Automata and Grammars

Prof. Dr. Friedrich Otto

Fachbereich Elektrotechnik/Informatik, Universität Kassel
34109 Kassel, Germany

E-mail: f.otto@uni-kassel.de

SS 2018

Lectures and Seminary SS 2018

Lectures:

Thursday 9:00 - 10:30, Room S 11

Start: Thursday, February 22, 2018, 9:00.

Seminary:

Thursday 10:40 - 12:10, Room S 11

Start: Thursday, February 22, 2018, 10:40.

Exercises:

From Thursday to Thursday next week!

To pass seminary:
At least two successful presentations in class (10 %)
and passing of midtem exam on April 5 (10:40 - 11:40) (20 %)

Final Exam: (dates have been fixed!)
Written exam (2 hours) on May 31 (9:00 - 11:30) in S 11

Oral exam (30 minutes per student) on June 14 (9:00 - 12:30) in S 10

Written exam (2. try) on June 21 (9:00 - 11:30) in S 11

Oral exam (2. try) on June 28 (9:00 - 12:30) in S 11

The written exam accounts for 50 % of the final grade, while the oral exam accounts for 20 % of the final grade.

Moodle: Course ”Automata and Grammars” NTIN071.

Literature:

P.J. Denning, J.E. Dennis, J.E. Qualitz;
Machines, Languages, and Computation.
Prentice-Hall, Englewood Cliffs, N.J., 1978.

M. Harrison; *Introduction to Formal Language Theory.*
Addison-Wesley, Reading, M.A., 1978.

J.E. Hopcroft, R. Motwani, J.D. Ullman; *Introduction to Automata Theory, Languages, and Computation.*
Addison-Wesley, Boston, 2nd. ed., 2001.

H.R. Lewis, Ch.H. Papadimitriou;
Elements of the Theory of Computation.
Prentice Hall, Englewood Cliffs, N.J., 1981.

G. Rozenberg, A. Salomaa (editors);
Handbook of Formal Languages, Vol.1: Word, Language, Grammar. Springer, Heidelberg, 1997.

1. Introduction

Formal Languages (not natural languages):
A formal language is a set of (finite) sequences of symbols (words, strings) from a fixed finite set of symbols (alphabet).

How to describe a formal language? There are various options:

- by a complete enumeration, but:

$$L = \{ w \in \{a, b, c\}^* \mid |w| = 20 \} : |L| = 3^{20} \sim 3.5 \times 10^9$$

- by a mathematical expression, but: how to check that a given word belongs to the language described?

- by a generative device: a grammar or a rewriting system.
A grammar tells you how to generate (all) the words of a language!
- by an analytical device: an automaton or an algorithm.
An automaton recognizes exactly the words of a given language!

The Roots of the Theory of Formal Languages:

- Combinatorics on Words (A. Thue 1906, 1912)
- Semigroup and Group Theory (M. Dehn 1911)
- Logic (A. Turing 1926, E. Post 1936, A. Church 1936)

Church's Thesis

A function is effectively computable iff there is a Turing machine that computes this function.

Let $f : \Sigma^* \rightsquigarrow \Gamma^*$ be a (partial) function.

$dom(f)$	$=$	$\{ u \in \Sigma^* \mid f(u) \text{ is defined} \}$:	domain
$range(f)$	$=$	$\{ v \in \Gamma^* \mid \exists u \in \Sigma^* : f(u) = v \}$:	range
$ker(f)$	$=$	$\{ u \in \Sigma^* \mid f(u) = \varepsilon \}$:	kernel
$graph(f)$	$=$	$\{ u\#f(u) \mid u \in dom(f) \}$:	graph

f is computable iff there is an "algorithm" that accepts the language $graph(f)$.

Classes of Automata: Restrictions of the Turing machine

- Linguistics	(N. Chomsky 1956)	:	phrase structure grammars
- Biology	(A. Lindenmayer 1968)	:	L-systems
	(T. Head 1987)	:	DNA-computing, H-systems
	(G. Paun 1999)	:	Membrane computing, P-systems

Overview:

Chapter 2: Regular Languages and Finite Automata
Chapter 3: Context-free Languages and Pushdown Automata
Chapter 4: Turing Machines and Recursively Enumerable and Context-Sensitive Languages
Chapter 5: Summary

Chapter 2:

Regular Languages and Finite Automata

2.1. Words, Languages, and Morphisms

An alphabet Σ is a finite set of symbols or letters.
For all $n \in \mathbb{N}$: $\Sigma^n = \{u : [1, n] \to \Sigma\}$: set of words of length n over Σ, that is, $u = (u(1), u(2), \ldots, u(n)) = u_1 u_2 \ldots u_n$.

$\Sigma^0 = \{\varepsilon\}$: $\varepsilon =$ empty word

$\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n$: set of all non-empty words over Σ

$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$: set of all words over Σ

The concatenation $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$ is defined through $u \cdot v = uv$.
For $u \in \Sigma^m$ and $v \in \Sigma^n$, $uv \in \Sigma^{m+n}$.

The length function $|.| : \Sigma^* \to \mathbb{N}$ is defined through
$|u| = n$ for all $u \in \Sigma^n, n \geq 0$.

Lemma 2.1

(a) *The operation of concatenation is associative, that is,* $(u \cdot v) \cdot w = u \cdot (v \cdot w)$.

(b) *For all* $u \in \Sigma^*$, $u \cdot \varepsilon = \varepsilon \cdot u = u$.

(c) *If* $u \cdot v = u \cdot w$, *then* $v = w$ *(Left cancellability).*

(d) *If* $u \cdot w = v \cdot w$, *then* $u = v$ *(Right cancellability).*

(e) *If* $u \cdot v = x \cdot y$, *then exactly one of the following cases holds:*

(1) $|u| = |x|$, $u = x$, *and* $v = y$.

(2) $|u| > |x|$, *and there exists* $z \in \Sigma^+$ *such that* $u = x \cdot z$ *and* $y = z \cdot v$.

(3) $|u| < |x|$, *and there exists* $z \in \Sigma^+$ *such that* $x = u \cdot z$ *and* $v = z \cdot y$.

Abbreviations:

uv stands for $u \cdot v$.

$u^0 = \varepsilon, u^1 = u, u^{n+1} = u^n u$ for all $u \in \Sigma^*, n \geq 1$.

Further Basic Notions:

The mirror function ${}^R : \Sigma^* \to \Sigma^*$ is defined through

$$\varepsilon^R = \varepsilon, (ua)^R = au^R \text{ for all } u \in \Sigma^*, a \in \Sigma.$$

Hence, $(abbc)^R = c(abb)^R = cb(ab)^R = cbba$.

If $uv = w$, then u is a prefix and v is a suffix of w.

If $u \neq \varepsilon$ ($v \neq \varepsilon$), then v is a proper suffix (proper prefix) of w.

If $uvz = w$, then v is a factor of w.

It is a proper factor, if $uz \neq \varepsilon$.

A language L over Σ is a subset $L \subseteq \Sigma^*$.

The cardinality of L is denoted by $|L|$.

Let $L, L_1, L_2 \subseteq \Sigma^*$.

$L_1 \cdot L_2 = \{ uv \mid u \in L_1, v \in L_2 \}$ is the product of L_1 and L_2.
It is also called the concatenation of L_1 and L_2.

$L^0 = \{\varepsilon\},\ L^1 = L,\ L^{n+1} = L^n \cdot L$ for all $(n \geq 1)$.

$L^+ = \bigcup_{n \geq 1} L^n$ and $L^* = \bigcup_{n \geq 0} L^n$ are the plus closure and the star closure (Kleene closure) of L.

$L^R = \{ w^R \mid w \in L \}$ is the mirror language of L.

Example:

(a) $\Sigma_1 := \{0,1\}$: $\varepsilon, 0, 10, 110 \in \Sigma_1^*$.

$L_1 = \{\, 10^i \mid i \geq 0 \,\}$:

$L_1^R = \{\, 0^j 1 \mid j \geq 0 \,\}$ und $L_1 \cdot L_1^R = \{\, 10^i 1 \mid i \geq 0 \,\}$.

$L_2 = \{\, 0^n 1^n \mid n \geq 1 \,\}$.

(b) $\Sigma_2 := \{a, b, c\}$:

$L_3 = \{\, wcw^R \mid w \in \{a,b\}^* \,\}$: marked palindromes of even length

$L_4 = \{\, ww \mid w \in \{a,b\}^* \,\}$: copy language

$L_5 = \{\, w \mid |w|_a \equiv 0 \bmod 2 \text{ and } |w|_b \equiv 1 \bmod 3 \,\}$.

(c) $\Sigma_3 := \{0, 1, \ldots, 9, ., +, -, e\}$: $112.45, -23e + 17 \in \Sigma_3^+$

$L_6 = \{\text{unsigned integer in PASCAL}\}$

Let Σ and Δ be two alphabets.

A mapping $h : \Sigma^* \to \Delta^*$ is a morphism, if

the equality $h(uv) = h(u) \cdot h(v)$ holds for all $u, v \in \Sigma^*$.

Lemma 2.2

If $h : \Sigma^* \to \Delta^*$ is a morphism, then $h(\varepsilon) = \varepsilon$.

For a set S, 2^S denotes the power set of S.

A mapping $\varphi : \Sigma^* \to 2^{\Delta^*}$ is a substitution, if

the two following conditions are satisfied:

– $\forall u, v \in \Sigma^* : \varphi(uv) = \varphi(u) \cdot \varphi(v)$.
– $\varphi(\varepsilon) = \{\varepsilon\}$.

For $L \subseteq \Sigma^*$, $h(L) = \{ h(w) \mid w \in L \}$ and $\varphi(L) = \bigcup_{w \in L} \varphi(w)$.

Example:

Let $\Sigma = \{a, b, c\}$ and $\Delta = \{0, 1\}$.
If $h : \Sigma^* \to \Delta^*$ is defined through $a \mapsto 01,\ b \mapsto 1,\ c \mapsto \varepsilon$,
then $h(baca) = 10101$.

If $\varphi : \Sigma^* \to 2^{\Delta^*}$ is defined through
$$a \mapsto \{01, 001\},\ b \mapsto \{1^i \mid i \geq 1\},\ c \mapsto \{\varepsilon\},$$
then $\varphi(baca) = \{\, 1^i0101, 1^i01001, 1^i00101, 1^i001001 \mid i \geq 1 \,\}$.

A morphism h is called ε-free, if $h(a) \neq \varepsilon$ for all $a \in \Sigma$.

A substitution φ is called ε-free, if $\varepsilon \notin \varphi(a)$ for all $a \in \Sigma$.

A substitution φ is called finite, if $\varphi(a)$ is a finite set for all $a \in \Sigma$.

$h^{-1} : \Delta^* \to 2^{\Sigma^*}$ is defined through $h^{-1}(v) = \{\, u \in \Sigma^* \mid h(u) = v \,\}$.
$\varphi^{-1} : \Delta^* \to 2^{\Sigma^*}$ is defined through $\varphi^{-1}(v) := \{\, u \in \Sigma^* \mid v \in \varphi(u) \,\}$.

These reverse mappings are in general not substitutions!

2.2 Regulare Grammars

A semi-Thue system (string-rewriting system) on an alphabet Σ is a (finite) set S of pairs of words over Σ :

$$S = \{\ell_1 \to r_1, \ldots, \ell_n \to r_n\}\ (n \geq 0, \ell_1, \ldots, \ell_n, r_1, \ldots, r_n \in \Sigma^*).$$

S induces a number of binary relations on Σ^*:

- the single-step derivation relation $\to_S$:
  $u \to_S v$ iff $\exists(\ell \to r) \in S\ \exists x, y \in \Sigma^* : u = x\ell y$ and $v = xry$.
- the derivation relation $\to_S^*$:
  $u \to_S^0 v$ iff $u = v$.
  $u \to_S^1 v$ iff $u \to_S v$.
  $u \to_S^{n+1} v$ iff $\exists w \in \Sigma^* : u \to_S^n w$ and $w \to_S v$.
  $u \to_S^+ v$ iff $\exists n \geq 1 : u \to_S^n v$.
  $u \to_S^* v$ iff $\exists n \geq 0 : u \to_S^n v$.

Lemma 2.3

(a) *The relation $\to_S^*$ is the smallest reflexive and transitive binary relation on Σ^* that contains $\to_S$.*

(b) *If $u \to_S^* v$, then $xuy \to_S^* xvy$ for all $x, y \in \Sigma^*$.*

A phrase-structure grammar G is a 4-tuple $G = (N, T, S, P)$, where

- N is a finite alphabet of nonterminals (variables),
- T is a finite alphabet of terminals (terminal symbols), where $N \cap T = \emptyset$,
- $S \in N$ is the start symbol, and
- $P \subseteq (N \cup T)^* \times (N \cup T)^*$ is a finite semi-Thue system on $N \cup T$, the elements of which are called productions.
  For each production $(\ell \to r) \in P$, it is required that $|\ell|_N \geq 1$.

For $A \in N$,

$$\hat{L}(G, A) = \{ \alpha \in (N \cup T)^* \mid A \to_P^* \alpha \}$$

is the set of A-sentential forms, and

$$L(G, A) = \{ w \in T^* \mid A \to_P^* w \}$$

is the set of terminal words that are derivable from A.

$\hat{L}_G = \hat{L}(G, S)$ is the set of sentential forms that are derivable in G and $L_G = L(G, S)$ is the language generated by G.

The grammar $G = (N, T, S, P)$ is called left regular, if $\ell \in N$ and $r \in T^* \cup N \cdot T^*$ for each production $(\ell \to r) \in P$.

G is called right regular, if $\ell \in N$ and $r \in T^* \cup T^* \cdot N$ for each production $(\ell \to r) \in P$.

Finally, G is called regular, if it is left or right regular.

Example:

(a) $G_1 = (N, T, S, P)$, where $N = \{S, A\}$, $T = \{0, 1\}$ and $P = \{S \to 0A, A \to 10A, A \to \varepsilon\}$.

G_1 is right regular, and $L(G_1) = \{\, 0(10)^i \mid i \geq 0 \,\}$.

(b) $G_2 = (N, T, S, P)$, where $N = \{S\}$, $T = \{0, 1\}$ and $P = \{S \to S10, S \to 0\}$.

G_2 is left regular, and $L(G_2) = \{\, 0(10)^i \mid i \geq 0 \,\} = L(G_1)$.

A language $L \subseteq \Sigma^*$ is called regular, if there exists a regular grammar G such that $L_G = L$.

$\mathrm{REG}(\Sigma)$	=	set of regular languages on Σ
REG	=	class of all regular languages

Remark 2.4

(a) *Each finite language is regular.*

(b) *If L is a regular language, then so is its mirror language L^R.*

(c) *If $L \in \mathrm{REG}(\Sigma)$, and if $h : \Sigma^* \to \Delta^*$ is a morphism, then $h(L) \in \mathrm{REG}(\Delta)$, that is, the class* REG *is closed under morphisms.*

(d) REG *is also closed under finite substitutions.*

A right regular grammar $G = (N, T, S, P)$ is in right normal form if $r \in T \cdot N \cup T$ for each production $(l \to r) \in P$. In addition, G may contain the production $(S \to \varepsilon)$, if S does not occur on the right-hand side of any production.

If G is in right normal form, then it does not contain any productions of the following forms:

- $A \to \varepsilon$ $(A \neq S)$: ε-rule,
- $A \to B$ $(A, B \in N)$: chain rule,
- $A \to wB$ or $A \to w$, where $w \in T^*$, $|w| \geq 2$.

Theorem 2.5

From a right regular grammar G, a grammar $\hat{G}$ in right normal form can be constructed such that $L(\hat{G}) = L(G)$.

Proof of Theorem 2.5.

Let $G = (N, T, S, P)$ be a right regular grammar that is not in right normal form. First we eliminate the ε-rules from P.

(1) Determine the set $N_1 = \{ A \in N \mid A \to_P^* \varepsilon \}$:
$N_1^{(1)} := \{ A \in N \mid (A \to \varepsilon) \in P \}$,
$N_1^{(k+1)} := N_1^{(k)} \cup \{ A \in N \mid \exists B \in N_1^{(k)} : (A \to B) \in P \}$.
Then $N_1 = \bigcup_{k \geq 1} N_1^{(k)} = N_1^{(|N|)}$.

(2) Remove all ε-rules.

(3) For each production $B \to wA$, where $|w| > 0$ and $A \in N_1$, add the production $B \to w$.

(4) If $S \in N_1$, then introduce a new start symbol $\hat{S}$ and the productions $\hat{S} \to \varepsilon$ and $\hat{S} \to \alpha$ for each production $S \to \alpha$.

Example:

Let $G = (N, T, S, P)$, where $N = \{S, A, B, C, D\}$, $T = \{a, b\}$, and

$$\begin{aligned} P \;=\; \{&S \to \varepsilon, \quad S \to abA, \quad S \to B, \quad A \to abS, \quad A \to B,\\ &B \to C, \quad B \to b^3C, \quad B \to D, \quad C \to A, \quad C \to aab,\\ &D \to \varepsilon, \quad D \to a, \quad D \to aab, \quad D \to abD\}. \end{aligned}$$

Elimination of ε-rules:

(1) $N_1^{(1)} = \{S, D\}$, $N_1^{(2)} = \{S, D, B\}$, $N_1^{(3)} = \{S, D, B, A\}$,
$N_1^{(4)} = \{S, D, B, A, C\} = N = N_1$.

(2) Delete productions $S \to \varepsilon$ and $D \to \varepsilon$.

(3) Add productions $S \to ab$, $A \to ab$, $B \to bbb$, and $D \to ab$.

(4) Introduce $\hat{S}$ and $\hat{S} \to \varepsilon$, $\hat{S} \to abA$, $\hat{S} \to ab$, and $\hat{S} \to B$.

Then $G_1 = (N \cup \{\hat{S}\}, T, \hat{S}, P_1)$ is equivalent to G, where

$$\begin{aligned} P_1 \;=\; \{&\hat{S} \to \varepsilon, \quad \hat{S} \to abA, \quad \hat{S} \to ab, \quad \hat{S} \to B, \quad S \to abA,\\ &S \to ab, \quad S \to B, \quad A \to abS, \quad A \to ab, \quad A \to B,\\ &B \to C, \quad B \to b^3C, \quad B \to b^3, \quad B \to D, \quad C \to A,\\ &C \to aab, \quad D \to a, \quad D \to aab, \quad D \to abD, \quad D \to ab\}. \end{aligned}$$

Proof of Theorem 2.5 (cont.).

Next we eliminate the chain-rules from P_1.

(1) Two nonterminals $A, B \in N_1$ are called equivalent ($A \leftrightarrow B$) if $A \to_{P_1}^+ B$ and $B \to_{P_1}^+ A$.
For $A \in N_1$, $[A] = \{ B \in N_1 \mid A \leftrightarrow B \}$.
Pick $A' \in [A]$ and replace all $B \in [A]$ in P_1 by A'.
Remove resulting productions of the form $(A' \to A')$.

(2) Order the remaining nonterminals such that $(A \to B) \in P_1$ implies $A < B$.
If B is the largest nonterminal occurring on the right-hand side of a chain-rule $A \to B$, then delete this chain-rule and add productions $A \to \alpha$ for all productions $B \to \alpha$.

(3) Repeat (2) until no chain-rules are left.

Example (cont.):

P_1 contains the chain-rules

$$\hat{S} \to B,\ S \to B,\ A \to B,\ B \to C,\ B \to D, \text{ and } C \to A.$$

Hence, $A \leftrightarrow B \leftrightarrow C$ and $[A] = \{A, B, C\}$.

We pick A as representative and replace B and C by A:

$$\begin{aligned} P_2 = \{&\hat{S} \to \varepsilon, & &\hat{S} \to abA, & &\hat{S} \to ab, & &\hat{S} \to A, & &S \to abA,\\ &S \to ab, & &S \to A, & &A \to abS, & &A \to ab, & &A \to A,\\ &A \to A, & &A \to b^3A, & &A \to b^3, & &A \to D, & &A \to A,\\ &A \to aab, & &D \to a, & &D \to aab, & &D \to abD, & &D \to ab\}. \end{aligned}$$

To eliminate the chain-rule $A \to D$,
add the productions $A \to a$ and $A \to abD$.

To eliminate the chain-rules $\hat{S} \to A$ and $S \to A$, add corresponding productions with left-hand sides $\hat{S}$ and S.

Example (cont.):

This process yields $G_3 = (N_3, T, \hat{S}, P_3)$ with $N_3 = \{\hat{S}, S, A, D\}$ and

$$\begin{aligned} P_3 \quad = \{ & \hat{S} \to \varepsilon \mid abA \mid ab \mid abS \mid b^3A \mid b^3 \mid aab \mid a \mid abD, \\ & S \to abA \mid ab \mid abS \mid b^3A \mid b^3 \mid aab \mid a \mid abD, \\ & A \to abS \mid ab \mid b^3A \mid b^3 \mid aab \mid a \mid abD, \\ & D \to a \mid aab \mid abD \mid ab\}. \end{aligned}$$

Proof of Theorem 2.5 (cont.).

Finally we eliminate long productions from P_3.

For each production $(A \to a_1 a_2 \cdots a_m B) \in P_3$, where $m \geq 2$, introduce new nonterminals $A_1, A_2, \ldots, A_{m-1}$, and replace the above production by $A \to a_1 A_1, A_1 \to a_2 A_2, \ldots, A_{m-2} \to a_{m-1} A_{m-1}, A_{m-1} \to a_m B$.

For each production $(A \to a_1 a_2 \cdots a_m) \in P_3$, where $m \geq 2$, introduce new nonterminals $A_1, A_2, \ldots, A_{m-1}$, and replace the above production by $A \to a_1 A_1, A_1 \to a_2 A_2, \ldots, A_{m-2} \to a_{m-1} A_{m-1}, A_{m-1} \to a_m$.

The resulting grammar $\hat{G}$ is in right normal form and $L(\hat{G}) = L(G)$. □

Example (cont.):

P_3 contains the following A-productions:

$A \to abS$, $A \to b^3A$, $A \to b^3$, $A \to aab$, $A \to ab$, $A \to a$, and $A \to abD$.

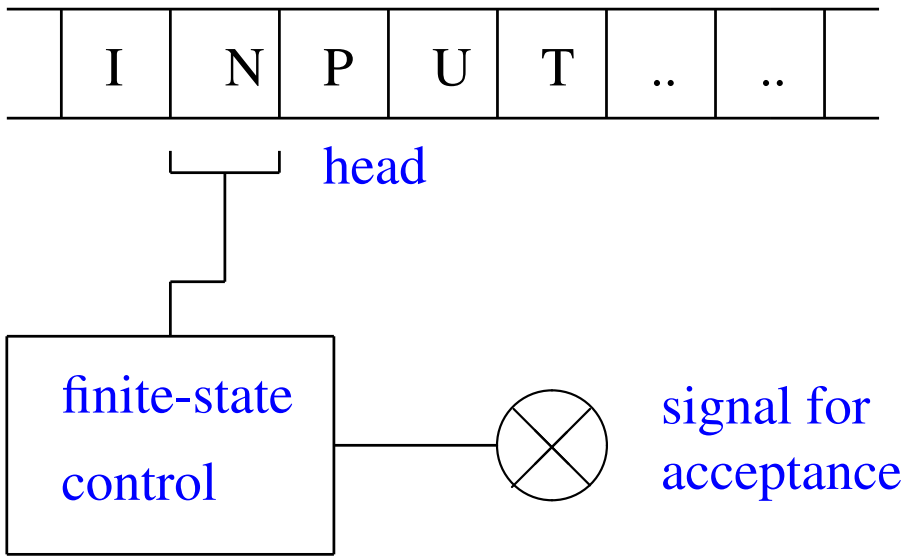
To replace the long ones, we introduce the nonterminals $A_1, A_2, \ldots, A_9$ and the following productions:

$$\begin{array}{lllll} A \to aA_1, & A_1 \to bS, & A \to bA_2, & A_2 \to bA_3, & A_3 \to bA, \\ A \to bA_4, & A_4 \to bA_5, & A_5 \to b, & A \to aA_6, & A_6 \to aA_7, \\ A_7 \to b, & A \to aA_8, & A_8 \to b, & A \to aA_9, & A_9 \to bD. \end{array}$$

For example: $A \to aA_1 \to abS$,
$A \to bA_2 \to bbA_3 \to bbbA$,
$A \to aA_6 \to aaA_7 \to aab$.

The long $\hat{S}$- and S-productions are replaced analogously. □

2.3 Deterministic Finite-State Automaton



A finite-state automaton reads an input word letter by letter from left to right and accepts or rejects.

finite-state automaton $\rightsquigarrow$ language of accepted words

Definition 2.6

A determinististic finite-state automaton (DFA) A is given through a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$*, where*

- Q *is a finite set of (internal) states,*
- Σ *is a finite (input) alphabet,*
- $\delta : Q \times \Sigma \rightsquigarrow Q$ *is a (partial) transition function,*
- $q_0 \in Q$ *is the initial state, and*
- $F \subseteq Q$ *is a set of accepting states.*

If δ *is defined for all* $(q, a) \in Q \times \Sigma$*, then A is a complete* DFA.

A DFA can be described by a state graph $G = (Kn, Ka)$:

- $Kn := Q$,
- Ka : If $\delta(q_1, a) = q_2$, then there is a directed edge with label a from q_1 to q_2.
- q_0 and $q \in F$ are marked in a special way.

Example:

$A = (Q, \Sigma, \delta, q_0, F)$, where

$Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $F = \{q_3\}$, and

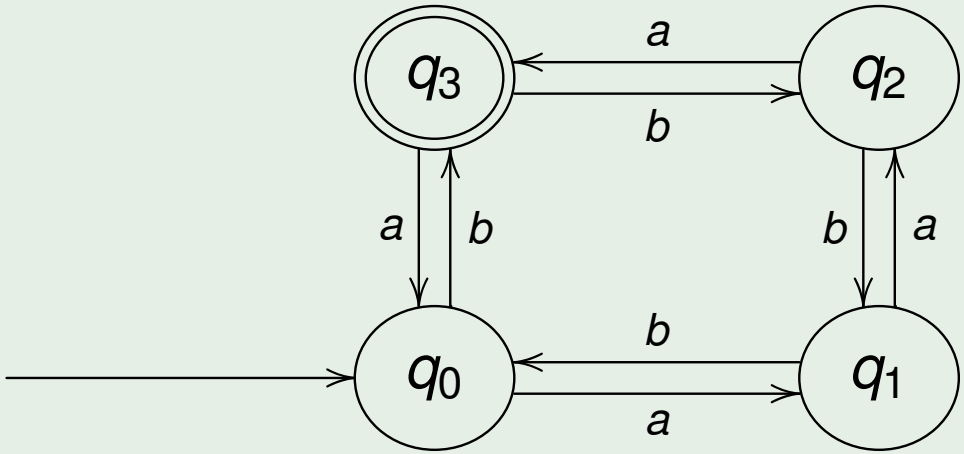
$\delta(q_0, a) = q_1$, $\delta(q_0, b) = q_3$

$\delta(q_1, a) = q_2$, $\delta(q_1, b) = q_0$

$\delta(q_2, a) = q_3$, $\delta(q_2, b) = q_1$

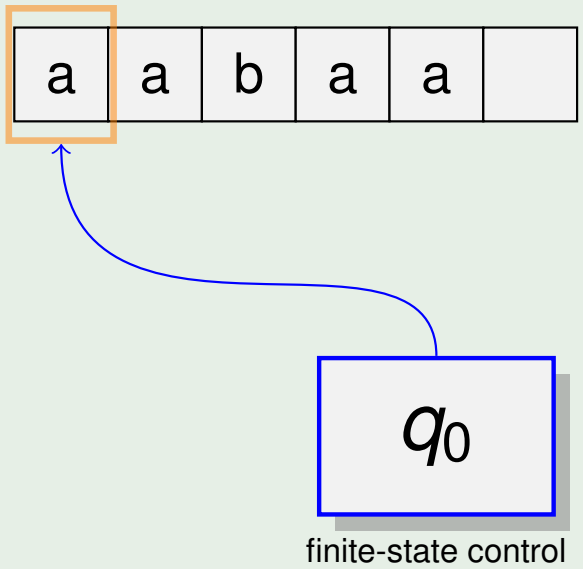
$\delta(q_3, a) = q_0$, $\delta(q_3, b) = q_2$

State graph:



Example (cont.):

Input: *aabaa*

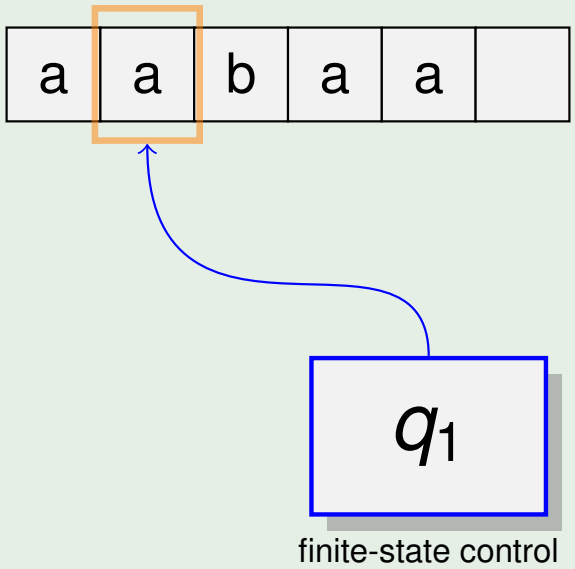


Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Example (cont.):

Input: *aabaa*

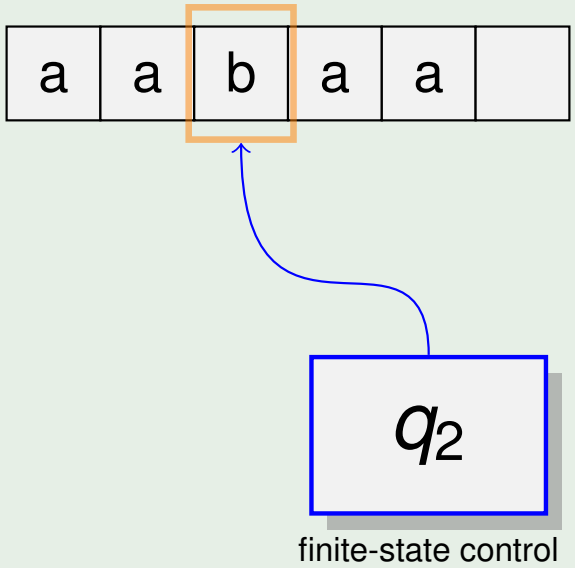


Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Example (cont.):

Input: *aabaa*

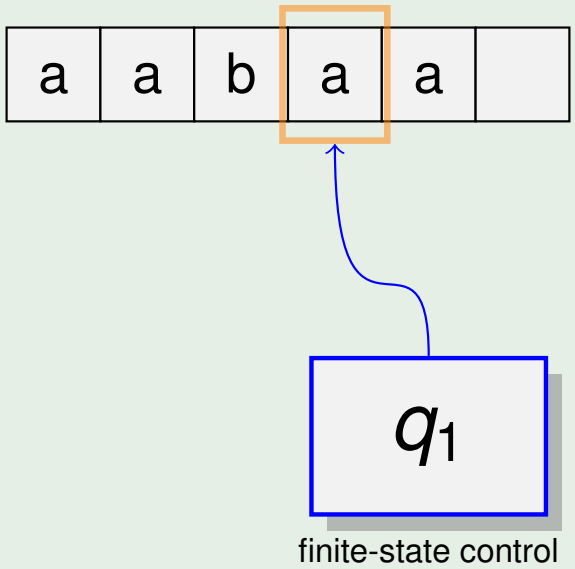


Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Example (cont.):

Input: *aabaa*

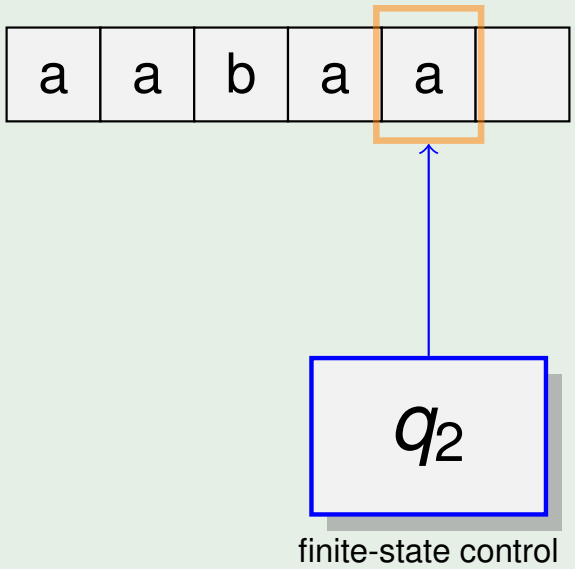


Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Example (cont.):

Input: *aabaa*

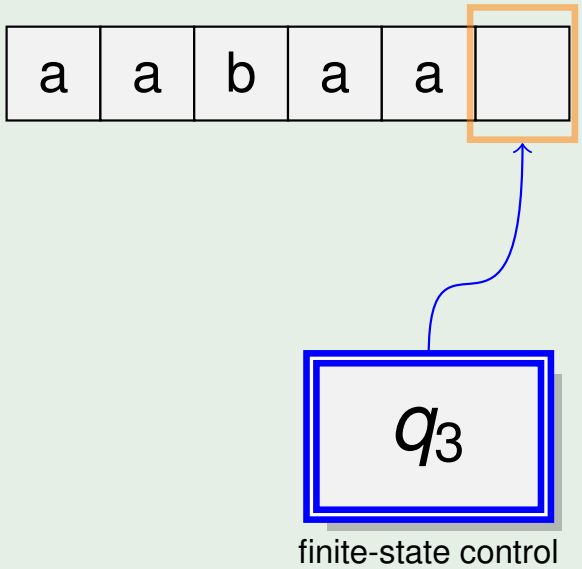


Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Example (cont.):

Input: *aabaa*



Computation: $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

Result: *aabaa* is accepted.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
The function $\delta : Q \times \Sigma \rightsquigarrow Q$ can be extended to a function $\hat{\delta} : Q \times \Sigma^* \rightsquigarrow Q$, where $q \in Q$, $u \in \Sigma^*$, and $a \in \Sigma$:

- $\hat{\delta}(q, \varepsilon) := q$,
- $\hat{\delta}(q, ua) = \begin{cases} \delta(\hat{\delta}(q, u), a), & \text{if } \hat{\delta}(q, u) \text{ is defined,} \\ \text{undefined}, & \text{otherwise.} \end{cases}$

$L_{A,q_1,q_2} := \{ u \in \Sigma^* \mid \hat{\delta}(q_1, u) = q_2 \}$ is the set of words that take A from state q_1 to state q_2.

