 3.18 [Ehrenfeucht–Fraissé games] Let Σ be a finite alphabet. We consider two players I und II, who play on words $u, v \in \Sigma^+$. A *move* chooses a position in either u or v. Player I has the first move and then there are alternating moves from II and I. If I moves in u then II must answer in v, and if I moves in v then II must answer in u.

A game of length n consists of n moves of I and of the n answer moves of II.

Let $(i_1, j_1), \dots, (i_n, j_n)$ be the chosen positions, where i_μ is the position in u and j_μ the position in v (independent on which player has chosen them).

Player II has won this game iff the following holds:
Let $u = a_1 \dots a_p$ and $v = b_1 \dots b_q$. Then:

- $a_{i_\mu} = b_{j_\mu}$ for $\mu = 1, \dots, n$, i.e. at the positions chosen in move μ we have the same letter.
- $i_\mu < i_{\mu'}$ iff $j_\mu < j_{\mu'}$, i.e. the relative placement of the positions must be the same.

Otherwise, II has lost and I has won.

Example 3.19 $n = 3$
Let $\Sigma = \{a\}$. Thus only the order of the chosen positions is relevant.

	1	2	3	4	5	6	7
$u =$	a	a	a	a	a	a	
		I,2		II,1			
$v =$	a	a	a	a	a	a	a
		II,2		I,1			

II, 1 on 5 or 6 would let I win the game by placing first 6 in v and then 7 in v. II, 2 could have also chosen 3 in v instead of 2

Whereever I moves in the third move, II can answer appropriately and thus II wins.

Example 3.20 $n = 3,\ \Sigma = \{a, b\}$

	1	2	3	4
$u =$	b	b	a	b
	I,3	I,2	I,1	
$v =$	b	a	b	b
	II,2	II,1		

II, 1 must choose the a in v and II,2 must choose the b to the left of a in v. II cannot answer the third move of I appropriately.

Definition 3.21 Let $n \geq 1$ and $u, v \in \Sigma^+$. We say that II has a *winning strategy* for games of length n on u, v iff II can answer all possible moves of I such that II wins.

To be able to use induction arguments, we will also consider games, that have already started.

Let $u, v \in \Sigma^+$ and $\vec{s} = s_1 \dots s_k \in \{1, \dots, |u|\}^k$ and $\vec{t} = t_1 \dots t_k \in \{1, \dots, |v|\}^k$. Then the pair $(u, \vec{s})$ and $(v, \vec{t})$ describes a game where already k moves have been made by each player. A game of length n on this pair is a continuation of this game by n moves. Player II wins this continuation game iff II wins the whole game.

Definition 3.22 Let $(u, \vec{s})$ and $(v, \vec{t})$ be given where $u, v \in \Sigma^+$ and $\vec{s} \in \{1, \dots, |u|\}^k$, $\vec{t} \in \{1, \dots, |v|\}^k$.

$(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ iff II has a winning strategy for games of length n on $(u, \vec{s})$ and $(v, \vec{t})$.

For $k = 0$ we extend $\sim_{0,n}$ to Σ^* by making $\{\epsilon\}$ a $\sim_{0,n}$–class.

Lemma 3.23 $\sim_{k,n}$ is an equivalence relation.

Proof:

reflexive: game played on $(u, \vec{s})$ and $(u, \vec{s})$. II just simulates the moves of I.

symmetric: clear since I can move in both words.

transitive: Let WS1 be the strategy that shows $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ and WS2 be the strategy that shows $(v, \vec{t}) \sim_{k,n} (w, \vec{r})$. The winning strategy for II on $(u, \vec{s})$ and $(w, \vec{r})$ works as follows:

- if I moves in u, then II answers first with WS1 in v, and then to this move with WS2 in w.
- if I moves in w, then II answers with WS2 in v, and to this move with WS1 in u ∎

We will show $\sim_{k,n} = \equiv_{k,n}$.

For $\sim_{k,n}$, the corresponding statement of Prop. 3.16 can easily be proved:

Lemma 3.24 Let $n \geq 0$, $u, v, u', v' \in \Sigma^*$ and $a \in \Sigma$.

$$u \sim_{0,n} u' \wedge v \sim_{0,n} v' \Rightarrow (uav, |u| + 1) \sim_{1,n} (u'av', |u'| + 1)$$

Proof: We consider the case where $u, v, u', v' \in \Sigma^+$. (The other cases can be treated analogously.)

Let WS1 be the startegy that yields $u \sim_{0,n} u'$ and WS2 the strategy that yields $v \sim_{0,n} v'$.
If I moves in u (u'), then II answers with WS1 in u' (u).
If I moves in v (v'), then II answers with WS2 in v' (v).
If I moves $|u| + 1$ in uav ($|u'| + 1$ in $u'av'$), then II answers $u' + 1$ in $u'av'$ ($|u| + 1$ in uav).
Obviously, this yields a winning strategy for II on $(uav, |u|+1)$ and $(u'av', |u'|+1)$ ∎

To prove Proposition 3.16, it is enough to show that $\sim_{k,n}$ and $\equiv_{k,n}$ coincide.

First, an intuitive argument. The chances for II to win are the greater the more similar the tuples are. If they are in the relation $\equiv_{k,n}$ they cannot be distinguished by formulae of quantifier–depth n, and are thus similar.

The connection between quantifier–depth and the number of moves is also quite clear: $\exists x.\varphi(x)$ says that there exists a position with certain porperties. A move picks a position.

Lemma 3.25 For all $n, k \geq 0$ we have

$$\sim_{k,n} = \equiv_{k,n}$$

i.e. $(u, \vec{s}) \sim_{k,n} (v, \vec{t})$ iff $(u, \vec{s}) \equiv_{k,n} (v, \vec{t})$.

Proof: by induction on n
Induction base $n = 0$: Let $(u, \vec{s})$ and $(v, \vec{t})$ be given.

1. Assume that $(u, \vec{s}) \equiv_{k,0} (v, \vec{t})$. Assume that the free variables are from the set $\{y_1, \ldots, y_k\}$ where y_i corresponds to the i–th component in $\vec{s}$ and $\vec{t}$. Let $u = a_1 \cdots a_p$, $v = b_1 \cdots b_q$. For the atomic formulae $y_i < y_j$, $y_i = y_j$, $Q_a(y_i)$, the equivalence $(u, \vec{s}) \equiv_{k,0} (v, \vec{t})$ implies that $(u, \vec{s})$ and $(v, \vec{t})$ behave the same on these formulae:

$$(*) \quad \begin{array}{lllllll} a) & (u, \vec{s}) & \models & y_i < y_j & \text{iff} & (v, \vec{t}) & \models & y_i < y_j \\ b) & (u, \vec{s}) & \models & y_i = y_j & \text{iff} & (v, \vec{t}) & \models & y_i = y_j \\ c) & (u, \vec{s}) & \models & Q_a(y_i) & \text{iff} & (v, \vec{t}) & \models & Q_a(y_i) \end{array}$$

This is equivalent to saying

$$(**) \quad \begin{array}{llll} a) & s_i < s_j & \text{iff} & t_i < t_j \\ b) & s_i = s_j & \text{iff} & t_i = t_j \\ c) & a_{s_i} = a & \text{iff} & b_{t_i} = a \end{array}$$

However, a) and c) of $(**)$ is exactly the condition that II has won the game (without additional moves since $n = 0$). And thus $(u, \vec{s}) \sim_{k,0} (u, \vec{t})$.

2. Conversely, assume that $(u, \vec{s}) \sim_{k,0} (v, \vec{t})$. Consequently II has won the game (without additional moves). This shows that a) and c) of $(**)$ hold. Condition b) follows from a) since $<$ is a total ordering. This yields a) b) c) of $(*)$, and thus $(u, \vec{s})$ and $(v, \vec{t})$ behave the same on atomic formulae. Thus, they behave the same on their Boolean combinations, which are all the elements of $L_{k,0}$. This shows $(u, \vec{s}) =_{k,0} (v, \vec{t})$.

Induction step $n \to n + 1$:

1. Assume that $(u, \vec{s}) \sim_{k,n+1} (v, \vec{t})$. Let $\varphi(y_1, \ldots, y_k) \in L_{k,n+1}$ and assume that $(u, \vec{s}) \models \varphi$. It is sufficient to show that this implies $(v, \vec{t}) \models \varphi$.

   Since Boolean operators are unproblematic, it is enough to assume that φ is of the form $\varphi = \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1})$ where $\psi \in L_{k+1,n}$.

   Since $(u, \vec{s}) \sim_{k,n+1} (v, \vec{t})$, II can answer appropriately the first move of I. Thus, for every $s_{k+1} \in \{1, \ldots, |u|\}$ there is a $t_{k+1} \in \{1, \ldots, |v|\}$ such that
   $$(u, \vec{s}s_{k+1}) \sim_{k+1,n} (v, \vec{t}t_{k+1}).$$
   Induction yields that for all s_{k+1} there exists a t_{k+1} such that
   $$(u, \vec{s}s_{k+1}) \equiv_{k+1,n} (v, \vec{t}t_{k+1}) \qquad (*)$$
   Since $(u, \vec{s}) \models \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1})$ there is an s_{k+1} with
   $$(u, \vec{s}s_{k+1}) \models \psi(y_1, \ldots, y_k, y_{k+1}) \qquad (**)$$
   Let t_{k+1} be such that $(*)$ holds. Since $\psi \in L_{k+1,n}$, $(*)$ and $(**)$ imply
   $$(v, \vec{t}t_{k+1}) \models \psi(y_1, \ldots, y_k, y_{k+1}).$$
   This shows that
   $$(v, \vec{t}) \models \exists y_{k+1}.\psi(y_1, \ldots, y_k, y_{k+1}).$$

2. Assume that $(u, \vec{s}) \not\sim_{k,n+1} (v, \vec{t})$. We are looking for a formula $\varphi \in L_{k,n+1}$ such that φ is satisfied by one of the two tuples $(u, \vec{s})$ and $(v, \vec{t})$, but not by the other one.

Since $(u, \vec{s}) \not\sim_{k,n+1} (v, \vec{t})$, there is a first move of I that II cannot answer appropriatly (i.e. after the first move II still does not have a winning strategy). Without loss of generality we assume that this first move is in u. Thus there is an s_{k+1} such that for all t_{k+1} we have

$$(u, \vec{s}s_{k+1}) \not\sim_{k+1,n} (v, \vec{t}t_{k+1}).$$

Induction yields:

$$(u, \vec{s}s_{k+1}) \not\equiv_{k+1,n} (v, \vec{t}t_{k+1}).$$

We now fix this s_{k+1}. For all t_{k+1} there is thus a formula $\psi_{t_{k+1}}(y_1, \ldots, y_{k+1}) \in L_{k+1,n}$ such that

$$(u, \vec{s}s_{k+1}) \models \psi_{t_{k+1}} \tag{$*$}$$

$$(v, \vec{t}t_{k+1}) \not\models \psi_{t_{k+1}} \tag{$**$}$$

From $(*)$ it follows that $\varphi_{t_{k+1}} := \exists y_{k+1}.\psi_{t_{k+1}}$ satisfies $(u, \vec{s}) \models \varphi_{t_{k+1}}$. Unfortunately, $(**)$ does *not* imply $(v, \vec{t}) \not\models \varphi_{t_{k+1}}$. The reason is that there may be a $t \neq t_{k+1}$ such that $(v, \vec{t}t) \models \psi_{t_{k+1}}$.

Instead of $\varphi_{t_{k+1}}$ we consider the formula

$$\varphi := \exists y_{k+1} \bigwedge_{1 \leq t_{k+1} \leq |v|} \psi_{t_{k+1}} \tag{3.1}$$

Because of $(*)$ we know that $(u, \vec{s}) \models \varphi$. Because of $(**)$, for every t_{k+1} there is one conjunct in φ that is not satisfied if we substitute y_{k+1} by t_{k+1}. Thus $(v, \vec{t}) \not\models \varphi$. ∎

Note: Lemma 3.25 also holds for infinite words. To obtain a finite conjunction in the definition of φ in (3.1), one uses the fact that every formula in $L_{k+1,n}$ is equivalent to a formula in the finite set $\Gamma_{k+1,n}$.

This finishes the proof of Theorem 3.16. Since not all regular languages are star–free (example $a(aa)^*$) there are regular languages that cannot be defined by formulae of PL1. To obtain all regular languages, we must add quantification over unary predicates (see Example 1.28). We will not show this here, but later on we will show it for the case of infinite words.

Simple application: a decidability result in logic
Theory of linear orderings

$$\mathcal{L}in \quad = \quad \left\{ \begin{array}{ll} \left.\begin{array}{l} \forall x \forall y \forall z (x < y \land y < z \to x < z) \\ \forall x . \neg (x < x) \end{array}\right\} & \text{strict partial order} \\ \forall x . \forall y . (x < y \lor x = y \lor y < x) \;\} & \text{linear} \end{array}\right.$$

Proposition 3.26 Let φ be a closed formula of PL1 that contains only the extra–logical symbols $<, =, P_1, \ldots P_n$ (unary). Then it is decidable whether $\mathcal{L}in \cup \{\varphi\}$ has a *finite* model.

Proof: We know that $L(\varphi)$ is a star–free language. The proof of Theorem 3.9 is constructive, i.e., given a formula φ, we can effectively construct a star–free expression (using finite languages, Boolean operations, and concatenation) for $L(\varphi)$. In principle, this is due to the fact that the finite set $\Gamma_{k,n}$ from Prop 3.12 can effectively be constructed (and thus the $\equiv_{k,n}$–classes, the formula $\varphi_W, \ldots$)

It is easy to see that $L(\varphi) \neq \emptyset$ iff $\mathcal{L}in \cup \{\varphi\}$ has a finite model. The star–free expression for $L(\varphi)$ can be transfered into a regular expression for $L(\varphi)$ (closure of $\mathcal{R}eg_\Sigma$ under $\cap, ^-$). For regular expressions, the emptiness problem is decidable. ∎

Sometimes, one would like to consider also infinite models of $\mathcal{L}in$. This is one motivation for also considering infinite words.

Chapter 4

Infinite words and Büchi–automata

Before introducing infinite words formally, let us reconsider finite words. Let Σ be an alphabet. A finite word $u \in \Sigma^+$ is a sequence $u = a_0 \cdots a_{k-1}$ of k elements $a_i \in \Sigma$. One can view u as a mapping

$$u : \{0, \ldots, k-1\} \to \Sigma : \quad i \mapsto a_i$$

Thus finite words are mappings from an initial segment of the natural numbers into the alphabet. Infinite words are mappings from the set of all natural numbers into the alphabet. Since we are only interested in the ordering of natural numbers (and not in the arithmetic operations), we denote the natural numbers by ω (omega) (ω is the order type of the natural numbers).

Definition 4.1

1. An *infinite word* over Σ is a mapping $\alpha : \omega \to \Sigma$.
2. Σ^ω denotes the set of all infinite words over Σ.
3. Subsets of Σ^ω (i.e. sets of infinite words) are called *?w-langauges@ω–languages*

We will often write infinite words as $\alpha = a_0 a_1 a_2 a_3 \cdots$ where $a_i = \alpha(i)$.

Example 4.2 $\Sigma = \{a, b\}$

$$\alpha(i) = \begin{cases} a & \text{if } i \text{ is even} \\ b & \text{if } i \text{ odd} \end{cases}$$

α can be written as $\alpha = ababab\cdots$.

Some operations on infinite words and ω–languages:

Segment: if $\alpha : \omega \to \Sigma$ is an infinite word then

- $\alpha(m, n)$ is the finite word $\alpha(m)\cdots\alpha(n)$
- $\alpha(m, \omega)$ is the infinite word $\alpha(m)\alpha(m+1)\alpha(m+2)\cdots$.

Concatenation: let $w = a_1 \cdots a_m$ be a finite word and $\alpha = \alpha(0)\alpha(1)\cdots$ an infinite word. Then $w \cdot \alpha$ is the infinite word

$$a_1 \cdots a_m \alpha(0)\alpha(1)\alpha(2)\cdots .$$

Note: It does note make sense to concatenate two infinite words.
As usual, concatenation can be extended from words to sets of words

Infinite iteration: let $L \subseteq \Sigma^*$ be a set of finite words

$$L^\omega := \{\alpha \in \Sigma^\omega \mid \alpha = w_0 w_1 w_2 \cdots \text{ for words } w_i \in L \setminus \{\epsilon\}\}$$

Example: $L = \{ab\}$

$$L^\omega = \{abababab\cdots\}. \text{ We often write } L^\omega = (ab)^\omega.$$

Limit: let $L \subseteq \Sigma^*$ be a set of finite words.

$$\lim L = \{\alpha \in \Sigma^\omega \mid \text{ there are infinitely many } n \text{ that } \alpha(0, n) \in L\}.$$

Example 4.3 $\Sigma = \{a, b\}$.

1. $L = a^*b$: $\lim L = \emptyset$ since any infinite word can have at most one initial segment in a^*b.
2. $L = ba^*$: $\lim L = \{baaa\cdots\} = ba^\omega$
3. $L = (a^*bb^*)^*$: $\lim L = \{\alpha \in \Sigma^\omega \mid$ after each occurrence of a in α there eventually follows an occurrence of $b\}$

4.1 Büchi–automata and ω–regular languages

Automata working on infinite words are defined like the "usual" finite automata. The distinction comes in when defining the acceptance condition.

Definition 4.4 A *Büchi–automaton* is a (non–det.) finite automaton $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$, i.e.,

- Q is a finite sets of states
- Σ is a finite alphabet
- $I \subseteq Q$ is the set of initial states
- $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation
- $F \subseteq Q$ is the set of final states

Since we are interested in infinite words, we consider *infinite paths* $q_0 \xrightarrow{a_1}_{\mathcal{A}} q_1 \xrightarrow{a_2}_{\mathcal{A}} q_2 \xrightarrow{a_3}_{\mathcal{A}} q_3 \xrightarrow{a_4}_{\mathcal{A}} q_4 \cdots$ where $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. The label of this infinite path is the infinite word $a_1 a_2 a_3 a_4 \cdots$.