If $\hat{\delta}(q_0, w) \in F$, then the word w is accepted by the DFA A.

The set $L(A) := \bigcup_{q \in F} L_{A,q_0,q}$ of all words that are accepted by A is called the language accepted by A.

Notice:

For all $q \in Q$ and $a_1, a_2, \ldots, a_n \in \Sigma$,

$$\begin{aligned} \hat{\delta}(q, a_1 a_2 \ldots a_n) &= \hat{\delta}(\delta(q, a_1), a_2 \ldots a_n) \\ &= \hat{\delta}(\delta(\delta(q, a_1), a_2), a_3 \ldots a_n) \\ &= \delta(\delta(\ldots \delta(\delta(q, a_1), a_2) \ldots, a_{n-1}), a_n) \end{aligned}$$

For all $q \in Q$ and $u, v \in \Sigma^*$, $\hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$.

Example (cont.):

$L(A) = \{ x \in \Sigma^* \mid |x|_a - |x|_b \equiv 3 \bmod 4 \}$.

A word of the form $qu \in Q \cdot \Sigma^*$ is called a configuration of A.

The DFA A induces a computation relation $\vdash_A^*$ on $Q \cdot \Sigma^*$.

It follows that $L(A) = \{ u \in \Sigma^* \mid q_0 u \vdash_A^* q \text{ for some } q \in F \}$.

Lemma 2.7

From a given DFA A, a complete DFA B can be constructed such that $L(B) = L(A)$.

Proof.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be an incomplete DFA.

We define a DFA $B = (Q \cup \{\bot\}, \Sigma, \delta', q_0, F)$ through

$$\begin{aligned} \delta'(q,a) &= \begin{cases} \delta(q,a), & \text{if } \delta(q,a) \text{ is defined,} \\ \bot, & \text{otherwise,} \end{cases} \\ \delta'(\bot,a) &= \bot \text{ for all } a \in \Sigma. \end{aligned}$$

Proof of Lemma 2.7 (cont.).)

Then the following equality holds for all $q \in Q$ and all $u \in \Sigma^*$:

$$\delta'(q,u) = \begin{cases} \delta(q,u), & \text{if } \delta(q,u) \text{ is defined,} \\ \bot, & \text{otherwise.} \end{cases}$$

Hence,

$$L(A) = \{\, u \in \Sigma^* \mid \delta(q_0,u) \in F \,\} = \{\, u \in \Sigma^* \mid \delta'(q_0,u) \in F \,\} = L(B). \qquad \square$$

Let $\mathcal{L}(\mathrm{DFA})$ denote the class of languages that are accepted by DFAs.

Theorem 2.8

$\mathcal{L}(\mathrm{DFA})$ *is closed under the operations of union, intersection, and complement.*

Proof.

Let $L = L(A)$, where $A = (Q, \Sigma, \delta, q_0, F)$ is a complete DFA.
Then $\overline{A} := (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a complete DFA for $\Sigma^* \setminus L(A) = L^C$.
Hence, $\mathcal{L}$(DFA) is closed under the operation of complement.

Let $A_i = (Q_i, \Sigma, \delta_i, q_0^{(i)}, F_i)$ be a complete DFA for L_i, $i = 1, 2$.
A DFA for the intersection $L_1 \cap L_2$ is obtained by taking
$A = (Q, \Sigma, \delta, q_0, F)$, where
$Q := Q_1 \times Q_2$, $q_0 := (q_0^{(1)}, q_0^{(2)})$, $F := F_1 \times F_2$, and
$\delta((q_1, q_2), a) := (\delta_1(q_1, a), \delta_2(q_2, a))$ for all $q_1 \in Q_1$, $q_2 \in Q_2$, $a \in \Sigma$.

Then $\delta(q_0, w) = (\delta_1(q_0^{(1)}, w), \delta_2(q_2^{(2)}, w)) \in F = F_1 \times F_2$
iff $w \in L_1$ and $w \in L_2$, that is, $L(A) = L_1 \cap L_2$.

A is called the product automaton of A_1 and A_2.

By De Morgan's law, closure under union follows. □

Theorem 2.9

If a language L is accepted by a DFA, then L is right regular.

Proof.

Let $L \subseteq \Sigma^*$ be accepted by a complete DFA $A = (Q, \Sigma, \delta, q_0, F)$.

We define a right regular grammar $G := (V, \Sigma, P, S)$ as follows:

– $V := Q$,

– $S := q_0$,

– $P := \{ q_1 \to aq_2 \mid \delta(q_1, a) = q_2 \}$
$\cup \{ q_1 \to a \mid \delta(q_1, a) = q_2 \text{ and } q_2 \in F \}$

If $q_0 \in F$, i.e., $\varepsilon \in L(A)$, then we add the new start symbol $\hat{q}_0$ and the following productions:

$$\{\hat{q}_0 \to \varepsilon\} \quad \text{and} \quad \{ \hat{q}_0 \to aq_2 \mid \delta(q_0, a) = q_2 \} \quad \text{and} \quad \{ \hat{q}_0 \to a \mid \delta(q_0, a) \in F \}.$$

Proof of Theorem 2.9 (cont.).

Claim.

$\forall x \in \Sigma^+ : x \in L(A)$ iff $x \in L(G)$.

Proof.

$x = a_1 a_2 \ldots a_n \in L(A)$
iff
$\exists q_1, q_2, \ldots, q_n \in Q : q_n \in F$ and $\delta(q_{i-1}, a_i) = q_i \ \ (1 \leq i \leq n)$
iff
$\exists q_1, \ldots, q_n \in V : q_0 \to a_1 q_1 \to \ldots \to a_1 \ldots a_{n-1} q_{n-1} \to a_1 \ldots a_{n-1} a_n$
iff
$x = a_1 a_2 \ldots a_n \in L(G)$. □□

There are many DFA that accept the same language.
Among these is there a smallest DFA?

Let $L \subseteq \Sigma^*$.

Definition 2.10

The Nerode relation $R_L \subseteq \Sigma^* \times \Sigma^*$ for L is defined through

$$x \; R_L \; y \text{ iff } \forall z \in \Sigma^* : (xz \in L \text{ iff } yz \in L).$$

Lemma 2.11

- *R_L is an equivalence relation on Σ^*.*
- *If $x \; R_L \; y$, then $xw \; R_L \; yw$ for all $w \in \Sigma^*$.*
- *R_L partitions Σ^* (and L) into disjoint equivalence classes.*

The index of R_L := number of equivalence classes of R_L.

Theorem 2.12 (Myhill, Nerode)

A language L is accepted by a DFA iff the relation R_L has finite index.

Proof.

"$\Rightarrow$": Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete DFA for L. We define a binary relation $R_A \subseteq \Sigma^* \times \Sigma^*$: $x \; R_A \; y$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$.

Claim. $R_A \subseteq R_L$.

Proof.

Assume that $x \; R_A \; y$, and let $z \in \Sigma^*$.

Then: $xz \in L$ iff $\hat{\delta}(q_0, xz) \in F$ iff $\hat{\delta}(\hat{\delta}(q_0, x), z) \in F$
iff $\hat{\delta}(\hat{\delta}(q_0, y), z) \in F$ iff $\hat{\delta}(q_0, yz) \in F$
iff $yz \in L$.

Hence: $x \; R_L \; y$. □

Thus: $\text{index}(R_L) \leq \text{index}(R_A) \leq |Q| < \infty$.

Proof of Theorem 2.12 (cont.)

"$\Leftarrow$": Assume that R_L has finite index.

Then there are finitely many words $x_1, x_2, \ldots, x_k \in \Sigma^*$ such that $\Sigma^* = [x_1] \cup [x_2] \cup \ldots \cup [x_k]$.

We define a complete DFA $A_0 := (Q_0, \Sigma, \delta, q_0, F)$:

– $Q_0 := \{ [x_1], [x_2], \ldots, [x_k] \}$,

– $q_0 := [\varepsilon]$,

– $F := \{ [x_i] \mid x_i \in L \}$,

– $\delta([x_i], a) := [x_i a]$ (for all $i = 1, 2, \ldots, k$ and $a \in \Sigma$).

Then $\hat{\delta}([\varepsilon], x) = [x]$ for all $x \in \Sigma^*$, that is,

$$\begin{aligned} x \in L(A_0) &\text{ iff } \hat{\delta}(q_0, x) \in F \\ &\text{iff } \hat{\delta}([\varepsilon], x) \in F \\ &\text{iff } [x] \in F \\ &\text{iff } x \in L. \end{aligned}$$

□

A_0 is called the equivalence class automaton for L.

Example 1:

$L = \{ a^n b^n \mid n \geq 1 \}$

Then:

$$\begin{aligned}
[ab] &= L \\
[a^2 b] &= \{a^2 b, a^3 b^2, a^4 b^3, \ldots\} \\
[a^3 b] &= \{a^3 b, a^4 b^2, a^5 b^3, \ldots\} \\
&\vdots \\
[a^k b] &= \{ a^{k+i-1} b^i \mid i \geq 1 \}
\end{aligned}$$

For all $i \neq j$, $a^i b$ and $a^j b$ are not equivalent w.r.t. R_L, that is, $\mathrm{index}(R_L) = \infty$.

Hence, L is not accepted by any DFA.

Example 2:

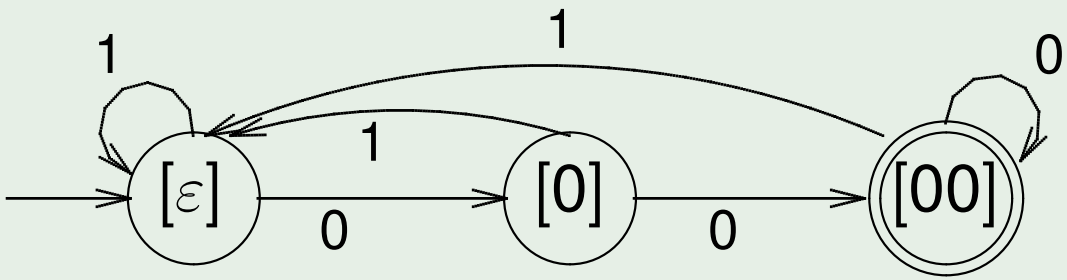
$L = \{ x \in \{0,1\}^* \mid x \text{ has suffix } 00 \}$

Then:

$$\begin{aligned} [\varepsilon] &= \{ x \mid x \text{ does not have suffix } 0 \} \\ [0] &= \{ x \mid x \text{ has suffix } 0, \text{ but not suffix } 00 \} \\ [00] &= \{ x \mid x \text{ has suffix } 00 \} \end{aligned}$$

$\{0,1\}^* = [\varepsilon] \cup [0] \cup [00]$, and hence, $\text{index}(R_L) = 3$.

Equivalence class automaton A_0 for L:



Assume that L is accepted by a complete DFA A, and let A_0 be the equivalence class automaton for L.

Then: $R_A \subseteq R_L = R_{A_0}$,

and so: $|Q| \geq |Q_0| = \text{index}(R_L)$.

In fact: If $|Q| = |Q_0|$, then $R_A = R_L$, that is, A and A_0 are isomorphic.

Thus: A_0 is the minimal automaton for L.

How to determine whether a given DFA A is minimal?

A is not minimal iff
$\exists q, q' \in Q, q \neq q' : \forall x \in \Sigma^* : \hat{\delta}(q, x) \in F$ iff $\hat{\delta}(q', x) \in F$.

Algorithm "minimal automaton"

Input: a complete DFA $A = (Q, \Sigma, \delta, q_0, F)$.

(0) Delete all states that are not reachable from q_0.

(1) Initialize a table of all pairs $\{q, q'\}$ such that $q \neq q'$.

(2) Mark all pairs $\{q, q'\}$ for which $\{q, q'\} \cap F \neq \emptyset$ and $\{q, q'\} \cap (Q \setminus F) \neq \emptyset$.

(3) For each unmarked pair $\{q, q'\}$ and each $a \in \Sigma$, check whether $\{\delta(q, a), \delta(q', a)\}$ is marked. If so, then mark $\{q, q'\}$ as well.

(4) Repeat (3) until no further pairs are marked.

(5) For each unmarked pair $\{q, q'\}$, merge the states q and q' into a joint state.

Output: The minimal automaton for $L(A)$

Example 1:

A :

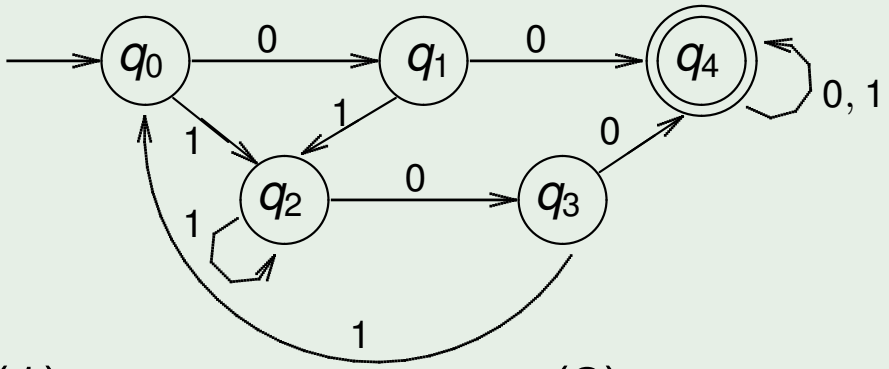
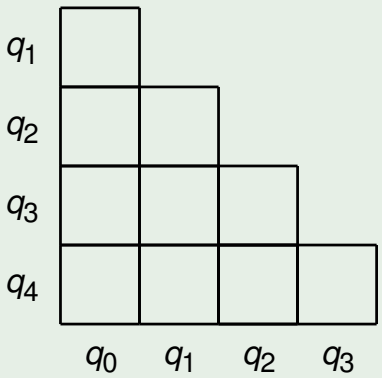
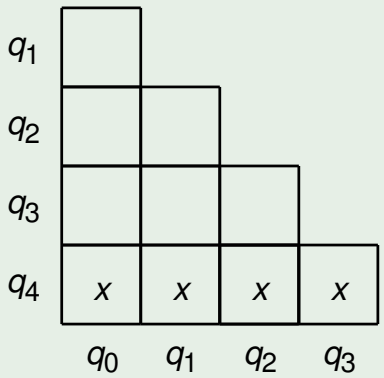


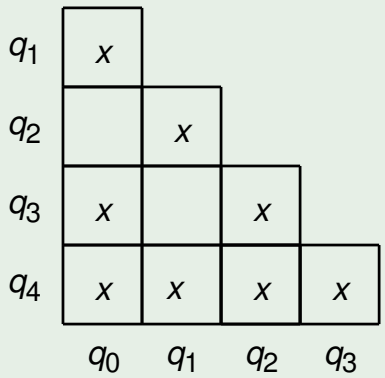
Table: (1)



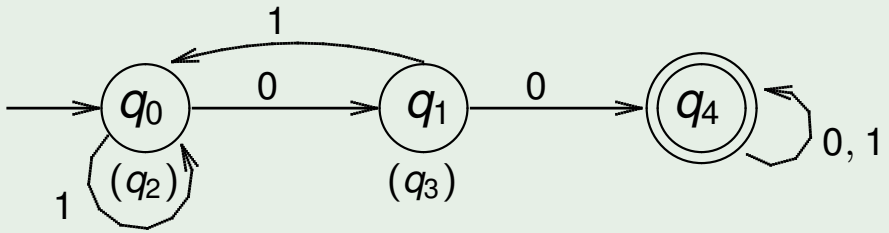
(2)



(3)

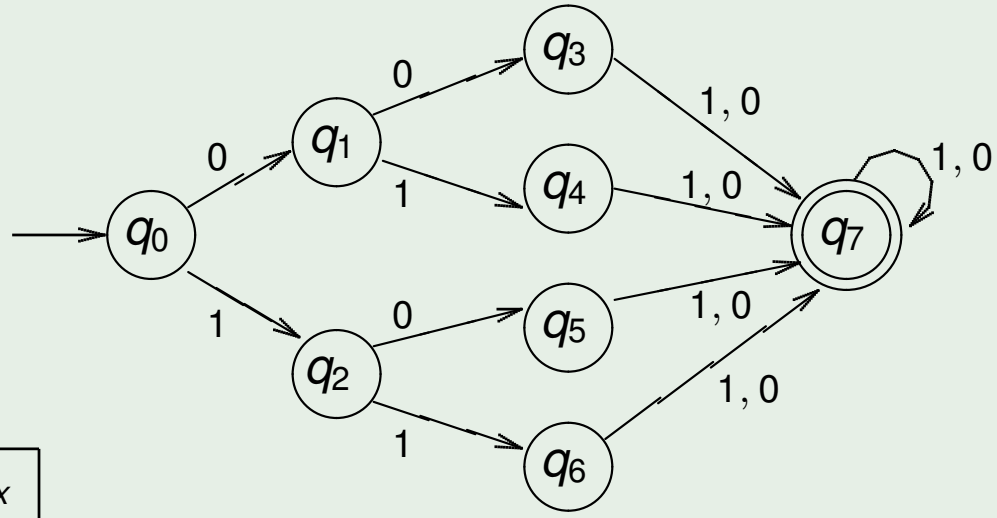


A_0 :



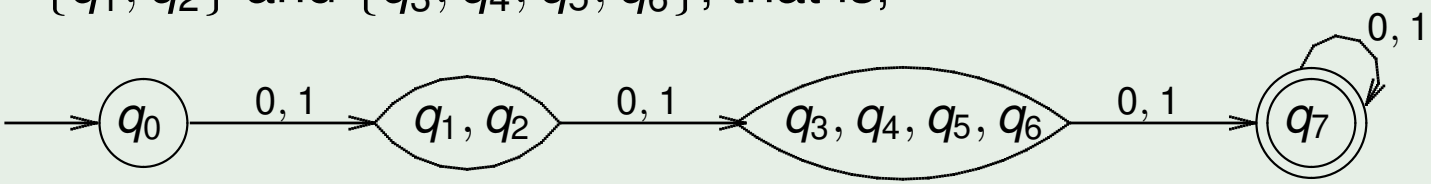
$L(A) = L(A_0) = \{ x \mid x \text{ contains 00 as a factor} \}$.

Example 2:



q_1	x						
q_2	x						
q_3	x	x	x				
q_4	x	x	x				
q_5	x	x	x				
q_6	x	x	x				
q_7	x	x	x	x	x	x	x
	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Hence: $\{q_1, q_2\}$ and $\{q_3, q_4, q_5, q_6\}$, that is,



$$L(A) = L(A_0) = \{\, w \in \{0,1\}^* \mid |w| \geq 3 \,\}$$

Let $h : \Sigma^* \to \Delta^*$ be a morphism. Then $h^{-1} : \Delta^* \to 2^{\Sigma^*}$ is defined through

$$h^{-1}(v) := \{ u \in \Sigma^* \mid h(u) = v \}.$$

For a language $L \subseteq \Delta^*$, $h^{-1}(L)$ is defined as

$$h^{-1}(L) := \{ u \in \Sigma^* \mid h(u) \in L \}.$$

Theorem 2.13

The language class $\mathcal{L}$(DFA) is closed under inverse morphisms, that is, if $h : \Sigma^* \to \Delta^*$ is a morphism and $L \subseteq \Delta^*$ is accepted by a DFA, then also $h^{-1}(L)$ is accepted by a DFA.

Proof of Theorem 2.13.

Let $A = (Q, \Delta, \delta, q_0, F)$ be a complete DFA such that $L(A) = L$ and let $h : \Sigma^* \to \Delta^*$ be a morphism.
We construct a DFA $B := (Q, \Sigma, \delta', q_0, F)$ by taking

$$\forall q \in Q \forall a \in \Sigma : \delta'(q, a) := \delta(q, h(a)).$$

Claim:

For all $x \in \Sigma^*$, $\delta'(q_0, x) = \delta(q_0, h(x))$.

Proof by induction on $|x|$:

If $x = \varepsilon$, then $\delta'(q_0, x) = q_0 = \delta(q_0, \varepsilon) = \delta(q_0, h(x))$.

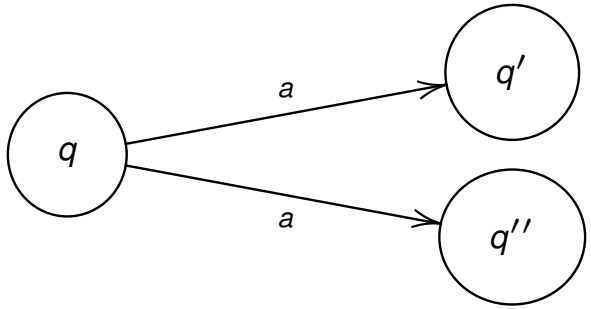
Assume that the claim has been proved for some $n \geq 0$,
let x be a word of length n, and let $a \in \Sigma$. Then:

$$\begin{aligned}\delta'(q_0, xa) &= \delta'(\delta'(q_0, x), a) \underset{\text{I.H.}}{=} \delta'(\delta(q_0, h(x)), a) = \delta(\delta(q_0, h(x)), h(a)) \\ &= \delta(q_0, h(x)h(a)) = \delta(q_0, h(xa)).\end{aligned}$$ □

Thus, for all $x \in \Sigma^*$: $x \in L(B)$ iff $\delta'(q_0, x) \in F$
iff $\delta(q_0, h(x)) \in F$
iff $h(x) \in L$
iff $x \in h^{-1}(L)$,

that is, $L(B) = h^{-1}(L)$. □

2.4 Nondeterministic Finite-State Automaton



Definition 2.14

A nondeterministic finite-state automaton (NFA) A is given through a 5-tuple $(Q, \Sigma, \delta, S, F)$*, where*

– Q *is a finite set of (internal) states,*
– Σ *is a finite (input) alphabet,*
– $Q \cap \Sigma = \emptyset$*,*
– $S \subseteq Q$ *is a set of initial states,*
– $F \subseteq Q$ *is a set of final states, and*
– $\delta : Q \times \Sigma \to 2^Q$ *is a transition relation.*

An NFA can be described by a state graph just like a DFA.

Definition 2.14 (cont.)

Let $A = (Q, \Sigma, \delta, S, F)$ be an NFA.
The transition relation δ is extended to a function $\hat{\delta}$: $2^Q \times \Sigma^* \to 2^Q$:

$$\begin{aligned} \hat{\delta}(P, \varepsilon) &:= P \text{ for all } P \subseteq Q, \\ \hat{\delta}(P, ax) &:= \bigcup_{q \in P} \hat{\delta}(\delta(q, a), x) \text{ for all } P \subseteq Q, a \in \Sigma, \text{ and } x \in \Sigma^*. \end{aligned}$$

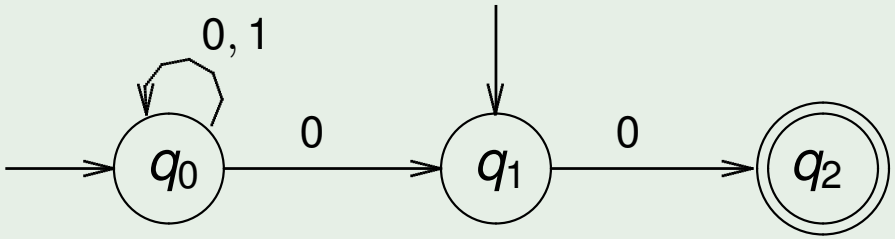
$L(A) := \{ x \in \Sigma^* \mid \hat{\delta}(S, x) \cap F \neq \emptyset \}$ is the language accepted by A.

Remark

$\hat{\delta}(S, a_1 a_2 \ldots a_n) \cap F \neq \emptyset$
iff
$\exists q_0, q_1, \ldots, q_n \in Q : q_0 \in S \ \wedge \ q_n \in F \ \wedge \ q_i \in \delta(q_{i-1}, a_i) \ \ (1 \leq i \leq n)$.
iff
$\exists q_0 \in S \, \exists q_n \in E$: In the state graph of A,
there is a directed path from q_0 to q_n with label $a_1 a_2 \ldots a_n$.

Example:

(a) A_1:



Let $x := 1100$. Then:

$S = \{q_0, q_1\} \xrightarrow{1} \{q_0\} \xrightarrow{1} \{q_0\} \xrightarrow{0} \{q_0, q_1\} \xrightarrow{0} \{q_0, q_1, q_2\}$

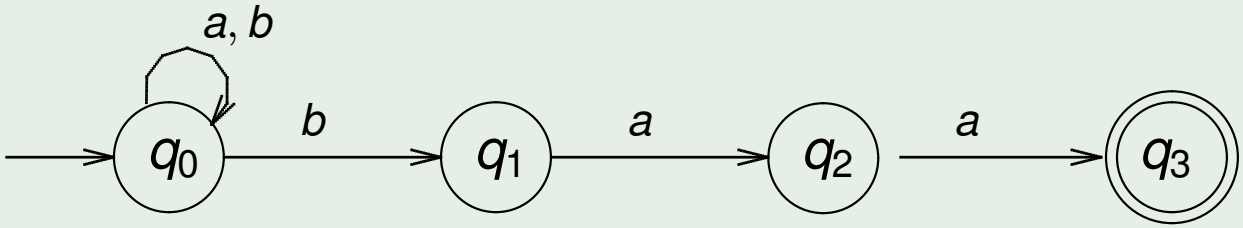
Hence, $x \in L(A_1)$.

In fact:

$L(A_1) = \{\ x \in \{0,1\}^* \mid x = 0 \text{ or } \exists y \in \{0,1\}^* : x = y00\ \}.$

Example (cont.):

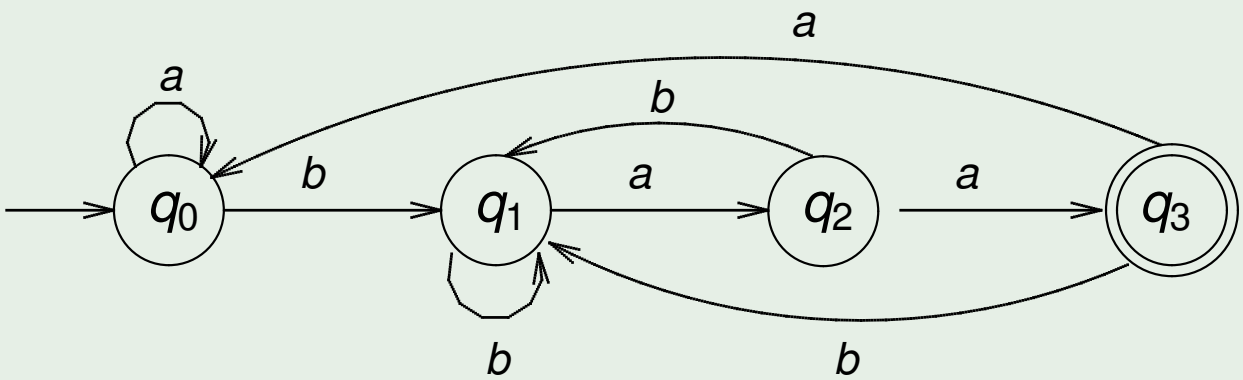
(b) A_2:



$L(A_2) = \{ w \in \{a, b\}^* \mid w \text{ has suffix } baa\}$.

A DFA for this language:

A_3:



Theorem 2.15 (Rabin,Scott)

From an NFA A, a DFA B can be constructed such that $L(B) = L(A)$.

Proof.

Let $A = (Q, \Sigma, \delta_A, S, F)$ be an NFA.

Goal: A DFA $B = (P, \Sigma, \delta_B, p_0, G)$ such that $L(B) = L(A)$.

Power set construction:

$P := 2^Q$, $p_0 := S$,
$G := \{ Q' \subseteq Q \mid Q' \cap F \neq \emptyset \}$,
$\delta_B(Q', a) := \bigcup_{q \in Q'} \delta_A(q, a) = \hat{\delta}_A(Q', a)$
for all $Q' \subseteq Q$ and all $a \in \Sigma$.

Proof of Theorem 2.15 (cont.)

Claim:

$L(B) = L(A)$.

Proof.

$\varepsilon \in L(A)$ iff $S \cap F \neq \emptyset$ iff $S \in G$ iff $\varepsilon \in L(B)$.

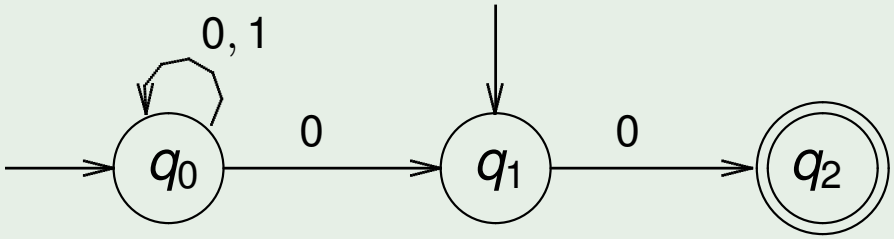
Now let $x = a_1 a_2 \ldots a_n \in \Sigma^+$.

Then: $x \in L(A)$ iff $\hat{\delta}_A(S, x) \cap F \neq \emptyset$
iff $\exists Q_1, Q_2, \ldots, Q_n \subseteq Q : \delta_B(S, a_1) = Q_1$,
$\delta_B(Q_1, a_2) = Q_2, \ldots, \delta_B(Q_{n-1}, a_n) = Q_n$ and $Q_n \cap F \neq \emptyset$
iff $\hat{\delta}_B(S, x) \in G$ iff $x \in L(B)$.

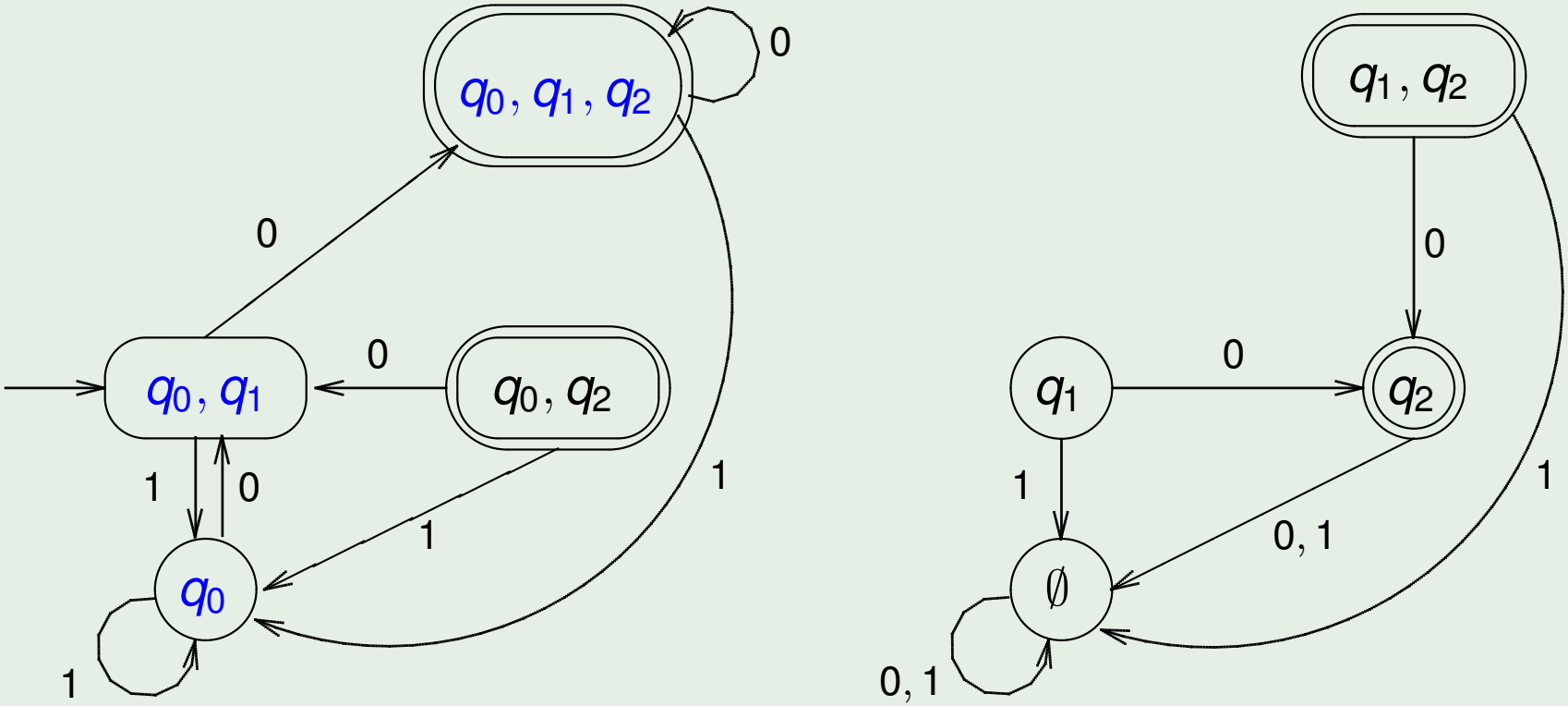
□□

Example (cont.):

A :



B :



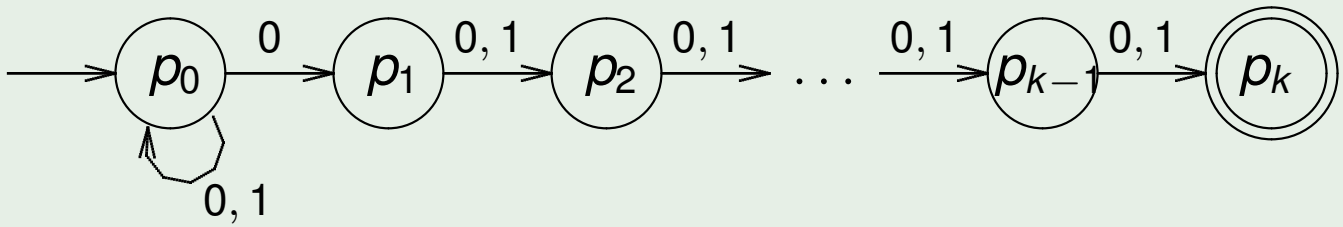
Remark

Only compute the subautomaton of B that is reachable from S!

Example:

$L_k := \{ x = x_1 \dots x_n \in \{0,1\}^* \mid |x| = n \geq k \text{ and } x_{n-k+1} = 0 \}$

An NFA A_k with $L(A_k) = L_k$:



Let $B = (Q, \{0,1\}, \delta, q_0, F)$ be a DFA such that $L(B) = L_k$.

Example (cont.):

Claim: $|Q| \geq 2^k$.

Proof.

Assume that $Q = \{q_0, q_1, \ldots, q_r\}$, where $r < 2^k - 1$.
Then: $\exists y_1, y_2 \in \{0,1\}^k : y_1 \neq y_2$ and $\hat{\delta}(q_0, y_1) = \hat{\delta}(q_0, y_2)$.

Let i be the leftmost position such that $y_1(i) \neq y_2(i)$,
w.l.o.g. $y_1 = u0v_1$ and $y_2 = u1v_2$.

Let $w \in \{0,1\}^{i-1}$ be arbitrary.

Then: $y_1 w = u0v_1 w$, $|v_1 w| = k - i + i - 1 = k - 1$
$\rightsquigarrow y_1 w \in L_k$
$y_2 w = u1v_2 w$, $|v_2 w| = k - 1$
$\rightsquigarrow y_2 w \notin L_k$.

But:
$$\begin{aligned} \hat{\delta}(q_0, y_1 w) &= \hat{\delta}(\hat{\delta}(q_0, y_1), w) \\ &= \hat{\delta}(\hat{\delta}(q_0, y_2), w) \\ &= \hat{\delta}(q_0, y_2 w), \text{ a contradiction!} \end{aligned}$$
□

Theorem 2.16

From a right regular grammar G, one can construct an NFA A such that $L(A) = L(G)$.

Proof.

Based on Theorem 2.5 we can first transform the grammar G into an equivalent grammar $G' = (N, T, S, P)$ that is in right normal form, that is, it only has productions of the form

$$A \to aB \text{ and } A \to a, \text{ where } A, B \in N \text{ and } a \in T,$$

and possibly the production $S \to \varepsilon$.

Proof of Theorem 2.16 (cont.)

We take $A := (Q, T, \delta, S', F)$, where

– $Q := N \cup \{X\}$ (X a new symbol),

– $S' := \{S\}$,

– $F := \begin{cases} \{S, X\}, & \text{if } (S \to \varepsilon) \in P, \\ \{X\}, & \text{otherwise,} \end{cases}$

– $\delta(A, a) := \{ B \mid (A \to aB) \in P \} \cup \{ X \mid (A \to a) \in P \}$
for all $A \in N$ and $a \in T$.

Then: $\varepsilon \in L(G)$ iff $(S \to \varepsilon) \in P$
iff $S \in F$
iff $\varepsilon \in L(A)$.

For all $n \geq 1$: $a_1 a_2 \ldots a_n \in L(G)$ iff $a_1 a_2 \ldots a_n \in L(G')$ iff
$\exists A_1, \ldots, A_{n-1} \in N : S \to a_1 A_1 \to \ldots \to a_1 a_2 \ldots a_{n-1} A_{n-1} \to a_1 \ldots a_n$
iff
$\exists A_1, \ldots, A_{n-1} \in N : A_1 \in \delta(S, a_1), A_2 \in \delta(A_1, a_2), \ldots, X \in \delta(A_{n-1}, a_n)$
iff $a_1 a_2 \ldots a_n \in L(A)$. □

Theorem 2.17

The class of languages $\mathcal{L}(\text{DFA})$ is closed under the operation of taking the mirror image.

Proof.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

By A^R we denote the NFA $A^R = (Q, \Sigma, \delta^R, F, \{q_0\})$,

where $\delta^R : Q \times \Sigma \to 2^Q$ is defined as follows:

$$\delta^R(p, a) := \{\, q \mid \delta(q, a) = p \,\} \quad (p \in Q, a \in \Sigma),$$

that is, initial and final states are interchanged,

and each transition is simply reversed.

Then $L(A^R) = (L(A))^R$. □

Corollary 2.18

For any language L, the following statements are equivalent:

(1) *L is a regular language.*

(2) *L is generated by a right regular grammar.*

(3) *L is generated by a left regular grammar.*

(4) *There exists a DFA A such that $L = L(A)$.*

(5) *There exists an NFA B such that $L = L(B)$.*

Proof.

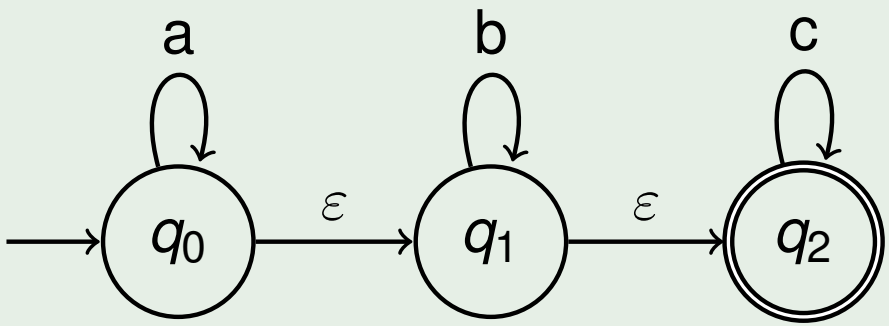
(2) $\rightarrow$ (5): Theorem 2.16.
(5) $\rightarrow$ (4): Theorem 2.15.
(4) $\rightarrow$ (2): Theorem 2.9.
(3) $\rightarrow$ (5): From a left regular grammar for L, we obtain a right regular grammar for L^R by reversing the right-hand sides of all productions. The above and Theorem 2.17 yield an NFA for $(L^R)^R = L$.
The remaining cases follow analogously. $\square$

Definition 2.19

A nondeterministic finite-state automaton with ε-transitions (ε-NFA) *A is given through a 5-tuple $A = (Q, \Sigma, \delta, S, F)$, where Q, Σ, S, and F are defined as for an NFA, while the transition relation δ has the form $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$.*

Example:

Let A be the ε-NFA that is given by the following state graph:



Then $L(A) = \{ a^i b^j c^k \mid i, j, k \geq 0 \}$.

Theorem 2.20

From an ε-NFA A, a DFA B can be constructed such that $L(B) = L(A)$.

Proof.

Let $A = (Q, \Sigma, \delta, S, F)$ be an ε-NFA. For $q \in Q$, ε-closure$(q) \subseteq Q$ denotes the set of states of A that can be reached from state q without reading any input symbol, that is,

$$\begin{aligned} \varepsilon\text{-closure}(q) \quad := \quad & \{\, p \in Q \mid \exists n \geq 0\, \exists\, p_0, p_1, \ldots, p_n \in Q : p_0 = q, \\ & p_n = p, \text{ and } p_{i+1} \in \delta(p_i, \varepsilon),\ i = 0, 1, \ldots, n-1 \,\}. \end{aligned}$$

Further, for $P \subseteq Q$, ε-closure$(P) := \bigcup_{q \in P} \varepsilon$-closure$(q)$.

We define the DFA $B := (2^Q, \Sigma, \delta_B, q_0, G)$ through a power set construction:

- $q_0 := S$,
- $G := \{\, P \subseteq Q \mid \varepsilon\text{-closure}(P) \cap F \neq \emptyset \,\}$,
- $\delta_B(P, a) := \delta(\varepsilon\text{-closure}(P), a)$ for all $P \subseteq Q$ and $a \in \Sigma$.

Proof of Theorem 2.20 (cont.)

Claim.

$L(B) = L(A)$.

Proof.

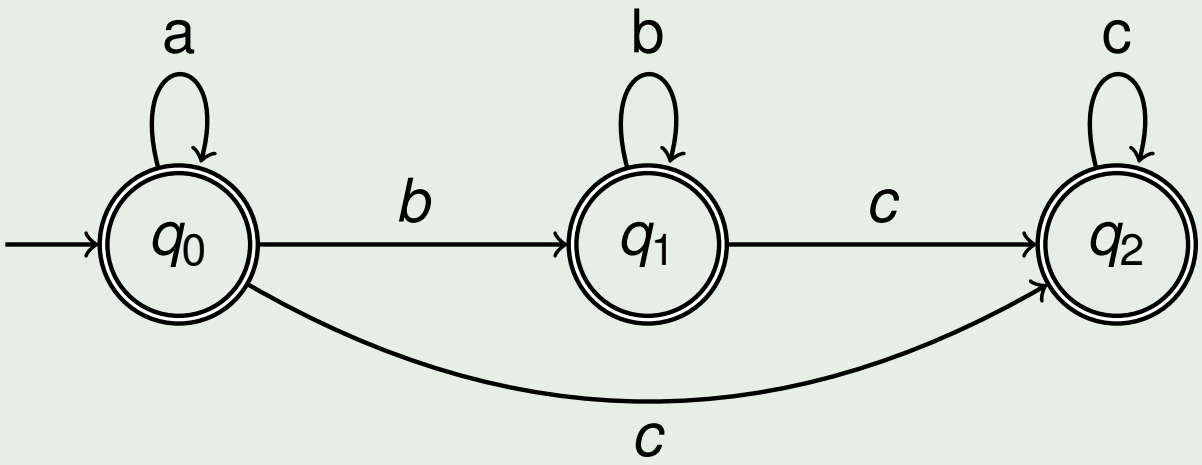
$\varepsilon \in L(A)$ iff ε-closure$(S) \cap F \neq \emptyset$ iff $q_0 \in G$ iff $\varepsilon \in L(B)$.

$$
\begin{aligned}
a_1 a_2 \ldots a_n \in L(A) \quad &\text{iff} \quad \exists s \in S \, \exists s_1, q_1, p_1, q_2, p_2, \ldots, q_n, p_n \in Q : \\
s \to^{\varepsilon^*} s_1 \to^{a_1} q_1 \to^{\varepsilon^*} p_1 \to^{a_2} q_2 \to^{\varepsilon^*} \cdots \to^{\varepsilon^*} p_{n-1} \to^{a_n} q_n \to^{\varepsilon^*} p_n \in F \\
&\text{iff} \quad \exists Q_1, Q_2, \ldots, Q_n \subseteq Q : \\
&\qquad Q_1 = \delta(\varepsilon\text{-closure}(S), a_1) \wedge Q_n \in G \wedge \\
&\qquad Q_{i+1} = \delta(\varepsilon\text{-closure}(Q_i), a_{i+1}) \; (1 \leq i \leq n-1) \\
&\text{iff} \quad a_1 a_2 \ldots a_n \in L(B).
\end{aligned}
$$

□

Example (cont.):

The ‘lazy’ form of this construction yields the following DFA from the given ε-NFA A:



Theorem 2.21

The language class $\mathcal{L}(\mathrm{DFA})$ is closed under the operations of union, product, and star.

Proof.

Union: Let $A_i = (Q_i, \Sigma, \delta_i, S_i, F_i)$, $i = 1, 2$, be two NFA s.t. $Q_1 \cap Q_2 = \emptyset$. Then $A := (Q_1 \cup Q_2, \Sigma, \delta, S_1 \cup S_2, F_1 \cup F_2)$, where

$$\delta(q, a) = \begin{cases} \delta_1(q, a), & \text{if } q \in Q_1, \\ \delta_2(q, a), & \text{if } q \in Q_2, \end{cases},$$

accepts the language $L(A) = L(A_1) \cup L(A_2)$.

Proof of Theorem 2.21 (cont.)

Product: Let $A_i = (Q_i, \Sigma, \delta_i, S_i, F_i)$, $i = 1, 2$, be two ε-NFA s.t. $Q_1 \cap Q_2 = \emptyset$.

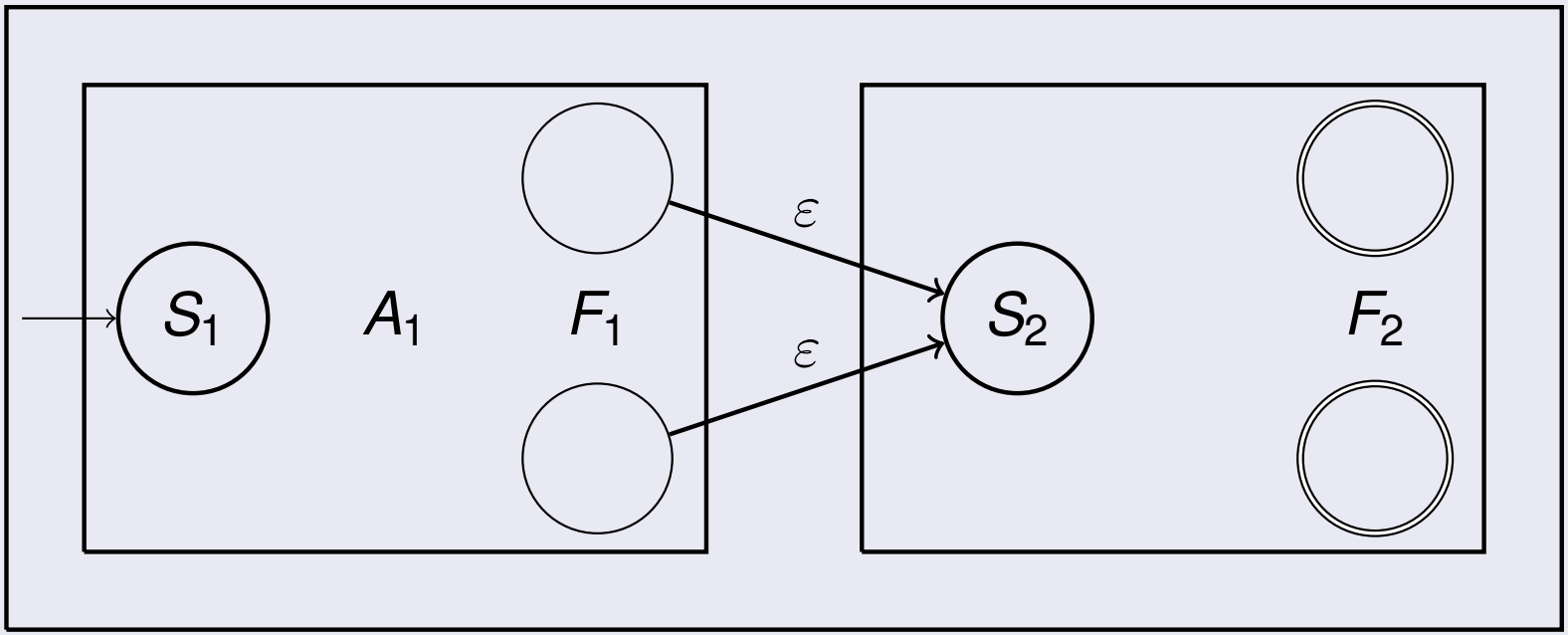
Define $A := (Q_1 \cup Q_2, \Sigma, \delta, S_1, F_2)$, where

$$\begin{aligned}
\delta(q,a) &= \delta_1(q,a) \quad \text{for all } q \in Q_1 \text{ and } a \in \Sigma, \\
\delta(q,\varepsilon) &= \begin{cases} \delta_1(q,\varepsilon) & \text{for all } q \in Q_1 \setminus F_1, \\ \delta_1(q,\varepsilon) \cup S_2 & \text{for all } q \in F_1, \end{cases} \\
\delta(q,a) &= \delta_2(q,a) \quad \text{for all } q \in Q_2 \text{ and } a \in \Sigma \cup \{\varepsilon\}.
\end{aligned}$$

Then A is an ε-NFA satisfying $L(A) = L(A_1) \cdot L(A_2)$.

Proof of Theorem 2.21 (cont.)

Graphical representation of A:



Proof of Theorem 2.21 (cont.)

Kleene Star:

Let $A_1 = (Q_1, \Sigma, \delta_1, S_1, F_1)$ be an ε-NFA.

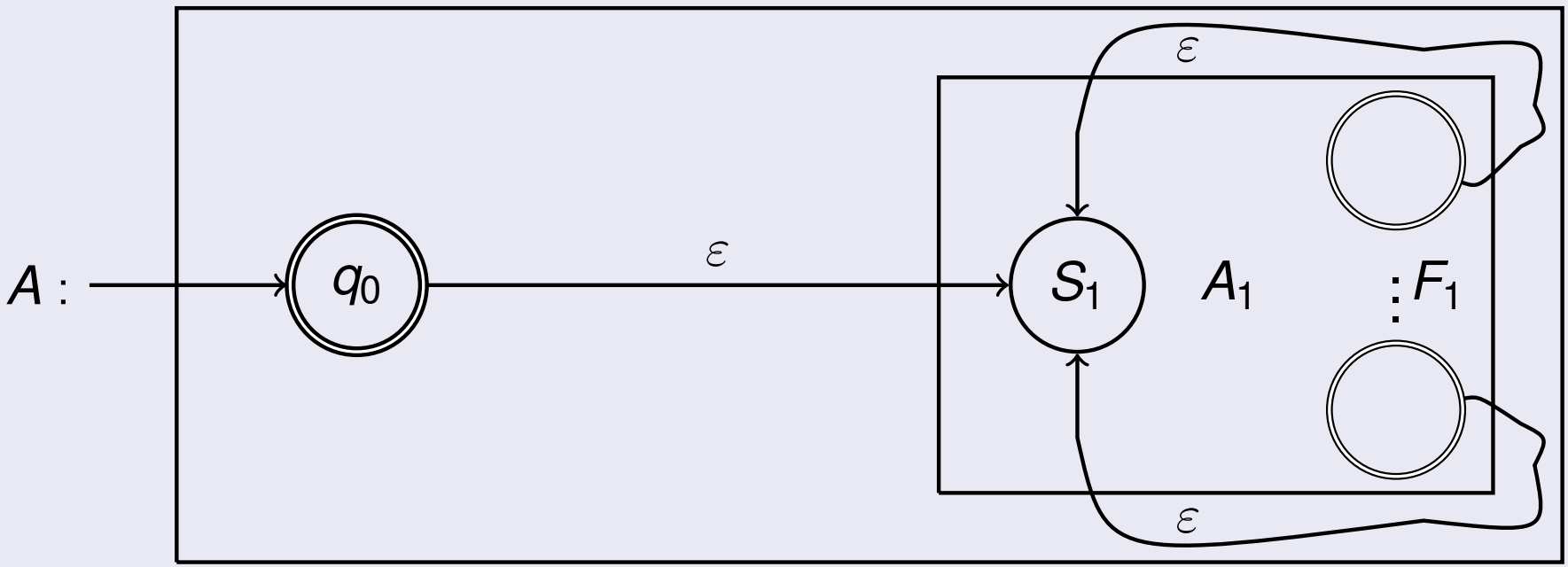
Define $A = (Q_1 \cup \{q_0\}, \Sigma, \delta, \{q_0\}, F_1 \cup \{q_0\})$, where

$$\begin{aligned}
\delta(q_0, \varepsilon) &:= S_1, \\
\delta(q, \varepsilon) &:= \delta_1(q, \varepsilon) \cup S_1 \quad \text{for all } q \in F_1, \\
\delta(q, \varepsilon) &:= \delta_1(q, \varepsilon) \quad \text{for all } q \in Q_1 \setminus F_1, \\
\delta(q, a) &:= \delta_1(q, a) \quad \text{for all } q \in Q_1 \text{ and } a \in \Sigma.
\end{aligned}$$

Then A is an ε-NFA satisfying $L(A) = (L(A_1))^*$.

Proof of Theorem 2.21 (cont.)

Graphical representation of A:



2.5 Two-Way Finite-State Automaton

Definition 2.22

A deterministic two-way finite-state automaton (2DFA) *A is defined through a 7-tuple* $A = (Q, \Sigma, \triangleright, \triangleleft, \delta, q_0, F)$*, where*

– Q, Σ, q_0*, and* F *are defined as for a DFA,*

– $\triangleright, \triangleleft \notin \Sigma$ *are two new letters that serve as border markers for the left and the right end of the tape, and*

$$\delta : Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \to Q \times \{L, R\}$$

is the transition function satisfying the following restrictions:

$$\forall q \in Q : \delta(q, \triangleright) \neq (q', L) \text{ and } \delta(q, \triangleleft) \neq (q', R).$$

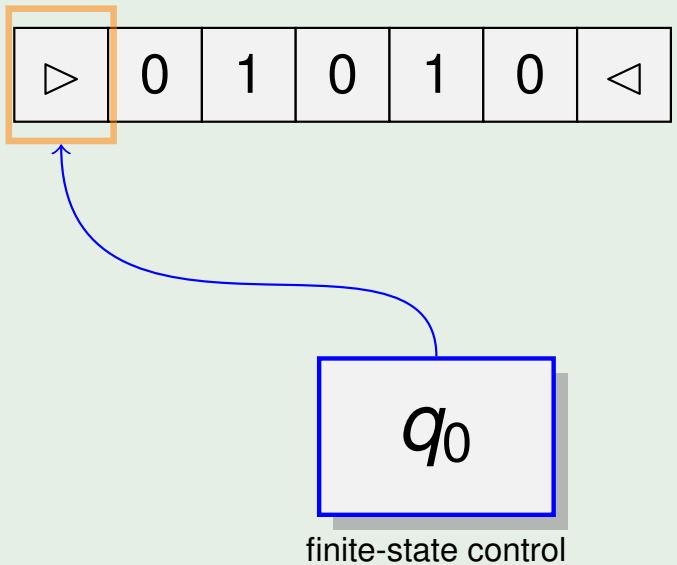
These restrictions ensure that A's head cannot fall off the tape.

Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

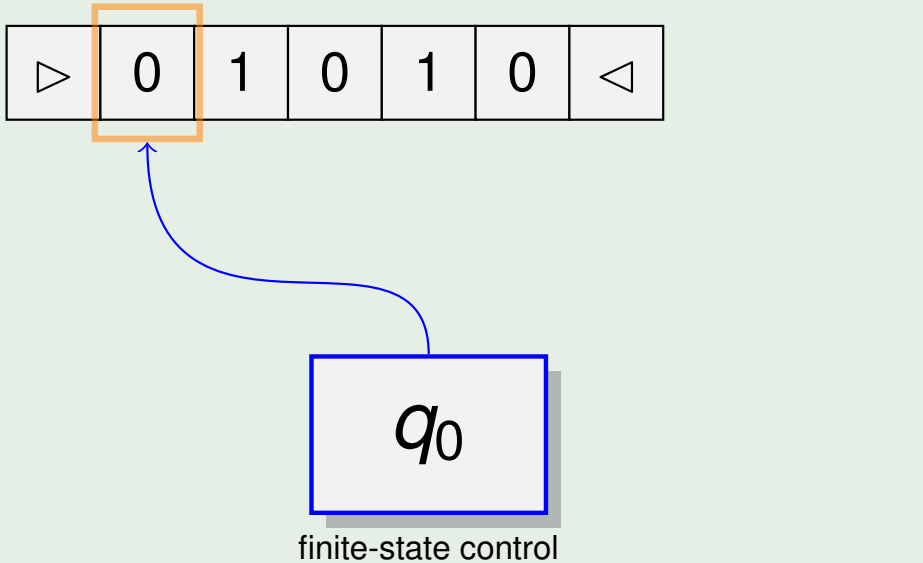


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

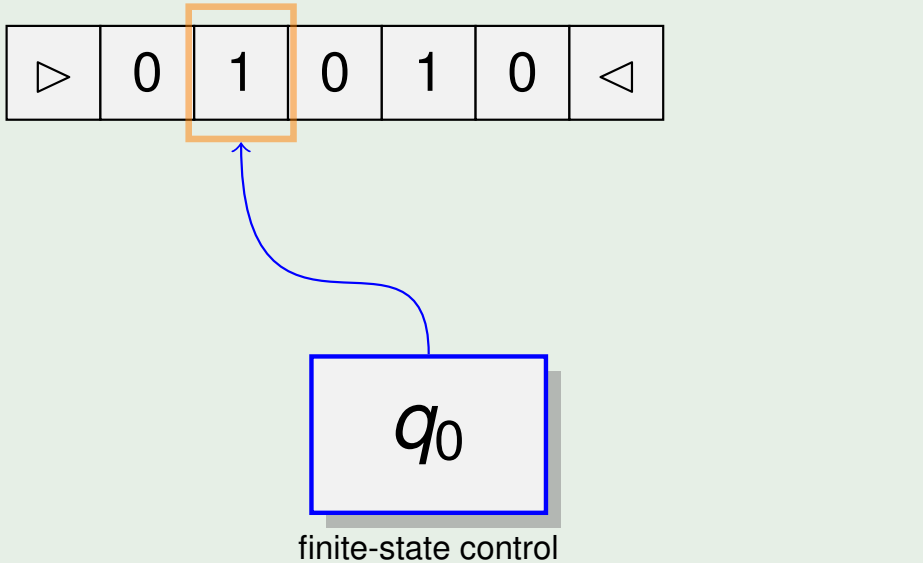


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

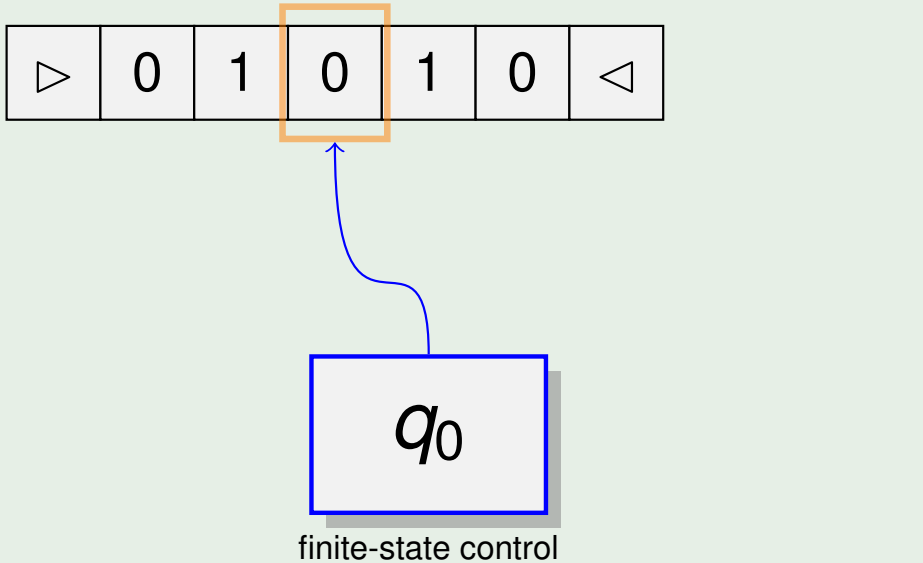


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

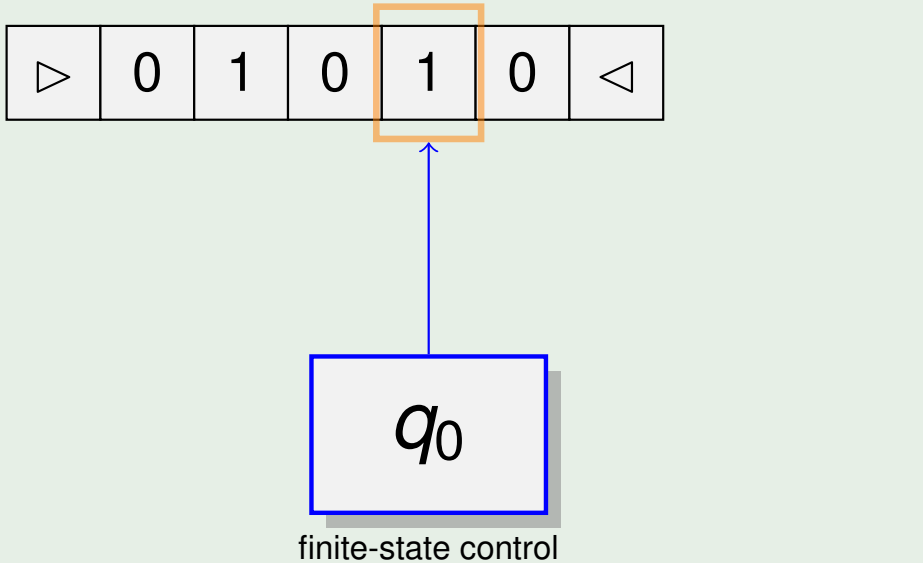


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

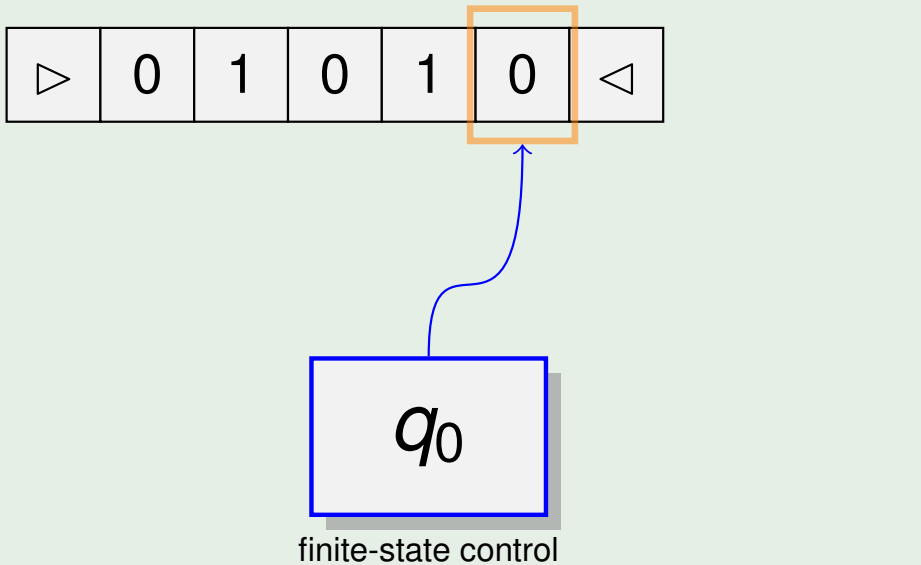


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

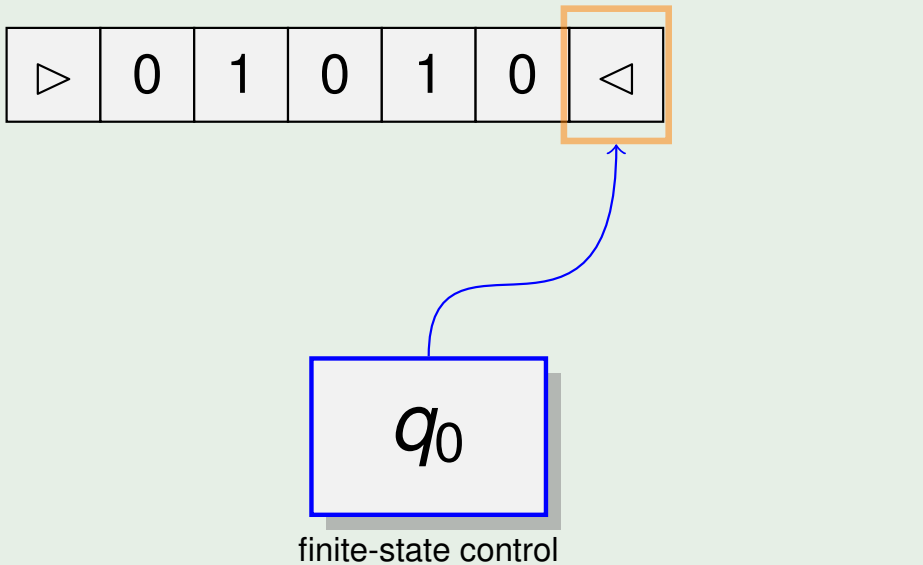


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

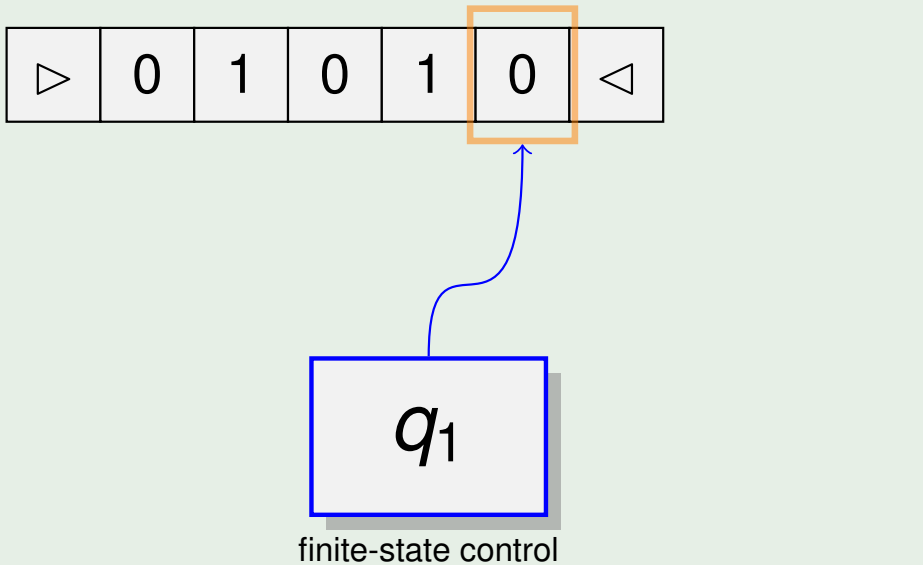


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

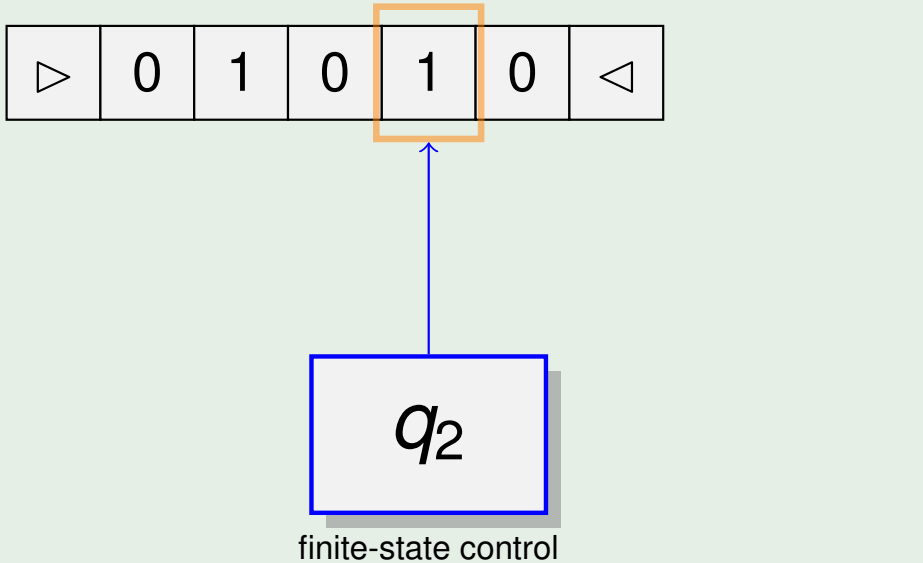


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

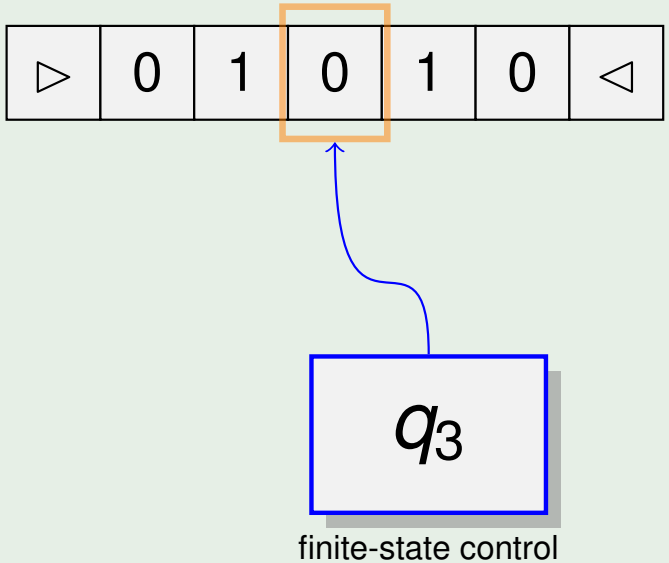


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

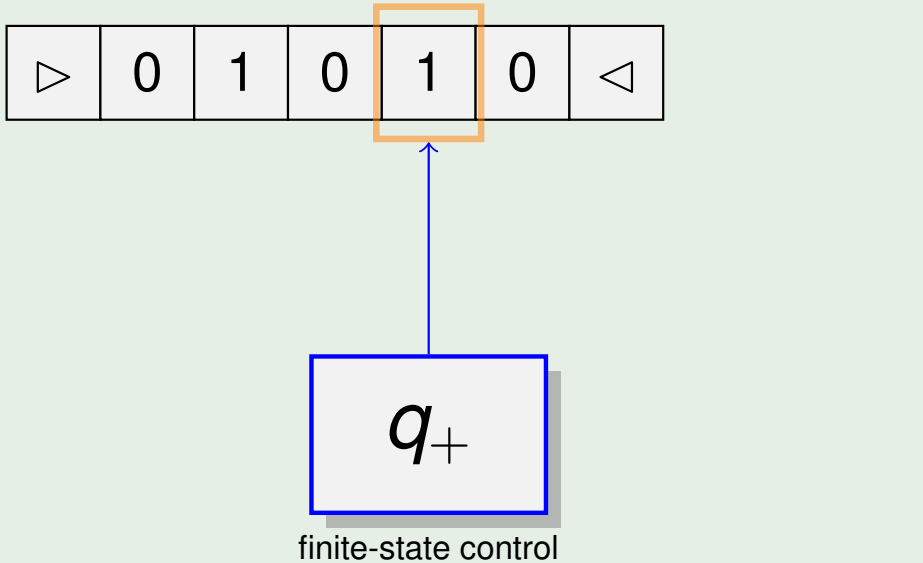


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

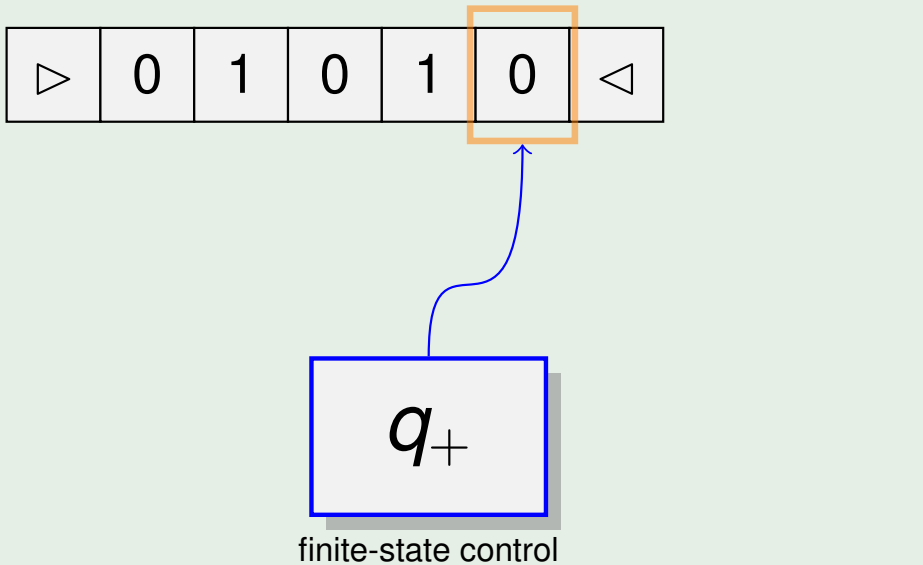


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:

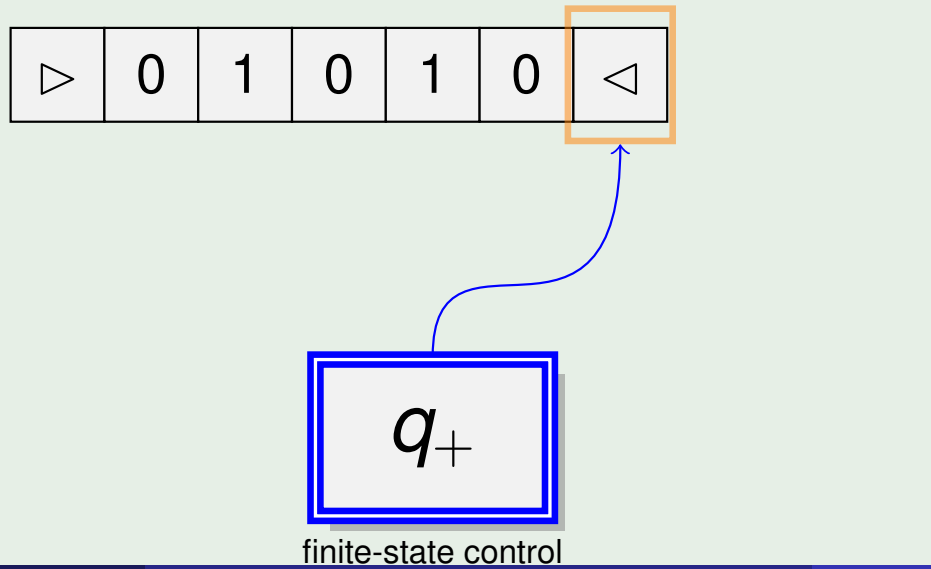


Example:

Let $A = (\{q_0, q_1, \ldots, q_k, q_+\}, \{0,1\}, \triangleright, \triangleleft, \delta, q_0, \{q_+\})$:

δ	$\triangleright$	0	1	$\triangleleft$
q_0	(q_0, R)	(q_0, R)	(q_0, R)	(q_1, L)
q_1	–	(q_2, L)	(q_2, L)	–
q_2	–	(q_3, L)	(q_3, L)	–
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
q_{k-1}	–	(q_k, L)	(q_k, L)	–
q_k	–	(q_+, R)	–	–
q_+	–	(q_+, R)	(q_+, R)	–

Let $k = 3$:



Definition 2.22 (cont.)

A configuration of the 2DFA A is described by a word of the form

$$q \triangleright w \triangleleft \quad \text{or} \quad \triangleright w_1 q w_2 \triangleleft,$$

where $q \in Q$ and $w, w_1, w_2 \in \Sigma^*$.

The transition function δ induces a computation relation $\vdash_A^*$ on the set of configurations, which is the reflexive and transitive closure of the following single-step computation relation $\vdash_A$:

$$\triangleright a_1 \dots a_{i-1} q a_i \dots a_n \triangleleft \vdash \begin{cases} \triangleright a_1 \dots a_{i-2} q' a_{i-1} \dots a_n \triangleleft, & \text{if } \delta(q, a_i) = (q', L), \\ \triangleright a_1 \dots a_i q' a_{i+1} \dots a_n \triangleleft, & \text{if } \delta(q, a_i) = (q', R). \end{cases}$$

The initial configuration for input $w \in \Sigma^*$ is $q_0 \triangleright w \triangleleft$, and a configuration of the form $\triangleright w q \triangleleft$, where $q \in F$, is an accepting configuration.

W.l.o.g. we can assume that $\delta(q, \triangleleft)$ is undefined for all $q \in F$.

Definition 2.22 (cont.)

The language $L(A)$ accepted by A is defined as follows:

$$L(A) = \{ w \in \Sigma^* \mid q_0 \triangleright w \triangleleft \vdash_A^* \triangleright wq \triangleleft \text{ for some } q \in F \},$$

and $\mathcal{L}(\text{2DFA})$ is the class of languages accepted by 2DEAs.

Example (cont.):

Let $k = 3$. On input $w = 01010$, A executes the following computation:

$$\begin{aligned} q_0 \triangleright 01010 \triangleleft \; &\vdash_A \; \triangleright q_0 01010 \triangleleft \; \vdash_A^5 \; \triangleright 01010 q_0 \triangleleft \\ &\vdash_A \; \triangleright 0101 q_1 0 \triangleleft \; \vdash_A \; \triangleright 010 q_2 10 \triangleleft \\ &\vdash_A \; \triangleright 01 q_3 010 \triangleleft \; \vdash_A \; \triangleright 010 q_+ 10 \triangleleft \\ &\vdash_A^2 \; \triangleright 01010 q_+ \triangleleft, \end{aligned}$$

that is, A accepts on input $w = 01010$.

In fact, it is easily seen that $L(A)$ is the language

$$L_k = \{ x = x_1 \dots x_n \in \{0,1\}^* \mid |x| = n \geq k \text{ and } x_{n-k+1} = 0 \}.$$

Example:

Let $A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \triangleright, \triangleleft, \delta, q_0, \{q_1, q_2, q_3\})$, where δ is given through the following table:

	$\triangleright$	0	1	$\triangleleft$
q_0	(q_1, R)	–	–	–
q_1	–	(q_1, R)	(q_2, R)	–
q_2	–	(q_2, R)	(q_3, L)	–
q_3	–	(q_1, R)	(q_3, L)	–

On input $w = 101001$, A executes the following computation:

$$\begin{aligned} q_0 \triangleright 101001 \triangleleft \;&\vdash_A\; \triangleright q_1 101001 \triangleleft \;\vdash_A\; \triangleright 1 q_2 01001 \triangleleft \\ &\vdash_A\; \triangleright 10 q_2 1001 \triangleleft \;\vdash_A\; \triangleright 1 q_3 01001 \triangleleft \\ &\vdash_A\; \triangleright 10 q_1 1001 \triangleleft \;\vdash_A\; \triangleright 101 q_2 001 \triangleleft \\ &\vdash_A^2\; \triangleright 10100 q_2 1 \triangleleft \;\vdash_A\; \triangleright 1010 q_3 01 \triangleleft \\ &\vdash_A\; \triangleright 10100 q_1 1 \triangleleft \;\vdash_A\; \triangleright 101001 q_2 \triangleleft, \end{aligned}$$

that is, A accepts on input $w = 101001$.

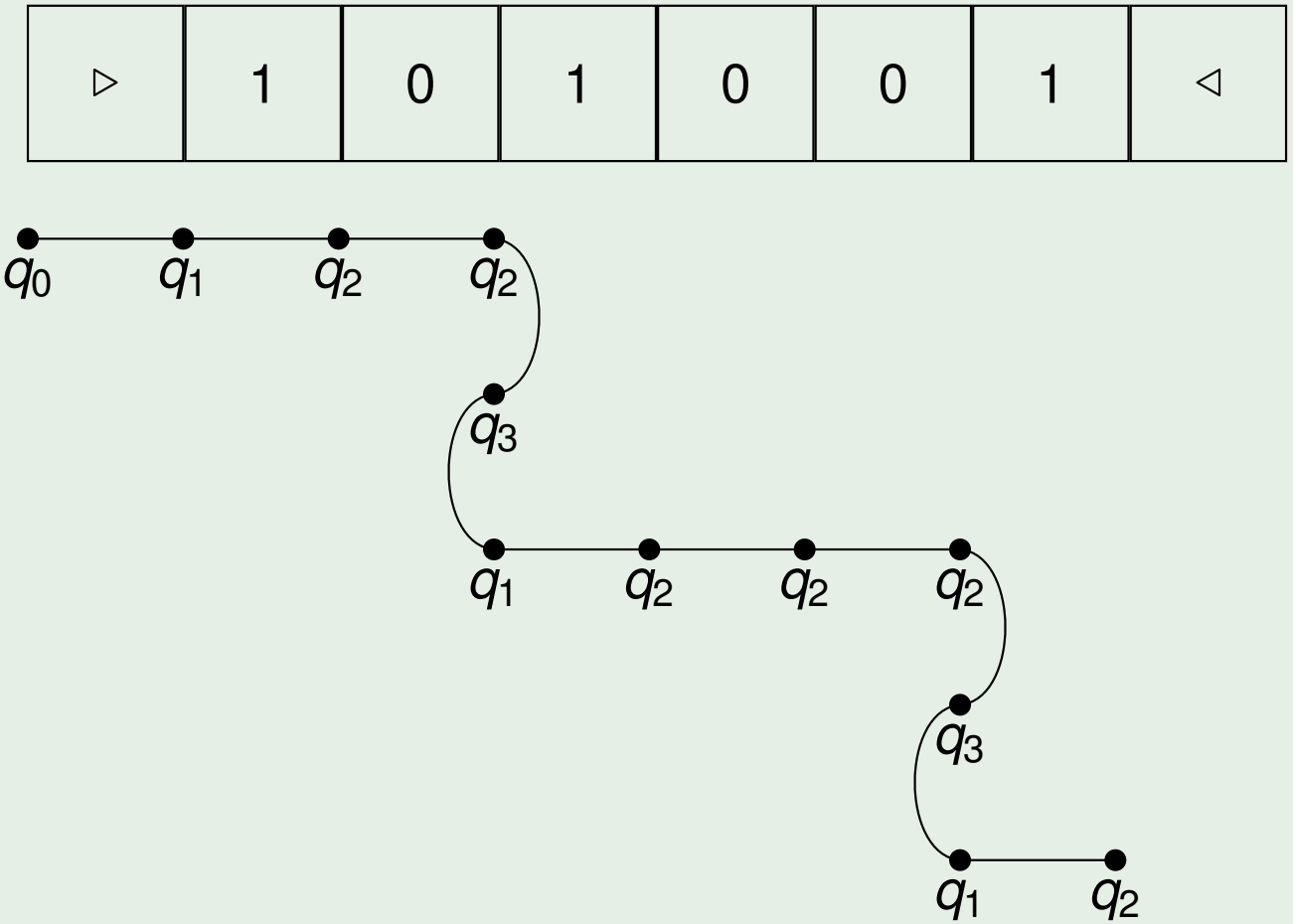
To describe the behaviour of a 2DFA $A = (Q, \Sigma, \triangleright, \triangleleft, \delta, q_0, F)$, we introduce the notion of a crossing sequence.

We consider a computation of A on an input $w \in \Sigma^*$, observing the path taken by the head of A. Each time the head crosses the boundary between two tape squares we note down the new state underneath the corresponding boundary. In this way we obtain a sequence of states for each boundary. Such a sequence is called the crossing sequence (CS) of A on input w at the current position.

Example (cont.):

The 2DFA A executes the above computation on input $w = 101001$, which yields the following graphical representation:

Example (cont.):



Then (q_1) is the CS between $\triangleright$ and 101001 $\triangleleft$,

(q_2, q_3, q_1) is the CS between $\triangleright$10 and 1001 $\triangleleft$,

and (q_2, q_3, q_1) is also the CS between $\triangleright$10100 and 1 $\triangleleft$.

Lemma 2.23

Let $(q_{i_1}, \ldots, q_{i_m})$ be a CS of a 2DFA A on input $w \in \Sigma^*$ between $\triangleright w_1$ and $w_2 \triangleleft$. Then the following statements hold:

(a) *In state q_{i_1}, A performs a move-right step from the last letter of $\triangleright w_1$ to the first letter of $w_2 \triangleleft$.*

(b) *In states q_{i_j}, where $j \equiv 1 \bmod 2$, A performs move-right steps, while in states q_{i_j}, where $j \equiv 0 \bmod 2$, it performs move-left steps.*

(c) *If $w \in L(A)$, then each CS of A on input w has odd length.*

(d) *If $w \in L(A)$, then $q_{i_j} \neq q_{i_{j+2r}}$ for all $j \geq 1$ and all $r \geq 1$.*

(c) holds, as A starts on the left delimiter $\triangleright$ and accepts on the right delimiter $\triangleleft$, and (d) holds, as $q_{i_j} = q_{i_{j+2r}}$ would imply that A has reached a cycle within its computation, that is, it will not terminate.

A sequence of states $(q_1, \ldots, q_k)$ is called a valid CS, if k is odd, all states with even index are pairwise distinct, and also all states with odd index are pairwise distinct.

If $s = |Q|$, then a valid CS has length at most $2s - 1$. Hence, there are only finitely many valid CSs for A.

Theorem 2.24

A language is regular iff it is accepted by a 2DFA, that is, $\mathrm{REG} = \mathcal{L}(\mathrm{2DFA})$.

Proof.

We construct an NFA B from a given 2DFA A such that $L(A) = L(B)$.

The states of B will correspond to the valid CSs of A.

For this construction we need to be able to check whether neighbouring CSs are consistent with each other and with the actual input symbol.

Proof of Theorem 2.24 (cont.)

Let $(q_1, q_2, \ldots, q_k)$ be the left CS of a tape square containing the letter $a \in \Sigma$, and let $(p_1, p_2, \ldots, p_\ell)$ be the right CS of that tape square.

We define right-matching and left-matching pairs of CSs:

1. The empty sequence left- and right-matches the empty sequence.
2. If $(q_3, \ldots, q_k)$ right-matches $(p_1, \ldots, p_\ell)$ and $\delta(q_1, a) = (q_2, L)$, then $(q_1, q_2, q_3, \ldots, q_k)$ right-matches $(p_1, \ldots, p_\ell)$.
3. If $(q_2, \ldots, q_k)$ left-matches $(p_2, \ldots, p_\ell)$ and $\delta(q_1, a) = (p_1, R)$, then $(q_1, q_2, \ldots, q_k)$ right-matches $(p_1, p_2, \ldots, p_\ell)$.
4. If $(q_1, \ldots, q_k)$ left-matches $(p_3, \ldots, p_\ell)$ and $\delta(p_1, a) = (p_2, R)$, then $(q_1, \ldots, q_k)$ left-matches $(p_1, p_2, p_3, \ldots, p_\ell)$.
5. If $(q_2, \ldots, q_k)$ right-matches $(p_2, \ldots, p_\ell)$ and $\delta(p_1, a) = (q_1, L)$, then $(q_1, q_2, \ldots, q_k)$ left-matches $(p_1, p_2, \ldots, p_\ell)$.

Example (cont.):

For the 2DFA A, we consider a tape square containing the letter 1. The empty sequence left-matches the empty sequence and $\delta(q_1, 1) = (q_2, R)$. Hence, by (3) (q_1) right-matches (q_2).

As $\delta(q_2, 1) = (q_3, L)$, (q_2, q_3, q_1) right-matches (q_2) by (2).

Proof of Theorem 2.24 (cont.)

Let $A = (Q, \Sigma, \triangleright, \triangleleft, \delta, q_0, F)$ and let $B = (Q', \Sigma \cup \{\triangleright, \triangleleft\}, \delta', \{q_0'\}, F')$ be the NFA that is defined as follows:

- Q' consists of all valid CSs of A;
- $q_0' = (q_0)$;
- F' consists of all valid CSs that end in a state from F;
- $\delta'((q_1, \ldots, q_k), a) = \{ (p_1, \ldots, p_\ell) \mid (p_1, \ldots, p_\ell)$ is a valid CS such that $(q_1, \ldots, q_k)$ right-matches $(p_1, \ldots, p_\ell)$ for the input letter $a \}$ for all $a \in \Sigma \cup \{\triangleright, \triangleleft\}$.

Proof of Theorem 2.24 (cont.)

While B reads the word $\triangleright w \triangleleft$ (where $w \in \Sigma^*$) letter by letter from left to right, it guesses valid CSs of A and checks whether the current CS, the new CS, and the current letter are compatible with each other.

It can now be shown that $L(B) = \triangleright \cdot L(A) \cdot \triangleleft$. Hence, the language $\triangleright \cdot L(A) \cdot \triangleleft$ is regular, from which it can be concluded that $L(A)$ is regular. □

The 2DFA can be generalized to the nondeterministic two-way finite-state automaton (2NFA).

Theorem 2.25

A language is regular iff it is accepted by a 2NFA, that is, $\mathrm{REG} = \mathcal{L}(\mathrm{2NFA})$.

2.6 Automata with Output

A Moore automaton is a DFA in which an output symbol is assigned to each state. Accordingly, a Moore automaton A is given through a 6-tuple $A = (Q, \Sigma, \Delta, \delta, \sigma, q_0)$, where

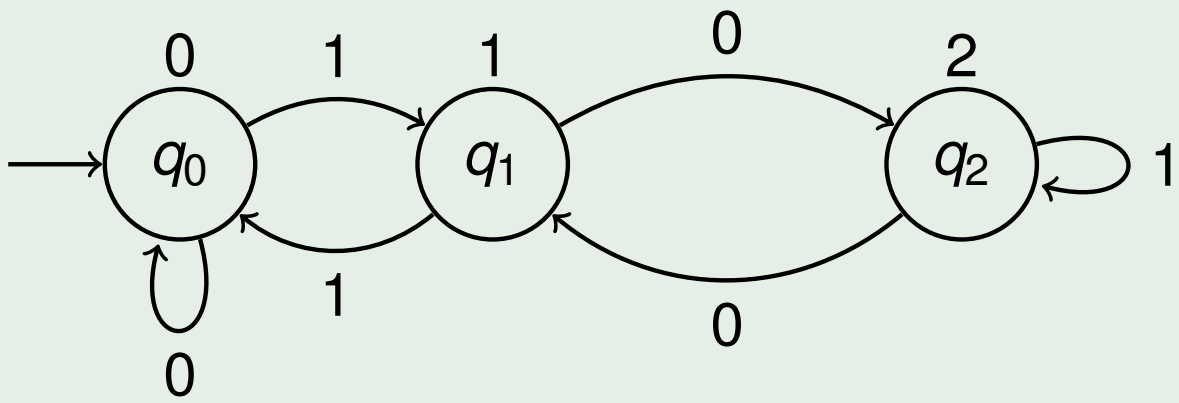
- Q is a finite set of (internal) states,
- Σ is a finite input alphabet,
- Δ is a finite output alphabet,
- $q_0 \in Q$ is the initial state,
- $\delta : Q \times \Sigma \to Q$ is the transition function, and
- $\sigma : Q \to \Delta$ is the output function.

For $u = a_1 a_2 \ldots a_n$, let $q_i := \delta(q_0, a_1 a_2 \ldots a_i)$, $i = 1, 2, \ldots, n$, that is, on input u, A visits the sequence of states $q_0, q_1, q_2, \ldots, q_n$. During this computation A generates the output

$$\sigma(q_0)\sigma(q_1)\sigma(q_2)\ldots\sigma(q_n) \in \Delta^{n+1}.$$

Example:

Let A be the Moore automaton that is given by the following graph, where the output symbols are written as external markings to the various states:



Example (cont.):

Claim:

If $u = b_1 b_2 \dots b_n$, then $\sigma(u) = r_0 r_1 \dots r_n$, where $r_i \equiv \sum_{j=1}^{i} b_j \cdot 2^{i-j} \bmod 3$.

If $u = \mathrm{bin}(m)$, that is, $m = \sum_{j=1}^{n} b_j \cdot 2^{n-j}$, then the last symbol r_n of the output $\sigma(u)$ is just the remainder of m mod 3.

Proof.

For $i = 0, 1, 2$, $\sigma(q_i) = i$. Hence, it suffices to prove the following:

$(*)$ For all $u = b_1 b_2 \dots b_n$, $\delta(q_0, u) = q_i$, where $i \equiv \sum_{j=1}^{n} b_j \cdot 2^{n-j} \bmod 3$.

If $n = 0$, then $u = \varepsilon$, und $\delta(q_0, u) = q_0$.

If $n = 1$, then $u = b_1 \in \{0, 1\}$. Hence, $\delta(q_0, u) = \begin{cases} q_0, & \text{for } b_1 = 0, \\ q_1, & \text{for } b_1 = 1. \end{cases}$

Example (cont.):

Proof (cont.)

Assume that the statement $(*)$ has been verified for some $n \geq 1$, and let $u = b_1 b_2 \ldots b_n b_{n+1}$.

Then $\delta(q_0, u) = \delta(\delta(q_0, b_1 b_2 \ldots b_n), b_{n+1})$.

For $i = n$, the statement holds by the induction hypothesis.

For index $n + 1$, the following can be checked by case analysis:

$$\delta(\delta(q_0, b_1 b_2 \ldots b_n), b_{n+1}) = q_i, \text{ where } i \equiv \sum_{j=1}^{n+1} b_j \cdot 2^{n+1-j} \bmod 3.$$

This completes the proof. □

A Mealey automaton is a DFA that outputs a symbol during each transition. Accordingly, a Mealey automaton A is specified by a 6-tuple $A = (Q, \Sigma, \Delta, \delta, \sigma, q_0)$, where $Q, \Sigma, \Delta, \delta$, and q_0 are defined as for a Moore automaton, while $\sigma : Q \times \Sigma \to \Delta$ is the output function.

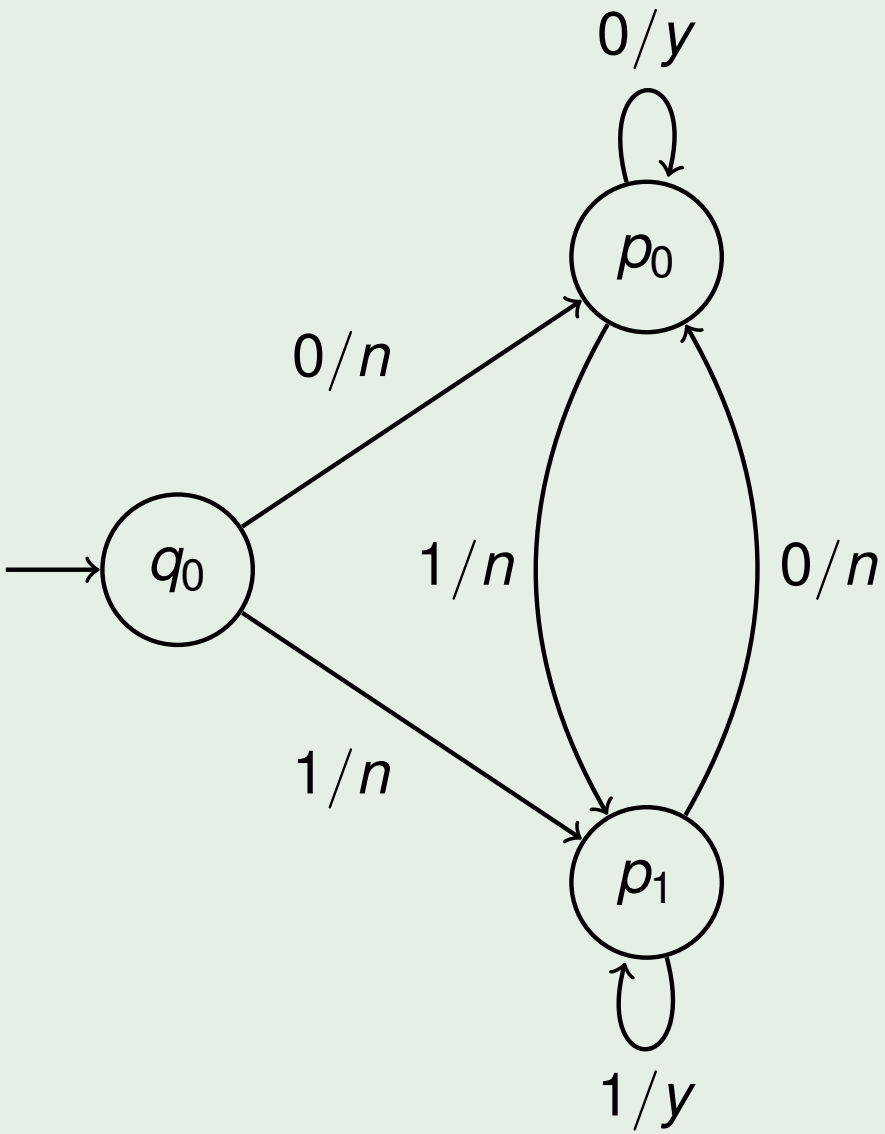
The output function σ can be extended to a function $\sigma : Q \times \Sigma^* \to \Delta^*$:

$$\begin{array}{lcll} \sigma(q, \varepsilon) & := & \varepsilon & \text{for all } q \in Q, \\ \sigma(q, ua) & := & \sigma(q, u) \cdot \sigma(\delta(q, u), a) & \text{for all } q \in Q, u \in \Sigma^*, a \in \Sigma. \end{array}$$

Example:

Let A be the Mealey automaton given through the following graph:

Example (cont.):



Example (cont.):

Let $L = \{0,1\}^* \cdot \{00,11\}$. On input $u = b_1 b_2 \ldots b_m \in \{0,1\}^m$, A produces output $v = c_1 c_2 \ldots c_m \in \{y,n\}^m$, where

$$c_i = \begin{cases} y, & \text{if } b_1 b_2 \ldots b_i \in L, \\ n, & \text{if } b_1 b_2 \ldots b_i \notin L. \end{cases}$$

In state p_0 or p_1, A "stores" the latest input symbol.

Let $A = (Q, \Sigma, \Delta, \delta, \sigma, q_0)$ be a Moore automaton and let $B = (Q', \Sigma, \Delta, \delta', \sigma', q_0')$ be a Mealey automaton. For $u \in \Sigma^*$, let $F_A(u) \in \Delta^*$ and $F_B(u) \in \Delta^*$ be the output words that are generated by A and B on input u. Then $|F_A(u)| = |u| + 1$ and $|F_B(u)| = |u|$.

The automata A und B are called equivalent, if

$$F_A(u) = \sigma(q_0) \cdot F_B(u)$$

for all $u \in \Sigma^*$.

Theorem 2.26

(a) *For each Moore automaton, there exists an equivalent Mealey automaton.*

(b) *For each Mealey automaton, there exists an equivalent Moore Automaton.*

Proof.

As an exercise! □

In fact, the equivalent automata can be constructed effectively!

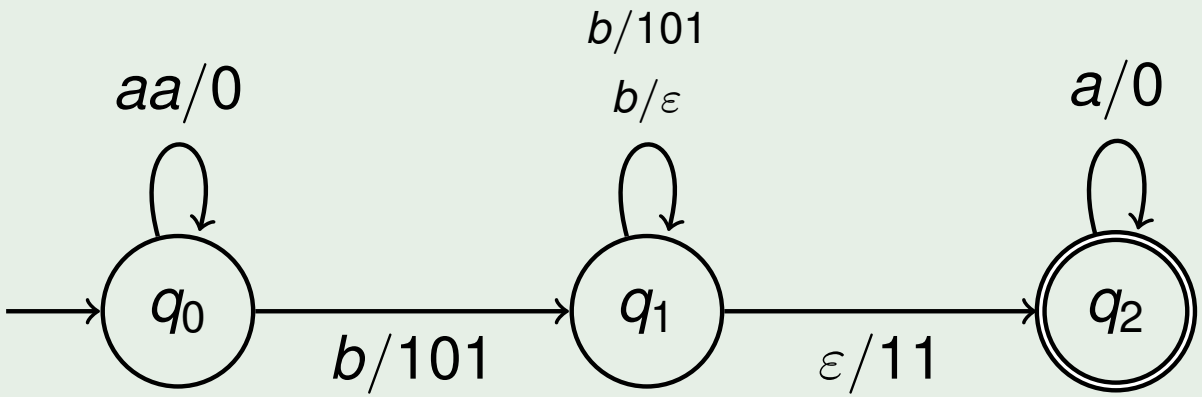
A finite-state transducer (FST) T is given through a 6-tuple

$$T = (Q, \Sigma, \Delta, \delta, q_0, F),$$

where Q, Σ, Δ, and q_0 are defined as for a Mealey automaton,
$F \subseteq Q$ is a set of final states,
and $\delta : D \to 2^{Q \times \Delta^*}$ is the transition and output function.
Here D is a finite subset of $Q \times \Sigma^*$, and
δ associates a finite subset of $Q \times \Delta^*$ to each pair $(q, u) \in D$.