For finite paths, we said that they are successful if

i) $q_0 \in I$

ii) $q_n \in F$

For infinite paths, there is no such final state q_n. Instead, we require that final states are reached infinitely often.
The infinite path $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \ldots$ is called *successful* iff

i) $q_0 \in I$

ii) There are *infinitely* many i such that $q_i \in F$

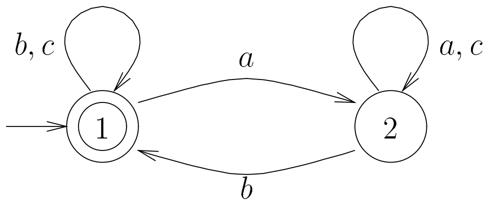
The Büchi–automaton $\mathcal{A}$ *accepts the ω–language*

$$L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ is the label of a successful path in } \mathcal{A}\}.$$

Such an ω–language is called *Büchi–recognizable*.

Example 4.5 $\Sigma = \{a, b, c\}$.

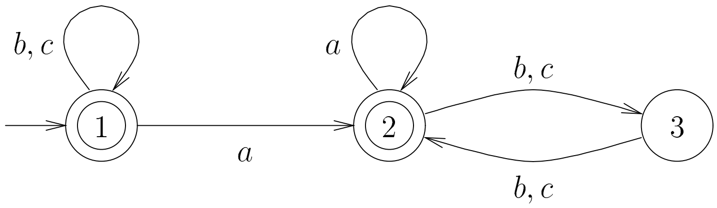
1. $\mathcal{A}_1$:



$$L_\omega(\mathcal{A}_1) = \{\alpha \in \Sigma^\omega \mid \text{after every } a \text{ there eventually is } b\}$$

The letter a leads to state ②. From ② the accepting state ① ist only reached through b.

2. $\mathcal{A}_2$:



$$L_\omega(\mathcal{A}_2) = \{\alpha \in \Sigma^\omega \mid \text{between two consecutive } a\text{'s there is an even number of } b\text{'s and } c\text{'s}\}$$

It is easy to see that $L_\omega(\mathcal{A}_2) = L_2$.

To investigate the languages accepted by Büchi–automata more closely, we introduce some notation.

Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi–automaton and $p, q \in Q$. We view $\mathcal{A}$ as a finite automaton where p is the initial and q is the final state. This automaton accepts the language

$$L_{p,q} := \{w \in \Sigma^* \mid w \text{ is the label of some finite path from } p \text{ to } q \text{ in } \mathcal{A}\}$$

Thus, $L_{p,q}$ are regular languages.

Lemma 4.6
$$L_\omega(\mathcal{A}) = \bigcup_{i \in I, f \in F} L_{i,f} \cdot L_{f,f}^\omega.$$

Proof:

"$\subseteq$" Let $\alpha = a_1a_2a_3a_4 \cdots \in L_\omega(\mathcal{A})$. By definition of $L_\omega(\mathcal{A})$ there is a path $I \ni q_0 \xrightarrow{a_1}_{\mathcal{A}} q_1 \xrightarrow{a_2}_{\mathcal{A}} q_2 \xrightarrow{a_3}_{\mathcal{A}} q_3 \xrightarrow{a_4}_{\mathcal{A}} \cdots$ such that infinitely often $q_i \in F$. Since F is *finite*, there is a single state $f \in F$ such that there are infinitely many indices $i_1 < i_2 < i_3 < \cdots$ with $q_{i_\nu} = f$. Thus, we have $a_1 \cdots a_{i_1} \in L_{q_0,f}$ and $a_{i_v+1} \cdots a_{i_{v+1}} \in L_{f,f} \setminus \{\varepsilon\}$ $(\nu \geq 1)$. This shows that $\alpha \in L_{q_0,f} \cdot L_{f,f}^\omega$ where $q_0 \in I$ and $f \in F$.

"$\supseteq$" Let $\alpha = w_0w_1w_2w_3 \cdots$ where $w_0 \in L_{i,f}$ and $w_i \in L_{f,f} \setminus \{\varepsilon\}$ $(i \geq 1)$. Thus, the path $i \xrightarrow{w_0}_{\mathcal{A}} f \xrightarrow{w_1}_{\mathcal{A}} f \xrightarrow{w_2}_{\mathcal{A}} f \xrightarrow{w_3}_{\mathcal{A}} \cdots$ is a successful path, which shows that $\alpha = w_0w_1w_2w_3 \cdots \in L_\omega(\mathcal{A})$. ∎

The next lemma states simple closure properties of Büchi–recognizable languages.

Lemma 4.7

1. If $U \subseteq \Sigma^*$ is regular, then U^ω is Büchi–recognizable.
2. If $U \subseteq \Sigma^*$ is regular and $L \subseteq \Sigma^\omega$ is Büchi–recognizable, then $U \cdot L$ is Büchi–recognizable.
3. If $L_1, L_2 \subseteq \Sigma^\omega$ are Büchi–recognizable, the so are $L_1 \cup L_2$ and $L_1 \cap L_2$.

Proof:

1. Recall that $U^\omega = \{\alpha \in \Sigma^\omega \mid \alpha = u_1u_2u_3 \cdots \text{ with } u_i \in U \setminus \{\varepsilon\}\}$. If U is regular, then $U \setminus \{\epsilon\}$ is also regular. It is easy to see that there is an automaton $\mathcal{A}$ for $U \setminus \{\varepsilon\}$ that satisfies the following:
   - $\mathcal{A} = (\Sigma, Q, \{q_0\}, \Delta, F)$, i.e., $\mathcal{A}$ has a single initial state.
   - for all $a \in \Sigma$, $q \in Q$ we have $(q, a, q_0) \notin \Delta$, i.e., q_0 cannot be reached.

Let $\mathcal{A}$ be such an automaton for $U \setminus \{\varepsilon\}$.
We define $\mathcal{A}' := (\Sigma, Q, \{q_0\}, \Delta', \{q_0\})$ where

$$\Delta' \;=\; \Delta \cup \{(q, a, q_0) \mid \exists f \in F: \; (q, a, f) \in \Delta\}.$$

It is easy to show (!) that $L_\omega(\mathcal{A}') = U^\omega$.

Note: the condition that q_0 is not reachable is necessary (!).

2. Let $\mathcal{A} = (Q_1, \Sigma, I_1, \Delta_1, F_1)$ be a finite automaton for U and $\mathcal{B} = (Q_2, \Sigma, I_2, \Delta_2, F_2)$ be a Büchi–automaton for L. We may assume that $Q_1 \cap Q_2 = \emptyset$.

$$\begin{array}{rcl} \mathcal{C} & = & (Q_1 \cup Q_2, \Sigma, I', \Delta', F_2) \text{ where} \\ I' & = & I_1 \cup \left\{ \begin{array}{ll} \emptyset & \text{if} \;\; I_1 \cap F_1 = \emptyset \\ I_2 & \text{if} \;\; I_1 \cap F_1 \neq \emptyset \end{array} \right. \\ \Delta' & = & \Delta_1 \cup \Delta_2 \cup \{(q, a, q') \mid \exists f \in F_1 \\ & & \bullet\ (q, a, f) \in \Delta_1 \\ & & \bullet\ q' \in I_2\} \end{array}$$

It is easy to show that $L_\omega(\mathcal{C}) = U \cdot L$.

3. **Union:** Exercise!

Intersection Let $\mathcal{A}_i = (Q_i, \Sigma, I_i, \Delta_i, F_i)$ be a Büchi–automaton for L_i $(i \in \{1, 2\})$. We define

$$\mathcal{B} := (Q_1 \times Q_2 \times \{0, 1, 2\}, \Sigma, I_1 \times I_2 \times \{0\}, \Delta, F)$$

where

$$\begin{array}{l} \Delta := \{((q_1, q_2, i), a, (q_1', q_2', j)) \;\mid \\ \quad - \; (q_1, a, q_1') \in \Delta_1 \text{ and } (q_2, a, q_2') \in \Delta_2 \\ \quad - \; \begin{array}{lcl} i = 0 \wedge q_1' \in F_1 & \Rightarrow & j = 1 \\ i = 1 \wedge q_2' \in F_2 & \Rightarrow & j = 2 \\ i = 2 & \Rightarrow & j = 0 \\ \text{otherwise,} & & i = j\} \end{array} \end{array}$$

This means the following: we start with 0 in the 3^{rd} component. If we reach for the first time some $f_1 \in F_1$, then the third component becomes 1. If after that we reach for the first time some $f_2 \in F_2$, then the third component becomes 2 and immediately after that 0. If we reach infinitely often elements of F_1 and infinitely often elements of F_2, then we go through this round infinitely often. Thus, we have infinitely often in the third component 2. Thus we must define:

$$F := Q_1 \times Q_2 \times \{2\}$$

■

Proposition 4.8 [Büchi, 1962]

1. An ω–language $L \subseteq \Sigma^\omega$ is Büchi–recognizable iff there are regular languages $U_1, \ldots, U_m, V_1, \ldots, V_m \subseteq \Sigma^*$ such that

$$L = \bigcup_{i=1}^{m} U_i \cdot V_i^\omega.$$

2. We can assume without loss of generality that $V_i \cdot V_i \subseteq V_i$.

Proof:

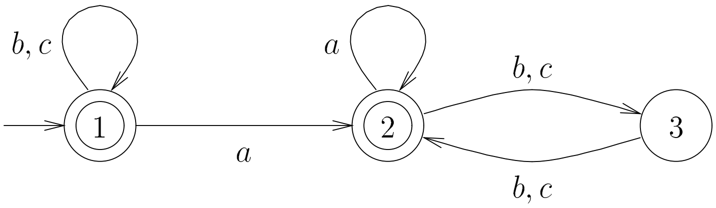
"$\Rightarrow$" of 1) as well as 2) follows from Lemma 4.6:

$$L = \bigcup_{i \in I, f \in F} L_{i,f} \cdot L_{f,f}^\omega \text{ and } L_{f,f} * L_{f,f} \subseteq L_{f,f}.$$

"$\Leftarrow$" of 1) is an immediate consequence of Lemma 4.7. ■

Because of this close connection to regular languages, Büchi–recognizable languages are called *ω–regular*. If U_i, V_i are given by regular expressions and if $L = \bigcup_{i=1}^m U_i \cdot V_i^\omega$, then we call $\bigcup_{i=1}^m U_i \cdot V_i^\omega$ an *ω–regular expression* for L.

Example 4.9 : consider the automaton $\mathcal{A}_2$ from Example 4.5:



$$\begin{aligned}
L_{1,1} &= (b \cup c)^*, \\
L_{1,2} &= (b \cup c)^* \cdot a \cdot L_{2,2}, \\
L_{2,2} &= (a \cup ((b \cup c) \cdot (b \cup c)))^*,
\end{aligned}$$

$$\begin{aligned}
L_\omega(\mathcal{A}_2) &= L_{1,1} \cdot L_{1,1}^\omega \cup L_{1,2} \cdot L_{2,2}^\omega \\
&= L_{1,1}^\omega \cup L_{1,2} \cdot L_{2,2}^\omega \\
&= ((b \cup c)^*)^\omega \cup (b \cup c)^* \cdot a \cdot L_{2,2} \cdot L_{2,2}^\omega \\
& (U^*)^\omega = U^\omega \\
& U \cdot U^\omega = U^\omega \\
&= (b \cup c)^\omega \cup (b \cup c)^* \cdot a \cdot (a \cup (b \cup c) \cdot (b \cup c))^\omega.
\end{aligned}$$

The characterization of Büchi–recognizable languages also shows that the emptiness problem is decidable.

Proposition 4.10

1. Given a Büchi–automaton $\mathcal{A}$, we can effectively decide whether $L_\omega(\mathcal{A}) = \emptyset$ or not.
2. If $L_\omega(\mathcal{A}) \neq \emptyset$, then $L_\omega(\mathcal{A})$ contains an ultimately periodic word, i.e. a word of the form $uvvvv\cdots$ for $u \in \Sigma^*, v \in \Sigma^+$.

Proof:

1. Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$. Then Lemma 4.6 says that $L_\omega(\mathcal{A}) = \bigcup_{i \in I, f \in F} L_{i,f} \cdot L_{f,f}$. We know that, $L_\omega(\mathcal{A}) \neq \emptyset$ iff there is an $i \in I, f \in F$ such that $L_{i,f} \neq \emptyset$ and $L_{f,f} \setminus \{\varepsilon\} \neq \emptyset$. There are finitely many such pairs i, f, and for each pair $L_{i,f}$ and $L_{f,f} \setminus \{\varepsilon\}$ are regular. The emptiness problem for regular languages is decidable. This shows 1.

2. If $L_\omega(\mathcal{A}) \neq \emptyset$ then there is an $i \in I, f \in F, u \in \Sigma^*$ and $v \in \Sigma^+$ such that $u \in L_{i,f}$ and $v \in L_{f,f} \setminus \{\varepsilon\}$. But then $uvvvv \cdots \in L_{i,f} \cdot L_{f,f}^\omega \subseteq L_\omega(\mathcal{A})$.

■

What about the equivalence problem "$L_1 = L_2$"?

For regular languages, decidability of the equivalence problem follows from the decidability of the emptiness problem since the class of regular languages is closed under $\cup, \cap, ^-$:

$$L_1 = L_2 \text{ iff } (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

For ω–regular languages, we still must show closure under $^-$.

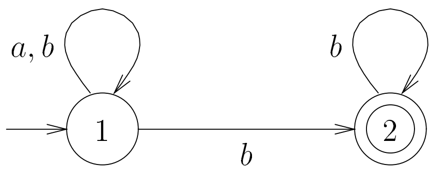
How can one show closure under complement for regular languages?

1. Make the finite automaton deterministic (power set construction)

2. In the deterministic automaton, exchange final states with non–final states.

For ω–languages, neither 1. nor 2. works.

Example 4.11 Let $\Sigma = \{a, b\}$. The non–deterministic Büchi–automaton shown below accepts the ω–regular language $L = (a \cup b)^* b^\omega$.



This language cannot be accepted by a deterministic Büchi–automaton.

Proof: Assume that $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ is a deterministic Büchi–automaton for L. Deterministic means: $\delta : Q \times \Sigma \to Q$ is a function. Since $ab^\omega \in L$, there is a $k_1 > 0$ and a $f_1 \in F$ such that $q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1$. Since $ab^{k_1}ab^{\omega} \in L$ and $\mathcal{A}$ is deterministic, there is a $k_2 > 0$ und $f_2 \in F$ such that

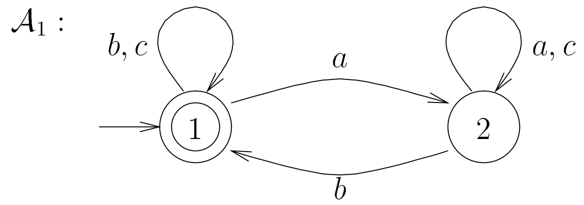
$$q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1 \xrightarrow{ab^{k_2}}_{\mathcal{A}} f_2.$$

Note that this is only true since $\mathcal{A}$ is deterministic!
By iterating this argument, we obtain $k_1, k_2, k_3, \ldots > 0$ and $f_1, f_2, f_3, \ldots \in F$ such that

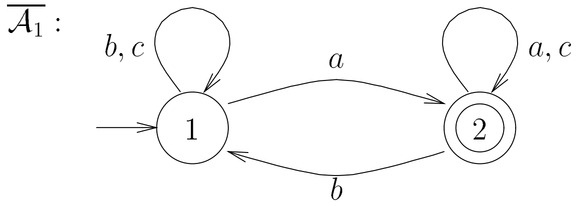
$$q_0 \xrightarrow{ab^{k_1}}_{\mathcal{A}} f_1 \xrightarrow{ab^{k_2}}_{\mathcal{A}} f_2 \xrightarrow{ab^{k_3}}_{\mathcal{A}} f_3 \xrightarrow{ab^{k_4}}_{\mathcal{A}} f_4 \cdots .$$

Thus, $\alpha = ab^{k_1}ab^{k_2}ab^{k_3}ab^{k_4} \cdots \in L_\omega(\mathcal{A})$. However $\alpha \notin L$ since it contains infinitely many a's. ∎

Example 4.12 Even for deterministic Büchi–automaton, exchanging final states states with non–final states does not work. Reconsider the automaton $\mathcal{A}_1$ of Example 4.5.



$L_\omega(\mathcal{A}_1) = \{\alpha \in \{a, b, c\} \mid \text{after each } a \text{ in } \alpha \text{ there eventually is } b \text{ in } \alpha\}$.



We have $(ab)^\omega \in L_\omega(\mathcal{A}_1) \cap L_\omega(\overline{\mathcal{A}_1})$, and thus $L_\omega(\overline{\mathcal{A}_1}) \neq \overline{L_\omega(\mathcal{A}_1)}$

Example 4.11 shows that the class of languages accepted by deterministic Büchi–automata is strictly smaller then the class of ω–regular languages. The next proposition characterizes this class.

Proposition 4.13 For $L \subseteq \Sigma^\omega$ the following are equivalent:

1. L is accepted by a deterministic Büchi–automaton.
2. $L = \lim U$ for a regular language U.

Proof:

"1→ 2" Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a deterministic Büchi–automaton with $L = L_\omega(\mathcal{A})$. Viewed as a finite automaton, $\mathcal{A}$ accepts a regular language $U := L(\mathcal{A})$.

Claim: For $a \in \Sigma^\omega$ the following are equivalent:

i) $\alpha \in L_\omega(\mathcal{A})$.

ii) $\alpha \in \lim U$, i.e., infinitely many initial segments of α belong to U.

This shows $L = \lim U$. It remains to prove the claim:

i)→ii) $\alpha \in L_\omega(\mathcal{A}) \Rightarrow$ there are $f_1, f_2, \ldots \in F$ and $u_1, u_2, \ldots \in \Sigma^+$ such that $q_0 \xrightarrow{u_1}_{\mathcal{A}} f_1 \xrightarrow{u_2}_{\mathcal{A}} f_2 \xrightarrow{u_3}_{\mathcal{A}} \cdots$. Thus, $u_1, u_1u_2, u_1u_2u_3, \ldots$ are initial segments of α that belong to U.

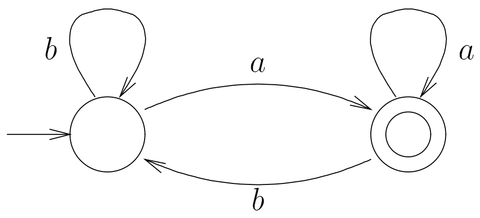
ii)→i) Let $\{u_1, u_1u_2, u_1u_2u_3, \ldots\}$ be initial segments of α that belong to $U = L(\mathcal{A})$ where $u_i \in \Sigma^+$ for $i \geq 1$. Since $\mathcal{A}$ is deterministic, this means that there are $f_1, f_2, f_3 \ldots \in F$ with $q_0 \xrightarrow{u_1}_{\mathcal{A}} f_1 \xrightarrow{u_2}_{\mathcal{A}} f_2 \xrightarrow{u_3}_{\mathcal{A}} f_3 \xrightarrow{u_4}_{\mathcal{A}} \cdots$. Thus $\alpha = u_1u_2u_3u_4 \cdots \in L_\omega(\mathcal{A})$.