Example:

Let T be the following finite-state transducer:



For $u \in \Sigma^*$, $v \in \Delta^*$ is a possible output of T, if u admits a factorisation of the form $u = u_1 u_2 \cdots u_n$ such that there are states $q_1, q_2, \ldots, q_n \in Q$ and transitions

$$\delta(q_0, u_1) \ni (q_1, v_1), \delta(q_1, u_2) \ni (q_2, v_2), \ldots, \delta(q_{n-1}, u_n) \ni (q_n, v_n)$$

such that $q_n \in F$ and $v = v_1 v_2 \cdots v_n$.
By $T(u) \subseteq \Delta^*$ we denote the set of all possible outputs of T for input u.
In this way T induces a (partial) mapping $T : \Sigma^* \to 2^{\Delta^*}$.

A mapping $\varphi : \Sigma^* \to 2^{\Delta^*}$ is called a finite transduction, if there exists a finite-state transducer T such that $T(w) = \varphi(w)$ for all $w \in \Sigma^*$.

Let $T = (Q, \Sigma, \Delta, \delta, q_0, F)$ be an FST.
For a language $L \subseteq \Sigma^*$,

$$T(L) := \bigcup_{w \in L} T(w)$$

is the image of L w.r.t. T.

For a language $L \subseteq \Delta^*$,

$$T^{-1}(L) := \{ u \in \Sigma^* \mid T(u) \cap L \neq \emptyset \}$$

is the preimage of L w.r.t. T.

By $R_T \subseteq \Sigma^* \times \Delta^*$ we denote the relation

$$R_T := \{ (u, v) \mid v \in T(u) \}.$$

Relations of this form are called rational relations.

Example (cont.):

$$\begin{aligned}
T(aabb) &= \{010111, 010110111\}, \\
T(bbba) &= \{101110, 1011101110, 10110111011110\}, \\
T(\varepsilon) &= \emptyset, \\
T(aaab) &= \emptyset, \\
T(\{b, ba\}) &= \{10111, 101110\}, \\
T^{-1}(\{10111, 101110\}) &= \{ b^{n+1}, b^{n+1}a \mid n \geq 0 \}.
\end{aligned}$$

Theorem 2.27 (Nivat 1968)

Let Σ and Δ be finite alphabets, and let $R \subseteq \Sigma^* \times \Delta^*$.
The relation R is a rational relation if and only if
there exist a finite alphabet Γ, a regular language $L \subseteq \Gamma^*$, and
morphisms $g : \Gamma^* \to \Sigma^*$ and $h : \Gamma^* \to \Delta^*$ such that
$R = \{ (g(w), h(w)) \mid w \in L \}$.

Proof.

"$\Rightarrow$": Let R be a rational relation, that is,

$$R = R_T = \{ (u, v) \mid u \in \Sigma^* \text{ and } v \in T(u) \}$$

for some FST $T = (Q, \Sigma, \Delta, \delta, q_0, F)$.
We must show that $R = \{ (g(w), h(w)) \mid w \in L \}$ for some $L \in \mathrm{REG}(\Gamma)$ and morphisms $g : \Gamma^* \to \Sigma^*$ and $h : \Gamma^* \to \Delta^*$.

Proof of Theorem 2.27 (cont.)

Let $\Gamma := \{ [q, u, v, p] \mid (p, v) \in \delta(q, u) \}$.

From the definition of T it follows that Γ is a finite set, the elements of which we interpret as letters.

We define morphisms $g : \Gamma^* \to \Sigma^*$ and $h : \Gamma^* \to \Delta^*$ through

$$g([q, u, v, p]) := u \text{ and } h([q, u, v, p]) := v.$$

Finally, let $L \subseteq \Gamma^*$ be defined as follows:
$[q_1, u_1, v_1, p_1][q_2, u_2, v_2, p_2] \cdots [q_n, u_n, v_n, p_n] \in L$ iff

1. $q_1 = q_0$,
2. $p_n \in F$, and
3. for all $i = 1, \ldots, n-1$, $p_i = q_{i+1}$.

One can easily define a DFA for this language, that is, $L \in \mathrm{REG}(\Gamma)$.

Proof of Theorem 2.27 (cont.)

The word

$$[q_1, u_1, v_1, p_1][q_2, u_2, v_2, p_2] \cdots [q_n, u_n, v_n, p_n] \in L$$

describes an accepting computation of T for input $u := u_1 u_2 \cdots u_n$ producing output $v := v_1 v_2 \cdots v_n$.

Thus, if $q_0 \notin F$, then

$$R = R_T = \{ (u, v) \mid v \in T(u) \} = \{ (g(w), h(w)) \mid w \in L \}.$$

If $q_0 \in F$, then we consider the language $L' := L \cup \{\varepsilon\}$, since then the empty computation is an accepting computation of T for input ε that produces the output ε.

Proof of Theorem 2.27 (cont.)

"$\Leftarrow$": Now let $R = \{ (g(w), h(w)) \mid w \in L \}$ for a regular language $L \subseteq \Gamma^*$. Then there is a DFA $A = (Q, \Gamma, \delta, q_0, F)$ such that $L(A) = L$. Let T be the FST $T = (Q, \Sigma, \Delta, \delta', q_0, F)$ that is obtained from A by replacing each transition $q \xrightarrow{c} q'$ of A by $q \xrightarrow{g(c)/h(c)} q'$. Then

$$R = \{ (g(w), h(w)) \mid w \in L \} = \{ (u, v) \mid v \in T(u) \}. \qquad \square$$

Corollary 2.28

If T is a finite transduction, then so is T^{-1}.

Corollary 2.29

The class REG *is closed under finite transductions and inverse finite transductions, that is, if* $T \subseteq \Sigma^* \times \Delta^*$ *is a finite transduction, then the following implications hold:*

1. *If* $L \in \mathrm{REG}(\Sigma)$, *then* $T(L) \in \mathrm{REG}(\Delta)$.
2. *If* $L \in \mathrm{REG}(\Delta)$, *then* $T^{-1}(L) \in \mathrm{REG}(\Sigma)$.

Proof.

By Corollary 2.28 it suffices to prove (1).
Let $T \subseteq \Sigma^* \times \Delta^*$ be a finite transduction, and let $L_1 \in \mathrm{REG}(\Sigma)$. We must prove that $T(L_1) \in \mathrm{REG}(\Delta)$ ist.

By Theorem 2.27, there are an alphabet Γ, a language $L \in \mathrm{REG}(\Gamma)$, and morphisms $g : \Gamma^* \to \Sigma^*$ and $h : \Gamma^* \to \Delta^*$ such that

$$T = \{ (g(w), h(w)) \mid w \in L \}.$$

Proof of Corollary 2.29 (cont.)

We obtain the following sequence of equivalent statements:

$$\begin{aligned} T(L_1) &= \{\, v \in \Delta^* \mid \exists u \in L_1 : v \in T(u) \,\} \\ &= \{\, h(w) \mid w \in L \text{ and } \exists u \in L_1 : g(w) = u \,\} \\ &= \{\, h(w) \mid w \in L \text{ and } g(w) \in L_1 \,\} \\ &= \{\, h(w) \mid w \in L \cap g^{-1}(L_1) \,\} \\ &= h(L \cap g^{-1}(L_1)). \end{aligned}$$

By Theorem 2.13, $g^{-1}(L_1) \in \mathrm{REG}(\Gamma)$,
by Theorem 2.8, REG is closed under intersection,
which yields $L \cap g^{-1}(L_1) \in \mathrm{REG}(\Gamma)$,
and by Remark 2.4(c), REG is closed under morphisms, that is,
$T(L_1) = h(L \cap g^{-1}(L_1))$ is a regular language. □

2.7 Regular Expressions

Let Σ be an alphabet, and let Γ be the following alphabet:

$$\Gamma := \Sigma \dot{\cup} \{\emptyset, \varepsilon, +, *, (,)\}.$$

The regular expressions $\mathrm{RA}(\Sigma)$ on Σ are defined as follows, where a language $L(r) \subseteq \Sigma^*$ is associated to each expression r:

(1)	$\emptyset \in \mathrm{RA}(\Sigma)$	$: L(\emptyset) := \emptyset,$
(2)	$\varepsilon \in \mathrm{RA}(\Sigma)$	$: L(\varepsilon) := \{\varepsilon\},$
(3)	$\forall a \in \Sigma : a \in \mathrm{RA}(\Sigma)$	$: L(a) := \{a\}.$
(4)	For all $r, s \in \mathrm{RA}(\Sigma), (r + s) \in \mathrm{RA}(\Sigma)$	$: L(r + s) := L(r) \cup L(s).$
(5)	For all $r, s \in \mathrm{RA}(\Sigma), (rs) \in \mathrm{RA}(\Sigma)$	$: L(rs) := L(r) \cdot L(s).$
(6)	For all $r \in \mathrm{RA}(\Sigma), (r^*) \in \mathrm{RA}(\Sigma)$	$: L(r^*) := (L(r))^*.$

Example:

$r = (0+1)^*00(0+1)^*$: $L(r) = \{ u \in \{0,1\}^* \mid u$ contains the factor $00 \}$.

$s = (0+\varepsilon)(1+10)^*$: $L(s) = \{0,1\}^* \setminus L(r)$.

$t = (a+b)((a+b)(a+b))^*$: $L(t) = \{ u \in \{a,b\}^* \mid |u| \equiv 1 \mod 2 \}$.

If $L = \{x_1, x_2, \ldots, x_m\}$, then $L = L((x_1 + (x_2 + (x_3 + \cdots + x_n) \cdots)))$.

Theorem 2.30

From a regular expression r, an ε-NFA A_r can be constructed such that $L(A_r) = L(r)$.

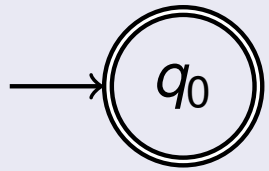
Proof.

By induction on the number of operations used to build r:

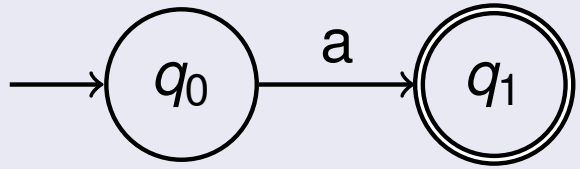
(1) For $r = \emptyset$, we take the DFA A_r: 

Proof of Theorem 2.30 (cont.)

(2) For $r = \varepsilon$, we take the DFA A_r:



(3) For $r = a\ (\in \Sigma)$, we take the DFA A_r:



(4-6) By the induction hypothesis, there are ε-NFAs A_1 and A_2 such that $L(r_1) = L(A_1)$ and $L(r_2) = L(A_2)$. By Theorem 2.21, $\mathcal{L}(\varepsilon\text{-NFA})$ is closed under union, product, and Kleene star. Hence, from A_1 and A_2, we can construct ε-NFAs for $L(r_1 + r_2)$, $L(r_1 r_2)$, and $L(r_1^*)$.

□

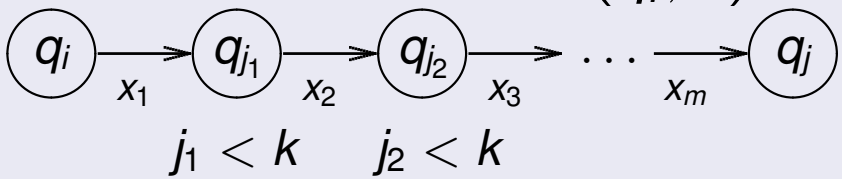
Theorem 2.31

From a DFA A, one can construct a regular expression r such that $L(A) = L(r)$.

Proof.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA for $L = L(A) \subseteq \Sigma^*$.
Assume that $Q = \{q_0, q_1, q_2, \ldots, q_n\}$.
For all $i, j \in \{0, 1, \ldots, n\}$ and all $k \in \{0, 1, \ldots, n+1\}$,

$$R_{i,j}^k := \{\, x \in \Sigma^* \mid \delta(q_i, x) = q_j, \text{ and for all } v, w \in \Sigma^+, \text{ if } x = vw \text{ and } \delta(q_i, v) = q_\ell, \text{ then } \ell < k \,\}.$$



Thus, $x \in R_{i,j}^k$ iff the DFA A, starting in state q_i and reading input x, reaches state q_j, and all states q_ℓ encountered during this computation satisfy the condition that $\ell < k$.

Then $L = \bigcup_{q_j \in F} R_{0,j}^{n+1}$.

Proof of Theorem 2.31 (cont.)

$R^0_{i,j} = \{ a \in \Sigma \mid \delta(q_i, a) = q_j \} \qquad (i \neq j)$

$R^0_{i,i} = \{ a \in \Sigma \mid \delta(q_i, a) = q_i \} \cup \{\varepsilon\}$

Thus: $R^0_{i,j}$ are finite languages, that is,
there exist regular expressions $\alpha^0_{i,j}$ for them.

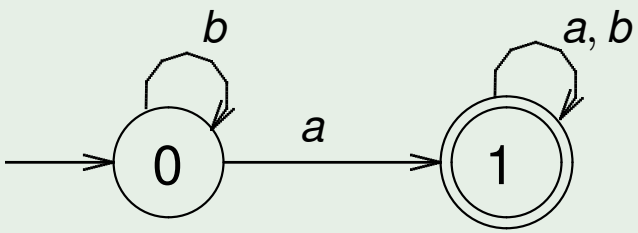
$R^{k+1}_{i,j} = R^k_{i,j} \cup R^k_{i,k} (R^k_{k,k})^* R^k_{k,j}$

$\alpha^{k+1}_{i,j} := (\alpha^k_{i,j} + \alpha^k_{i,k} (\alpha^k_{k,k})^* \alpha^k_{k,j})$

Now $L = \bigcup_{q_j \in F} R^{n+1}_{0,j}$,
that is, if $F = \{q_{i_1}, q_{i_2}, \ldots, q_{i_m}\}$, then
$\gamma = (\alpha^{n+1}_{0,i_1} + (\alpha^{n+1}_{0,i_2} + \cdots + \alpha^{n+1}_{0,i_m}))$. □

Example:

A:



$R^k_{i,j} : i, j \in \{0, 1\}, k \in \{0, 1, 2\}.$

Example (cont.):

$$R^0_{0,0} = \{\varepsilon, b\}, \quad R^0_{0,1} = \{a\}, \quad R^0_{1,0} = \emptyset, \quad R^0_{1,1} = \{\varepsilon, a, b\}$$

$$R^1_{0,0} = R^0_{0,0} \cup R^0_{0,0} \cdot (R^0_{0,0})^* \cdot R^0_{0,0} = \{\varepsilon, b\} \cup \{\varepsilon, b\} \cdot \{\varepsilon, b\}^* \cdot \{\varepsilon, b\} = b^*$$

$$R^1_{0,1} = R^0_{0,1} \cup R^0_{0,0} \cdot (R^0_{0,0})^* \cdot R^0_{0,1} = \{a\} \cup \{\varepsilon, b\} \cdot \{\varepsilon, b\}^* \cdot \{a\} = b^* \cdot a$$

$$R^1_{1,0} = R^0_{1,0} \cup R^0_{1,0} \cdot (R^0_{0,0})^* \cdot R^0_{0,0} = \emptyset \cup \emptyset \cdot \{\varepsilon, b\}^* \cdot \{\varepsilon, b\} = \emptyset$$

$$R^1_{1,1} = R^0_{1,1} \cup R^0_{1,0} \cdot (R^0_{0,0})^* \cdot R^0_{0,1} = \{\varepsilon, a, b\} \cup \emptyset \cdot \{\varepsilon, b\}^* \cdot \{a\} = \{\varepsilon, a, b\}$$

$$R^2_{0,0} = R^1_{0,0} \cup R^1_{0,1} \cdot (R^1_{1,1})^* \cdot R^1_{1,0} = b^* \cup b^*a \cdot \{\varepsilon, a, b\}^* \cdot \emptyset = b^*$$

$$R^2_{0,1} = R^1_{0,1} \cup R^1_{0,1} \cdot (R^1_{1,1})^* \cdot R^1_{1,1} =$$
$$b^*a \cup b^*a \cdot \{\varepsilon, a, b\}^* \cdot \{\varepsilon, a, b\} = b^*a \cdot \{a, b\}^*$$

$$R^2_{1,0} = R^1_{1,0} \cup R^1_{1,1} \cdot (R^1_{1,1})^* \cdot R^1_{1,0} = \emptyset \cup \ldots \cdot \emptyset = \emptyset$$

$$R^2_{1,1} = R^1_{1,1} \cup R^1_{1,1} \cdot (R^1_{1,1})^* \cdot R^1_{1,1} =$$
$$\{\varepsilon, a, b\} \cup \{\varepsilon, a, b\} \cdot \{\varepsilon, a, b\}^* \cdot \{\varepsilon, a, b\} = \{a, b\}^*.$$

$$L(A) = R^2_{0,1} = b^* \cdot a \cdot \{a, b\}^* = \{w \in \{a, b\}^* \mid |w|_a \geq 1\}$$ □

Corollary 2.32 (Kleene's Theorem)

A language L is regular iff there exists a regular expression r such that $L(r) = L$.

A substitution $\varphi : \Sigma^* \to 2^{\Delta^*}$ is called regular, if $\varphi(a) \in \mathrm{REG}(\Delta)$ for all $a \in \Sigma$.

Corollary 2.33

The language class REG is closed under regular substitutions, that is, if $L \in \mathrm{REG}(\Sigma)$ and if $\varphi : \Sigma^* \to 2^{\Delta^*}$ is a regular substitution, then $\varphi(L) \in \mathrm{REG}(\Delta)$.

Proof.

For $L \in \mathrm{REG}(\Sigma)$, there exists a regular expression $r \in \mathrm{RA}(\Sigma)$ such that $L = L(r)$.

For each letter $a \in \Sigma$, there exists a regular expression $r_a \in \mathrm{RA}(\Delta)$ such that $\varphi(a) = L(r_a)$.

Proof of Corollary 2.33 (cont.)

Let $s \in \mathrm{RA}(\Delta)$ be the regular expression that we obtain from r by replacing each occurrence of each letter $a \in \Sigma$ by the expression r_a.

Claim:

$L(s) = \varphi(L)$.

Proof by induction on the structure of r:

1. If $r = \emptyset$, then $L = \emptyset = \varphi(L)$ and $s = \emptyset$, that is, $\varphi(L) = L(s)$.
2. If $r = \varepsilon$, then $L = \{\varepsilon\} = \varphi(L)$ and $s = \varepsilon$.
3. If $r = a \in \Sigma$, then $L = \{a\}$ and $s = r_a$. Hence,
$$\varphi(L) = \varphi(a) = L(r_a) = L(s).$$

Proof (cont.)

4 If $r = (r_1 + r_2)$, then $L = L(r_1) \cup L(r_2)$ and $s = (s_1 + s_2)$.
By the ind. hyp., $L(s_i) = \varphi(L(r_i))$, $i = 1, 2$. Hence,

$$\varphi(L) = \varphi(L(r_1)) \cup \varphi(L(r_2)) = L(s_1) \cup L(s_2) = L(s).$$

5 If $r = (r_1 r_2)$, then $L = L(r_1) \cdot L(r_2)$ and $s = (s_1 s_2)$.
By the ind. hyp., $L(s_i) = \varphi(L(r_i))$, $i = 1, 2$. Hence,

$$\varphi(L) = \varphi(L(r_1)) \cdot \varphi(L(r_2)) = L(s_1) \cdot L(s_2) = L(s).$$

6 If $r = (r_1^*)$, then $L = (L(r_1))^*$ and $s = (s_1^*)$.
By the ind. hyp., $L(s_1) = \varphi(L(r_1))$. Hence,

$$\varphi(L) = (\varphi(L(r_1)))^* = (L(s_1))^* = L(s).$$

It follows that $\varphi(L) \in \mathrm{REG}(\Delta)$.

2.8 The Pumping Lemma

Theorem 2.34 (Pumping Lemma)

Let L be a regular language. Then there exists a positive integer n such that each word $x \in L$ of length $|x| \geq n$ admits a factorization of the form $x = uvw$ that satisfies all of the following properties:

(1) $|v| \geq 1$,

(2) $|uv| \leq n$,

(3) $uv^i w \in L$ *for all* $i \in \mathbb{N}$.

Proof.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA for L.
We choose $n := |Q|$, and consider a word $x \in L$ satisfying $|x| \geq n$.

$$A : q_0 \xrightarrow[x_1]{} q_1 \xrightarrow[x_2]{} \cdots \xrightarrow[x_{n-1}]{} q_{n-1} \xrightarrow[x_n]{} q_n \xrightarrow[x']{*} q' \in F,$$

where $x = x_1 x_2 \ldots x_n x'$, $x_1, x_2, \ldots, x_n \in \Sigma$ and $x' \in \Sigma^*$.

Proof of Theorem 2.34 (cont.)

Then there are r and s such that $0 \leq r < s \leq n$ and $q_r = q_s$.
Hence, x can be written as $x = uvw$, where

$$\begin{aligned} |u| &= r, \\ 1 \leq |v| &= s - r \leq n, \\ \hat{\delta}(q_0, u) &= q_r, \\ \hat{\delta}(q_r, v) &= q_s = q_r, \\ \hat{\delta}(q_s, w) &= q' \in F. \end{aligned}$$

It follows that

$$\begin{aligned} \hat{\delta}(q_0, uv^i w) = \hat{\delta}(q_r, v^i w) &= \hat{\delta}(q_s, v^{i-1} w) \\ &= \hat{\delta}(q_r, v^{i-1} w) = \hat{\delta}(q_s, w) = q' \in F, \end{aligned}$$

that is, $uv^i w \in L(A) = L$. □

Example 1:

Claim: $L = \{ a^m b^m \mid m \geq 1 \}$ is not regular.

Proof (indirect):

Assume that L were regular. Then L satisfies the Pumping Lemma, that is, $\exists n \in \mathbb{N}_+ \forall x \in L : |x| \geq n \rightsquigarrow \exists x = uvw :$
$|v| \geq 1$, $|uv| \leq n$, and $uv^i w \in L$ for all $i \geq 0$.

Consider the word $x := a^n b^n : x \in L$ and $|x| = 2n > n$.
Hence: $\exists x = uvw$ s. t. $|v| \geq 1$, $|uv| \leq n$, and $uv^i w \in L$ for all $i \geq 0$.

$x = a^n b^n = uvw$, where $|uv| \leq n$
$\rightsquigarrow$ $u = a^r$, $v = a^s$, and $w = a^{n-s-r} b^n$
for certain integers r, s satisfying $r \geq 0$, $s \geq 1$, $r + s \leq n$.
Thus: $uv^0 w = a^r a^{n-s-r} b^n = a^{n-s} b^n \notin L$, a contradiction!

As L does not satisfy the Pumping Lemma, it is not regular. □

Example 2:

Claim: $L = \{ 0^m \mid m \text{ is a square number} \}$ is not regular.

Proof (indirect)

Assume that L were regular.
Let n be the constant for L from the Pumping Lemma.
Consider the word $x := 0^{n^2} : x \in L$ and $|x| = n^2 > n$.
Hence: $\exists x = uvw$ s. t. $|v| \geq 1$, $|uv| \leq n$, and $uv^i w \in L$ for all $i \geq 0$.

Now consider the word uv^2w:

$$\begin{array}{rcl@{\qquad}rcl} n^2 & = & |uvw| & < & |uv^2w| \\ & = & |uvw| + |v| & = & n^2 + |v| \\ & \leq & n^2 + n & < & n^2 + 2n + 1 \\ & = & (n+1)^2, & & \end{array}$$

that is, $|uv^2w|$ is not a square number, and so, $uv^2w \notin L$.
This contradiction shows that L is not regular. □

Example 3:

Let $L = \{ c^m a^n b^n \mid m, n \geq 0 \} \cup \{a, b\}^*$.

Claim: L satisfies the Pumping Lemma with the constant $k = 1$.

Proof.

Let $x \in L$, where $|x| \geq k$.

(i) $x \in \{a, b\}^*$: obvious.

(ii) $x = c^m a^n b^n$ for some $m \geq 1$:
Choose $u := \varepsilon$, $v := c$, $w := c^{m-1} a^n b^n$.
Then: $x = uvw$, $1 \leq |v|$, $|uv| = 1 \leq k$,
$uv^i w = c^{i+m-1} a^n b^n \in L$ for all $i \geq 0$. □

Claim: L is not regular.

Proof.

L has infinite index, and hence, by Theorem 2.12, it is not regular. □

2.9 Decision Problems

The membership problem for a regular language:

INSTANCE : $x \in \Sigma^*$.
QUESTION : Is x in L?

This problem is solvable in time $|x|$ using a DFA.

The emptiness problem for a DFA (NFA):

INSTANCE : A DFA (NFA) A.
QUESTION : Is $L(A) = \emptyset$?

This is decidable in time $O(|A|^2)$, as $L(A) \neq \emptyset$ iff a final state is reachable from the initial state in the graph of A.

The finiteness problem for a DFA (NFA):

INSTANCE : A DFA (NFA) A.
QUESTION : Is $L(A)$ finite?

This is decidable in polynomial time, as $L(A)$ is infinite iff

$\exists q_0$ initial state $\exists q_1 \exists q_2$ final state: $q_0 \to^* q_1 \to^+ q_1 \to^* q_2$,

which can be checked using the graph of A.

The intersection emptiness problem for regular grammars (or NFAs):

INSTANCE : Two grammars G_1 and G_2.
QUESTION : Is $L(G_1) \cap L(G_2)$ empty?

This is decidable in quadratic time:
Construct an NFA (or a grammar) for $L(G_1) \cap L(G_2)$
and test for emptiness.

The inclusion problem for regular languages:

INSTANCE : Two regular grammars G_1 and G_2.
QUESTION : Is $L(G_1)$ a subset of $L(G_2)$?

Decidable, as $L(G_1) \subseteq L(G_2)$ iff $L(G_1) \cap \overline{L(G_2)} = \emptyset$,
and a DFA for $L(G_1) \cap \overline{L(G_2)}$ can be constructed from G_1 and G_2.

If $L(G_1)$ and $L(G_2)$ are given through DFAs, then this problem is decidable in quadratic time.

The equivalence problem for regular languages:

INSTANCE : Two regular grammars G_1 and G_2.
QUESTION : Are $L(G_1)$ and $L(G_2)$ equal?

Decidable, as $L(G_1) = L(G_2)$ iff $L(G_1) \subseteq L(G_2)$ and $L(G_2) \subseteq L(G_1)$.

Chapter 3:

Context-Free Languages and Pushdown Automata

3.1. Context-Free Grammars

A phrase-structure grammar $G = (N, T, S, P)$ is called context-free, if $\ell \in N$ for each production $(\ell \to r) \in P$.

A language $L \subseteq T^*$ is called context-free, if there exists a context-free grammar G satisfying $L(G) = L$.

By CFL(Σ) we denote the class of all context-free languages over Σ, and CFL is the class of all context-free languages.

Obiously, we have REG $\subseteq$ CFL.

Let $G = (N, T, S, P)$ be a context-free grammar, and
let $\alpha_0 \to_G \alpha_1 \to_G \ldots \to_G \alpha_n$ be a derivation in G.
Then there exist $\beta_i, \gamma_i \in (N \cup T)^*$ and $(A_i \to r_i) \in P$ such that

$$\alpha_i = \beta_i A_i \gamma_i \text{ and } \alpha_{i+1} = \beta_i r_i \gamma_i \quad (0 \leq i \leq n-1).$$

This derivation is called a left derivation if $|\beta_i| \leq |\beta_{i+1}|$ for all i,
it is called a right derivation if $|\gamma_i| \leq |\gamma_{i+1}|$ for all i,
and it is called a leftmost derivation if $\beta_i \in T^*$ for all i.
If $\alpha_i \to \alpha_{i+1}$ is a step in a leftmost derivation, this is denoted as
$\alpha_i \to_{\mathrm{lm}} \alpha_{i+1}$.

Remark:

If $\alpha_0 \in N$ and $\alpha_n \in T^*$, then each left derivation $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_n$ is necessarily leftmost.

Example:

(a) $G_1 := (\{A\}, \{a\}, A, \{A \to AA, A \to a\})$.

(b) $G_2 := (\{S\}, \{a, b\}, S, \{S \to aSa, S \to bSb, S \to \varepsilon\})$.

(c) $G_3 := (\{A\}, \{[,], a, \#, \uparrow\}, A, \{A \to [A\#A], A \to [A \uparrow A], A \to a\})$.

(a): The derivation $A \to AA \to Aa \to AAa \to aAa \to aaa$
is neither a left nor a right derivation.
(b): Each derivation that starts with S is a left and a right derivation.
(c): The derivation

$$A \to [A\#A] \to [A\#[A \uparrow A]] \to [a\#[A \uparrow A]] \to [a\#[a \uparrow A]]$$

is neither a left nor a right derivation, but the derivation

$$A \to [A\#A] \to [a\#A] \to [a\#[A \uparrow A]] \to [a\#[a \uparrow A]]$$

is a leftmost derivation.

The grammar $G = (N, T, S, P)$ is called unambiguous w.r.t. $A \in N$ if there exists exactly one left derivation $A \to_P^* \alpha$ for each $\alpha \in L(G, A)$.

It is called unambiguous if it is unambiguous w.r.t. all its nonterminals.

It is called ambiguous if it is not unambiguous.

A context-free language is called unambiguous if it is generated by a context-free grammar $G = (N, T, S, P)$ that is unambiguous.

A context-free language L is called inherently ambiguous if it is not generated by any unambiguous grammar.

Example (cont.):

(a) G_1 is not unambiguous, as

$$A \to AA \to AAA \to aAA \to aaA \to aaa$$

and

$$A \to AA \to aA \to aAA \to aaA \to aaa$$

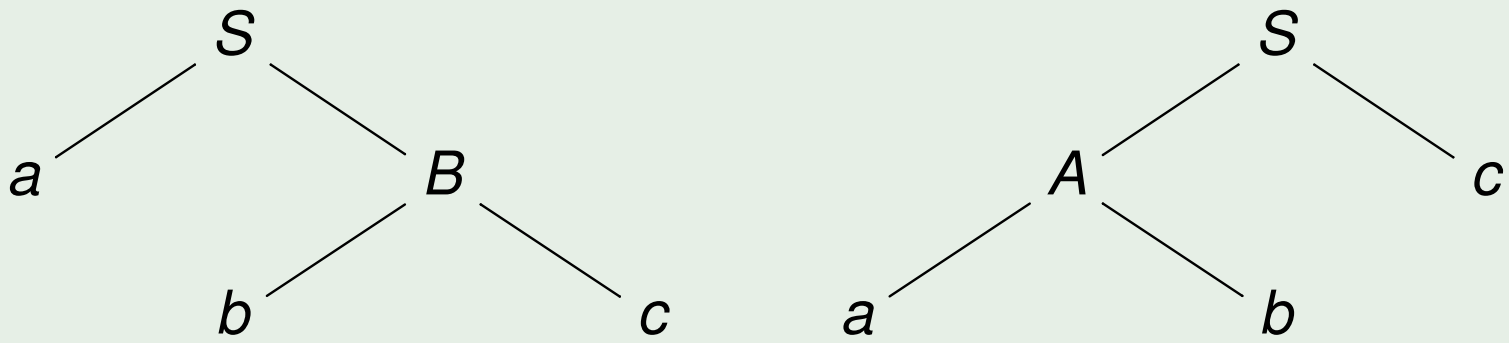
are two different left derivations for aaa.

(b) G_2 is trivially unambiguous.

(c) It can be shown that G_3 is unambiguous.

Example:

Let $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, where
$P := \{S \to aB, S \to Ac, A \to ab, B \to bc\}$:



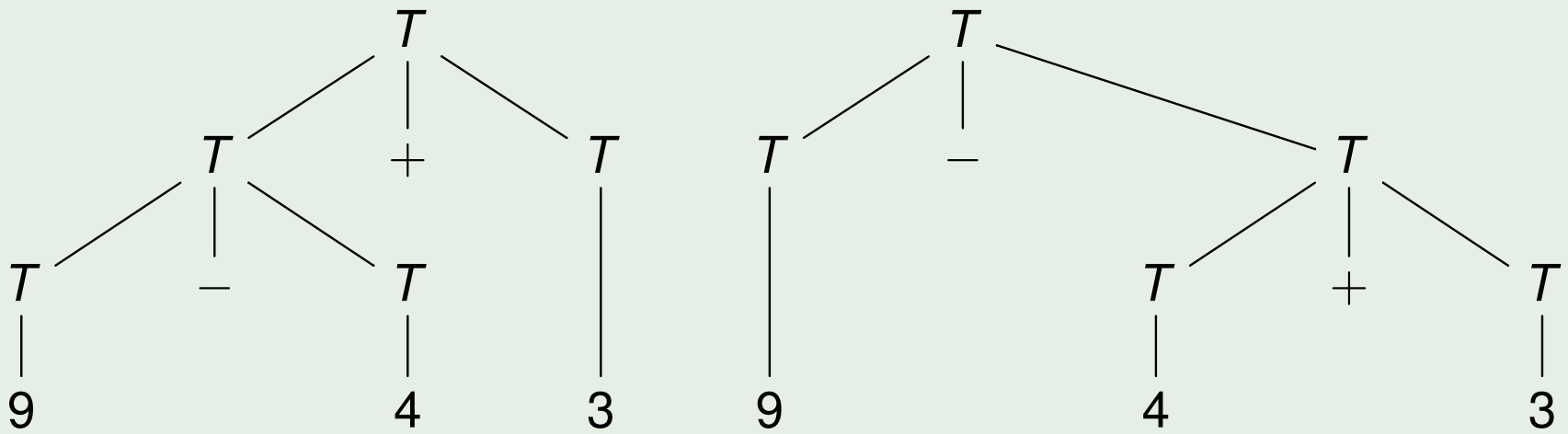
G is ambiguous.