"2→ 1" Let $L = \lim U$ for a regular language U. Let $\mathcal{A}$ be a deterministic finite automaton for U. Viewing $\mathcal{A}$ as a Büchi–automaton yields an ω–regular language $L_\omega(\mathcal{A})$. Now i) ↔ ii) from above shows that $L = \lim U = L_\omega(\mathcal{A})$, and thus L is accepted by a deterministic Büchi–automaton. ∎

Corollary 4.14 The class of languages accepted by deterministic Büchi–automaton is *not* closed under complement.

Proof: In Example 4.11 we have shown that $L = (a \cup b)^* b^\omega$ is *not* accepted by a deterministic Büchi–automaton. What is $\overline{L}$?

$\overline{L} = \{a,b\}^\omega \setminus L$ consists of those words $\alpha \in \{a,b\}^\omega$ such that α contains infinitely many a's. Thus, the following is a deterministic Büchi–automaton for $\overline{L}$:



Another way of showing that $\overline{L}$ is accepted by a deterministic Büchi–automaton is the following:

$$\overline{L} = \lim(b^* a)^*$$

■

4.2 Closure under complement

The class of ω–regular languages is closed under complement. However, the proof is more complicated than the one for regular languages.

Main Theorem 4.15 If $L \subseteq \Sigma^\omega$ is ω–regular, then $\Sigma^\omega \setminus L$ is also ω–regular.

Idea underlying the proof: we show that L und $\overline{L}$ can be written as a finite union of languages $U \cdot V^\omega$ where U, V are regular languages. The languages U, V are obtained as equivalence classes of a congruence $\sim_{\mathcal{A}}$of finite index where $\mathcal{A}$ is a Büchi–automaton for L.

Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi–automaton with $L_\omega(\mathcal{A}) = L$. We write $p \xrightarrow[w]{F}_{\mathcal{A}} q$ to indicate there is a path in $\mathcal{A}$ from p to q with label w that contains at least one state from F. We define:

$$L^F_{p,q} := \{w \in \Sigma^* \mid p \xrightarrow[w]{F}_{\mathcal{A}} q\}$$

The languages $L^F_{p,q}$ are regular since

$$L^F_{p,q} = \bigcup_{f \in F} L_{p,f} \cdot L_{f,q}.$$

Definition 4.16 $\sim_{\mathcal{A}}$ is defined as follows: for all $u, v \in \Sigma^*$

$$\begin{array}{llll} u \sim_{\mathcal{A}} v & \text{iff} & \forall p, q \in Q & 1)\ p \xrightarrow{u}_{\mathcal{A}} q \quad \text{iff} \quad p \xrightarrow{v}_{\mathcal{A}} q \\ & & & 2)\ p \xrightarrow[u]{F}_{\mathcal{A}} q \quad \text{iff} \quad p \xrightarrow[v]{F}_{\mathcal{A}} q \end{array}$$

Lemma 4.17 $\sim_{\mathcal{A}}$ is a congruence relation of finite index.

Proof:

1. Obviously, $\sim_{\mathcal{A}}$ is an equivalence relation (since "iff" is reflexive, transitive, and symmetric).
   Congruence: $u \sim_{\mathcal{A}} v \Rightarrow xuy \sim_{\mathcal{A}} xvy$ for all words x, y.
   Assume that $u \sim_{\mathcal{A}} v$ and that $p \xrightarrow[xuy]{F}_{\mathcal{A}} q$. We want to show that this implies $p \xrightarrow[xvy]{F}_{\mathcal{A}} q$. There are states p', q' such that $p \xrightarrow{x}_{\mathcal{A}} p' \xrightarrow{u}_{\mathcal{A}} q' \longrightarrow yq$.

   Case 1: $p \xrightarrow[x]{F}_{\mathcal{A}} p'$

   Since $u \sim_{\mathcal{A}} v$ and $p' \xrightarrow{u}_{\mathcal{A}} q'$, we know that $p' \xrightarrow{v}_{\mathcal{A}} q'$, and thus

   $p \xrightarrow[x]{F}_{\mathcal{A}} p' \xrightarrow{v}_{\mathcal{A}} q' \xrightarrow{y}_{\mathcal{A}} q$, i.e. $p \xrightarrow[xvy]{F}_{\mathcal{A}} q$.

   Case 2: $q' \xrightarrow[y]{F}_{\mathcal{A}} q$ can be treated similarly.

Case 3: $p' \xrightarrow[u]{F}_{\mathcal{A}} q'$. Since $u \sim_{\mathcal{A}} v$, this implies $p' \xrightarrow[v]{F}_{\mathcal{A}} q'$, and thus $p \xrightarrow[F]{xvy}_{\mathcal{A}} q$.

Thus, in all cases $p \xrightarrow[xuy]{F}_{\mathcal{A}} q$ implies $p \xrightarrow[xvy]{F}_{\mathcal{A}} q$.
The other cases can be handled similarly.

2. Finite index: the $\sim_{\mathcal{A}}$–equivalence class of w is uniquely determined by the following pair of sets: $(\{(p,q) \mid p \xrightarrow{w}_{\mathcal{A}} q\}, \{(p,q) \mid p \xrightarrow[w]{F}_{\mathcal{A}} q\})$.
Thus, there are at most $2^{|Q\times Q|} \cdot 2^{|Q\times Q|}$ $\sim_{\mathcal{A}}$–classes. ∎

How do the $\sim_{\mathcal{A}}$–classes look like?

Lemma 4.18

1. $[w] = \bigcap_{\substack{p,q \in Q \\ w \in L_{p,q}}} L_{p,q} \cap \bigcap_{\substack{p,q \in Q \\ w \notin L_{p,q}}} \overline{L_{p,q}} \cap \bigcap_{\substack{p,q \in Q \\ w \in L^F_{p,q}}} L^F_{p,q} \cap \bigcap_{\substack{p,q \in Q \\ w \notin L^F_{p,q}}} \overline{L^F_{p,q}}$.
2. In particular, the $\sim_{\mathcal{A}}$–classes are regular languages.

Proof:

2. is an immediate consequence of 1. since the languages $L_{p,q}$ and $L^F_{p,q}$ are regular, and regular languages are closed under $\cap$ and $^{-}$.

1. "$\subseteq$" Let $u \in [w]$, i.e. $u \sim_{\mathcal{A}} w$. If $w \in L_{p,q}$ $(\overline{L_{p,q}}, L^F_{p,q}, \overline{L^F_{p,q}})$, then $u \in L_{p,q}$ $(\overline{L_{p,q}}, L^F_{p,q}, \overline{L^F_{p,q}})$.

"$\supseteq$" Assume that u is in the intersection on the right–hand side. We must show $u \sim_{\mathcal{A}} w$.

- $p \xrightarrow{w}_{\mathcal{A}} q \Rightarrow w \in L_{p,q} \Rightarrow u \in L_{p,q} \Rightarrow p \xrightarrow{u}_{\mathcal{A}} q$
- $p \not\xrightarrow{w}_{\mathcal{A}} q \Rightarrow w \in \overline{L_{p,q}} \Rightarrow u \in \overline{L_{p,q}} \Rightarrow p \not\xrightarrow{u}_{\mathcal{A}} q$.
- $\xrightarrow[w]{F}$ can be treated in the same way. ∎

Proposition 4.19 Let $\mathcal{A}$ be a Büchi–automaton.

1. For all $\sim_{\mathcal{A}}$–classes U, V we have:

   a) $UV^\omega \cap L_\omega(\mathcal{A}) \neq \emptyset \Rightarrow UV^\omega \subseteq L_\omega(\mathcal{A})$

   b) $UV^\omega \cap \overline{L_\omega(\mathcal{A})} \neq \emptyset \Rightarrow UV^\omega \subseteq \overline{L_\omega(\mathcal{A})}$

2. For every $\alpha \in \Sigma^\omega$ there exist $\sim_{\mathcal{A}}$–classes U, V such that $\alpha \in UV^\omega$.

First, we show that this implies that $\overline{L_\omega(\mathcal{A})}$ is ω–regular.

1. and 2. of the proposition imply that

- $$L_\omega(\mathcal{A}) = \bigcup_{\substack{U,V \sim_{\mathcal{A}}\text{–classes} \\ UV^\omega \subseteq L_\omega(\mathcal{A})}} UV^\omega$$
- $$\overline{L_\omega(\mathcal{A})} = \bigcup_{\substack{U,V \sim_{\mathcal{A}}\text{–classes} \\ UV^\omega \subseteq \overline{L_\omega(\mathcal{A})}}} UV^\omega$$

The non–trivial part is the inclusion "$\subseteq$", which needs both 2. and 1.

Since the $\sim_{\mathcal{A}}$–classes U, V are regular, this shows that $\overline{L_\omega(\mathcal{A})}$ is ω–regular.

We could prove the proposition in an ad hoc manner, but it is more elegant to use a nice combinatorial result: *Ramsey's theorem.*

Definition 4.20 For a set M, we denote by $[M]^2$ the *set of all 2–element subsets* of M. Let $[M]^2 = A_1 \cup A_2 \cup \cdots \cup A_n$ be a partition of $[M]^2$ into n disjoint classes. The set $X \subseteq M$ is called *homogeneous* for this partition if there is an $i, 1 \leq i \leq n$, such that $[X]^2 \subseteq A_i$.

Example: $[\mathbb{N}]^2 = A \cup B$ where

$$\begin{array}{rcl} A & = & \{\{i,j\} \mid i \neq j \text{ and } i \equiv j \bmod 2\} \\ B & = & \{\{i,j\} \mid i \neq j \text{ and } i \not\equiv j \bmod 2\} \end{array}$$

$$\begin{array}{rcll} G & = & \{i \in \mathbb{N} \mid i \text{ is even}\} & \text{are both homogeneous since} \\ U & = & \{j \in \mathbb{N} \mid j \text{ is odd}\} & [G]^2 \subseteq A \text{ and } [U]^2 \subseteq A \end{array}$$

Proposition 4.21 [Ramsey] Let $[\mathbb{N}]^2 = A_1 \cup A_2 \cup \cdots \cup A_n$ be a partition of $[\mathbb{N}]^2$. Then there is an *infinite* set $X \subseteq \mathbb{N}$ that is homogeneous for this partition.

Proof: see J.G.Rosenstein: Linear Orderings, Academic Press, 1982, p. 111,112.

■

Proof of Prop. 4.19

2. Let $\alpha \in \Sigma^\omega$. Together with $\sim_\mathcal{A}$, α defines a partition of $[\mathbb{N}]^2$.

   Let $U_1, U_2, \ldots, U_n$ be the (finitely many) $\sim_\mathcal{A}$–classes.

$$A_\nu = \{\{i,j\} \mid i < j \text{ and } \alpha(i+1,j) \in U_\nu\}.$$

   Since every word $\alpha(i+1,j)$ belongs to one of the $\sim_\mathcal{A}$–classes and since the $\sim_\mathcal{A}$–classes are disjoint, $[\mathbb{N}]^2 = A_1 \dot\cup A_2 \dot\cup \cdots \dot\cup A_n$ is a partition. By Ramsey, there is an infinite $X \subseteq \mathbb{N}$ that is homogeneous for this partition, i.e. there is a $k, 1 \leq k \leq n$ such that for all $i,j \in X$ with $i < j$ we have $\alpha(i+1,j) \in U_k$.

   Since X is infinite, there is an infinite sequence $i_1, i_2, i_3, \ldots$ in X such that $i_j + 1 < i_{j+1}$. Then we know that $\alpha(i_j+1, i_{j+1}) \in U_k \setminus \{\epsilon\}$. Let U be the $\sim_\mathcal{A}$–class of $\alpha(0,i_1)$. Then we have

$$\alpha = \alpha(0,i_1)\alpha(i_1+1,i_2)\alpha(i_2+1,i_3)\ldots \in U \cdot U_k^\omega.$$

1. Let $\alpha \in UV^\omega \cap L_\omega(\mathcal{A})$. This means

   i) $\alpha = uv_1v_2v_3\cdots$ where $u \in U$ and $v_i \in V \setminus \{\epsilon\}$.

   ii) There is a successful path
   $I \ni q_0 \xrightarrow{u}_\mathcal{A} q_1 \xrightarrow{v_1}_\mathcal{A} q_2 \xrightarrow{v_2}_\mathcal{A} q_3 \xrightarrow{v_3}_\mathcal{A} \cdots$

   with label $\alpha = uv_1v_2v_3\cdots$

   Since this path is successful we reach infinitely often a final state. Thus there are infinitely many $i \geq 1$ such that $q_i \xrightarrow[F]{v_i}_\mathcal{A} q_{i+1}$.

   Let $\beta \in UV^\omega$ be arbitrary. Then β is of the form $\beta = u'v_1'v_2'v_3'\cdots$ with $u' \in U$ and $v_i' \in V \setminus \{\varepsilon\}$.

Since U and V are $\sim_{\mathcal{A}}$–classes, we know that $u \sim_{\mathcal{A}} u'$ and $v_i \sim_{\mathcal{A}} v_i{}'$. Thus there is a path of the form $q_0 \xrightarrow{u'}_{\mathcal{A}} q_1 \xrightarrow{v_1'}_{\mathcal{A}} q_2 \xrightarrow{v_2'}_{\mathcal{A}} q_3 \xrightarrow{v_3'}_{\mathcal{A}} \cdots$ with label $\beta = uv_1'v_2'v_3' \cdots$ in $\mathcal{A}$ such that there are infinitely many $i \geq 1$ with $q_i \xrightarrow[F]{v_i'}_{\mathcal{A}} q_{i+1}$. This shows that $\beta \in L_\omega(\mathcal{A})$.

This shows a) of 1. Part b) of 1. is an immediate consequence: assume that $\alpha \in UV^\omega \cap \overline{L_\omega(\mathcal{A})}$, but there is $\beta \in UV^\omega \cap L_\omega(\mathcal{A})$. Now a) implies $\alpha \in L_\omega(\mathcal{A})$. ∎

Corollary 4.22 For every Büchi–automaton $\mathcal{A}$ we can effectively *construct* a Büchi–automaton $\mathcal{B}$ such that $L_\omega(\mathcal{B}) = \overline{L_\omega(\mathcal{A})}$.

Proof:

1. The $\sim_{\mathcal{A}}$–classes (to be more precise: finite automata accepting them) can effectively be constructed: Lemma 4.18 shows how they can be obtained from the languages $L_{p,q}$ and $L^F_{p,q}$.
2. For a given pair U, V of $\sim_{\mathcal{A}}$–classes we can decide whether $UV^\omega \cap L_\omega(\mathcal{A}) \neq \emptyset$. In fact, the emptiness problem for ω–regular languages is decidable (Prop 4.10).
3. For finite unions of the language UV^ω we can effectively construct a Büchi–automaton. ∎

Corollary 4.23 The equivalence problem for ω–regular languages is decidable.

Proof:

$$L_\omega(\mathcal{A}_1) = L_\omega(\mathcal{A}_2) \text{ iff}$$
$$\underbrace{(L_\omega(\mathcal{A}_1) \setminus L_\omega(\mathcal{A}_2)) \cup (L_\omega(\mathcal{A}_2) \setminus L_\omega(\mathcal{A}_1))}_{\text{for this we can construct a Büchi–automaton.}} \quad = \quad \emptyset$$

The emptiness problem for Büchi–automaton is decidable (Prop. 4.10). ∎

4.3 Muller–automata

We know that deterministic Büchi–automaton are weaker than non–deterministic ones. Can we get an automata model where the deterministic automata are as powerful as non–deterministic Büchi–automata?

Definition 4.24 [Muller–automata] A *Muller–automaton* is of the form $\mathcal{A} = (Q, \Sigma, I, \Delta, \mathcal{F})$ where

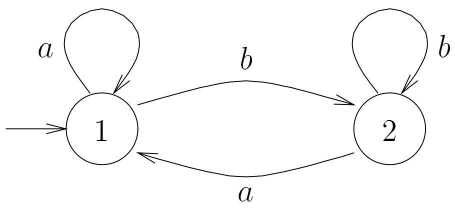
- Q, Σ, I, Δ is as for Büchi–automata.
- $\mathcal{F} \subseteq 2^Q$ is a set of sets of final states.

The infinite path $p_0 \xrightarrow{a_0}_{\mathcal{A}} p_1 \xrightarrow{a_0}_{\mathcal{A}} p_2 \xrightarrow{a_2}_{\mathcal{A}} \cdots$ is successful iff

- $p_0 \in I$
- $\{p \in Q \mid \text{there are infinitely many } i \text{ with } p = p_i\} \in \mathcal{F}$.

$L_\omega(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ is the label of a successful path in } \mathcal{A}\}$.

Example 4.25 $L = (a \cup b)^* b^\omega$.
In Example 4.11 we have shown that L cannot be accepted by a deterministic Büchi–automaton. The following is a deterministic Muller–automaton for L:



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

If the set of states reached infinitely often is $\{2\}$ then 1 is reached only a finite number of times. Thus, we have only finitely may a's.

Note: considered as a Büchi–automaton with $F = \{2\}$, this automaton also accepts words not in L, like $(ab)^\omega$.

Proposition 4.26 For an ω–regular–language L the following are equivalent:

1. L is ω–regular.
2. L is accepted by a deterministic Muller–automaton.

Proof:

"2 → 1" is simple. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ be a deterministic Muller–automaton. Then we have:

$$L_\omega(\mathcal{A}) \quad = \quad \bigcup_{F \in \mathcal{F}} \left(\bigcap_{q \in F} \quad \underbrace{\lim L_{q_0,q}}_{q \text{ is reached infinitely often}} \quad \cap \quad \bigcap_{q \in Q \setminus F} \overline{\lim L_{q_0,q}} \right)$$

$\lim L_{q_0,q}$ contains exactly those words that label paths in $\mathcal{A}$ on which q is reached infinitely often. This is only true since $\mathcal{A}$ is deterministic.

We know that the languages $\lim L_{q_0,q}$ are ω–regular (Prop. 4.13). The ω–regular languages are closed under $\cup, \cap, ^-$.