$G' := (\{S\}, \{a, b, c\}, \{S \to abc\}, S)$ is unambiguous and
$L(G') = L(G)$.

The language $L := \{ a^i b^j c^k \mid i = j \text{ or } j = k \}$ is inherently ambiguous.

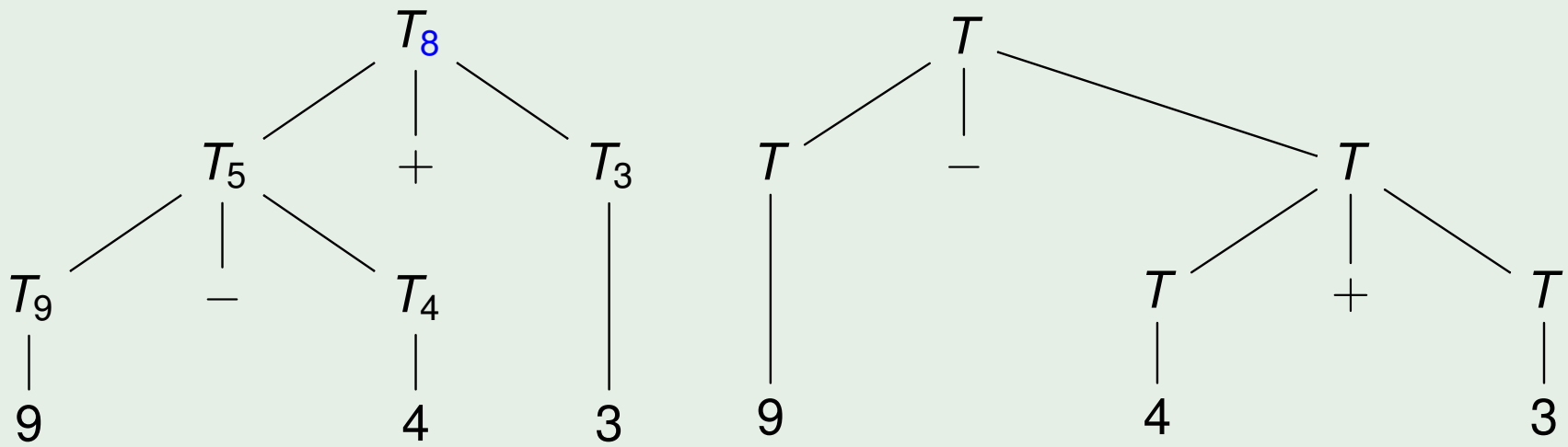
Example:

$G = (\{T\}, \{1, 2, \ldots, 9, +, -\}, \{T \to T + T \mid T - T \mid 1 \mid 2 \mid \cdots \mid 9\}, T)$:



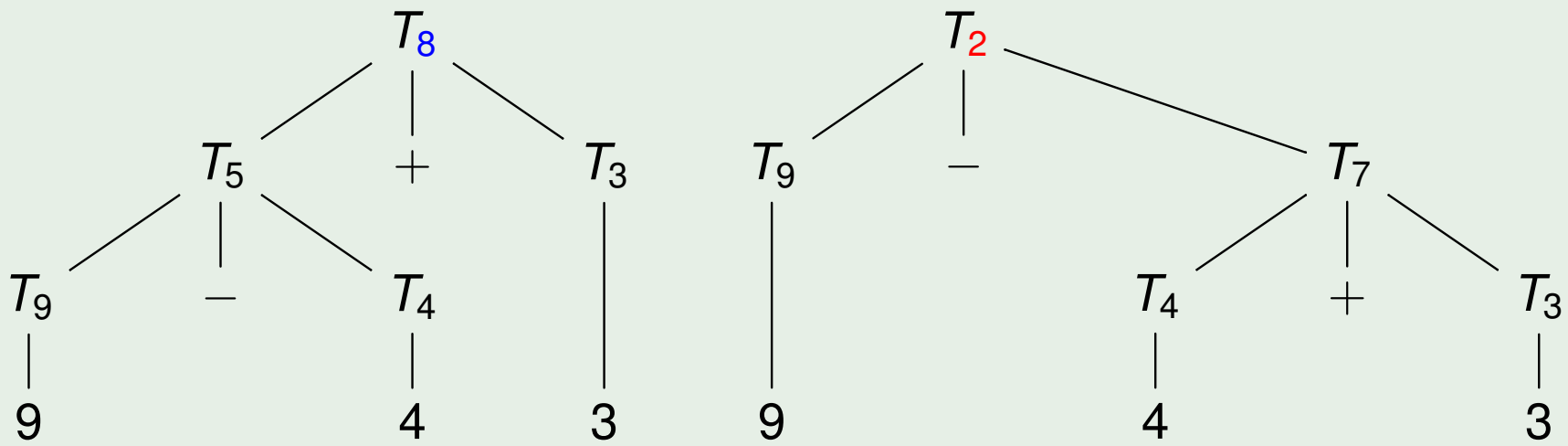
Example:

$G = (\{T\}, \{1, 2, \ldots, 9, +, -\}, \{T \to T + T \mid T - T \mid 1 \mid 2 \mid \cdots \mid 9\}, T)$:



Example:

$G = (\{T\}, \{1, 2, \ldots, 9, +, -\}, \{T \to T + T \mid T - T \mid 1 \mid 2 \mid \cdots \mid 9\}, T)$:



Syntax Trees

Let $G = (V, T, P, S)$ be a context-free grammar, and let $S = x_0 \to x_1 \to x_2 \to \ldots \to x_n = x$ be a derivation in G for a word $x \in L(G)$.

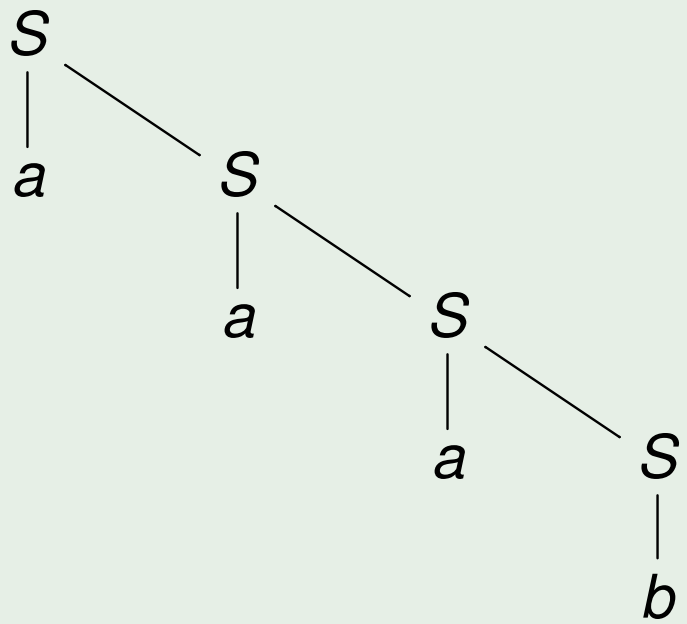
We can describe this derivation through a syntax tree:

Start: Introduce a root with label S.

Step i: If $x_{i-1} = uAv \to urv = x_i$, where $u, v \in (V \cup \Sigma)^*$ and $(A \to r) \in P$, then add $|r|$ children to the node with label A which are labelled from left to right with the symbols of r.

Example:

$G = (\{S\}, \{a, b\}, \{S \to aS, S \to b\}, S)$:

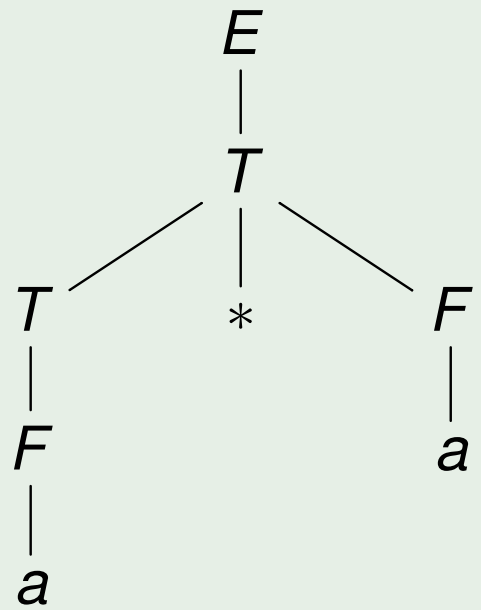


Beispiel:

Let $G = (\{E, F, T\}, \{a, *\}, E, \{E \to T, T \to F, T \to T * F, F \to a\})$:

(1.) $\underline{E} \to \underline{T} \to \underline{T} * F \to \underline{F} * F \to a * \underline{F} \to a * a$

(2.) $\underline{E} \to \underline{T} \to T * \underline{F} \to \underline{T} * a \to \underline{F} * a \to a * a$



(1) Left derivation
(2) Right derivation

Theorem 3.1

Let $G = (N, T, S, P)$ be a context-free grammar.

(a) *If $A \in N$ and $x \in (N \cup T)^*$ such that $A \to^* x$, then there exists a syntax tree with root labelled A and leaves labelled with x.*

(b) *If there exists a syntax tree with root labelled $A \in N$ and leaves labelled with $x \in T^*$, then $A \to_{\mathrm{lm}}^* x$.*

(c) *For each nonterminal $A \in N$ and each word $x \in L(G, A)$, the number of leftmost derivations of x from A coincides with the number of syntax trees with root labelled A and leaves labelled with x.*

Corollary 3.2

A context-free grammar G is unambiguous if and only if there exists a unique syntax tree with root labelled A and leaves labelled x for each $A \in N$ and each word $x \in L(G, A)$.

We could consider rightmost derivations instead of leftmost derivations. Theorem 3.1 also holds for rightmost derivations.

Thus, for each word $w \in L(G, A)$, there are as many leftmost derivations as there are rightmost derivations.

A context-free grammar $G = (N, T, S, P)$ is called proper, if it satisfies the following conditions:

(1) $\forall A \in N : L(G, A) = \{ w \in T^* \mid A \to_P^* w \} \neq \emptyset$, und

(2) $\forall A \in N \, \exists u, v \in T^* : S \to_P^* uAv$.

Thus, in a proper context-free grammar, there are no useless nonterminals.

Lemma 3.3

It is decidable whether a given context-free grammar $G = (N, T, S, P)$ is proper.

Proof of Lemma 3.3.

Inductively, we first determine the set of nonterminals $V_{\text{term}} = \{ A \in N \mid L(G, A) \neq \emptyset \}$:

$$\begin{aligned} V_1 &:= \{ A \in N \mid \exists x \in T^* : (A \to x) \in P \}; \\ V_{i+1} &:= V_i \cup \{ A \in N \mid \exists \alpha \in (V_i \cup T)^* : (A \to \alpha) \in P \} \ (i \geq 1). \end{aligned}$$

Then $V_{\text{term}} = \bigcup_{i \geq 1} V_i = V_{|N|}$.

Next, we inductively determine the set of reachable nonterminals $V_{\text{reach}} = \{ A \in N \mid \exists \alpha, \beta \in (V_{\text{term}} \cup T)^* : S \to_P^* \alpha A \beta \}$:

$$\begin{aligned} U_1 &:= \{ A \in V_{\text{term}} \mid \exists \alpha, \beta \in (V_{\text{term}} \cup T)^* : (S \to \alpha A \beta) \in P \}; \\ U_{i+1} &:= U_i \cup \{ A \in V_{\text{term}} \mid \exists B \in U_i \, \exists \alpha, \beta \in (V_{\text{term}} \cup T)^* : \\ &\qquad (B \to \alpha A \beta) \in P \} \ (i \geq 1). \end{aligned}$$

Then $V_{\text{reach}} = \bigcup_{i \geq 1} U_i = U_{|N|}$.

Now G is proper iff $V_{\text{reach}} = V_{\text{term}} = N$. □

Lemma 3.4

From any given context-free grammar $G = (N, T, S, P)$, one can construct a proper context-free grammar $G' = (N', T, S, P')$ such that $L(G') = L(G)$.

Proof.

Just take $N' = V_{\text{reach}}$ and

$$P' = \{ (A \to \alpha) \in P \mid A \in V_{\text{reach}} \text{ and } \alpha \in (V_{\text{reach}} \cup T)^* \}.$$

Then G' is a proper context-free grammar and $L(G') = L(G)$. □

A production $(\ell \to r) \in P$ is called terminal if $r \in T^*$.
It is called an ε-production if $r = \varepsilon$.

If $\varepsilon \in L(G)$, then G must contain at least one ε-production.
In fact, it can be required that $(S \to \varepsilon)$ is the only ε-production in G.

Lemma 3.5

From a given context-free grammar $G = (N, T, S, P)$, one can construct an equivalent context-free grammar $G' = (N', T, S', P')$ such that S' does not occur on the righthand side of any production in P', and G' contains no ε-production with the only possible exception of $(S' \to \varepsilon)$. In fact, $(S' \to \varepsilon) \in P'$ iff $\varepsilon \in L(G) = L(G')$.

Proof.

First we consider the case that $\varepsilon \notin L(G)$.

Task: Elimination of all ε-productions.

(1.) Determine $V_1 := \{ A \in N \mid A \to_G^* \varepsilon \}$:

$$V_1^{(1)} := \{ A \in N \mid (A \to \varepsilon) \in P \};$$
$$V_1^{(i+1)} := V_1^{(i)} \cup \{ A \in N \mid \exists r \in V_1^{(i)*} : (A \to r) \in P \}.$$

Then $V_1 = \bigcup_{i \geq 1} V_1^{(i)} = V_1^{(|N|)}$.

(2.) Remove all ε-productions.

Proof of Lemma 3.5 (cont.)

(3.) $\forall B \to xAy$ such that $A \in V_1$ and $xy \neq \varepsilon$:
Introduce the new production $B \to xy$.

Then $G' = (N, T, S, P')$ does not contain any ε-productions and $L(G') = L(G)$.

If $\varepsilon \in L(G)$, then first apply the construction above, which yields a context-free grammar $G' = (N, T, S, P')$ for $L(G) \setminus \{\varepsilon\}$ without ε-productions.
Then introduce a new nonterminal S' and the productions $(S' \to \varepsilon)$ and $(S' \to S)$, where S' is taken as the new start symbol.
The resulting grammar $G'' = (N, T, S', P'')$ generates $L(G)$, $(S' \to \varepsilon)$ is the only ε-production in P'', and the start symbol S' does not occur on the righthand side of any production. □

Let $G = (N, T, S, P)$ be a context-free grammar.
A production $(A \to B) \in P$, where $A, B \in N$, is called a chain rule.

Example:

A grammar for arithmetic expressions with brackets:
$G = (N, T, E, P)$, where $N = \{E, T, F\}$, $T = \{a, +, -, *, /, (,)\}$, and P contains the following productions:

$E \to T \mid E + T \mid E - T$,
$T \to F \mid T * F \mid T/F$,
$F \to a \mid (E)$.

This grammar contains the chain rules $(E \to T)$ and $(T \to F)$.

Lemma 3.6

From a given context-free grammar $G = (N, T, S, P)$, one can construct an equivalent context-free grammar G' that does not contain any chain rules.

Proof of Lemma 3.6.

By Lemma 3.5, G does not contain any ε-productions.

We define an equiv. relation $\sim$ on N: $A \sim B$ iff $A \to_P^* B$ and $B \to_P^* A$.

Let $[A] = \{ B \in N \mid A \sim B \}$ denote the equivalence class of A.

For each $A \in N$, choose a unique representative $A_0 \in [A]$, replace all occurrences of all nonterminals $B \in [A]$ within P by A_0, and delete all productions of the form $(A_0 \to A_0)$.

Let $V = \{A_1, A_2, \ldots, A_n\}$ be the remaining nonterminals. W.l.o.g. we can assume for each remaining chain rule $(A_i \to A_j)$ that $i < j$.

For $k = n-1, n-2, \ldots, 2, 1$:
If $(A_k \to A_{k'}) \in P$ (where $k' > k$) and if $A_{k'} \to x_1 \mid x_2 \mid \ldots \mid x_m$ are all $A_{k'}$-productions, then replace $(A_k \to A_{k'})$ by $A_k \to x_1 \mid x_2 \mid \ldots \mid x_m$.
(By the I.H. $|x_i| \geq 2$ or $x_i \in T$, $i = 1, 2, \ldots, m$).

The resulting grammar is equivalent to G and it does not contain any chain-rules. □

Example (cont.):

$E \to T \mid E + T \mid E - T$
$T \to F \mid T * F \mid T/F$
$F \to a \mid (E)$

Remove the chain rules $(T \to F)$ and $(E \to T)$:

There are no equivalent nonterminals.

We order the set N through $E < T < F$:

$(T \to F)$ is replaced by: $T \to a \mid (E) \mid T * F \mid T/F$,

$(E \to T)$ is replaced by: $E \to a \mid (E) \mid T * F \mid T/F \mid E + T \mid E - T$.

Thus, the new grammar has 12 productions, while the given one had only 8.

Definition 3.7

Let T be a (terminal) alphabet, and let $\overline{T} := \{ \bar{a} \mid a \in T \}$ be a set of 'copies' of T such that $T \cap \overline{T} = \emptyset$.

The Dyck language D_T^* on T is generated by the grammar $G = (\{S, A\}, T \cup \overline{T}, S, P)$, where P contains the following productions:

$$P = \{ S \to AS, S \to \varepsilon, A \to aS\bar{a} \mid a \in T \}.$$

The words from the language $D_T := L(G, A)$ are called Dyck primes.

Examples:

$ab a\bar{a}\bar{b}\bar{a}, a\bar{a}, ba\bar{a}\bar{b} \in D_T$ and $ab\bar{b}\bar{a}ba\bar{a}\bar{b}, \varepsilon \in D_T^* \setminus D_T$.

As $T = \{a, b\}$ contains just two letters, D_T and D_T^* are denoted as D_2 and D_2^*.

Corollary 3.8

REG $\subsetneq$ CFL.

Proof.

As observed before, REG $\subseteq$ CFL.

On the other hand,
$D_T^* \cap \{ a^n \bar{a}^m \mid n, m \geq 1 \} = \{ a^n \bar{a}^n \mid n \geq 1 \}$ is not regular.

The class REG is closed under intersection (Theorem 2.8).

If $D_T^* \in$ REG, then also $\{ a^n \bar{a}^n \mid n \geq 1 \} \in$ REG, a contradiction.

Thus, $D_T^* \in$ CFL $\setminus$ REG. □

3.2. Normal Forms of Context-Free Grammars

A context-free grammar $G = (N, T, S, P)$ is in weak Chomsky Normal Form, if $r \in N^* \cup T \cup \{\varepsilon\}$ for each production $(\ell \to r) \in P$.

The grammar G is in Chomsky Normal Form (CNF), if $r \in N^2 \cup T \cup \{\varepsilon\}$ for each production $(\ell \to r) \in P$).

Theorem 3.9 (Chomsky 1959)

Given a context-free grammar G, one can effectively construct an equivalent context-free grammar G' that is in Chomsky Normal Form. In addition, it can be ensured that the start symbol S' of G' does not occur on the righthand side of any production and that G' does not contain any ε-production apart from possibly $(S' \to \varepsilon)$.

Proof of Theorem 3.9.

From the previous subsection we recall that we can construct a context-free grammar $G_1 = (N_1, T, S_1, P_1)$ from G such that $L(G_1) = L(G)$ and G_1 contains no chain rules and satisfies the condition on ε-productions.

Let $N_1' := \{ A_a \mid a \in T \}$ be a new alphabet of nonterminals.
We take $G_2 := (N_2, T, S_2, P_2)$, where

$$N_2 := N_1 \cup N_1', \ S_2 := S_1, \text{ and } P_2 := P_{2,1} \cup \{ A_a \to a \mid a \in T \}.$$

Here $P_{2,1}$ is obtained from P_1 by replacing in each righthand side r, where $|r| > 1$, each occurrence of a terminal symbol $a \in T$ by $A_a \in N_1'$.

Then P_2 contains three types of productions:

Proof of Theorem 3.9 (cont.).

(1.) possibly the ε-production $(S_2 \to \varepsilon)$,

(2.) the terminal productions of the form $(A \to b) \in P_1$, and the new terminal productions $(A_a \to a)$ $(a \in T)$,

(3.) nonterminal productions of the form $(A \to r)$, where $r \in N^*$ and $|r| \geq 2$.

Thus, G_2 is in weak Chomsky Normal Form.

If $(A \to B_1 B_2 \cdots B_m)$ is a nonterminal production s.t. $m > 2$, then we introduce new nonterminals $A^{(1)}, A^{(2)}, \ldots, A^{(m-2)}$, and we replace this production by the following ones:

$$(A \to B_1 A^{(1)}), (A^{(1)} \to B_2 A^{(2)}), \ldots, (A^{(m-2)} \to B_{m-1} B_m).$$

We obtain a grammar G' in CNF that is equivalent to G_2 and therewith to G. In addition, G' satisfies the condition on ε-productions. $\square$

Example:

Let $G = (\{S, A, B\}, \{a, b\}, S, P)$, where P is the following system:

$$P := \{S \to bA \mid aB, A \to bAA \mid aS \mid a, B \to aBB \mid bS \mid b\}.$$

Then G is a proper context-free grammar.

The first step yields the grammar $G_2 = (N_2, \{a, b\}, S, P_2)$, where $N_2 = \{S, A, B, A_a, A_b\}$ and

$$\begin{aligned} P_2 \quad = \quad & \{ S \to A_bA \mid A_aB, A \to A_bAA \mid A_aS \mid a, B \to A_aBB \mid A_bS \mid b, \\ & A_a \to a, A_b \to b \}, \end{aligned}$$

which is in weak CNF.

Example (cont.):

From G_2 we obtain the grammar $G' = (N', \{a, b\}, S, P')$, where $N' = \{S, A, B, A_a, A_b, C_1, C_2\}$ and

$$\begin{aligned} P' \;=\; \{S \to A_bA \mid A_aB,\;\; & A \to A_bC_1 \mid A_aS \mid a, C_1 \to AA, \\ & B \to A_aC_2 \mid A_bS \mid b, C_2 \to BB, \\ & A_a \to a, A_b \to b\}. \end{aligned}$$

This grammar is in CNF, and it is equivalent to G. □

A context-free grammar $G = (N, T, S, P)$ is in

Greibach Normal Form,

if $r \in T \cdot N^*$ holds for each production $(\ell \to r) \in P$.

Theorem 3.10 (Greibach 1965)

Given a context-free grammar G such that $\varepsilon \notin L(G)$, one can effectively construct an equivalent context-free grammar G' that is in Greibach Normal Form.

Lemma 3.11

Let $G = (N, T, S, P)$ be a context-free grammar, let $(A \to \alpha_1 B \alpha_2) \in P$, and let $(B \to \beta_1), (B \to \beta_2), \ldots, (B \to \beta_r) \in P$ be the set of all B-produktions in G. Further, let $G_1 = (N, T, S, P_1)$ be the grammar that is obtained G by replacing the production $(A \to \alpha_1 B \alpha_2)$ by the productions

$$(A \to \alpha_1 \beta_1 \alpha_2), (A \to \alpha_1 \beta_2 \alpha_2), \ldots, (A \to \alpha_1 \beta_r \alpha_2).$$

Then $L(G_1) = L(G)$.

Lemma 3.12

Let $G = (N, T, S, P)$ be a context-free grammar and let

$$(A \to A\alpha_1), (A \to A\alpha_2), \ldots, (A \to A\alpha_r) \in P$$

be the set of A-produktions the righthand side of which has the prefix A. Let $(A \to \beta_1), (A \to \beta_2), \ldots, (A \to \beta_s) \in P$ be the other A-productions of G. The grammar $G_1 = (N \cup \{B\}, T, S, P_1)$ is obtained from G by introducing the new nonterminal B and by replacing the A-productions of G by the following productions:

$$\begin{array}{llllll} (A \to \beta_1), & \ldots, & (A \to \beta_s), & (A \to \beta_1 B), & \ldots, & (A \to \beta_s B), \\ (B \to \alpha_1), & \ldots, & (B \to \alpha_r), & (B \to \alpha_1 B), & \ldots, & (B \to \alpha_r B). \end{array}$$

Then $L(G_1) = L(G)$.

Proof of Theorem 3.10.

Let G be in CNF with $N = \{A_1, A_2, \ldots, A_m\}$.

(1.) Modify the productions such that $(A_i \to A_j\alpha) \in P$ implies $i < j$:

FOR $i := 1$ TO m DO
  FOR $j := 1$ TO $i - 1$ DO
    FOR ALL $(A_i \to A_j\alpha) \in P$ DO
      Let $A_j \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ be all A_j-productions.
      Add the productions $A_i \to \beta_1\alpha \mid \beta_2\alpha \mid \ldots \mid \beta_n\alpha$
      and delete $(A_i \to A_j\alpha)$
    END;
  END;

(2.) Remove left recursive productions using Lemma 3.12:

  IF there is a production of the form $(A_i \to A_i\alpha)$ THEN
    use Lemma 3.12 with the new nonterminal B_i
  END
END.

Proof of Theorem 3.10 (cont.).

(3.) The righthand side of each A_m-production begins with a terminal symbol. We now enforce this also for all A_i-productions, $i = m-1, m-2, \ldots, 1$:

FOR $i := m-1$ DOWNTO 1 DO
  FOR ALL $(A_i \to A_j\alpha) \in P$, $j > i$, DO
    Let $A_j \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ be all A_j-productions.
    Add the productions $A_i \to \beta_1\alpha \mid \beta_2\alpha \mid \ldots \mid \beta_n\alpha$
    and delete $(A_i \to A_j\alpha)$
  END
END

(4.) Modify the B_i-productions $(B_i \to A_j\alpha)$ $(1 \leq i \leq n)$:

Let $A_j \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ be all A_j-productions.
Add the productions $B_i \to \beta_1\alpha \mid \beta_2\alpha \mid \ldots \mid \beta_n\alpha$
and delete $(B_i \to A_j\alpha)$. □

Example:

Let $G = (\{A_1, A_2, A_3\}, \{a, b\}, A_1, P)$, where

$$P : (A_1 \to A_2A_3), (A_2 \to A_3A_1), (A_2 \to b), (A_3 \to A_1A_2), (A_3 \to a).$$

Step 1: The A_1- and A_2-productions are already in the correct form,

$(A_3 \to A_1A_2)$ is replaced by $(A_3 \to A_2A_3A_2)$,

$(A_3 \to A_2A_3A_2)$ is replaced by
$(A_3 \to A_3A_1A_3A_2)$ and $(A_3 \to bA_3A_2)$.

Step 2: Lemma 3.12 is applied to the A_3-productions:

$(A_3 \to bA_3A_2), (A_3 \to a), (A_3 \to bA_3A_2B_3), (A_3 \to aB_3),$
$(B_3 \to A_1A_3A_2), (B_3 \to A_1A_3A_2B_3).$

Example (cont.):

Step 3: All A_i-productions are brought into Greibach form:

$$(A_3 \to bA_3A_2), (A_3 \to a), (A_3 \to bA_3A_2B_3), (A_3 \to aB_3),$$
$$(A_2 \to bA_3A_2A_1), (A_2 \to aA_1), (A_2 \to bA_3A_2B_3A_1),$$
$$(A_2 \to aB_3A_1), (A_2 \to b),$$
$$(A_1 \to bA_3A_2A_1A_3), (A_1 \to aA_1A_3),$$
$$(A_1 \to bA_3A_2B_3A_1A_3), (A_1 \to aB_3A_1A_3), (A_1 \to bA_3).$$

Step 4: The B_3-productions are brought into Greibach form:

$$(B_3 \to bA_3A_2A_1A_3A_3A_2), (B_3 \to aA_1A_3A_3A_2),$$
$$(B_3 \to bA_3A_2B_3A_1A_3A_3A_2), (B_3 \to aB_3A_1A_3A_3A_2),$$
$$(B_3 \to bA_3A_3A_2), (B_3 \to bA_3A_2A_1A_3A_3A_2B_3),$$
$$(B_3 \to aA_1A_3A_3A_2B_3), (B_3 \to bA_3A_2B_3A_1A_3A_3A_2B_3),$$
$$(B_3 \to aB_3A_1A_3A_3A_2B_3), (B_3 \to bA_3A_3A_2B_3).$$

While G has only 5 short productions, G' has 24 long ones. □

A context-free grammar $G = (N, T, S, P)$ is in

quadratic Greibach Normal Form,

if $r \in T \cdot N^*$ and $|r| \leq 3$ for each production $(\ell \to r) \in P$, that is, r contains at most two nonterminals.

Theorem 3.13 (Rosenkrantz 1967)

Given a context-free grammar G such that $\varepsilon \notin L(G)$, one can effectively construct an equivalent context-free grammar G' that is in quadratic Greibach Normal Form.

3.3. A Pumping Lemma for Context-Free Languages

Theorem 3.14 (Pumping Lemma: Bar-Hillel, Perles, Shamir 1961)

Let L be a context-free language on Σ. Then there exists a constant k that depends on L such that each word $z \in L$, $|z| \geq k$, has a factorization of the form $z = uvwxy$ that satisfies all of the following conditions:

(1) $|vx| \geq 1$,

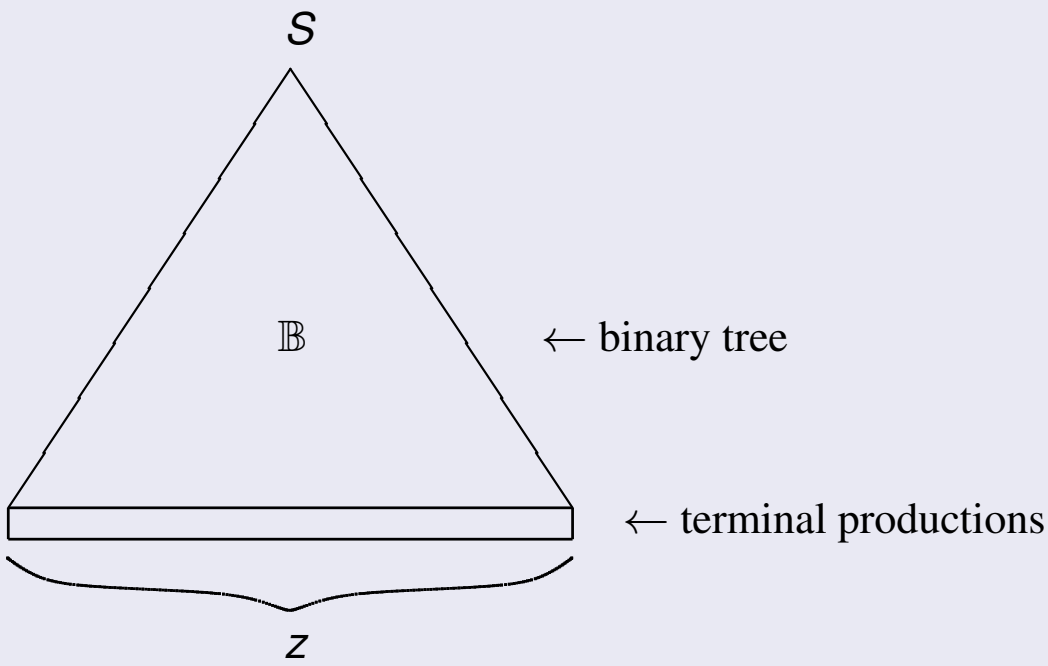
(2) $|vwx| \leq k$,

(3) $uv^i wx^i y \in L$ *for all* $i \geq 0$.

Proof of Theorem 3.14.

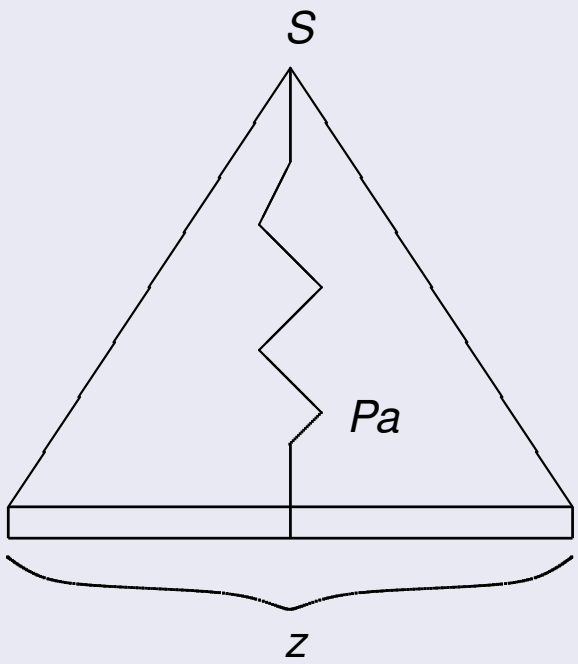
Let $G = (N, \Sigma, P, S)$ be a context-free grammar in CNF for $L - \{\varepsilon\}$, and let $n = |N|$.
We choose $k := 2^n$.
Now let $z \in L$ such that $|z| \geq k$.