"1 → 2" is as hard as showing complementation for Büchi–automata. Reason: it is easy to show that the class of languages accepted by deterministic Muller–automata is closed under complement. We don't give the the proof for "1 → 2" here. ∎

Proposition 4.27 If $L \subseteq \Sigma^\omega$ is accepted by a deterministic Muller–automaton, then so is $\overline{L}$.

Proof: Let $L = L_\omega(\mathcal{A})$ for a deterministic Muller–automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$. It is easy to see that $\mathcal{B} = (Q, \Sigma, q_0, \delta, 2^Q \setminus \mathcal{F})$ accepts $\overline{L}$. ∎

Chapter 5

Infinite words and logical formulae

5.1 S1S logic and ω–regular languages

Goal: describe a set of formulae that can exactly define the ω–regular languages.

Just as in the case of finite words, first–order predicate logic is *not* enough to get all ω–regular languages.

Definition 5.1 Formulae of monadic second–order logic of one successor (S1S) are built using:

- n unary predicate symbols $P_1, P_2, \dots, P_n$,
- a unary function symbol s,
- a constant symbol $\underline{0}$,
- a binary predicate symbol $=$,
- a binary predicate symbol $<$,
- Boolean operations $\wedge, \vee, \neg$,

- first–order quantifiers $\exists x, \forall x$ ranging over elements of the domain
- second–order quantifiers $\exists X, \forall X$ ranging over subsets of the domain

As interpretation domain we take the natural numbers ω, where we interpret

- $\underline{0}$ as 0
- s as the successor function: $n \mapsto n+1$
- $=$ as equality
- $<$ as the usual ordering on ω

As in the finite case we take as alphabet $\Sigma = \{0,1\}^n$.

An interpretation $P_1^I, P_2^I, \ldots, P_n^I$ of the unary predicate symbols corresponds to an ω–word

$$\begin{array}{l} \alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma^\omega \text{ where} \\ \alpha(m) = (b_{m_1}, \ldots, b_{m_n}) \text{ with } b_{m_i} = \begin{cases} 1 & \text{if } m \in P_i^I \\ 0 & \text{if } m \notin P_i^I \end{cases} \end{array}$$

For a closed S1S–formuale φ and $\alpha \in \Sigma^\omega$ we write $\alpha \models \varphi$ to say that the interpretation corresponding to α makes φ true. The *ω–language accepted* by φ is defined as

$$L_\omega(\varphi) = \{\alpha \in \Sigma^\omega \mid \alpha \models \varphi\}.$$

Example 5.2 Let $n = 1$, i.e. $\Sigma = \{0,1\}$.

1. $\varphi = P_1(\underline{0}) \wedge (\forall x. P_1(x) \Rightarrow \neg P_1(s(x))) \wedge (\forall x. \neg P_1(x) \Rightarrow P_1(s(x)))$
   $L_\omega(\varphi) = \{10101010\cdots\} = (10)^\omega$
2. $L_1 = \{\alpha \in \Sigma^\omega \mid \text{after every 1 in } \alpha \text{ there eventually is } 0\}$
   For the formula $\varphi_1 = \forall x. (P_1(x) \Rightarrow \exists y. x < y \wedge \neg P_1(y))$ we have $L_\omega(\varphi_1) = L_1$.

3. $L_2 = \{\alpha \in \Sigma^\omega \mid$ between two consecutive 1s there is an even number of 0s$\}$

$$\begin{aligned}\varphi_2 = \forall x.\forall y.(x < y \wedge P_1(x) \wedge P_1(y) \wedge \forall z.(x < z \wedge z < y \Rightarrow \neg P_1(z))) \\ \Rightarrow \exists X.\exists Y.\forall z.((x < z \wedge z \leq y) \Rightarrow \\ (\neg(X(z) \wedge Y(z)) \wedge \\ (X(z) \Rightarrow Y(s(z))) \wedge (Y(z) \Rightarrow X(s(z))) \wedge \\ X(s(x)) \wedge X(y))).\end{aligned}$$

As in the finite case, we use $Q_a(x)$ as an abbreviation for the formula that says that $a \in \Sigma$ is at position x.

Next we show that we can dispense with the symbols $\underline{0}$ and $<$ without losing expressive power.

Lemma 5.3 Both $\underline{0}$ and $<$ can be expressed in S1S using the other symbols.

Proof:

- $x = \underline{0}$ is equivalent to $\neg\exists y.(y < x)$
- $x < y$ is equivalent to $\exists X.(\neg X(x) \wedge X(y) \wedge \forall z.(X(z) \Rightarrow X(s(z))))$.

■

Proposition 5.4 For an ω–language $L \subseteq \Sigma^\omega$ the following are equivalent:

1. L is ω–regular.
2. $L = L_\omega(\varphi)$ for a closed S1S–formula φ.

Proof:

"1 → 2" Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi–automaton such that $L = L_\omega(\mathcal{A})$. We express the existence of a successful path with the help of an S1S–formula. Let $Q = \{q_0, \ldots, q_m\}$ be the states of $\mathcal{A}$. For each state q_i we introduce a second–order variable Y_i with the intended meaning:

$$Y_i(x) \widehat{=} \text{ at position } x \text{ in the path we have state } q_i$$

$\exists Y_0 \cdots \exists Y_m. \big(\forall x. \bigwedge_{0 \le i < j \le m} \neg(Y_i(x) \wedge Y_j(x))\big) \wedge$	the sets are disjoint
$\bigvee_{q_i \in I} Y_i(\underline{0}) \wedge$	the path starts with an initial state
$\forall x. \bigvee_{(q_i,a,q_j) \in \Delta} Y_i(x) \wedge Q_a(x) \wedge Y_j(s(x)) \wedge$	the transition from q_i at x with a to q_j at $s(x)$ must be admissible in Δ
$\bigvee_{q_i \in F} \forall x. \exists y. (x < y \wedge Y_i(y))$	one of the final states is reached infinitely often

By construction, a word $\alpha \in \Sigma^\omega$ satisfies this formula iff there is a successful path in $\mathcal{A}$ with label α.

"$2 \to 1$" First, we transform S1S–formulae into an appropriate normal form:

1. These formulae contain only second–order varibales (no first–order varibales)
2. Atomic formulae are of the following form:
   - $X_i \subseteq X_j$ (with the semantics $\forall x. X_i(x) \Rightarrow X_j(x)$)
   - $\mathrm{Succ}(X_i) = X_j$ (with the semantics that X_i and X_j are singleton sets $\{n_i\}$ and $\{n_j\}$ such that $n_j = n_i + 1$)

Formulae that are built from these atomic formulae using Boolean operations and second–order quantifiers are called $\mathrm{S1S}_0$–formulae.

Claim: Every S1S–formula can be transformed into an equivalent $\mathrm{S1S}_0$–formula.

Proof of the claim:

i) We have already seen that $\underline{0}$ and $>$ can be eliminated.

ii) Nested applications of s can be eliminated:

$$x = \underbrace{s(s(\cdots s}_{m>1}(y)\cdots))$$

is equivalent to

$$\exists y_1 \ldots \exists y_{m-1}. (x = s(y_1) \wedge y_1 = s(y_2) \wedge \ldots \wedge y_{m-1} = s(y)).$$

iii) Thus we may assume that all atomic formulae are of the form:

$$x = y,\ s(x) = y,\ P_i(x),\ X(x)$$

In the final step, we use the following abbreviations:

- $X = Y$ for "$X \subseteq Y \wedge Y \subseteq X$"
- $X \neq Y$ for "$\neg(X = Y)$"
- Singleton(X) =

  "$\exists Y. (Y \subseteq X \wedge Y \neq X \wedge \forall Z. (Z \subseteq X \Rightarrow (Z = X \vee Z = Y)))$"

  X has exactly one strict subset, which is the case iff X is a singleton set.

iv) First–order varibales can be eliminated as illustrated by the following example:

$$\forall x. \exists y. s(x) = y \wedge Z(y)$$

is transformed into

$$\forall X. \text{Singleton}(X) \Rightarrow \exists Y. (\text{Singleton}(Y) \wedge \text{Succ}(X) = Y \wedge Y \subseteq Z).$$

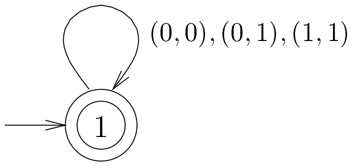
■ Claim

We show by induction on the structure of S1S$_0$–formulae that they define ω–regular languages. We also consider S1S$_0$–formulae with free second–order variables. When defining languages these free varibales are treated like unary predicate symbols: e.g. $\exists Y. (X \subseteq Y \wedge P_1 \subseteq Y)$ yields an ω–language over $\Sigma = \{0,1\}^2$ since P_1 and X occur free.

Induction base: atomic formulae of the form $X \subseteq Y$ and Succ$(X) = Y$ yield ω–languages over $\Sigma = \{0,1\}^2$ (we assume that the first component stands for X and the second for Y).

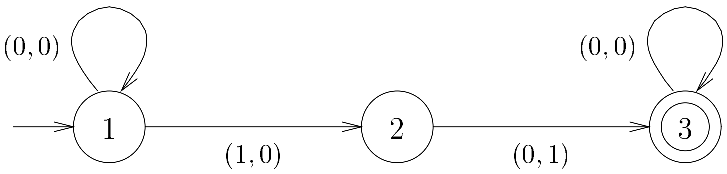
$$L_\omega(X \subseteq Y) \;=\; \{\alpha = \alpha_0\alpha_1\alpha_2 \cdots \mid \text{where } \alpha_i = (b_{i_1}, b_{i_2}) \text{ we have } b_{i_1} = 1 \Rightarrow b_{i_2} = 1\}$$

Thus, $L_\omega(X \subseteq Y)$ is accepted by



$$L_\omega(\mathrm{Succ}(X) = Y) \;=\; \{\alpha = \alpha_0\alpha_1\alpha_2\cdots \mid \begin{array}{l} \text{where } \alpha_i = (b_{i_1}, b_{i_2}) \text{ we have that} \\ \text{there exists a } k \text{ such that} \\ \bullet\ b_{k_1} = 1 \wedge b_{(k+1)_2} = 1 \\ \bullet\ b_{j_1} = 0 \text{ for } j \neq k \\ \bullet\ b_{j_2} = 0 \text{ for } j \neq k+1\}. \end{array}$$

Thus, $L_\omega(\mathrm{Succ}(X) = Y)$ is accepted by



Induction step: It is sufficient to consider $\neg, \vee, \exists X$.

i) $L_\omega(\neg\varphi) = \Sigma^\omega \setminus L_\omega(\varphi)$. By induction, we know that $L_\omega(\varphi)$ is ω–regular, and thus $\Sigma^\omega \setminus L_\omega(\varphi)$ is also ω–regular (Main Theorem 4.15).

ii) in principle, $\vee$ corresponds to union. However, if $\varphi = \varphi_1 \vee \varphi_2$, then φ_1 and φ_2 may be based on different predicates/second–order varibales.

Example:

$$\varphi(\underbrace{X_1, X_2, X_3}_{\text{free variables or unary predicates}}) = \varphi_1(X_1, X_2) \vee \varphi_2(X_2, X_3)$$

Both φ_1 and φ_2 define a language over $\Sigma = \{0,1\}^2$, but the first component for φ_1 corresponds to X_1 whereas the first component for φ_2 corresponds to X_2.

We extend φ_1 and φ_2 by the missing variables, e.g. $\hat{\varphi}_1 = \varphi_1 \wedge X_3 \subseteq X_3$ and $\hat{\varphi}_2 = \varphi_2 \wedge X_1 \subseteq X_1$.
If $\mathcal{A}_1$ is a Büchi–automaton for φ_1, then we obtain a Büchi–automaton for φ_1 as follows: $\hat{\mathcal{A}}_1$ has a transition $q \xrightarrow{(b_1,b_2,b_3)} q'$ iff $q \xrightarrow{(b_1,b_2)} q'$ in $\mathcal{A}_1$.

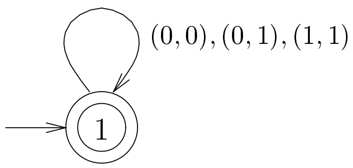
$$\begin{aligned} L_\omega(\varphi) &= L_\omega(\varphi_1 \vee \varphi_2) \\ &= L_\omega(\hat{\varphi}_1 \vee \hat{\varphi}_2) \\ &= L_\omega(\hat{\varphi}_1) \cup L_\omega(\hat{\varphi}_2) \\ &= \underbrace{L_\omega(\hat{\mathcal{A}}_1) \cup L_\omega(\hat{\mathcal{A}}_2)}_{\omega--regular} \end{aligned}$$

iii) $\varphi(X_1, \ldots, X_n) = \exists Y.\psi(Y, X_1, \ldots, X_n)$.
If $\mathcal{A}$ is a Büchi–automaton accepting $L_\omega(\psi(Y, X_1, \ldots, X_n))$, then we obtain a Büchi–automaton for $L_\omega(\exists Y.\psi(Y, X_1, \ldots, X_n))$ by replacing every transition $q \xrightarrow{(b_0,b_1,\ldots,b_n)} q'$ by $q \xrightarrow{(b_1,\ldots,b_n)} q'$ ∎

Example 5.5 (illustrates "2 → 1")

$\varphi = \exists Y.(X \subseteq Y \vee \mathrm{Succ}(Y) = Z)$

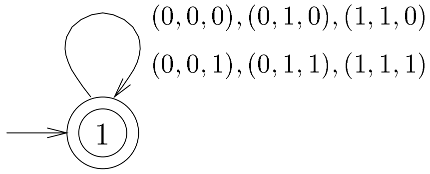
$\mathcal{A}_1$:



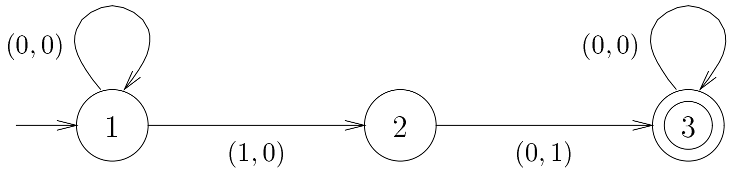
is an automaton for $X \subseteq Y$;

adding a component for Z:

$\hat{\mathcal{A}}_1$:



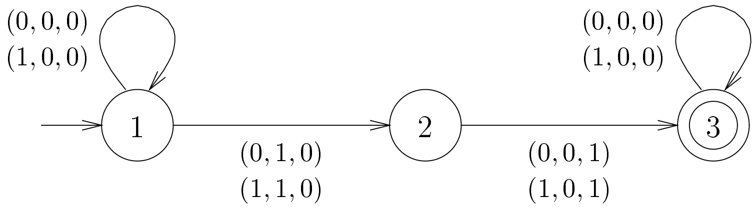
$\mathcal{A}_2$:



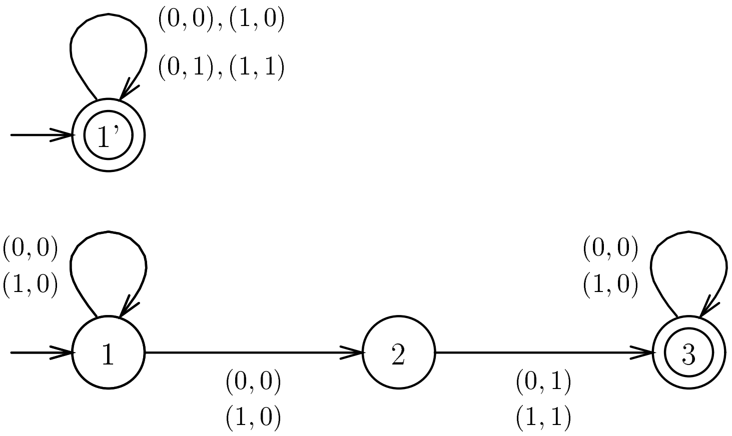
is an automaton for $\mathrm{Succ}(Y) = Z$;

adding a component for X:

$\hat{\mathcal{A}}_2$:



$\hat{\mathcal{A}}_1 \dot{\cup} \hat{\mathcal{A}}_2$ is an automaton for $X \subseteq Y \vee \mathrm{Succ}(Y) = Z$. The automaton $\mathcal{A}$ for φ thus looks as follows:



The proof of the proposition shows that for every S1S–formula φ we can effectively construct a Büchi–automaton $\mathcal{A}$ such that $L_\omega(\varphi) = L_\omega(\mathcal{A})$.

Corollary 5.6 Validity in S1S is decidable.

Proof: If φ is a (closed) S1S–formula, then φ is valid iff $\neg\varphi$ does not have a model, i.e., $L_\omega(\neg\varphi) = \emptyset$. We can effectively construct a Büchi–automaton

$\mathcal{A}$ with $L_\omega(\neg\varphi) = L_\omega(\mathcal{A})$ and the emptiness problem for Büchi–automata is decidable. ∎

The proof of Prop 5.4 can be modified such that it works for finite words.

Corollary 5.7 For a language $L \subseteq \Sigma^*$ the following are equivalent

1. L is regular.
2. $L \setminus \{\varepsilon\} = L(\varphi)$ for a closed S1S–formula φ.

As introduced in Chapter 1.3, finite words correspond to finite interpretations. In the proof of "1 $\to$ 2" we have to take the different acceptance condition into account:

$$\text{“}\bigvee_{q_i \in F} \forall x. \exists y. x < y \wedge Y_i(y)\text{”}$$

is replaced by

$$\text{“}\bigvee_{q_i \in F} \forall x. \underbrace{\mathrm{Max}(x)}_{\text{abbrev. of } s(x) = x} \Rightarrow Y_i(x)\text{”}$$

In the proof of "2 $\to$ 1" we use the closure properties of regular languages.

Corollary 5.8 For a closed S1S–formula φ it is decidable whether φ holds for all finite models.

Note: there are formulae that hold in all finite interpretations, but not in infinite ones.

Example:

$$\exists y \forall x. (x \leq y)$$

Definition 5.9 The ω–language $L \subseteq \Sigma^\omega$ is called *star–free* iff $L = \bigcup_{i=1}^{n} U_i V_i^\omega$ where $U_i, V_i \subseteq \Sigma^*$ are star–free.