We consider a syntax tree for z:



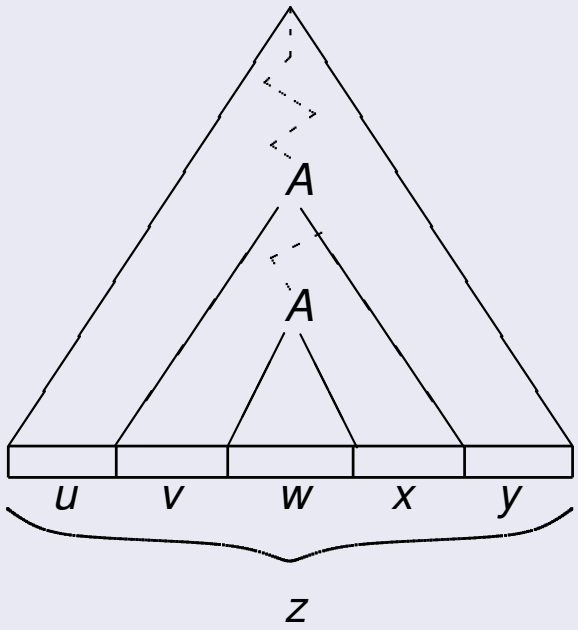
Proof of Theorem 3.14 (cont.)

$\mathbb{B}$ has $|z| \geq k = 2^n$ leaves. Thus, $\mathbb{B}$ contains a path of length $\ell \geq n$.
We consider a path Pa of maximal length ℓ:



Proof of Theorem 3.14 (cont.)

Pa contains $\ell + 1 \geq n + 1$ nodes labelled with nonterminals, that is, at least one nonterminal occurs twice at the nodes of *Pa*.



We choose the first such repetition in *Pa* starting from the leaf.

Proof of Theorem 3.14 (cont.)

$S \to^* uAy$

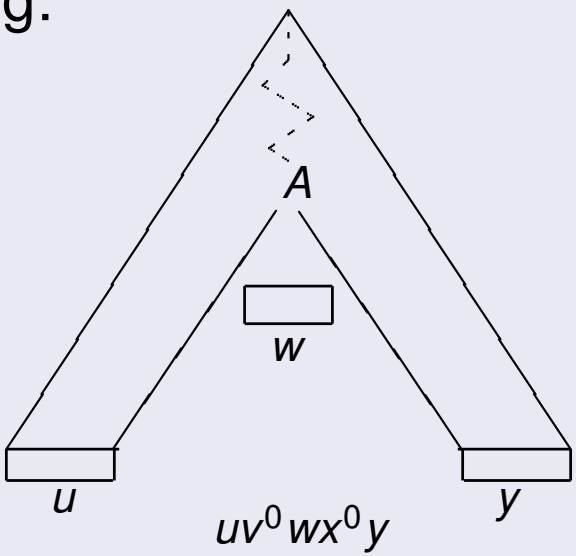
$A \to BC \to^* vAx$

$A \to^* w$

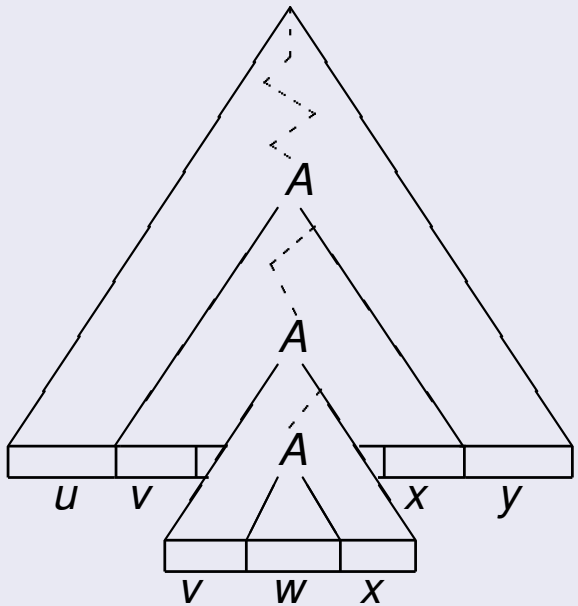
Chomsky Normal Form : $|vx| \geq 1$

Height of upper node with label $A \leq n$: $|vwx| \leq 2^n = k$

Pumping:



uv^0wx^0y



uv^2wx^2y

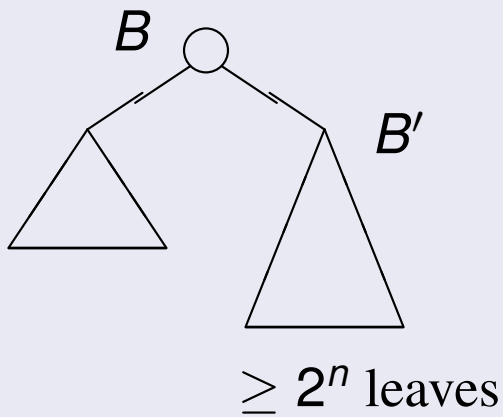
Lemma 3.15

Let B be a binary tree, each inner node of which has two sons. If B has at least 2^n leaves, then B contains a path of length at least n.

Proof by induction on n:

$n = 0$: Number of leaves is $\geq 2^0 = 1$.
B contains a path of length ≥ 0.

$n \rightsquigarrow n+1$: B has $\geq 2^{n+1}$ leaves:



I.H.: B' contains a path of length $\geq n$.
Hence: B contains a path of length $\geq n+1$. □

Example:

Claim: $L = \{ a^m b^m c^m \mid m \geq 1 \}$ is not context-free.

Proof (indirect).

Assume that L is context-free. Then L satisfies the Pumping Lemma, that is, $\exists k \in \mathbb{N}_+ \, \forall z \in L : |z| \geq k \rightsquigarrow \exists z = uvwxy :$
$|vx| \geq 1$, $|vwx| \leq k$, and $uv^i wx^i y \in L$ for all $i \geq 0$.

Consider the word $z := a^k b^k c^k : z \in L$ and $|z| = 3k \geq k$.
Hence:
$\exists z = uvwxy$ s.t. $vx \neq \varepsilon$, $|vwx| \leq k$, and $uv^i wx^i y \in L$ for all $i \geq 0$.
$|vwx| \leq k : |vx|_a = 0$ or $|vx|_c = 0$
$\rightsquigarrow uv^0 wx^0 y = uwy \notin L$. Contradiction!
Thus, L is not context-free. □

Example:

The language $L := \{ a^n b^m c^n d^m \mid n, m \geq 1 \}$ is not context-free, as for $z = a^k b^k c^k d^k$, no factor vwx satisfying $|vwx| \leq k$ can possibly contain a's and c's or b's and d's. □

Example:

Let $L := \{ a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 0, \text{ and } i > 0 \text{ implies } j = k = \ell \}$, and let $n > 0$ be a constant.

If $z = b^j c^k d^\ell$, $|z| \geq n$, then we choose vwx as a factor of b^j, c^k or d^ℓ. For all $m \geq 0$, $uv^m wx^m y \in L$.

If $z = a^j b^j c^j d^j$, $|z| \geq n$ and $i > 0$, then we choose vwx as a factor a^j, and it follows that $uv^m wx^m y \in L$ for all $m \geq 0$.

Thus, L satisfies the Pumping Lemma, but we will see later that L is not context-free. □

Theorem 3.16

Each context-free language over a one-letter alphabet is regular.

Proof.

Let L be a context-free language over $\{a\}$, and let k be the constant from the Pumping Lemma for L:

$\forall z \in L : |z| \geq k \rightsquigarrow \exists z = uvwxy : vx \neq \varepsilon, |vwx| \leq k,$ and $uv^i wx^i y \in L$ f.a. $i \geq 0$.

However: $L \subseteq \{a\}^* : |z| = m \rightsquigarrow z = a^m$.

$\forall m \geq k : \exists n, \ell \geq 0 : m = n + \ell, 1 \leq \ell \leq k,$ and $a^n a^{i \cdot \ell} \in L$ f.a. $i \geq 0$.

Choose $q := k!$: Each ℓ_i divides q.
Choose $q' \geq q$ such that the following condition is met:

$\forall m \geq q : a^m \in L \rightsquigarrow \exists q \leq p \leq q' : m \equiv p \bmod q$ and $a^p \in L$.

Then $L := \{ x \in L \mid |x| < q \} \cup \{ a^r a^{iq} \mid q \leq r \leq q', a^r \in L, i \in \mathbb{N} \}$, which shows that L is regular. □

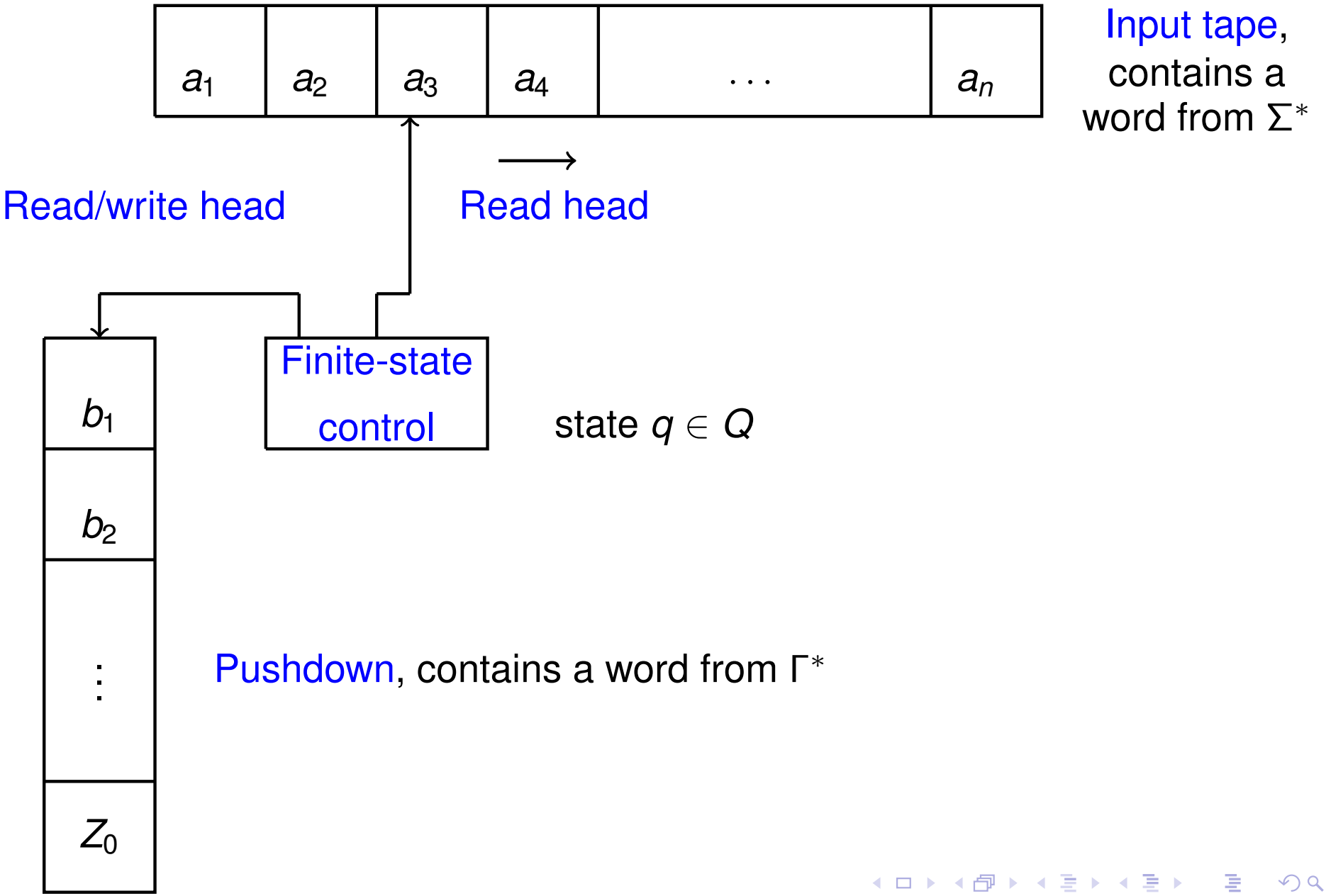
3.4. Pushdown Automata

A pushdown automaton (PDA) is an ε-NFA that has an additional external memory in the form of a pushdown.

A PDA M is defined through a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

- Q is a finite set of (internal) states,
- Σ is a finite input alphabet,
- Γ is a finite pushdown alphabet,
- $q_0 \in Q$ is the initial state,
- $Z_0 \in \Gamma$ is the bottom marker of the pushdown,
- $F \subseteq Q$ is the set of accepting states, and
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$ is the transition relation.
  For each $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $b \in \Gamma$,
  $\delta(q, a, b)$ is a finite subset of $Q \times \Gamma^*$.

A PDA can be picturered as follows:



A configuration of M is a triple $(q, \gamma, w) \in Q \times \Gamma^* \times \Sigma^*$, where

q is the current state,

γ is the current content of the pushdown, and

w is the remaining input.

Here the last symbol of γ is the topmost symbol on the pushdown.

The PDA M induces a computation relation $\vdash_M^*$ on the set $\mathrm{CONF} := Q \times \Gamma^* \times \Sigma^*$ of configurations, which is the reflexive and transitive closure of the following single-step relation $\vdash_M$:

$$\begin{array}{lll} (q, \gamma Z, aw) \vdash_M (p, \gamma\beta, w), & \text{if } (p, \beta) \in \delta(q, a, Z) \text{ , and} \\ (q, \gamma Z, w) \vdash_M (p, \gamma\beta, w), & \text{if } (p, \beta) \in \delta(q, \varepsilon, Z). \end{array}$$

Example:

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $F = \{q_3\}$, $\Sigma = \{a, b\}$, $\Gamma = \{A, B, Z_0\}$, and let δ be given by the following table:

$$\begin{aligned}
\delta(q_0, a, Z_0) &= \{(q_1, Z_0A)\},\\
\delta(q_0, b, Z_0) &= \{(q_1, Z_0B)\},\\
\delta(q_0, \varepsilon, Z_0) &= \{(q_3, \varepsilon)\},\\
\delta(q_1, a, Z) &= \{(q_1, ZA)\},\\
\delta(q_1, b, Z) &= \{(q_1, ZB)\},\\
\delta(q_1, \varepsilon, Z) &= \{(q_2, Z)\},\\
\delta(q_2, a, A) &= \{(q_2, \varepsilon)\},\\
\delta(q_2, b, B) &= \{(q_2, \varepsilon)\},\\
\delta(q_2, \varepsilon, Z_0) &= \{(q_3, \varepsilon)\}.
\end{aligned}$$

On input *abba*, the PDA M can execute the following computation:

$$\begin{aligned}
(q_0, Z_0, abba) &\vdash (q_1, Z_0A, bba) \vdash (q_1, Z_0AB, ba) \vdash (q_2, Z_0AB, ba)\\
&\vdash (q_2, Z_0A, a) \vdash (q_2, Z_0, \varepsilon) \vdash (q_3, \varepsilon, \varepsilon).
\end{aligned}$$

Depending on its mode of operation, a PDA M accepts one of two possible languages:

$L(M)$ denotes the language

$$L(M) := \{ w \in \Sigma^* \mid (q_0, Z_0, w) \vdash_M^* (p, \gamma, \varepsilon) \text{ for some } p \in F \text{ and } \gamma \in \Gamma^* \},$$

that is, $L(M)$ is the language that M accepts with final states,

and $N(M)$ denotes the language

$$N(M) := \{ w \in \Sigma^* \mid (q_0, Z_0, w) \vdash_M^* (p, \varepsilon, \varepsilon) \text{ for some } p \in Q \},$$

that is, $N(M)$ is the language that M accepts with empty pushdown.

Example (cont.):

$L(M) = N(M) = \{ uu^R \mid u \in \{a, b\}^* \}$.

Theorem 3.17

For each PDA M_1, there exists a PDA M_2 such that $L(M_1) = N(M_2)$.

Proof.

Let $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, and let $L = L(M_1)$, that is,

$$L = \{ w \in \Sigma^* \mid (q_0, Z_0, w) \vdash_{M_1}^* (p, \gamma, \varepsilon) \text{ for some } p \in F \text{ and } \gamma \in \Gamma^* \}.$$

The PDA M_2 will simulate the PDA M_1 step by step.

Essentially M_2 must solve the following two problems:

- M_2 must be able to recognize when M_1 empties its pushdown without being in a final state, as in this situation, M_2 must not accept.

- M_2 must empty its pushdown when M_1 accepts.

Proof of Theorem 3.17 (cont.)

Let $M_2 = (Q \cup \{q_\ell, q_0'\}, \Sigma, \Gamma \cup \{X_0\}, \delta', q_0', X_0, \emptyset)$ be defined as follows:

(1) $\delta'(q_0', \varepsilon, X_0) = \{(q_0, X_0 Z_0)\}$,

(2) $\delta'(q, a, Z) \supseteq \delta(q, a, Z)$ for all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $Z \in \Gamma$,

(3) $\delta'(q, \varepsilon, Z) \ni (q_\ell, Z)$ for all $q \in F$ and $Z \in \Gamma \cup \{X_0\}$,

(4) $\delta'(q_\ell, \varepsilon, Z) = \{(q_\ell, \varepsilon)\}$ for all $Z \in \Gamma \cup \{X_0\}$.

By (1) M_2 enters the initial configuration of M_1, with the symbol X_0 below the bottom marker of M_1.

By (2) M_2 simulates the computation of M_1 step by step.

If and when M_1 reaches a final state, then M_2 empties its pushdown using (3) and (4). It follows that $L(M_1) \subseteq N(M_2)$.

If M_1 empties its pushdown without being in a final state, then M_2 gets stuck in the corresponding configuration (q, X_0, aw).

It can now be shown that $N(M_2) = L(M_1)$. □

Theorem 3.18

For each PDA M_2, there exists a PDA M_1 such that $L(M_1) = N(M_2)$.

Theorem 3.19

From a given context-free grammar G, one can effectively construct a PDA M such that $N(M) = L(G)$.

Proof.

Let $G = (N, \Sigma, S, P)$ be a context-free grammar. By Theorem 3.10 we can assume that G is in Greibach Normal form. To simplify the discussion we assume that $\varepsilon \notin L(G)$.

We define a PDA $M = (\{q\}, \Sigma, N, \delta, q, S, \emptyset)$ by taking $\delta(q, a, A) := \{ (q, \gamma^R) \mid (A \to a\gamma) \in P \}$ for all $a \in \Sigma$ and $A \in N$.

Proof of Theorem 3.19 (cont.)

Claim:

$\forall x \in \Sigma^* \, \forall \alpha \in N^* : S \to_{\mathrm{lm}}^* x\alpha$ iff $(q, S, x) \vdash_M^* (q, \alpha^R, \varepsilon)$.

Proof.

By induction on i, we will prove the following:

$(*)$ If $(q, S, x) \vdash_M^i (q, \alpha^R, \varepsilon)$, then $S \to_{\mathrm{lm}}^i x\alpha$.

If $i = 0$, then $\alpha = S$ and $x = \varepsilon$.

Assume that $(*)$ has already been shown for some integer $i - 1$, and assume that $(q, S, x) \vdash_M^{i-1} (q, \beta^R, x') \vdash_M (q, \alpha^R, \varepsilon)$.

As M has no ε-transitions, $x = ya$ and $x' = a$ for some $y \in \Sigma^{i-1}$ and $a \in \Sigma$, that is, the computation above has the form

$$(q, S, ya) \vdash_M^{i-1} (q, \beta^R, a) \vdash_M (q, \alpha^R, \varepsilon).$$

Proof of Theorem 3.19 (cont.)

Proof of Claim (cont.)

Thus, on input y, M can execute the following computation:

$$(q, S, y) \vdash_M^{i-1} (q, \beta^R, \varepsilon).$$

By our I.H. this gives a derivation $S \to_{\mathrm{lm}}^{i-1} y\beta$.

As $(q, \beta^R, a) \vdash_M (q, \alpha^R, \varepsilon)$, $\beta^R = \gamma^R A$ for some $A \in N$ and $\alpha^R = \gamma^R \eta^R$ for a production $(A \to a\eta) \in P$.

Hence, we obtain the following derivation in G:

$$S \to_{\mathrm{lm}}^{i-1} y\beta = yA\gamma \to_{\mathrm{lm}} ya\eta\gamma = x\alpha.$$

This proves the implication from right to left.

Proof of Theorem 3.19 (cont.)

Proof of Claim (cont.)

Now we establish the converse implication:

$(**)$ If $S \to_{\mathrm{lm}}^{i} x\alpha$, then $(q, S, x) \vdash_M^i (q, \alpha^R, \varepsilon)$,

again proceeding by induction on i.

If $i = 0$, then $x = \varepsilon$ and $\alpha = S$.
As $(q, S, \varepsilon) \vdash_M^0 (q, S, \varepsilon)$, the claim holds in this situation.

Assume that the implication has already been proved for some integer $i - 1$, and let $S \to_{\mathrm{lm}}^{i-1} yA\gamma \to_{\mathrm{lm}} ya\eta\gamma$ be a derivation in G, where $(A \to a\eta) \in P$.

Then $x = ya$ and $\alpha = \eta\gamma$. By our I.H. $(q, S, y) \vdash_M^{i-1} (q, \gamma^R A, \varepsilon)$, which yields $(q, S, x) = (q, S, ya) \vdash_M^{i-1} (q, \gamma^R A, a)$.

Proof of Theorem 3.19 (cont.)

Proof of Claim (cont.)

Because of the production $(q, \eta^R) \in \delta(q, a, A)$, we obtain the following computation of M:

$$(q, S, x) = (q, S, ya) \vdash_M^{i-1} (q, \gamma^R A, a) \vdash_M (q, \gamma^R \eta^R, \varepsilon) = (q, \alpha^R, \varepsilon).$$

□

For all $x \in \Sigma^+$, we have the following sequence of equivalent statements:

$$\begin{aligned} x \in L(G) \quad &\text{iff} \quad S \to_{\mathrm{lm}}^i x \text{ for some } i \geq 1 \\ &\text{iff} \quad (q, S, x) \vdash_M^i (q, \varepsilon, \varepsilon) \text{ for some } i \geq 1 \\ &\text{iff} \quad x \in N(M). \end{aligned}$$

Thus, $N(M) = L(G)$ follows. □

Example:

$(S \to aBC), (B \to bC), (C \to cDD), (D \to a) \in P$:

$S \to aBC \to abCC \to abcDDC \to^2 abcaaC \to abcaacDD \to^2 abcaacaa$.

Computation of M:

$$\begin{aligned}(q, S, abcaacaa) &\vdash (q, CB, bcaacaa) \vdash (q, CC, caacaa)\\ &\vdash (q, CDD, aacaa) \vdash (q, CD, acaa)\\ &\vdash (q, C, caa) \vdash (q, DD, aa)\\ &\vdash (q, D, a) \vdash (q, \varepsilon, \varepsilon).\end{aligned}$$

Thus, each context-free language is accepted by a PDA that has only a single state and that uses no ε-transitions.

Theorem 3.20

The language $N(M)$ is context-free for each PDA M.

Proof.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$ be a PDA, and let $L = N(M)$.

W.l.o.g. we can assume the following:

For all $(q', B_1 B_2 \cdots B_k) \in \delta(q, a, A)$, we have $k \leq 2$.

Otherwise, we replace $(q', B_1 B_2 \cdots B_k) \in \delta(q, a, A)$ $(k > 2)$ by the following transitions:

$$\begin{array}{lcl} (q_1, B_1 B_2) & \in & \delta(q, a, A), \\ \{(q_2, B_2 B_3)\} & = & \delta(q_1, \varepsilon, B_2), \\ & \vdots & \\ \{(q', B_{k-1} B_k)\} & = & \delta(q_{k-2}, \varepsilon, B_{k-1}), \end{array}$$

where $q_1, q_2, \ldots, q_{k-1}$ are new states.

Proof of Theorem 3.20 (cont.)

Idea: A grammar G that simulates the computations of M through leftmost derivations.

Let $G := (N, \Sigma, P, S)$, where

$$
\begin{aligned}
N &:= \{S\} \cup [Q \times \Gamma \times Q], \\
P &:= \{\, S \to (q_0, \#, q) \mid q \in Q \,\} \\
&\cup \{\, [q, A, q'] \to a \mid (q', \varepsilon) \in \delta(q, a, A) \,\} \\
&\cup \{\, [q, A, q'] \to a[q_1, B, q'] \mid (q_1, B) \in \delta(q, a, A), q' \in Q \,\} \\
&\cup \{\, [q, A, q'] \to a[q_1, C, q_2][q_2, B, q'] \mid \\
&\qquad\qquad (q_1, BC) \in \delta(q, a, A), q', q_2 \in Q \,\}.
\end{aligned}
$$

Claim:

$\forall p, q \in Q \forall A \in \Gamma \forall x \in \Sigma^* : [q, A, p] \to_{\mathrm{lm}}^* x$ iff $(q, A, x) \vdash_M^* (p, \varepsilon, \varepsilon)$.

Proof of Theorem 3.20 (cont.)

Proof of Claim.

First we prove by induction on i that
$(q, A, x) \vdash_M^i (p, \varepsilon, \varepsilon)$ implies that $[q, A, p] \to_{\mathrm{lm}}^i x$.

For $i = 1$,

if $(q, A, x) \vdash_M (p, \varepsilon, \varepsilon)$, then $(p, \varepsilon) \in \delta(q, x, A)$ and $x \in \Sigma \cup \{\varepsilon\}$.
Hence, G contains the production $[q, A, p] \to x$, and so, $[q, A, p] \to_{\mathrm{lm}} x$.

Now assume that the above implication has been established for $i - 1$, let $x = ay$, and let

$$(q, A, ay) \vdash_M (q_1, B_2B_1, y) \vdash_M^{i-1} (p, \varepsilon, \varepsilon),$$

that is, $(q_1, B_2B_1) \in \delta(q, a, A)$.

Proof of Theorem 3.20 (cont.)

Proof of Claim (cont.)

As M can only read and replace the topmost symbol on the pushdown, it follows that $y = y_1 y_2$, and that the above computation can be factored as follows:

$$(q_1, B_1, y_1) \vdash_M^{k_1} (q_2, \varepsilon, \varepsilon) \text{ and } (q_2, B_2, y_2) \vdash_M^{k_2} (p, \varepsilon, \varepsilon),$$

where $q_2 \in Q$, and $k_1 + k_2 = i - 1$.

From the I.H. we obtain the leftmost derivations

$$[q_1, B_1, q_2] \to_{\mathrm{lm}}^{k_1} y_1 \text{ and } [q_2, B_2, p] \to_{\mathrm{lm}}^{k_2} y_2.$$

Proof of Theorem 3.20 (cont.)

Proof of Claim (cont.)

Because of the transition in step 1, we have

$$([q, A, p] \to a[q_1, B_1, q_2][q_2, B_2, p]) \in P,$$

and therewith we obtain the following leftmost derivation in G:

$$\begin{aligned} [q, A, p] &\to_{\mathrm{lm}} a[q_1, B_1, q_2][q_2, B_2, p] \\ &\to_{\mathrm{lm}}^{k_1} ay_1[q_2, B_2, p] \\ &\to_{\mathrm{lm}}^{k_2} ay_1y_2 = ay = x, \end{aligned}$$

that is, we have $[q, A, p] \to_{\mathrm{lm}}^{i} x$.

Proof of Theorem 3.20 (cont.)

Proof of Claim (cont.)

Now we prove the converse implication, again by induction on i.

For $i = 1$, $[q, A, p] \to x$ implies that $x \in \Sigma \cup \{\varepsilon\}$ and $(p, \varepsilon) \in \delta(q, x, A)$, that is, $(q, A, x) \vdash_M (p, \varepsilon, \varepsilon)$.

Now assume that the implication has been proved for $i - 1$, and assume that

$$[q, A, p] \to a[q_1, B_1, q_2][q_2, B_2, p] \to_{\mathrm{lm}}^{i-1} x \in \Sigma^*.$$

Then $x = ax_1x_2$, where $[q_1, B_1, q_2] \to_{\mathrm{lm}}^{k_1} x_1$, $[q_2, B_2, p] \to_{\mathrm{lm}}^{k_2} x_2$, and $k_1 + k_2 = i - 1$.

From the I.H. we obtain $(q_1, B_1, x_1) \vdash_M^{k_1} (q_2, \varepsilon, \varepsilon)$ and $(q_2, B_2, x_2) \vdash_M^{k_2} (p, \varepsilon, \varepsilon)$.

Proof of Theorem 3.20 (cont.)

Proof of Claim (cont.)

Hence, we have

$$(q_1, B_2B_1, x_1) \vdash_M^{k_1} (q_2, B_2, \varepsilon).$$

As $(q_1, B_2B_1) \in \delta(q, a, A)$, this yields the following computation:

$$\begin{aligned}(q, A, x) &= (q, A, ax_1x_2) \vdash_M (q_1, B_2B_1, x_1x_2) \\ &\quad \vdash_M^{k_1} (q_2, B_2, x_2) \vdash_M^{k_2} (p, \varepsilon, \varepsilon),\end{aligned}$$

that is, we have $(q, A, x) \vdash_M^{i} (p, \varepsilon, \varepsilon)$. □

Proof of Theorem 3.20 (cont.)

For all $x \in \Sigma^*$ we have the following equivalent statements:

$$\begin{aligned} x \in L(G) \quad &\text{iff} \quad S \to_{\mathrm{lm}}^* x \\ &\text{iff} \quad S \to [q_0, \#, q] \to_{\mathrm{lm}}^i x \text{ for some } q \in Q \text{ and } i \geq 1 \\ &\text{iff} \quad (q_0, \#, x) \vdash_M^i (q, \varepsilon, \varepsilon). \end{aligned}$$

It follows that $L(G) = N(M)$. □

Corollary 3.21

For each language $L \subseteq \Sigma^*$, the following statements are equivalent:

(1) $L \in \mathrm{CFL}(\Sigma)$, *that is, L is generated by a context-free grammar.*

(2) *L is generated by a context-free grammar in CNF.*

(3) *L is generated by a context-free grammar in Greibach NF.*

(4) *There exists a PDA M such that $L = N(M)$.*

(5) *There exists a PDA M such that $L = L(M)$.*

3.5. Closure Properties

Theorem 3.22

The class of context-free languages CFL is closed under the operations of union, product, Kleene star, and morphism.

Proof.

(a) Let $G_1 = (N_1, \Sigma, P_1, S_1)$, $G_2 = (N_2, \Sigma, P_2, S_2)$, $N_1 \cap N_2 = \emptyset$.
For $G_3 := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 | S_2\}, S)$,
we have $L(G_3) = L(G_1) \cup L(G_2)$.

(b) For $G_4 := (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$,
we have $L(G_4) = L(G_1) \cdot L(G_2)$.

(c) W.l.o.g.: S_1 does not occur on the right-hand side of P_1.
For $G_5 := (N_1 \cup \{S\}, \Sigma, P_1 \cup \{S \to \varepsilon | S_1 | SS_1\} \setminus \{S_1 \to \varepsilon\}, S)$,
we have $L(G_5) = (L(G_1))^*$.

Proof of Theorem 3.22 (cont.)

(d) Let $h : \Sigma^* \to \Gamma^*$ be a morphism.
For $G_6 := (N_1, \Gamma, \{ A \to h(r) \mid (A \to r) \in P_1 \}, S_1)$, where h is extended to a morphism $h : (N_1 \cup \Sigma)^* \to (N_1 \cup \Gamma)^*$ by taking $h(A) = A$ for all $A \in N_1$, we have $L(G_6) = h(L(G_1))$. □

Theorem 3.23

The class CFL is not closed under intersection nor under complement.

Proof.

$L_1 := \{ a^i b^j c^j \mid i, j > 0 \} \in$ CFL and $L_2 := \{ a^i b^i c^j \mid i, j > 0 \} \in$ CFL,
but $L_1 \cap L_2 = \{ a^i b^i c^i \mid i > 0 \} \notin$ CFL, that is,
CFL is not closed under intersection.
$L_1 \cap L_2 = \overline{(\overline{L}_1 \cup \overline{L}_2)}$, that is,
CFL is not closed under complement, either. □

Theorem 3.24

The class CFL is closed under intersection with regular languages, that is, if $L \in$ CFL and $R \in$ REG, then $L \cap R \in$ CFL.

Proof.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, and
let $A = (P, \Sigma, \eta, p_0, G)$ be a DFA.

We define a PDA $M' := (Q \times P, \Sigma, \Gamma, \delta', (q_0, p_0), Z_0, F \times G)$ by taking

$$\begin{aligned} \delta'((q,p),a,A) &= \{((q',p'),\alpha) \mid (q',\alpha) \in \delta(q,a,A), p' = \eta(p,a)\}, \\ \delta'((q,p),\varepsilon,A) &= \{((q',p),\alpha) \mid (q',\alpha) \in \delta(q,\varepsilon,A)\}, \end{aligned}$$

where $q, q' \in Q$, $p, p' \in P$, $a \in \Sigma$, $A \in \Gamma$, and $\alpha \in \Gamma^*$.
Then it is easily seen that $L(M') = L(M) \cap L(A)$. □

Theorem 3.25

Let $L \subseteq \Sigma^*$ be a context-free language, and let $w \in \Sigma^*$. Then the left quotient $w \setminus L := \{ u \in \Sigma^* \mid wu \in L \}$ is a context-free language.

Proof.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA without ε-transitions, and let $w \in \Sigma^*$. Actually, we only consider the special case that $w = b \in \Sigma$.

We define a PDA $M' := (Q \cup \{p_0\}, \Sigma, \Gamma, \delta', p_0, Z_0, F)$ by taking

$$\begin{aligned} \delta'(p_0, \varepsilon, Z_0) &= \delta(q_0, b, Z_0) \text{ and} \\ \delta'(q, a, A) &= \delta(q, a, A) \text{ for all } q \in Q, a \in \Sigma, \text{ and } A \in \Gamma. \end{aligned}$$

Then $N(M') = b \setminus N(M)$. □

Remark:

We saw in Section 3 that the language

$$L := \{ a^i b^j c^k d^\ell \mid i, j, k, \ell \geq 0, \text{ and } i > 0 \text{ implies } j = k = \ell \}$$

satisfies the Pumping Lemma for context-free languages.