Proposition 5.10 For an ω–language $L \subseteq \Sigma^\omega$ the following are equivalent

1. L is star–free.
2. $L = L_\omega(\varphi)$ for a closed formula of S1S not containing second–order quantifiers.

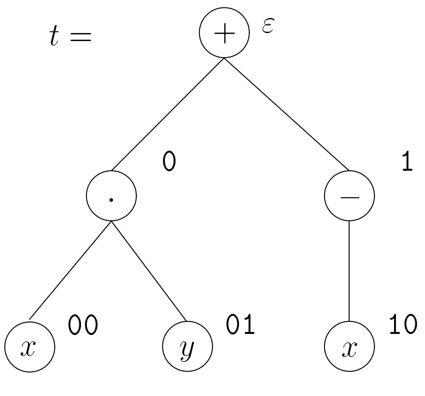
Chapter 6

Automata on finite trees

We consider trees where the number of successor nodes is determined by the arity of the node label.

6.1 Finite trees

Example 6.1 $\Sigma = \{+, \cdot, -, x, y\}$ where $+, \cdot$ have arity 2, $-$ has arity 1, and x, y have arity 0.



is a Σ–labelled tree. The nodes can uniquely be addressed using words over the alphabet $\{0, 1\}$. Thus, the tree t can be viewed as a partial function

$$t : \{\mathtt{0}, \mathtt{1}\}^* \to \Sigma$$

with domain $\mathrm{dom}(t) = \{\varepsilon, \mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{01}, \mathtt{10}\}$; e.g. $t(\mathtt{0}) = \cdot$, $t(\mathtt{10}) = x$.

Definition 6.2 Let Σ be an alphabet and $\nu : \Sigma \to \omega$ a function that assigns with every $a \in \Sigma$ an *arity* $\nu(a)$ (alphabet with arity function). For $n \in \omega$ let $\Sigma_n = \{a \in \Sigma \mid \nu(a) = n\}$. A Σ–*tree* is a partitial function $t : \omega^* \to \Sigma$ whose domain $\mathrm{dom}(t)$ satisfies the following:

1. $\varepsilon \in \mathrm{dom}(t)$,
2. For all $u \in \omega^*$ and $i \in \omega$ we have

$$ui \in \mathrm{dom}(t) \text{ iff } u \in \mathrm{dom}(t) \text{ and } i < \nu(t(u)).$$

1) means that every tree has a root.

2) says that every node $\neq \varepsilon$ has a predecessor node and that every node has the right number of successors.

A *leaf* of t is a node $u \in \mathrm{dom}(t)$ such that $\nu(t(u)) = 0$, i.e. u does not have successor nodes. The tree t is *finite* if $\mathrm{dom}(t)$ is finite. By T_Σ we denote the set of all finite trees over Σ. Let $\prec$ be the *prefix relation* on ω^*, i.e.

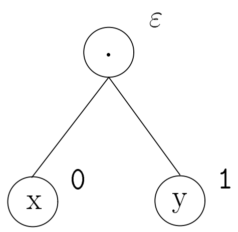
$$u \prec v \quad \text{iff } \exists u' \in \omega^+ \text{ with } uu' = v.$$

Because of 2 in Def. 6.2, the set $\mathrm{dom}(t)$ for a tree t is closed under building prefix, i.e. $v \in \mathrm{dom}(t)$ and $u \prec v \Rightarrow u \in \mathrm{dom}(t)$.

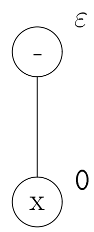
Definition 6.3

1. A *path through* t is a maximal and totally ordered subset of $\mathrm{dom}(t)$. In Example 6.1, $\{\varepsilon, \mathtt{0}, \mathtt{00}\}, \{\varepsilon, \mathtt{0}, \mathtt{01}\}$, and $\{\varepsilon, \mathtt{1}, \mathtt{10}\}$ are all the paths. $\{\varepsilon, \mathtt{00}\}$ is not maximal and $\{\varepsilon, \mathtt{0}, \mathtt{00}, \mathtt{01}\}$ is not totally ordered.
2. The *subtree* of t at position $u \in \mathrm{dom}(t)$ is the tree t_u with
   - $\mathrm{dom}(t_u) = \{v \mid uv \in \mathrm{dom}(t)\}$
   - $t_u(v) = t(uv)$.

For example, $t_0 =$

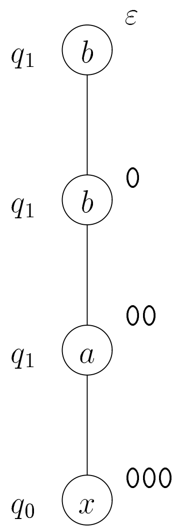


$t_1 =$

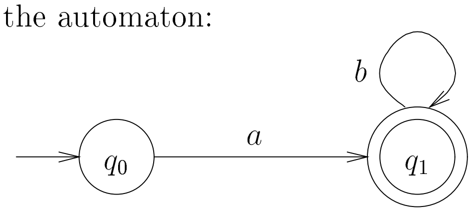


6.2 Automata on finite trees

A word $w \in \Sigma^*$ can be viewed as a finite tree over an alphabet $\widehat{\Sigma} = \Sigma \cup \{x\}$ where $\nu(a) = 1$ for all $a \in \Sigma$ and $\nu(x) = 0$. E.g. abb can be viewed as the tree:



A path with label abb of a finite automaton corresponds to a labelling of the nodes of the tree with states of the automaton:



This can be generalized to trees with branching factor > 1.

Definition 6.4 An *LR–tree automaton* (leaf to root) $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ consists of:

- a finite set of states Q
- a finite alphabet Σ with arity function
- an initial assignment $I : \Sigma_0 \to 2^Q$
- a transition assignment Δ, which assigns to every $a \in \Sigma$ of arity $n > 0$ a function $\Delta_a : Q^n \to 2^Q$
- a set of final states F

A *run* of this automaton on the tree $t \in T_\Sigma$ is a mapping $\ell : \mathrm{dom}(t) \to Q$ such that

$$\ell(u) \in \Delta_a(\ell(u0), \ldots, \ell(u(n-1)))$$

where $a = t(u)$ has arity n.

The ℓ run is *successful* iff

- $\ell(u) \in I(t(u))$ for all leafs u,
- $\ell(\varepsilon) \in F$.

The *tree language accepted by* $\mathcal{A}$ is

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid \text{ there is a successful run of } \mathcal{A} \text{ on } t\}.$$

$\mathcal{A}$ is *deterministic* iff

- $|I(a)| = 1$ for all $a \in \Sigma_0$
- $|\Delta_a(q_1, q_2, \ldots, q_n)| = 1$ for all $n > 0, a \in \Sigma_n$ and $q_1, q_2, \ldots, q_n \in Q$

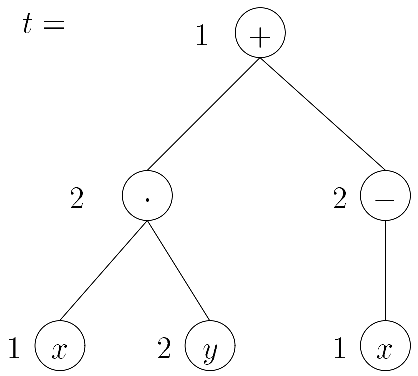
In this case we write I as a function $I : \Sigma_0 \to Q$ and Δ_a as a function $\Delta_a : Q^n \to Q$

Example 6.5 $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{x, y\}, \Sigma_1 = \{-\}, \Sigma_2 = \{+, \cdot\}$

$\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ where

- $Q = \{0, 1, 2\}$,
- $I(x) = 1, I(y) = 2$,
- $$\begin{array}{lcl} \Delta_-(q) & = & -q \bmod 3, \\ \Delta_+(q_1, q_2) & = & q_1 + q_2 \bmod 3, \\ \Delta_\cdot(q_1, q_2) & = & q_1 \cdot q_2 \bmod 3, \end{array}$$
- $F = \{1\}$.

$t =$



The automaton evaluates arithmetic expressions with $x = 1$ and $y = 2$ modulo 3, and accepts if the value is 1.

Just as in the case of words, non–det. automata can be transformed into deterministic ones using the powerset construction.

Proposition 6.6 For a tree language $L \subseteq T_\Sigma$ the following are equivalent:

1. L is accepted by a non–deterministic LR–tree automaton
2. L is accepted by a deterministic LR–tree automaton

(Exercise)

Instead of working from the leafs to the root, we can also work in the other direction:

Definition 6.7 An *RL–tree automaton* $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ consists of

- a finite set of states Q
- an alphabet with arity function Σ
- a set $I \subseteq Q$ of initial states
- a transition assignment Δ that assigns to each $a \in \Sigma_n$ for $n > 0$ a function $\Delta_a : Q \to 2^{Q^n}$
- a final assignment $F : \Sigma_0 \to 2^Q$.

A *run* of $\mathcal{A}$ on $t \in T_\Sigma$ is a mapping $\ell : \mathrm{dom}(t) \to Q$ such that

$$(\ell(u\mathtt{0}), \ldots, \ell(u(n-1))) \in \Delta_a(\ell(u)) \text{ where } a = t(u) \text{ has arity } n.$$

This run is *successful* iff

- $\ell(\varepsilon) \in I$,
- $\ell(u) \in F(t(u))$ for all leafs u.

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid \text{ there is a successful run of } \mathcal{A} \text{ on } t\}.$$

$\mathcal{A}$ is *deterministic* iff

- $|I| = 1$
- $|\Delta_a(q)| = 1$ for all $n > 0, a \in \Sigma_n$, and $q \in Q$

Proposition 6.8 For a tree language $L \subseteq T_\Sigma$ the following are equivalent:

1. L is accepted by an LR–tree automaton.
2. L is accepted by an RL–tree automaton

Proof:

"$1 \Rightarrow 2$" Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be an LR tree–automaton. We consider the RL tree–automaton $\mathcal{B} = (Q, \Sigma, F, \Delta', I)$ where

$$\Delta'_a : q \mapsto \{(q_1, \ldots, q_n) \mid q \in \Delta_a(q_1, \ldots, q_n)\}$$

It is easy to see that any successful run of $\mathcal{A}$ on a tree t is also a successful run of B on this tree and vice versa.

"$2 \Rightarrow 1$" can be shown accordingly. ∎

Example 6.9 Let $\mathcal{A}$ be the LR–tree automaton of Example 6.5. The corresponding RL–tree automaton $\mathcal{B} = (Q, \Sigma, I', \Delta', F')$ is defined as follows

- $Q = \{0, 1, 2\}$
- $\Sigma = \{x, y, -, +, \cdot\}$,
- $I' = \{1\}$
- $$\begin{aligned} \Delta_-(q) &= \{q' \in Q \mid \Delta_-(q') = q\} \\ &= \{q' \mid -q' \bmod 3 = q\} = \{q' \in Q \mid q' = -q \bmod 3\} \end{aligned}$$

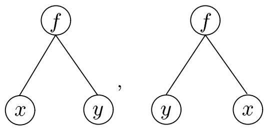
- $\Delta_+(q) = \{(q', q'') \mid q = q' + q'' \bmod 3\}$,
- $\Delta_\cdot(q) = \{(q', q'') \mid q = q' \cdot q'' \bmod 3\}$,
- $F'(x) = 1$ and $F'(y) = 2$

Note: Although the LR–tree automaton $\mathcal{A}$ is deterministic, the corresponding RL–tree automaton is *not* deterministic:

$$\text{e.g. } \Delta_+(1) = \{(0,1), (1,0), (2,2)\}$$

We will show that deterministic RL–tree automata are weaker than non–deterministic ones.

Example 6.10 Not every language accepted by a non–deterministic RL–tree automaton can also be accepted by a deterministic RL–tree automaton.

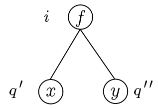
$\Sigma = \{\underbrace{x,\ y,}_{\text{arity 0}}\ \underbrace{f}_{\text{arity 2}}\}$, $L = \{$  $\}$

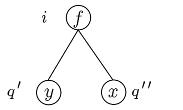
- The following non–deterministic RL–tree automaton accepts L:

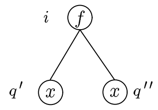
$$\mathcal{A} = (\{q_0, q_1, q_x, q_y\}, \Sigma, \underbrace{\{q_0, q_1\}}_{\text{non–det.}}, \Delta, F)$$

where

$$\begin{aligned} \Delta_f(q_0) &= \{(q_x, q_y)\} \\ \Delta_f(q_1) &= \{(q_y, q_x)\} \\ \Delta_f(q_x) &= \Delta_f(q_y) = \emptyset \\ F(x) &= \{q_x\}, \\ F(y) &= \{q_y\}. \end{aligned}$$

- Assume that $\mathcal{B} = (Q', \Sigma, \{i\}, \Delta', F')$ is a *deterministic* RL–tree automaton for L. Let $(q', q'') = \Delta'_f(i)$. Since  $\in L$, we know that

$q' \in F(x)$. Since 

$\in L$, we know that $q'' \in F(x)$. But then $\mathcal{B}$ also accepts 

$\notin L$.

Deterministic and non–deterministic LR–tree automata as well as non–deterministic RL–tree automata accept the same class of tree languages. Deterministic RL–tree automata accept a smaller class.

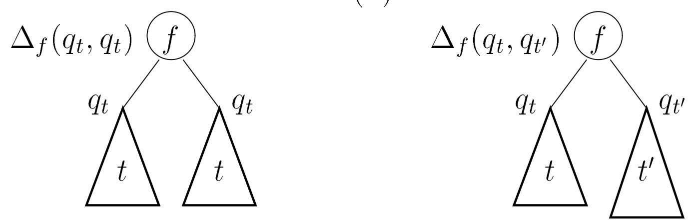
Definition 6.11 The tree language $L \subseteq T_\Sigma$ is called *recognizable* iff it is accpeted by an LR–tree automaton.

Example 6.12 There are non–recognizable tree languages. To show this, we consider an alphabet Σ with arity function such that $|\Sigma_0| > 0$ and $|\Sigma_2| > 0$. Thus T_Σ is infinite.

For $f \in \Sigma_2$ we define $L = \{f(t,t) \mid t \in T_\Sigma\}$ and show that L is *not* recognizable.

Assume that L is recognizable. Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a deterministic LR–tree automaton for L. For every tree $t \in T_\Sigma$ we consider the run ℓ on t that labels each leaf u with $\ell(u) = I(t(u))$. Let $q_t = \ell(\varepsilon)$. Since Q is finite and T_Σ is infinite, there are trees $t \neq t'$ such that $q_t = q_{t'}$. Consider the run of $\mathcal{A}$ on the following trees:

(a) (b)



Since the tree in (a) belongs to L we have that $\Delta_f(q_t, q_t) \in F$. But then the automaton $\mathcal{A}$ accepts also the tree in (b), which does not belong to L.

6.3 Regular tree languages

Recognizable languages of finite words can be described using regular expressions. A similar characterization can be shown for recognizable tree languages. To this purpose we must introduce appropriate operations on tree languages.

Recall:

- $\emptyset \in Reg_\Sigma$, $\{a\} \in Reg_\Sigma$ for all $a \in \Sigma$
- $L_1, L_2 \in Reg_\Sigma \Rightarrow L_1 \cup L_2, L_1 \cdot L_2, {L_1}^* \in Reg_\Sigma$

Proposition 6.13

1. The empty tree language is recognizable
2. For every $a \in \Sigma_0$ the language $\{ⓐ\}$ is recognizable.

Proof:

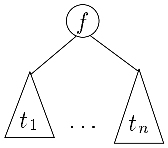
1. Use an LR–tree automaton with $F = \emptyset$.
2. LR–tree automaton where:
   - $Q = \{0, 1\}$
   - $I(a) = 1$, $I(b) = 0$ for all $b \in \Sigma_0 \setminus \{a\}$
   - $\Delta_f(q_1, \ldots, q_n) = \emptyset$ for all $(q_1, \ldots q_n) \in Q^n, f \in \Sigma_n, n > 0$
   - $F = \{1\}$

Proposition 6.14 The class of recognizable tree languages is closed under union, intersection, and complement.

Proof: similar to the case of words

1. Union: take the union of the automata (Exercise).
2. Complement: use deterministic LR–tree automata and exchange final states with non–final states. ∎

Next, we define *concatenation of tree languages*.

Notation: a tree  is written as $f(t_1, \ldots, t_n)$.

Definition 6.15 Let Σ be an alphabet with arity function and let $\overline{x} = (x_1, \ldots, x_k)$ be a k–tuple of elements of Σ_0.

1. For $t \in T_\Sigma$ and $L_1, \ldots, L_k \subseteq T_\Sigma$ we define $t \cdot^{\overline{x}} (L_1, \ldots, L_k) \subseteq T_\Sigma$ by induction:

   - $t \in \Sigma_0 :$ $\begin{array}{lcl} t = x_i & : & t \cdot^{\overline{x}} (L_1, \ldots, L_k) := L_i \\ t \neq x_i \text{ for all } i & : & t \cdot^{\overline{x}} (L_1, \ldots, L_k) := \{t\} \end{array}$
   - $t = f(t_1, \ldots, t_n) :$

   $$t \cdot^{\overline{x}} (L_1, \ldots, L_k) := \{f(t'_1, \ldots, t'_n) \mid t'_i \in t_i \cdot^{\overline{x}} (L_1, \ldots, L_k)\}$$

2. For $L, L_1, \ldots, L_k \subseteq T_\Sigma$ we define

   $$L \cdot^{\overline{x}} (L_1, \ldots, L_k) := \bigcup_{t \in L} t \cdot^{\overline{x}} (L_1, \ldots, L_k)$$

A tree in $L \cdot^{\overline{x}} (L_1, \ldots, L_k)$ is obtained from a tree $t \in L$ by replacing each leaf with label x_i by some tree in L_i.

Example:

$$\{\underset{\in \Sigma_2}{f}(x, x)\} \cdot^{x} \{\underset{\in \Sigma_0}{a, b}\} = \{f(a, a), f(a, b), f(b, a), f(b, b)\}.$$

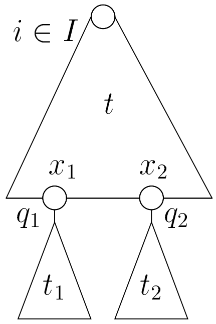
Note: different occurrences if x_i may be replaced by different elements of L_i. In particular:

$$\begin{array}{rcl} \{f(x, x)\} \cdot^{x} T_\Sigma & = & \{f(t, t') \mid t, t' \in T_\Sigma\} \\ & \neq & \{f(t, t) \mid t \in T_\Sigma\}. \end{array}$$

Proposition 6.16 If $L, L_1, L_2, \ldots, L_k$ are recognizable tree languages, then so $L \cdot^{\overline{x}} (L_1, \ldots, L_k)$.

Proof: Given an RL–tree automaton for $L, L_1, \ldots, L_k$ we construct an RL–tree automaton for the concatenation.

Idea:



$q_1 \in F(x_1)$
$q_2 \in F(x_2)$

here we continue with the automata for L_1 and for L_2

Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be an RL–tree automaton for L and $\mathcal{A}_i = (Q^{(i)}, \Sigma, I^{(i)}, \Delta^{(i)}, F^{(i)})$ be an RL–tree automaton for L_i ($i = 1, \ldots, k$). W.l.o.g. the sets of states are disjoint. The following automaton

$$\mathcal{B} = (Q \cup Q^{(1)} \cup \ldots \cup Q^{(k)}, \Sigma, I, \Delta', F')$$

is an RL–tree automaton for the concatenation.

- for $a \in \Sigma_n$ with $n > 0$:
  - for $q \in Q^{(j)}$: $\Delta'_a(q) = \Delta^{(j)}_a(q)$
  - for $q \in Q$: $\Delta'_a(q) = \Delta_a(q) \cup \bigcup_{\substack{j \text{ with} \\ q \in F(x_j)}} \{\Delta^{(j)}_a(i) \mid i \in I^{(j)}\}$
- for $a \in \Sigma_0$:
  - for $a \notin \{x_1, \ldots, x_k\}$:

$$\begin{aligned} F'(a) &= F(a) \cup F^{(1)}(a) \cup F^{(2)}(a) \cup \ldots \cup F^{(k)}(a) \cup \\ &\quad \{q \in Q \mid q \in F(x_j) \text{ for some } j, 1 \le j \le k \text{ and} \\ &\quad \underbrace{F^{(j)}(a) \cap I^{(j)} \neq \emptyset}_{a \,\in\, L_j}\} \end{aligned}$$

  - for $a \in \{x_1, \ldots, x_k\}$:

$$\begin{aligned} F'(a) &= F^{(1)}(a) \cup F^{(2)}(a) \cup \ldots \cup F^{(k)}(a) \cup \\ &\quad \{q \in Q \mid q \in F(x_j) \text{ for some } j, 1 \le j \le k \text{ and} \\ &\quad F^{(j)}(a) \cap I^{(j)} \neq \emptyset\} \end{aligned}$$

■

When defining regular languages, we will consider two special cases of the general concatenation introduced in Definition 6.15:

1. Applying an $f \in \Sigma_k$, $k > 0$, to tree languages $L_1, \ldots, L_k$:

$$f(L_1, \ldots, L_k) := f(x_1, \ldots, x_k) \cdot^{(x_1, \ldots, x_k)} (L_1, \ldots, L_k)$$

2. $\overline{x} = x$, i.e. tuple of length 1 : $L \cdot^x L'$.

The Kleene–Star can also be generalized to tree languages:

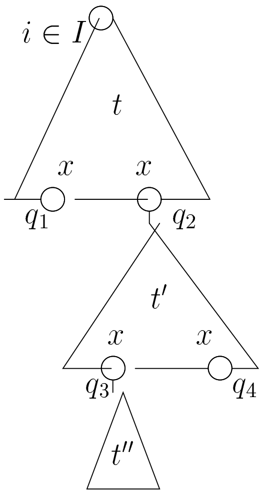
Definition 6.17 Let $L \subseteq T_\Sigma$ and $x \in \Sigma_0$. We define :

- $L^{0,x} \quad = \quad \{x\}$
- $L^{n+1,x} \quad = \quad L^{n,x} \cup L \cdot^x L^{n,x}$
- $L^{*,x} \quad = \quad \bigcup_{n \geq 0} L^{n,x}$

Proposition 6.18 If L is a recognizable tree language, then so is $L^{*,x}$.

Proof: Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be an RL–tree automaton for L.

Idea:



If $q_i \in F(x)$, then we can either stop or continue with an initial state.

We define $\mathcal{B} = (Q \cup \{\hat{\imath}\}, \Sigma, I', \Delta', F')$ where $\{\hat{\imath}\} \notin Q$ and

- for $a \in \Sigma_n$ with $n > 0$:

$$\Delta'_a(q) = \Delta_a(q) \quad \cup \quad \begin{cases} \emptyset & q \notin F(x) \\ \bigcup\limits_{i \in I} \Delta_a(i) & q \in F(x) \end{cases}$$

$$\Delta'_a(\hat{\imath}) = \emptyset$$

- for all $a \in \Sigma_0$
  - $a \neq x$:

$$F'(a) = F(a) \cup \{q \in Q \mid q \in F(x) \text{ and } \underbrace{F(a) \cap I}_{a \in L} \neq \emptyset\}$$

  - for $a = x$:

$$F'(x) = F(x) \cup \{\hat{\imath}\}$$

- $I' = I \cup \{\hat{\imath}\}$

■

Note: the state $\hat{\imath}$ in $F'(x)$ and I' ensures that x is accepted.

Definition 6.19 Let Σ be an alphabet with arity function, Z a set of symbols of arity 0 with $\Sigma \cap Z = \emptyset$, and define $\widehat{\Sigma} := \Sigma \cup Z$. $\mathcal{R}eg(T_\Sigma, Z)$ is the smallest class of tree languages over $\widehat{\Sigma}$ such that

1. $\emptyset \in \mathcal{R}eg(T_\Sigma, Z)$,
2. $\{x\} \in \mathcal{R}eg(T_\Sigma, Z)$ for all $x \in \Sigma_0 \cup Z$,
3. $L_1, L_2 \in \mathcal{R}eg(T_\Sigma, Z) \Rightarrow L_1 \cup L_2 \in \mathcal{R}eg(T_\Sigma, Z)$,
4. $L_1, L_2 \in \mathcal{R}eg(T_\Sigma, Z)$ and $z \in Z \Rightarrow L_1 \cdot^z L_2 \in \mathcal{R}eg(T_\Sigma, Z)$,
5. $L \in \mathcal{R}eg(T_\Sigma, Z)$ and $z \in Z \Rightarrow L^{*,z} \in \mathcal{R}eg(T_\Sigma, Z)$,
6. $n > 0, f \in \Sigma_n, L_1, \ldots, L_n \in \mathcal{R}eg(T_\Sigma, Z) \Rightarrow f(L_1, \ldots L_n) \in \mathcal{R}eg(T_\Sigma, Z)$.

The language $L \subseteq T_\Sigma$ is *regular* iff there is a set of auxiliary symbols Z of arity 0 such that $L \in \mathcal{R}eg(T_\Sigma, Z)$.

Proposition 6.20 For $L \subseteq T_\Sigma$ the following are equivalent:

1. L is regular
2. L is recognizable

Proof:

"$1 \Rightarrow 2$" Follows from what we have shown.

"$2 \Rightarrow 1$" Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be an RL–tree automaton with $L = L(\mathcal{A})$. W.l.o.g. Q is of the form $\{1, \ldots, k\}$ and $Q \cap \Sigma = \emptyset$. We define $Z := Q$ (where $q \in Q$ is assumed to be of arity 0). Let $\mathcal{A}' = (Q, \Sigma \cup Z, I, \Delta, F')$ where $F'(q) = \{q\}$. Thus, subtrees can be replaced by a leaf q if the corresponding node in the run gets label q.

For $K \subseteq Q, 0 \leq h \leq k$ and $i \in Q$ let $L(K, h, i)$ be the set of all trees $t \in T_{\Sigma \cup K}$ such that there is a run ℓ of $\mathcal{A}'$ on t with

- $\ell(\epsilon) = i$
- $\ell(u) \leq h$ for all $u \neq \varepsilon$ that are not leafs
- $\ell(u) \in F'(t(u))$ for all leafs u

$L(K, h, i)$ consists of the trees that may have additional leafs labelled with elements of K. A run of $\mathcal{A}'$ on this tree that begins with i must stop with a state in $F'(t(u))$ for each leaf u (in particular, if $t(u) = q$ then $\ell(u) = q$) and the intermediate states must be $\leq h$. Obviously, the following holds:

$$L(\mathcal{A}) = \bigcup_{i \in I} L(\emptyset, k, i)$$

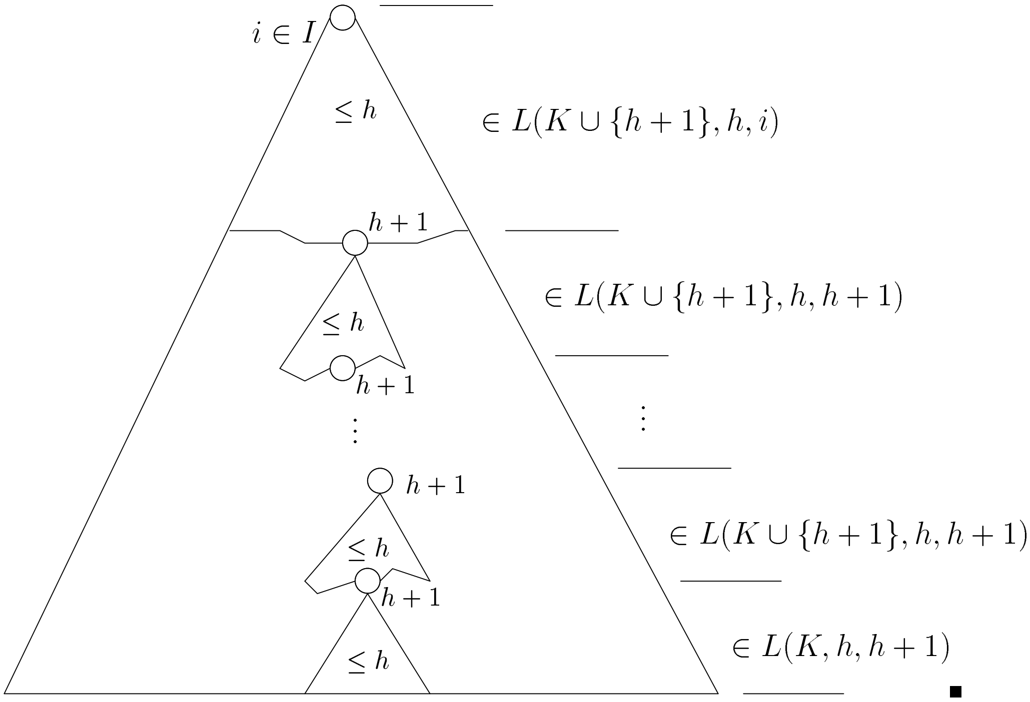
Thus, it is sufficient to show that all the languages $L(K, h, i)$ belong to $\mathcal{R}eg(T_\Sigma, Z)$

Induction base $h = 0$: In this case, $L(K, 0, i)$ cannot contain trees having a node that is neither the root nor a leaf (since a run must label such an intermediate node with a state $q > 0$). Thus, $L(k, 0, i)$ is finite. It is easy to see that finite sets of trees are regular.

Induction step $h > 0$: we have the following:

$$\begin{aligned} L(K, h+1, i) &= L(K, h, i) \cup \\ & \quad L(K \cup \{h+1\}, h, i) \cdot^{h+1} \\ & \quad L(K \cup \{h+1\}, h, h+1)^{*,h+1} \cdot^{h+1} L(K, h, h+1) \end{aligned}$$

By induction and the definition of $\mathcal{R}eg(T_\Sigma, Z)$ this shows that $L(k, h+1, i) \in \mathcal{R}eg(T_\Sigma, Z)$



■

Another interesting closure property of recognizable tree languages is closure under *alphabet renaming*: Let $\Sigma^{(1)}, \Sigma^{(2)}$ be alphabets with arity functions and $\varphi : \Sigma^{(1)} \to \Sigma^{(2)}$ a mapping, such that $\varphi(\Sigma_n^{(1)}) \subseteq \Sigma_n^{(2)}$ for all $n \geq 0$. For a tree $t \in T_{\Sigma^{(1)}}$ we define $\varphi(t) \in \Sigma^{(2)}$ as follows:

$$\varphi(t) : \mathrm{dom}(t) \to \Sigma^{(2)} \quad \varphi(t)(u) = \varphi(t(u))$$

Proposition 6.21 If $L \subseteq T_{\Sigma^{(1)}}$ is recognizable, then so is $\varphi(L) = \{\varphi(t) \mid t \in L\} \subseteq T_{\Sigma^{(2)}}$.

Proof: Let $\mathcal{A} = (Q, \Sigma^{(1)}, I, \Delta, F)$ be an LR–tree automaton for L. Then $\mathcal{A}' = (Q, \Sigma^{(2)}, I', \Delta', F)$ is an LR–tree automaton for $\varphi(L)$ where

- for $a \in \Sigma_0^{(2)}$:
$$I'(a) = \bigcup_{\substack{a' \in \Sigma_0^{(1)} \\ \varphi(a') = a}} I(a'),$$
- for $a \in \Sigma_n^{(2)}$, $n > 0$:
$$\Delta'_a(q_1, \ldots, q_n) = \bigcup_{\substack{a' \in \Sigma_n^{(1)} \\ \varphi(a') = a}} \Delta_{a'}(q_1, \ldots, q_n).$$

∎

Proposition 6.22 For regular languages, the equivalence and emptiness problem is decidable.

Proof: Since the regular/recognizable tree languages are closed under Boolean operations, the equivalence problem can be reduced to the emptiness problem.

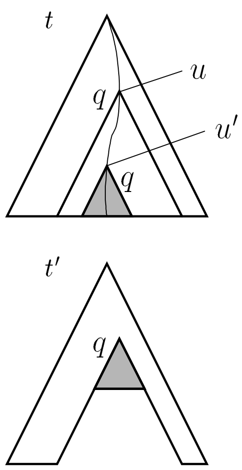
Emptiness problem: Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a deterministic LR–tree automaton for the language L with $|Q| = k$.

Claim: $L(\mathcal{A}) \neq \emptyset$ iff there is a tree t of depth $\leq k$ with $t \in L(\mathcal{A})$.
Since Σ is finite, there are only finitely many trees $t \in T_\Sigma$ of depth $\leq k$. For each of these trees we can effectively test $t \in L(\mathcal{A})$.

Proof of the claim:

"$\Leftarrow$" is trivial.

"⇒" Let t be a tree of minimal size with $t \in L(\mathcal{A})$. Assume that $\mathcal{A}$ has depth $> k$, i.e. t contains at least one path of length $> k$. Let ℓ be a successful run of $\mathcal{A}$ on t. Since we have only k states there are two different positions u, u' on the path such that $q := \ell(u) = \ell(u')$.



If we replace in t the subtree t_u by $t_{u'}$, then we get a smaller tree t' for which $\mathcal{A}$ also has a successful run. This contradicts the minimality of t. ∎

Note: this yields an exponential algorithm for the emptiness problem. There is a linear–time algorithm for the emptiness problem (Exercise).

Chapter 7

Automata on infinite trees

For the sake of simplicity, we restrict the attention to binary trees, i.e. Σ is an alphabet with arity function such that $\Sigma = \Sigma_2$. All the results can easily be extended to the general case.

An *infinite tree over* Σ is a mapping $\{0,1\}^* \to \Sigma$. With T_Σ^ω we denote the set of all infinite trees over Σ. An infinite tree over Σ is called ω–tree. An ω*–tree language* is a subset of T_Σ^ω.

ω–tree languages can be obtained by *infinite iteration*.

Definition 7.1 Let $Z = \{z_1, \ldots, z_k\}$ be a set of symbols of arity 0 and Σ an alphabet of binary symbols. Let $U, U_1, \ldots, U_k \subseteq T_{\Sigma \cup Z}$ be tree languages over $\Sigma \cup Z$. The ω–tree language

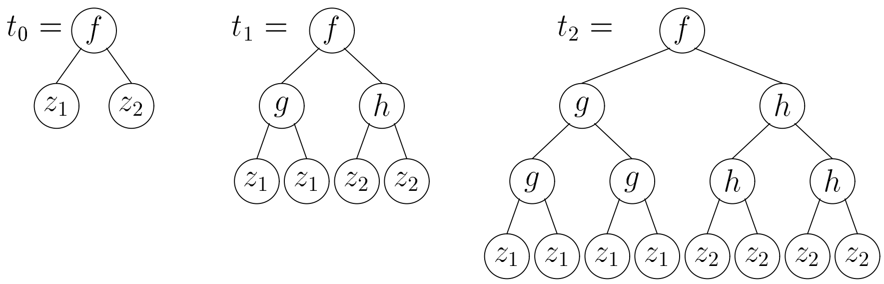
$$U \cdot^{(z_1,\ldots,z_k)} (U_1, \ldots, U_k)^{\omega,(z_1,\ldots,z_k)}$$

consists of all ω–trees $t \in T_\Sigma^\omega$ for which there exists a sequence $t_0, t_1, t_2, \ldots$ of trees in $T_{\Sigma \cup Z}^\omega$ such that

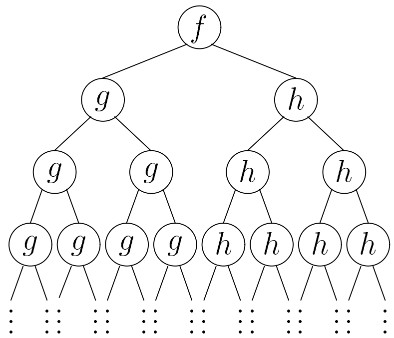
1. $t_0 \in U$
2. For all $i \geq 0$: $t_{i+1} \in t_i \cdot^{(z_1,\ldots,z_k)} (U_1, \ldots, U_k)$
3. t is the *limit* of the sequence, i.e. for all $u \in \{\mathtt{0},\mathtt{1}\}^*$ there is an $m \geq 0$ such that

- $u \in \mathrm{dom}(t_m)$ and $t(u) \in \Sigma$ (i.e. u is not a leaf of $t_m(u)$)
- $t(u) = t_m(u)$

Example 7.2 $Z = \{z_1, z_2\}, U = \{f(z_1, z_2)\}, U_1 = \{g(z_1, z_1)\}, U_2 = \{h(z_2, z_2)\}$



There is only one possible sequence, whose limit is:



7.1 Büchi– and Rabin–tree automata

Since our infinite trees have no leafs, our automata start at the root, i.e. they generalize RL–tree automata.

Definition 7.3

1. A *Büchi–tree automaton* over the alphabet Σ (with $\Sigma = \Sigma_2$) is of the form $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ where
   - Q, Σ, I, Δ are as for RL–tree automata
   - $F \subseteq Q$ is a set of final states

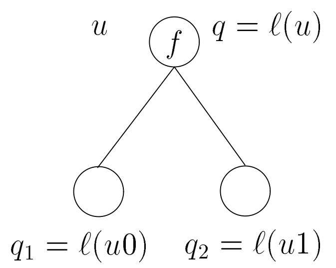
2. A *Rabin–tree automaton* over the alphabet Σ (with $\Sigma = \Sigma_2$) is of the form $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$, where

   - Q, Σ, I, Δ are as for RL–tree automata
   - $\Omega = \{(F_1, G_1), \ldots, (F_n, G_n)\}$ with $F_i, G_i \subseteq Q$

   (Compare this to exercise 55.)

A *run* ℓ of a (Büchi– or Rabin–) tree automaton on the tree $t \in T_\Sigma^\omega$ is defined as follows:

$\ell : \{0,1\}^* \to Q$ such that $(\ell(u\mathtt{0}), \ell(u\mathtt{1})) \in \Delta_f(\ell(u))$ where $f = t(u)$



$(q_1, q_2) \in \Delta_f(q)$

Thus a run is itself an infinite tree over the alphabet Q (where all $q \in Q$ have arity 2). The run ℓ of a Büchi tree–automaton is called *successful* iff

- $\ell(\varepsilon) \in I$
- every path in ℓ contains infinitely often final states

The run ℓ of the Rabin–tree automaton is called *successful* iff

- $\ell(\varepsilon) \in I$
- for every path in ℓ there is an i $(1 \le i \le n)$ such that
  - the path contains infinitely often states from F_i
  - none of the states in G_i occurs infinitely often in the path

Thus, the acceptance condition for Büchi/Rabin automata on ω–words is applied to paths in the infinite tree.

$$L_\omega(\mathcal{A}) := \{t \in T_\Sigma^\omega \mid \text{there is a successful run of } \mathcal{A} \text{ on } t\}$$

$L \subseteq T_\Sigma^\omega$ is called *Büchi–recognizable* (*Rabin–recognizable*) iff there is a Büchi– (Rabin–) tree automaton $\mathcal{A}$ such that $L_\omega(\mathcal{A}) = L$

Proposition 7.4 Every Büchi–recognizable language is also Rabin–recognizable.

Proof: Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi tree–automaton with $L = L_\omega(\mathcal{A})$. The Rabin–tree automaton $\mathcal{A}' := (Q, \Sigma, I, \Delta, \{(F, \emptyset)\})$ obviously accepts L.

■

The following examples illustrate the difference between Büchi– and Rabin–tree automata.

Example 7.5 $\Sigma = \{a, b\}$ and

$$L_1 = \{t \in T_\Sigma^\omega \mid \text{ there is a path in } t \text{ containing infinitely many } a\text{'s}\}.$$

We want to design a Büchi tree automata for L_1.

Idea: the automata "guesses" the path containing infinitely many a's

$$\mathcal{A}_1 = (\{\underset{\substack{\text{states on}\\\text{the guessed}\\\text{path}}}{i, f}, \underset{\substack{\text{other}\\\text{paths}}}{\Box}\}, \Sigma, \{i\}, \Delta_1, \{\underset{\substack{\text{have}\\\text{seen a}}}{f}, \underset{\substack{\text{on another}\\\text{path}}}{\Box}\})$$

$$\begin{array}{llll} \Delta_{1a}: & i \mapsto \{(f, \Box), (\Box, f)\} & \qquad \Delta_{1b}: & i \mapsto \{(i, \Box), (\Box, i)\} \\ & f \mapsto \{(f, \Box), (\Box, f)\} & & f \mapsto \{(i, \Box), (\Box, i)\} \\ & \Box \mapsto \{(\Box, \Box)\} & & \Box \mapsto \{(\Box, \Box)\} \end{array}$$

The "guessed" path is labelled by states from $\{i, f\}$ with label f immediately after a node with a was reached. Thus the path in the run contains infinitely many f's iff on the corresponding path in the tree there are infinitely many a's. The other paths are labelled with $\Box$ except for a finite initial segment. Thus, such paths in the run contain infinitely often $\Box$.

Example 7.6 $\Sigma = \{a, b\}$ and

$$L_2 = \{t \in T_\Sigma^\omega \mid \text{ every path in } t \text{ contains only finitely many } a\text{'s}\}.$$

Obviously, $L_2 = T_\Sigma^\omega \setminus L_1$. The following is a Rabin–tree automaton for L_2:

$$\mathcal{A}_2 = (\{i, f\}, \Sigma, \{i\}, \Delta_2, \{(\{i, f\}, \Omega_2\}) \text{ where}$$

$$\begin{array}{lcl} \Delta_{2a} & : & \begin{array}{l} i \mapsto \{(f, f)\} \\ f \mapsto \{(f, f)\} \end{array} \end{array} \qquad f \text{ means "have seen } a\text{"}$$

$$\begin{array}{lcl} \Delta_{2b} & : & \begin{array}{l} i \mapsto \{(i, i)\} \\ f \mapsto \{(i, i)\} \end{array} \end{array} \qquad i \text{ means "have seen } b\text{"}$$

$$\Omega_2 \quad = \quad \{(\underbrace{\{i, f\}}_{\substack{\text{no condition}\\ \text{on states seen}\\ \text{infinitely often}}} , \underbrace{\{f\}}_{\substack{f \text{ restricted to}\\ \text{occur finitely}\\ \text{often in each path}}})\}$$

For every path, this path contains infinitely many a's iff in the corresponding run this path contains infinitely many f's.

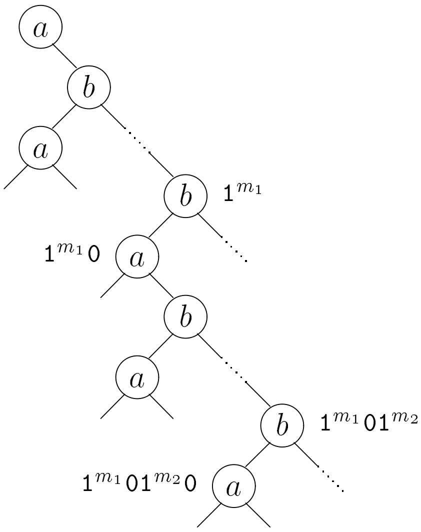
Proposition 7.7 The language L_2 of Example 7.6 is Rabin–recognizable but *not* Büchi–recognizable.

Proof: It remains to show that L_2 is not Büchi–recognizable. Assume that $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ is a Büchi tree–automaton for L_2. Let n be such $|Q| < n$. We construct a tree $t^{(n)} : \{0, 1\}^* \to \Sigma$ as follows:

$$t^{(n)}(u) \quad := \quad \begin{cases} a & u \in U_n \\ b & u \notin U_n \end{cases} \quad \text{where}$$

$$\begin{aligned} U_n \quad := \quad & \{\varepsilon\} \cup \{\mathtt{1}^{m_1}\mathtt{0} \mid m_1 > 0\} \cup \{\mathtt{1}^{m_1}\mathtt{0}\mathtt{1}^{m_2}\mathtt{0} \mid m_1 > 0, m_2 > 0\} \\ & \cup \ldots \cup \{\mathtt{1}^{m_1}\mathtt{0}\mathtt{1}^{m_2}\mathtt{0} \cdots \mathtt{1}^{m_n}\mathtt{0} \mid m_1 > 0 \ldots m_n > 0\} \end{aligned}$$

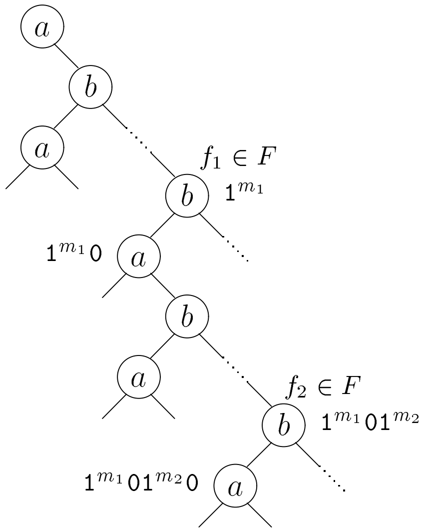
This tree contains infinitely many a's, but in every path there are at most $n + 1$ a's. Thus, $t^{(n)} \in L_2$.



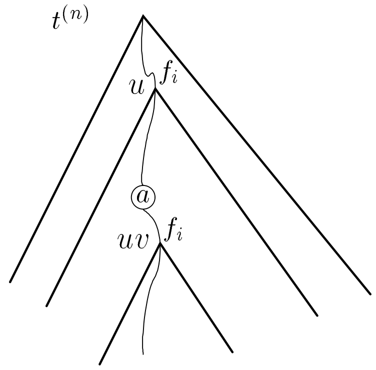
To reach an a that is not at the root, we must go at least once to the right, and then to the left. The next a is reached in the same way. After going n times to the left, the final a is reached.

Since $t^{(n)} \in L_2$, there is a successful run ℓ of $\mathcal{A}$ on $t^{(n)}$. We use ℓ to construct a path in $t^{(n)}$:

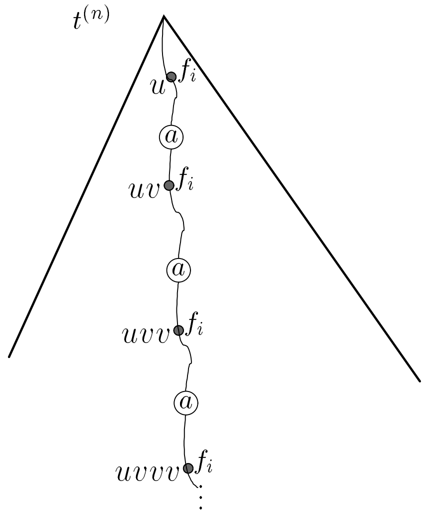
- let $m_1 > 0$ be minimal with $\ell(1^{m_1}) = f_1 \in F$. Such a final state exists since every path in ℓ contains infinitely many final states.
- assume that $m_1, m_2, \ldots, m_i > 0$ $(i < n)$ are already defined. Let $m_{i+1} > 0$ be minimal with $\ell(1^{m_1}01^{m_2}0 \cdots 1^{m_i}01^{m_{i+1}}) = f_{i+1} \in F$. This defines $m_1, \ldots, m_n > 0$ such that the following holds:



Since $|Q| < n$, there are $i < j$ such that $f_i = f_j$. Thus, we have the following situation:



If we replace in $t^{(n)}$ the tree at position uv by $t_u^{(n)}$, then we get a new tree which still has a successful run. If we iterate this an infinite number of times, we obtain a tree that is also accepted by $\mathcal{A}$ and has a path containing infinitely many a's. ↯ ∎



Corollary 7.8 The class of Büchi–recognizable ω–tree–languages is *not* closed under complement.

Proof: L_1 is Büchi–recognizable, but $L_2 = T_\Sigma^\omega \setminus L_1$ is not. ∎

Proposition 7.9 The class of Rabin–recognizable tree–languages is closed under complement.

The proof is quite involved. There are several approaches for proving this (Handbook article by W. Thomas). Why is this harder to prove than for Büchi–automata on words? The reason lies in the quantifier on paths in the definition of a successful run:

"*For all paths* the acceptance condition is satisfied."

If we negate this, we obtain:

"There *exists a path*, such that the acceptance condition is *not satisfied*."

In addition to transforming "*not satisfied*" into "*satisfied*", one must also transform "*exists a path*" into "*for all paths*".

7.2 Decidability results

Goal: reduce logical satisfiability problems to the emptiness problem for the automata.

Thus, we want the emptiness problem to be decidable. We first show decidability for Büchi tree–automata since the proof is simpler and also yields a characterization of Büchi–recognizable tree–languages.

Proposition 7.10 The emptiness problem for Büchi–recognizable ω–tree languages is decidable.

First we show another result, from which Prop. 7.10 can easily be deduced.

Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi tree–automaton, where $F = \{f_1, \ldots, f_m\}$. Let $\ell : \{\mathtt{0}, \mathtt{1}\}^* \to Q$ be a successful run of $\mathcal{A}$ on the tree $t \in T^{\omega}_{\Sigma}$. We decompose t into finite subtrees, which are accepted by automata working on finite trees.

Let $u \in \{\mathtt{0}, \mathtt{1}\}^*$. We are interested in where the run reaches for the first time a final state below u:

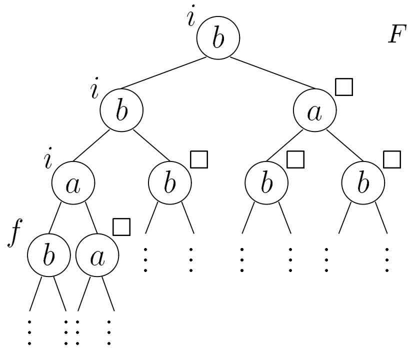
$$D_u := \{w \in \{\mathtt{0}, \mathtt{1}\}^* \mid \ell(uv) \notin F \text{ for all } \varepsilon < v \leq w\}$$

Since D_u is closed under prefix, it can be viewed as the domain of the tree. This tree is finitely branching and it does not containing an infinite path (otherwise ℓ would not be successful). König's Lemma implies that D_u is finite.

$$D_u^+ := D_u \cup \{w\sigma \mid \sigma \in \{\mathtt{0}, \mathtt{1}\} \wedge w \in D_u \wedge w\sigma \notin D_u\}$$

By definition of D_u we have $\ell(w\sigma) \in F$ for all $w\sigma \in D_u^+ \setminus D_u$.

Example: consider a run of the Büchi tree–automaton of Ex. 7.5 on the following tree:



$F = \{\square, f\}$

$$\begin{array}{rcl} D_\varepsilon & = & \{\varepsilon, \mathtt{0}, \mathtt{00}\} \\ D_\varepsilon^+ \setminus D_\varepsilon & = & \{\mathtt{1}, \mathtt{01}, \mathtt{000}, \mathtt{001}\} \end{array} \qquad \begin{array}{rcl} D_1 & = & \{\varepsilon\} \\ D_1^+ \setminus D_1 & = & \{\mathtt{0}, \mathtt{1}\}. \end{array}$$

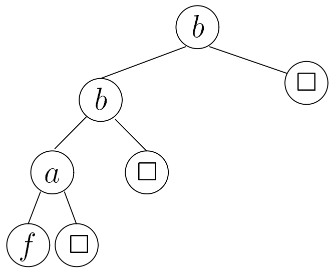
For $u \in \{\mathtt{0}, \mathtt{1}\}^*$ we define the *finite* tree $\widehat{t}_u : D_u^+ \to \Sigma \cup F$

$$\widehat{t}_u(w) \quad = \quad \begin{cases} t(uw) & \text{if} \quad w \in D_u \\ \ell(uw) & \text{if} \quad w \in D_u^+ \setminus D_u \end{cases}$$

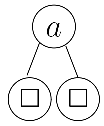
Thus, $\widehat{t}_u \in T_{\Sigma \cup F}$, where $f \in F$ is a symbol of arity 0.

Example:

$\hat{t}_\epsilon$



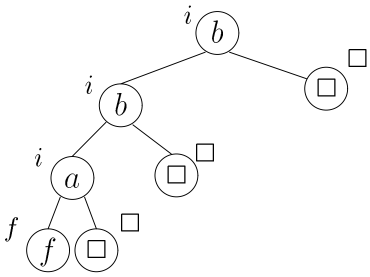
$\hat{t}_1$



For every $q \in Q$ we define the RL–tree automaton $\mathcal{A}_q = (Q, \Sigma \cup F, \{q\}, \Delta, \widehat{F})$ where $\widehat{F}(f) = f$ for all $f \in F$.

Let $L_q := L(\mathcal{A}_q) \subseteq T_{\Sigma \cup F}$. Then the following holds: If $\ell(u) = q$, then $\widehat{t}_u \in L_q$.

Example:



successful run of $\mathcal{A}_i$

This shows that:

$$t \in \widehat{L} := (\bigcup_{i \in I} L_i) \cdot^{(f_1,\ldots,f_m)} (L_{f_1},\ldots,L_{f_m})^{\omega,\,(f_1,\ldots,f_m)}$$

Conversely, it is easy to see that any element of $\widehat{L}$ belongs to $L_\omega(\mathcal{A})$. Thus, we have shown that $\widehat{L} = L_\omega(\mathcal{A})$.

Proposition 7.11 For an ω–tree language $L \subseteq T_\Sigma^\omega$ the following are equivalent:

1. L is Büchi–recognizable
2. There are recognizable tree languages $L_0,\ldots,L_m \subseteq T_{\Sigma\,\cup\,F}$ for some alphabet $F = \{f_1,\ldots,f_m\}$ of symbols of arity 0 such that:

$$L = L_0 \cdot^{(f_1,\ldots,f_m)} (L_1,\ldots,L_m)^{\omega,(f_1,\ldots,f_m)}$$

Proof:

"$1 \Rightarrow 2$" we have just shown.

"$2 \Rightarrow 1$" use RL–tree automata for $L_0,\ldots,L_m$ to construct a Büchi tree–automaton for L (exercise). ∎

Proof of Prop 7.10: Let $\mathcal{A} = (Q, \Sigma, I, \Delta, F)$ be a Büchi tree–automaton for L. Let the tree languages L_q be defined as above. We successively eliminate states that cannot occur in a successful run.

1. The automaton $\mathcal{A}_1 = (Q_1, \Sigma, I_1, \Sigma_1, F_1)$ is obtained from $\mathcal{A}$ by eliminating those states $q \in Q$ with $L_q = \emptyset$. Since $\mathcal{A}_q$ is an automaton on finite trees, $L_q = L(\mathcal{A}_q) = \emptyset$ is decidable (Prop 6.22). To be more precise:

$$\begin{array}{lcl} Q_1 & = & Q \setminus \{q \in Q \mid L_q = \emptyset\} \\ I_1 & = & I \cap Q_1 \\ F_1 & = & F \cap Q_1 \\ \Delta_{1a} & : & Q_1 \to 2^{Q_1 \times Q_1} : q \mapsto \Delta_a(q) \cap Q_1 \times Q_1, \end{array}$$

   Why does $L_q = \emptyset$ imply that q cannot occur on a successful run? If ℓ is a successful run and $\ell(u) = q$, then $\widehat{t_u} \in L_q$, and thus $L_q = \emptyset$. This shows that $L_\omega(\mathcal{A}_1) = L_\omega(\mathcal{A})$.

2. By iterating this, we obtain a sequence $\mathcal{A}_1, \mathcal{A}_2, \ldots$ of Büchi tree–automata with $L_\omega(\mathcal{A}_i) = L_\omega(\mathcal{A})$. Since Q is finite, this sequence becomes stable after a finite number of steps, i.e. one reaches an automaton $\mathcal{A}_n$ such that

   $L_\omega(\mathcal{A}) = L_\omega(\mathcal{A}_n)$ and $L_q \neq \emptyset$ for all $q \in Q_n$ (note: $Q_n = \emptyset$ is possible).

3. We **claim**: $L_\omega(\mathcal{A}_n) \neq \emptyset$ iff $I_n \neq \emptyset$.

   Proof of the claim: We know from Prop 7.11:

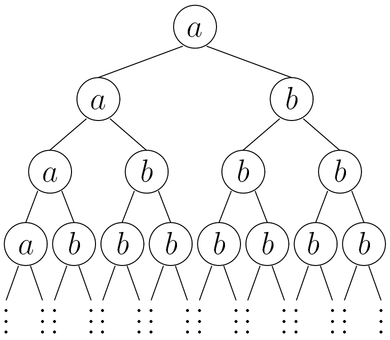
$$L_\omega(\mathcal{A}_n) = \left(\bigcup_{i \in I_n} L_i \right) \cdot^{(f_1, \ldots, f_m)} (L_{f_1}, \ldots, L_{f_m})^{\omega, (f_1, \ldots, f_m)},$$

   Obviously, $I_n = \emptyset$ implies that this expression is empty. If $I_n \neq \emptyset$, then $\bigcup_{i \in I} L_i \neq \emptyset$ since $L_i \neq \emptyset$ for all $i \in I_n$. $I_n \neq \emptyset$ also implies that $F_n \neq \emptyset$ (since for $F_n = \emptyset$ all the sets L_q are empty). Thus, there are trees $t_0 \in \bigcup_{i \in I_n} L_i, t_1 \in L_{f_1}, \ldots, t_m \in L_{f_m}$ for $F_n = \{f_1, \ldots, f_m\} \neq \emptyset$. But then the tree

$$\{t_0\} \cdot^{(f_1, \ldots, f_m)} (\{t_1\}, \ldots, \{t_m\})^{\omega, (f_1, \ldots, f_m)}$$

   belongs to $L_\omega(\mathcal{A})$. ■

Example: automaton from Example 7.5:



$$\in \{a(f,\Box)\} \cdot^{(f,\Box)} (\{a(f,\Box)\},\{b(\Box,\Box)\})^{\omega,(f,\Box)}$$

Proposition 7.12 The emptiness problem for Rabin–recognizable ω–tree languages is decidable.

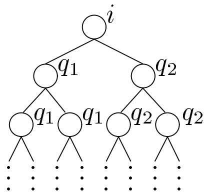
Proof: Let $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$ be a Rabin–tree automaton. A state $q \in Q$ is called *active* iff it can be reached from some state and does not only reproduce itself. To be more precise: $q \in Q$ is active if there exist $a, b \in \Sigma$ and states $q_0, q_1, q_2, q' \in Q$ such that

- $(q, q') \in \Delta_b(q_0)$ or $(q', q) \in \Delta_b(q_0)$ (reachable)
- $(q_1, q_2) \in \Delta_a(q)$ where $\{q_1, q_2\} \neq \{q\}$ (does not reproduce itself)

Otherwise, q is *passive*. Passive states allow only transitions of the form $\Delta_a(q) = \{(q, q)\}$ or they can only occur at the root of a successful run. Note that it is obviously decidable whether a given state is active or not. We show decidability of the emptiness problem by induction on the number of active states in $\mathcal{A}$.

Induction base: no active states

A successful run then has the following form:



This means that there are states $i, q_1, q_2 \in Q$ such that

- there are $a, b, c \in \Sigma$ with
  - $(q_1, q_2) \in \Delta_a(i)$
  - $(q_1, q_1) \in \Delta_b(q_1)$
  - $(q_2, q_2) \in \Delta_c(q_2)$
- $i \in I$
- there are $(F, G), (F', G') \in \Omega$ with $q_1 \in F$ and $q_1 \notin G$, $q_2 \in F'$ and $q_2 \notin G'$.

Obviously it is decidable whether such a triple i, q_1, q_2 exists.

Induction step: $n > 0$ active states

For a successful run $\ell : \{\mathtt{0}, \mathtt{1}\}^* \to Q$ there are three possibilities.

Case 1: one of the active states does not occur in ℓ

Then ℓ is also a successful run of the automaton $\mathcal{A}_q^-$ obtained from $\mathcal{A}$ by deleting q.

$$\mathcal{A}_q^- = (Q', \Sigma, I \cap Q', \Delta', \Omega') \text{ where}$$

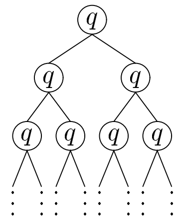
$$\begin{aligned}
Q' &= Q \setminus \{q\} \\
\Delta'_a &: Q' \to 2^{Q' \times Q'} \quad : \quad \Delta'_a(p) = \Delta_a(p) \cap (Q' \times Q') \\
\Omega' &= \{(F', G') \mid \\
&\qquad \text{there is } (F, G) \in \Omega \text{ such that } F' = F \cap Q' \text{ and } G' = G \cap Q'\}
\end{aligned}$$

It can be decided whether this case holds for some successful run by considering for every state q the automaton $\mathcal{A}_q^-$ and then deciding the emptiness problem for $\mathcal{A}_q^-$ (by induction this is decidable since $\mathcal{A}_q^-$ has less active states).

Case 2: In ℓ there is a node u such that $\ell(u) = q$ and q is active and the subtree ℓ_u does not contain the active state q' (except possibly at the root if $q = q'$).

Let $\widehat{\ell}_q$ be the "tree" obtained from ℓ by pruning all branches immediately below the first occurrence of q in each path. (Note: $\widehat{\ell}_q$ may have both finite and infinite paths). Then $\widehat{\ell}_q \cdot^q \ell_u$ is still a successful run! How can we test whether such runs $\widehat{\ell}_q$ and ℓ_u exist?

1. Existence of $\widehat{\ell}_q$
   $\widehat{\ell}_q$ is modified to $\tilde{\ell}_q$ by replacing the leafs labelled with q by the tree:

   An automaton that has $\tilde{\ell}_q$ as successful run can be obtained as follows:
   $$\tilde{\mathcal{A}}_q = (Q, \Sigma, I, \Delta', \Omega') \text{ where}$$
   - $\Delta'_a(q) = \{(q, q)\}$ and $\Delta'_a(p) = \Delta_a(p)$ for all $p \neq q$
   - $\Omega' = \Omega \cup \{(\{q\}, \emptyset)\}$

   We have $L_\omega(\tilde{\mathcal{A}}_q) \neq \emptyset$ iff there exists a run $\tilde{\ell}_q$. Obviously $\tilde{\mathcal{A}}_q$ has one active state less than $\mathcal{A}$, and thus $L_\omega(\tilde{\mathcal{A}}_q) \neq \emptyset$ is decidable by induction.

2. ℓ_u is a successful run of the automaton $\mathcal{A}^-_{q,\,q'}$, which is obtained from $\mathcal{A}$ by removing q' from Δ. To be more precise:
   $$\begin{aligned} \mathcal{A}^-_{q,\,q'} &= (Q, \Sigma, \{q\}, \Delta', \Omega) \text{ where} \\ & \quad \Delta'_a(p) = \Delta_a(p) \cap (Q' \times Q') \text{ for all } a \in \Sigma, p \in Q \text{ where} \\ & \quad Q' = Q \setminus \{q'\} \end{aligned}$$

   Again, $\mathcal{A}^-_{q,\,q'}$ has one active state less than $\mathcal{A}$ since q' is no longer active.

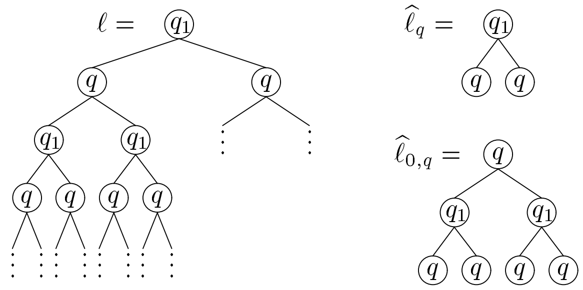
By induction, we can test for all pairs (q, q') of active states, whether $\tilde{\mathcal{A}}_q$ and $\mathcal{A}^-_{q,\,q'}$ accept non–empty languages. If this is the case, then $\mathcal{A}$ has a successful run satisfying Case 2.

Case 3: There is at least one active state and below every active state in ℓ every other active state occurs.

Thus, there is a path π in ℓ such that *every* active state occurs infinitely often on π. Except at the beginning of π, no passive states can occur on π. There must be a pair $(F_0, G_0) \in \Omega$ that accepts π. Thus, F_0 contains at least one active state and none of the active states occurs in G_0 (i.e. G_0 contains only passive states). Let $q \in F_0$ be an active state.

The "tree" $\widehat{\ell}_q$ is defined as in Case 2. Let $u \in \{0,1\}^*$ be such that $\ell(u) = q$. The tree $\widehat{\ell}_{u,q}$ is obtained from ℓ_u by pruning below every occurrence of q that is *not* at the root.

Example:



Obviously, $\widehat{\ell}_q \cdot^q \widehat{\ell}_{u,q}{}^{\omega,q}$ is a run of $\mathcal{A}$. Why is it successful? Let π be a path in this run.

Case a: π is an infinite path in $\widehat{\ell}_q$ or an infinite final segment of π belongs to $\widehat{\ell}_{u,q}$. Then this path is accepted by some pair in Ω since an infinite final segment of it also occurs in a path in ℓ.

Case b: Otherwise, q occurs infinitely often in π. In addition, a passive state can only occur at the beginning of π. Thus, (F_0, G_0) accepts π.

Existence of $\widehat{\ell}_q$ can be tested as shown in Case 2.

Existence of $\widehat{\ell}_{u,q}$: Since q in $\widehat{\ell}_{u,q}$ has two different functions (at the root and the leafs), we rename q at the root to a new state q_0. $\widehat{\ell}\,^{q_0}_{u,q}$ is obtained from $\widehat{\ell}_{u,q}$ by labelling the root with q_0. As in the second case, we modify $\widehat{\ell}\,^{q_0}_{u,q}$ to $\tilde{\ell}\,^{q_0}_{u,q}$. The automaton $\tilde{\mathcal{A}}_{q_0,q} = (Q \cup \{q_0\}, \Sigma, \{q_0\}, \Delta', \Omega')$ with

- $q_0 \notin Q$
- $$\begin{aligned} \Delta'_a(q_0) &:= \Delta_a(q) \\ \Delta'_a(q) &:= \{(q,q)\} \\ \Delta'_a(q') &:= \Delta_a(q') \text{ for } q' \notin \{q, q_0\} \end{aligned}$$

- $\Omega' = \Omega \cup \{(\{q\}, \emptyset)\}$

has $\tilde{\ell}\,^{q_0}_{u,q}$ as a successful run. It has one active state less than $\mathcal{A}$ since q, q_0 are passive.

To sum up:

For a Rabin–tree automaton $\mathcal{A}$ we have $L_\omega(\mathcal{A}) = \emptyset$ iff the following holds:

- If $\mathcal{A}$ does not contain active states, then there does not exists a triple $i, q_1, q_2 \in Q$ such that
  - $i \in I$ and
  - there is $a, b, c \in \Sigma$ with $(q_1, q_2) \in \Delta_a(i)$, $(q_1, q_1) \in \Delta_b(q_1)$, $(q_2, q_2) \in \Delta_c(q_2)$
  - there exist $(F, G), (F', G') \in \Omega$ such that $q_1 \in F$ and $q_1 \notin G$, $q_2 \in F'$, $q_2 \notin G'$
- If $\mathcal{A}$ contains at least one active state, then for all active states q:
  - $L(\mathcal{A}_q^-) = \emptyset$
  - $L(\tilde{\mathcal{A}}_q) = \emptyset$ or $L(\mathcal{A}_{q,q'}^-) = \emptyset$ for all active states q', and
  - if there is a pair $(F_0, G_0) \in \Omega$ with $q \in F_0$ and G_0 contains only passive states, then $L(\tilde{\mathcal{A}}_q) = \emptyset$ or $L(\tilde{\mathcal{A}}\,_{q_0,q}) = \emptyset$. ∎

Chapter 8

Tree–automata and logical formulae

8.1 S2S logic

We extend S1S to a logic with 2 successor functions. Instead of the interpretation domain $\mathbb{N}$ (ω) we use the infinite binary tree.

Definition 8.1

1. Formulae of the logic S2S are built like formulae of S1S, with the only difference that
   - the successor function s is replaced by two successor functions s_0 and s_1.
   - The constant $\underline{0}$ is replaced by the constant $\underline{\varepsilon}$.
2. As interpretation domain we take the set $\{0,1\}^*$ (the domain of infinite binary trees), where
   - $\underline{\varepsilon}$ is interpreted as ε (the root)
   - s_0,s_1 are interpreted as $\mathrm{s}_0 : u \mapsto u\mathtt{0}$, $\mathrm{s}_1 : u \mapsto u\mathtt{1}$

- $<$ is interpreted as the prefix order on $\{\mathtt{0},\mathtt{1}\}^*$ i.e. $u < v$ iff $\exists w \in \{\mathtt{0},\mathtt{1}\}^+.\ uw = v$
- $P_1, \ldots, P_n$ are interpreted as subsets of $\{\mathtt{0},\mathtt{1}\}^*$

S2S–formulae can be used to define ω–tree languages. As in the case of S1S we use the alphabet $\Sigma = \{0,1\}^n$. Every element of Σ has arity 2. An S2S interpretation I can be viewed as an ω-tree t_I with labels from Σ:

$$t_I(u) := (b_1, \ldots, b_n) \text{ where } b_i = \begin{cases} 1 & u \in P_i^I \\ 0 & u \notin P_i^I. \end{cases}$$

Definition 8.2 Let φ be a closed S2S–formula. Then,

$$L_\omega(\varphi) := \{t_I \in T_\Sigma^\omega \mid I \models \varphi\}.$$

Some examples of S2S–formulae:

- Chain(X) : describes subsets X of $\{\mathtt{0},\mathtt{1}\}^*$ such that all elements are prefix–comparable.

  $$\text{Chain}(X) : \forall x. \forall y. (X(x) \wedge X(y) \Rightarrow x < y \vee x = y \vee y < x)$$

- Path(X) : paths are maximal chains (no holes)

  $$X \subseteq Y : \forall x. X(x) \Rightarrow Y(x)$$

  $$X = Y : \forall x. X(x) \Leftrightarrow Y(x)$$

  $$\text{Path}(X) : \text{Chain}(X) \wedge \forall Y. (X \subseteq Y \wedge \text{Chain}(Y) \Rightarrow X = Y)$$

- InfiniteChain(X)

  $$\text{InfiniteChain}(X) : \text{Chain}(X) \wedge \forall x. (X(x) \Rightarrow \exists y. (x < y \wedge X(y)))$$

- $Z = \text{PrefixClosure}(X)$

  $$\forall z. (Z(z) \Leftrightarrow \exists x. (X(x) \wedge (z \leq x)))$$

- Finite(X)

  $$\text{Finite}(X) : \forall Z. (Z = \text{prefixClosure}(X) \Rightarrow \neg \exists Y. (Y \subseteq Z \wedge \text{InfiniteChain}(Y)))$$

Why does this express finiteness of X ?

- If X is infinite, then so is its prefix closure Z. Thus Z can be viewed as infinite tree. By König's lemma it contains an infinite path Y.
- If X is finite, then its prefix closure is finite, and thus cannot contain an infinite chain.

Example 8.3 $n = 1$, i.e. $\Sigma = \{0, 1\}$.

1. $L_1 = \{t \in T^\omega_\Sigma \mid \text{there is a path in } t \text{ containing infinitely many 1's}\}$ (see Example 7.5)

$$\varphi_1 = \exists Y.\mathrm{Path}(Y) \wedge \forall x.(Y(x) \Rightarrow \exists y.(Y(y) \wedge x < y \wedge P_1(y)))$$

2. $L_2 = \overline{L_1}$

$$L_2 = L_\omega(\neg\varphi_1)$$

Proposition 8.4 (Rabin) For an ω–tree language $L \subseteq T^\omega_\Sigma$ the following are equivalent

1. L is Rabin–recognizable.
2. $L = L_\omega(\varphi)$ for a closed S2S–formula φ.

Proof: very similar to the proof of Prop. 5.4.

"1 $\Rightarrow$ 2" Let $\mathcal{A} = (Q, \Sigma, I, \Delta, \Omega)$ be a Rabin–tree automaton with $Q = \{q_1, \ldots \ldots, q_m\}$. For every q_j we introduce a second–order variable Y_j with the intended interpretation:

$$x \text{ belongs to } Y_j \text{ if the run labels } x \text{ by } q_j.$$

The existence of a successful run can be expressed as follows:

$$
\begin{aligned}
&\exists Y_1 \ldots \exists Y_m. \\
&\quad \Big(\bigvee_{q_i \in I} Y_i(\underline{\varepsilon}) \wedge \\
&\quad \forall x. \bigwedge_{i \neq j} \neg (Y_i(x) \wedge Y_j(x)) \wedge \\
&\quad \forall x. \bigvee_{\substack{(q_i, q_j) \in \Delta_a(q_k) \\ a \in \Sigma}} Y_k(x) \wedge Q_a(x) \wedge Y_i(\mathrm{s}_0(x)) \wedge Y_j(\mathrm{s}_1(x)) \wedge \\
&\quad \forall Z. \mathrm{Path}(Z) \Rightarrow \\
&\qquad \bigvee_{(F,G) \in \Omega} \Big(\bigvee_{q_i \in F} \forall x. (Z(x) \Rightarrow \exists y. (Z(y) \wedge x < y \wedge Y_i(y))) \wedge \\
&\qquad\qquad \bigwedge_{q_i \in G} \exists x. \forall y. (Z(y) \wedge x < y \Rightarrow \neg Y_i(y))) \Big)
\end{aligned}
$$

“2 ⇒ 1” As in the case of S1S we reduce S2S–formulae to $\mathrm{S2S}_0$–formulae. Then the proof is by induction on the structure of $\mathrm{S2S}_0$–formulae. For the induction step one uses closure under Boolean operations and alphabet renaming for Rabin–recognizable languages. ∎

Corollary 8.5 Validity in S2S is decidable.

Instead of binary trees one can also consider k–ary trees ($k \geq 1$). The results for $k = 2$ can easily be generalized to arbitrary k. In particular, SkS (k successor functions) is decidable.