Claim:

L is not context-free.

Proof.

Assume that L is context-free. Then also the language

$$L' := L \cap a \cdot b^* \cdot c^* \cdot d^* = \{ ab^n c^n d^n \mid n \geq 0 \}$$

is context-free, and so is the language $a \setminus L' = \{ b^n c^n d^n \mid n \geq 0 \}$, which, however, is not context-free by the Pumping Lemma, a contradiction!

Thus, it follows that L is not context-free. □

Theorem 3.26

The class CFL is closed under inverse morphisms, that is, if $L \in \mathrm{CFL}(\Delta)$, *and if* $h : \Sigma^* \to \Delta^*$ *is a morphism, then* $h^{-1}(L) = \{ w \in \Sigma^* \mid h(w) \in L \} \in \mathrm{CFL}(\Sigma)$.

Proof.

Let $h : \Sigma^* \to \Delta^*$ be a morphism, and
let $M = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$ be a PDA s.t. $L(M) = L$.

We construct a PDA $M' = (Q', \Sigma, \Gamma, \delta', q_0', Z_0, F')$ for $L' := h^{-1}(L) = \{ w \in \Sigma^* \mid h(w) \in L \}$:

- $Q' := \{ [q, x] \mid q \in Q$, and x is a suffix of $h(a), a \in \Sigma \}$,
- $q_0' := [q_0, \varepsilon]$,
- $F' := \{ [q, \varepsilon] \mid q \in F \}$, and
- δ' is defined by:

Proof of Theorem 3.26 (cont.)

$$
\begin{aligned}
(1)\quad \delta'([q,ax],\varepsilon,A) &= \{\,([p,x],\gamma) \mid (p,\gamma) \in \delta(q,a,A)\,\} \\
&\quad \cup \{\,([p,ax],\gamma) \mid (p,\gamma) \in \delta(q,\varepsilon,A)\,\}, \\
(2)\quad \delta'([q,\varepsilon],a,A) &= \{([q,h(a)],A)\} \text{ for all } a \in \Sigma.
\end{aligned}
$$

The second component of the states of M' is used as a "buffer" for storing the word $h(a)$ for a letter $a \in \Sigma$ read. This word is then processed through a simulation of a computation of M.

It follows that $h^{-1}(L) \subseteq L(M')$.

Concersely, let $w = a_1 a_2 \cdots a_n \in L(M')$.

A transition reading a letter $a \in \Sigma$ can only be applied if and when the buffer is empty, that is, each accepting computation of M' on input w can be written as follows:

Proof of Theorem 3.26 (cont.)

$$
\begin{aligned}
&([q_0, \varepsilon], Z_0, a_1 a_2 \cdots a_n) \\
\vdash^*_{M'} \quad &([p_1, \varepsilon], \alpha_1, a_1 a_2 \cdots a_n) && \text{(simulating } \varepsilon\text{-transitions of M)} \\
\vdash_{M'} \quad &([p_1, h(a_1)], \alpha_1, a_2 \cdots a_n) && (h(a_1) \text{ is “read”}) \\
\vdash^*_{M'} \quad &([p_2, \varepsilon], \alpha_2, a_2 \ldots a_n) && (\text{for } (p_1, \alpha_1, h(a_1)) \vdash^*_M (p_2, \alpha_2, \varepsilon)) \\
&\vdots \\
\vdash_{M'} \quad &([p_n, h(a_n)], \alpha_n, \varepsilon) && (h(a_n) \text{ is “read”}) \\
\vdash^*_{M'} \quad &([p_{n+1}, \varepsilon], \alpha_{n+1}, \varepsilon) && (\text{for } (p_n, \alpha_n, h(a_n)) \vdash^*_M (p_{n+1}, \alpha_{n+1}, \varepsilon)).
\end{aligned}
$$

It follows that

$$(q_0, Z_0, h(a_1 a_2 \cdots a_n)) = (q_0, Z_0, h(a_1)h(a_2) \cdots h(a_n)) \vdash^*_M (p_{n+1}, \alpha_{n+1}, \varepsilon).$$

As M' accepts, $p_{n+1} \in F$, which implies that M accepts, too.

Thus, $L(M') = h^{-1}(L)$, which yields $h^{-1}(L) \in \text{CFL}(\Sigma)$. □

In analogy to Corollary 2.29 we have the following closure property.

Corollary 3.27

The class CFL *is closed under finite transductions, that is, if* $T \subseteq \Sigma^* \times \Delta^*$ *is a finite transduction, and if* $L \in \mathrm{CFL}(\Sigma)$*, then* $T(L) \in \mathrm{CFL}(\Delta)$*.*

3.6. Decision Problems

Theorem 3.28

The following Emptiness Problem is decidable in polynomial time:

INSTANCE: *A context-free grammar G.*
QUESTION: *Is $L(G) \neq \emptyset$?*

Proof.

Let $G = (N, T, S, P)$ be a context-free grammar. In polynomial time we can determine the set V_{term} of usefull nonterminals of G (Lemma 3.3), where

$$V_{\text{term}} = \{ A \in N \mid L(G, A) \neq \emptyset \}.$$

As $L(G) \neq \emptyset$ iff $S \in V_{\text{term}}$, this yields the desired algorithm. □

Theorem 3.29

The following Finiteness Problem is decidable:

INSTANCE: *A context-free grammar G.*
QUESTION: *Is $L(G)$ a finite language?*

Proof.

First we transform the given grammar into a grammar $G_1 = (N, T, S, P)$ in CNF s.t. $L(G_1) = L(G) \cap T^+$ (Theorem 3.9).

In addition, we can assume that G_1 is proper and that it contains no ε-productions.

From G_1 we construct a directed graph (V, E) as follows:

- $V := N$, that is, there exists a node for each nonterminal, and
- $(A \to B) \in E$ iff there exists a nonterminal C such that P contains the production $A \to BC$ or $A \to CB$ (or both).

Proof of Theorem 3.29 (cont.)

Claim:

The language $L(G)$ is infinite iff the graph (V, E) contains a cycle.

Proof.

If $A_0, A_1, \ldots, A_m, A_0$ is a cycle in (V, E), then we have a derivation of the following form in G_1:

$$\begin{aligned} A_0 &\to_P \alpha_1 A_1 \beta_1 \to_P \alpha_1 \alpha_2 A_2 \beta_2 \beta_1 \to_P \cdots \to_P \alpha_1 \cdots \alpha_m A_m \beta_m \cdots \beta_1 \\ &\to_P \alpha_1 \cdots \alpha_m \alpha_{m+1} A_0 \beta_{m+1} \cdots \beta_1, \end{aligned}$$

where $\alpha_i, \beta_i \in N^*$ and $|\alpha_i| + |\beta_i| = 1$ for all $i = 1, 2, \ldots, m+1$.

As G_1 is proper, $S \to_P^* u_0 A_0 v_0$ for some $u_0, v_0 \in T^*$

and $\alpha_i \to_P^* u_i \in T^*$, $\beta_i \to_P^* v_i \in T^*$, $i = 1, 2, \ldots, m+1$.

As G_1 contains no ε-production, it follows that $|u_i v_i| \geq |\alpha_i| + |\beta_i| = 1$.

Proof of Theorem 3.29 (cont.)

Proof of Claim (cont.)

Thus, we obtain the following derivations in G_1:

$$\begin{aligned} S &\to_P^* u_0 A_0 v_0 \to_P^* u_0 u_1 \cdots u_{m+1} A_0 v_{m+1} \cdots v_1 v_0 \\ &\to_P^* u_0 (u_1 \cdots u_{m+1})^k A_0 (v_{m+1} \cdots v_1)^k v_0 \\ &\to_P^* u_0 (u_1 \cdots u_{m+1})^k x (v_{m+1} \cdots v_1)^k v_0 \in T^* \end{aligned}$$

for some $x \in T^*$ and all $k \geq 1$. As $|u_1 \cdots u_{m+1} v_{m+1} \cdots v_1| \geq m+1$, all these words differ from one another, that is, $L(G)$ is infinite.

If (V, E) does not contain any cycle, then each path in each syntax tree of each derivation from S to some word $v \in T^*$ has length at most $|N| + 1$. Thus, there are only finitely many syntax trees for G, and hence, $L(G)$ is finite. □

It is decidable in time $O(|N|^2)$ whether (V, E) contains a cycle. □

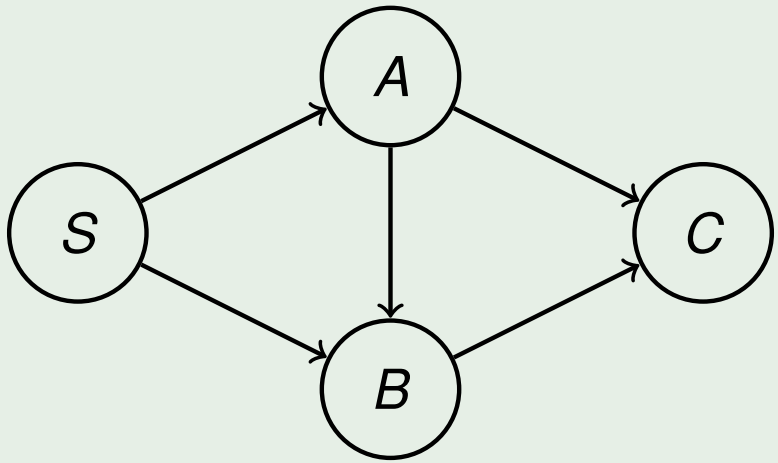
Example:

Let $G = (\{S, A, B, C, \}, \{a, b\}, S, P)$, where

$$P = \{S \to AB, A \to BC, A \to a, B \to CC, B \to b, C \to a\}.$$

G is a proper context-free grammar in CNF without ε-productions.

The directed graph(V, E) for G looks as follows:



(V, E) does not contain any cycle, that is, $L(G)$ is finite.

In fact, it is easily seen that $L(G) = \{ab, a^3, a^3b, ba^3, bab, a^5\}$. □

Theorem 3.30

The following Membership Problem is decidable in polynomial time:

INSTANCE: *A context-free grammar G in CNF, and a word $w \in T^*$.*

QUESTION: *Is $w \in L(G)$?*

Proof.

Let $G = (N, T, S, P)$ be a context-free grammar in CNF, let $N = \{A_1, A_2, \ldots, A_m\}$, and let $S = A_1$.

We can assume that $(S \to \varepsilon)$ is the only ε-production (if any), and that A_1 does not occur on the righthand side of any production.

Hence, $\varepsilon \in L(G)$ iff $(A_1 \to \varepsilon) \in P$.

Proof of Theorem 3.30 (cont.)

Let $w = x_1 x_2 \cdots x_n$, where $x_1, x_2, \ldots, x_n \in T$.

For all $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, n+1-i\}$,

let $V_{i,j} \subseteq N$ be defined as follows:

$$V_{i,j} := \{ A \in N \mid A \to_P^* x_i x_{i+1} \cdots x_{i+j-1} \}.$$

Then $w \in L(G)$ iff $S = A_1 \in V_{1,n}$.

We now compute all the sets $V_{i,j}$ through the method of dynamic programming.

This algorithm goes back to J. Cocke, T. Kasami, and D. Younger (1967).

Proof of Theorem 3.30 (cont.)

procedure CKY;
begin
  for $i := 1$ **to** n **do** $V_{i,1} := \{ A \mid (A \to x_i) \in P \}$;
  for $j := 2$ **to** n **do**
    for $i := 1$ **to** $n - j + 1$ **do**
    begin
      $V_{i,j} := \emptyset$;
$(*)$ **for** $k := 1$ **to** $j - 1$ **do**
        $V_{i,j} := V_{i,j} \cup \{ A \mid (A \to BC) \in P, B \in V_{i,k}, C \in V_{i+k,j-k} \}$
    end
end.

As $A \to_P^* x_i \cdots x_{i+j-1}$ iff $\exists (A \to BC) \in P$ s.t. $B \to_P^* x_i \cdots x_{i+k-1}$ and $C \to_P^* x_{i+k} \cdots x_{i+j-1}$, the set $V_{i,j}$ is computed correctly in $(*)$.

Obviously, this algorithm runs in polynomial time. □

Example:

Let $G = (\{S, A, B, C, D, E, F\}, \{a, b, c\}, P, S)$,
where $P = \{ S \to AB, A \to CD, A \to CF, B \to c, B \to EB,$
$C \to a, D \to b, E \to c, F \to AD \}$

Let $x = aaabbbcc$.

$x =$		a	a	a	b	b	b	c	c
j	1	C	C	C	D	D	D	B,E	B,E
$\downarrow$	2			A				B	
	3			F					
	4		A						
	5		F						
	6	A							
	7	S							
	8	S							

As $S \in V_{1,8}$, it follows that $x \in L(G)$. □

Theorem 3.31

The following problems cannot be solved algorithmically:

INSTANCE: Two context-free grammars G_1, G_2.

(1.) *QUESTION: Is $L(G_1) \cap L(G_2) = \emptyset$?*

(2.) *QUESTION: Is $|L(G_1) \cap L(G_2)| = \infty$?*

(3.) *QUESTION: Is $L(G_1) \cap L(G_2)$ context-free ?*

(4.) *QUESTION: Is $L(G_1) \subseteq L(G_2)$?*

(5.) *QUESTION: Is $L(G_1) = L(G_2)$?*

(6.) *QUESTION: Is G_1 unambiguous?*

(7.) *QUESTION: Is $L(G_1)^c$ context-free?*

(8.) *QUESTION: Is $L(G_1)$ regular?*

(9.) *QUESTION: Is $L(G_1)$ deterministic context-free?*

Proof: Later!

3.7. Deterministic Context-Free Languages

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic, that is, M is a DPDA, if in each configuration there is at most one applicable transition.

This is equivalent to the following two conditions, where $q \in Q$ and $z \in \Gamma$:

(1) For all $a \in \Sigma \cup \{\varepsilon\}$, $|\delta(q, a, z)| \leq 1$.

(2) If $\delta(q, \varepsilon, z) \neq \emptyset$, then $\delta(q, a, z) = \emptyset$ for all $a \in \Sigma$.

A language L is deterministic context-free, if there exists a DPDA M such that $L = L(M)$.

DCFL denotes the class of deterministic context-free languages.

Remark:

For a DPDA M, if $L' = N(M)$, that is, L' is accepted by empty pushdown, then L' is prefix-free: for all $u \in L'$, $u \cdot \Sigma^+ \cap L' = \emptyset$.

Hence, $a^+ \neq N(M)$ for each DPDA M.

Example:

$L := \{ a^m b^m a^n b^n \mid m, n \geq 0 \} \in$ DCFL,

but $L \neq N(M)$ for each DPDA M.

A DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is in normal form, if one of the following statements holds for each $\delta(q, a, z) = (p, \gamma)$:

(i) $\gamma = \varepsilon$, that is, the symbol z is popped from the pushdown,

(ii) $\gamma = z$, that is, the pushdown is not changed, or

(iii) $\gamma = zz'$ for some $z' \in \Gamma$, that is, the symbol z' is pushed onto the pushdown.

Theorem 3.32

If $L \in$ DCFL(Σ), then there exists a DPDA M in normal form such that $L = L(M)$.

Let M be a DPDA, and let $u \in L(M)$.
Then M reads input u completely and accepts.

For $v \in \Sigma^* \setminus L(M)$, M will in general not read input v completely.
In fact, one of the following cases can occur before M has read v completely:

- M reaches a configuration to which no transition applies,
- M empties its pushdown completely,
- M enters an infinite computation consisting entirely of ε-transitions.

Lemma 3.33

For each DPDA M, there exists an equivalent DPDA M' that always reads its input completely.

Proof of Lemma 3.33.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA in normal form.

- We introduce a new pushdown symbol X_0 and a new initial state q_0' together with the transition $\delta(q_0', \varepsilon, Z_0) = (q_0, X_0 Z_0)$.

- We add a new state d s.t. $\delta(d, a, z) = (d, z)$ for all $a \in \Sigma$ and $z \in \Gamma$. If $\delta(q, a, z) \cup \delta(q, \varepsilon, z) = \emptyset$ for some $q \in Q$, $a \in \Sigma$, and $z \in \Gamma$, then we take $\delta(q, a, z) = (d, z)$.

The resulting DPDA M_1 accepts the same language as M.

If M_1 does not read an input completely, this means that, starting in some state q, M_1 executes an infinite sequence of ε-transitions without removing the topmost symbol z from the pushdown.

- In this situation we take $\delta(q, \varepsilon, z) = (d, z)$, provided that no final state is reached through this sequence of ε-steps.
If, however, a final state is reached, then we take $\delta(q, \varepsilon, z) := (e, z)$ for a new final state e and $\delta(e, \varepsilon, z) := (d, z)$.

The DPDA M' obtained in this way has the desired property. □

Remark:

The construction in the proof of Lemma 3.33 is effective.

Theorem 3.34

The language class DCFL *is closed under complementation, that is, for each* $L \in \mathrm{DCFL}(\Sigma)$, $L^c := (\Sigma^* \setminus L) \in \mathrm{DCFL}(\Sigma)$, *too.*

Proof.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA such that $L(M) = L$, and assume that M always reads its input completely.

Unfortunately, L^c does in general not coincide with the language accepted by the DPDA $M' = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, Q \setminus F)$, as M may have a computation of the following form:

$$(q_0, Z_0, w) \vdash_M^* (q, \alpha, \varepsilon) \vdash_M (q', \beta, \varepsilon) \text{ for some } q \in F \text{ and } q' \notin F.$$

Proof of Theorem 3.34 (cont.)

We must ensure that the DPDA M cannot execute ε-transitions in a final state.

We define a DPDA $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_1, Z_0, F_1)$ as follows:

– $Q_1 := \{ [q,k] \mid q \in Q \text{ and } k \in \{1,2,3\} \}$,

– $q_1 := \left\{ \begin{array}{ll} [q_0,1], & \text{if } q_0 \in F, \\ [q_0,2], & \text{if } q_0 \notin F, \end{array} \right\}$, – $F_1 := \{ [q,3] \mid q \in Q \}$,

– and the transition function δ_1 is defined by:

(1) $\delta_1([q,k],\varepsilon,z) := ([p,k'],\gamma)$ if $\delta(q,\varepsilon,z) = (p,\gamma)$, $k \in \{1,2\}$ and $k' = \left\{ \begin{array}{ll} 1, & \text{if } k = 1 \text{ or } p \in F, \\ 2, & \text{otherwise,} \end{array} \right.$

(2) $\delta_1([q,2],\varepsilon,z) := ([q,3],z)$ if $\delta(q,a,z) = (p,\gamma)$,

(3) $\left. \begin{array}{l} \delta_1([q,1],a,z) := ([p,k],\gamma) \\ \delta_1([q,3],a,z) := ([p,k],\gamma) \end{array} \right\}$ if $\delta(q,a,z) = (p,\gamma)$ and $k = \left\{ \begin{array}{ll} 1, & \text{if } p \in F, \\ 2, & \text{otherwise.} \end{array} \right.$

Proof of Theorem 3.34 (cont.)

Claim:

$L(M_1) = \Sigma^* \setminus L$.

Proof.

Let $u = a_1 a_2 \cdots a_n \in L$. On input u, M executes a computation of the following form:

$$(q_0, Z_0, u) \vdash_M^* (p_1, \alpha, a_n) \vdash_M (p_2, \beta, \varepsilon) \vdash_M^* (p_3, \gamma, \varepsilon) \text{ where } p_3 \in F,$$

but p_2 and the subsequent states before p_3 are from $Q \setminus F$.

For M_1, we have the following computation:

$$([q_0, .], Z_0, u) \vdash_{M_1}^* ([p_1, .], \alpha, a_n) \vdash_{M_1}^{\leq 2} ([p_2, 2], \beta, \varepsilon) \vdash_{M_1}^* ([p_3, 1], \gamma, \varepsilon),$$

that is, further ε-transitions cannot lead to a final state of M_1.

Thus, $L(M_1) \subseteq \Sigma^* \setminus L$.

Proof of Theorem 3.34 (cont.)

Proof of Claim (cont.)

Conversely, for $u = a_1 a_2 \cdots a_n \notin L$, M has a computation of the form:

$$(q_0, Z_0, u) \vdash_M^* (p_1, \alpha, a_n) \vdash_M (p_2, \beta, \varepsilon) \vdash_M^* (p_3, \gamma, \varepsilon), \text{ where } p_2, \ldots, p_3 \notin F,$$

and in $(p_3, \gamma, \varepsilon)$, no further ε-transition is applicable.

As M reads each input completely, there exists a letter $a \in \Sigma$ s.t. $\delta(p_3, a, \mathrm{top}(\gamma))$ is defined. Hence, M_1 executes the following computation:

$$([q_0, .], Z_0, u) \vdash_{M_1}^* ([p_2, 2], \beta, \varepsilon) \vdash_{M_1}^* ([p_3, 2], \gamma, \varepsilon) \vdash_{M_1} ([p_3, 3], \gamma, \varepsilon)$$

where $[p_3, 3] \in F_1$, that is, $\Sigma^* \setminus L = L(M_1)$. □

This shows that $L^c \in \mathrm{DCFL}(\Sigma)$. □

Corollary 3.35

Each language $L \in$ DCFL is accepted by a DPDA that does not execute any ε-transitions in any final state.

Corollary 3.36

The class DCFL *is closed under intersection with regular languages.*

Proof.

Let $L_1 \in$ DCFL. The there exists a DPDA M that accepts L_1 and that reads each input completely.

For $L_2 \in$ REG, there exsists a DFA A such that $L(A) = L_2$.

From M and A, one can construct a DPDA for $L_1 \cap L_2$, that is, $L_1 \cap L_2 \in$ DCFL. □

Obviously, DCFL $\subseteq$ CFL. Now we will prove that this is a proper inclusion. Let

$$L_{\mathrm{Gl}} := \{\, w¢w^R¢w \mid w \in \{a,b\}^* \,\}$$

be the so-called Gladkij language [Gladkij 1964].

Using the Pumping Lemma 3.14 it is easily shown that L_{Gl} is not context-free.

Let $L_{\mathrm{Gl}}^c := (\{a,b,¢\}^* \setminus L_{\mathrm{Gl}})$.

Lemma 3.37

$L_{\mathrm{Gl}}^c \in \mathrm{CFL} \setminus \mathrm{DCFL}$.

Proof.

If $L_{\mathrm{Gl}}^c \in \mathrm{DCFL}$, then by Theorem 3.34, $L_{\mathrm{Gl}} \in \mathrm{DCFL}$.
As $L_{\mathrm{Gl}} \notin \mathrm{CFL}$, we see that $L_{\mathrm{Gl}}^c \notin \mathrm{DCFL}$.

It remains to prove that $L_{\mathrm{Gl}}^c \in \mathrm{CFL}$.

For a word $w \in \{a, b, ¢\}^*$, we have $w \in L_{\mathrm{Gl}}^c$ iff one of the following conditions is met:

(1) $|w|_¢ \neq 2$, or

(2) $w = w_1 ¢ w_2 ¢ w_3$, where $w_1^R \neq w_2$, or

(3) $w = w_1 ¢ w_2 ¢ w_3$, where $w_3^R \neq w_2$.

Let $H_1 := \{ u \in \{a, b, ¢\}^* \mid |w|_¢ \neq 2 \}$,

$H_2 := \{ w_1 ¢ w_2 ¢ w_3 \mid w_1, w_2, w_3 \in \{a, b\}^*, w_1^R \neq w_2 \}$ and

$H_3 := \{ w_1 ¢ w_2 ¢ w_3 \mid w_1, w_2, w_3 \in \{a, b\}^*, w_3^R \neq w_2 \}$.

Then $H_1 \in \mathrm{REG}$ and $H_2, H_3 \in \mathrm{CFL}$, implying that $L_{\mathrm{Gl}}^c = H_1 \cup H_2 \cup H_3 \in \mathrm{CFL}$. □

Corollary 3.38

DCFL $\subsetneq$ CFL.

Each DFA can be interpreted as a DPDA that does not use its pushdown. As $\{ a^n b^n \mid n \geq 1 \} \in \text{DCFL} \setminus \text{REG}$, we obtain the following proper inclusion.

Corollary 3.39

REG $\subsetneq$ DCFL.

Theorem 3.40 (Ogden's Lemma for DCFL (see Har78))

Let $L \in \mathrm{DCFL}(\Sigma)$. Then there exists a constant k that depends on L such that each word $z \in L$ containing $\delta(z) \geq k$ marked positions has a factorization $z = uvwxy$ that satisfies all of the following properties:

(1) $v \neq \varepsilon$,

(2) $uv^i wx^i y \in L$ *for all* $i \geq 0$,

(3) u, v *and* w *or* w, x *and* y *contain marked positions,*

(4) $\delta(vwx) \leq k$,

(5) *if* $y \neq \varepsilon$, *then the following equivalence holds for all* $m, n \geq 0$ *and all* $\alpha \in \Sigma^*$: $uv^{m+n}wx^n\alpha \in L$ *iff* $uv^m w\alpha \in L$.

Theorem 3.41

$L := \{ a^n b^n, a^n b^{2n} \mid n \geq 1 \} \in \mathrm{CFL} \setminus \mathrm{DCFL}$.

Proof.

It is easily seen that L is context-free. In fact, L is the union of the two deterministic context-free languages $L_1 := \{ a^n b^n \mid n \geq 1 \}$ and $L_2 := \{ a^n b^{2n} \mid n \geq 1 \}$.

We claim that L is not deterministic context-free.

Assume that L is deterministic context-free.

Let k be the corresponding constant from Thm. 3.40, and let $p := k!$.

Let $z := a^p b^p \in L$, where we mark all occurrences of b.

Then $z = a^p b^p$ has a factorization $z = a^p b^p = uvwxy$ that satisfies conditions (1) to (5) of the theorem.

As $z' = uwy \in L$, we have $v = a^i$ und $x = b^i$ for some $i \in \{1, 2, \ldots, k\}$.

Proof of Theorem 3.41 (cont.)

Because of (3) this implies that w, x and y contain marked positions, that is,

$$u = a^{n_1},\ v = a^i,\ w = a^{p-n_1-i}b^j,\ x = b^i \text{ and}$$
$$y = b^{p-i-j} \text{ for some } j \in \{1, 2, \ldots, k\}.$$

In fact, we have $i + j < k$. In particular, we have $y \neq \varepsilon$.

Now we choose $m = 1$ and $n = 1$, and we take $\alpha = b^{p-j+p+i}$. Then

$$uv^{n+m}wx^n\alpha = a^{n_1}a^{i\cdot 2}a^{p-n_1-i}b^jb^ib^{p-j+p+i} = a^{p+i}b^{p+i+p+i} \in L,$$

but $uv^mw\alpha = a^{n_1}a^ia^{p-n_1-i}b^jb^{p-j+p+i} = a^pb^{p+p+i} = a^pb^{2p+i} \notin L$, a contradiction!

Thus, it follows that $L \notin$ DCFL. □

Corollary 3.42

The language class DCFL *is not closed under union.*

By using the same technique it can be shown that

$$L' := \{ a^n b^n c, a^n b^{2n} d \mid n \geq 1 \}$$

is not in DCFL. On the other hand, the language

$$L'^R = \{ c b^n a^n, d b^{2n} a^n \mid n \geq 1 \}$$

belongs obviously to DCFL.

Corollary 3.43

The class DCFL *is not closed under reversal.*

Also $L'' := \{ ca^n b^n, da^n b^{2n} \mid n \geq 1 \}$ is in DCFL.
The morphism $\varphi : c \mapsto \varepsilon, d \mapsto \varepsilon, a \mapsto a, b \mapsto b$ maps L'' onto L.

Corollary 3.44

The class DCFL *is not closed under morphisms.*

Theorem 3.45

The class DCFL *is closed under inverse morphisms.*

Theorem 3.46

The class DCFL *is not closed under product and Kleene star.*

Theorem 3.47

The following problems are decidable:

(1) *INSTANCE:* $L \in \mathrm{DCFL}(\Sigma)$ *and* $R \in \mathrm{REG}(\Sigma)$.
QUESTION: *Is* $L = R$*?*

(2) *INSTANCE:* $L \in \mathrm{DCFL}(\Sigma)$ *and* $R \in \mathrm{REG}(\Sigma)$.
QUESTION: *Is* $R \subseteq L$*?*

(3) *INSTANCE:* $L \in \mathrm{DCFL}(\Sigma)$.
QUESTION: *Is* $L^c = \emptyset$*?*

(4) *INSTANCE:* $L \in \mathrm{DCFL}(\Sigma)$.
QUESTION: *Is* L *regular?*

(5) *INSTANCE:* $L_1, L_2 \in \mathrm{DCFL}(\Sigma)$.
QUESTION: *Is* $L_1 = L_2$*?*

Proof.

(1) Let $L_1 := (L \cap R^c) \cup (L^c \cap R)$.

Then $L = R$ iff $L_1 = \emptyset$.

From a DPDA for L and a DFA for R one can construct a PDA for L_1. By Theorem 3.28 it is decidable whether $L_1 = \emptyset$.

(2) $R \subseteq L$ iff $L^c \cap R = \emptyset$. In analogy to (1) this is decidable.

(3) This is obvious, as by Theorem 3.28 emptiness of context-free languages is decidable.

(4) See (Stearns 1967).

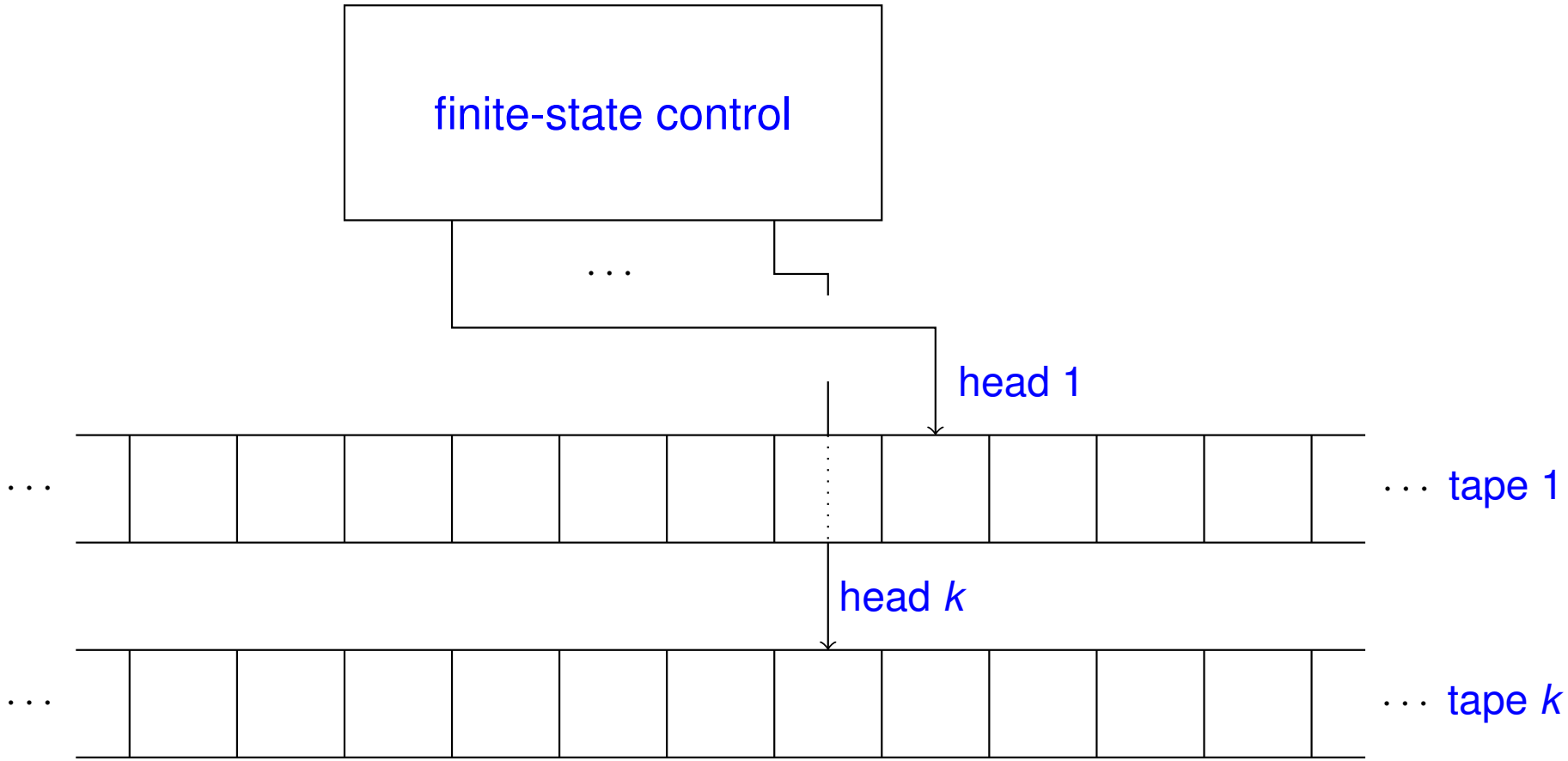
(5) See (Senizergues 1997). □

Chapter 4:

Turing Machines and Recursively Enumerable and Context-Sensitive Languages

4.1. Turing Machines

The Turing machine is a mathematical model of a computing device. Informally, it looks as follows:



Formally a Turing machine (TM) with $k \geq 1$ tapes is defined through a 7-tuple $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$, where

- Q is a finite set of (internal) states,
- Σ is a finite input alphabet,
- $\Gamma \supsetneq \Sigma$ is a finite tape alphabet,
- $\Box \in \Gamma \setminus \Sigma$ is the blank symbol,
- $\delta : Q \times \Gamma^k \rightsquigarrow Q \times \Gamma^k \times \{L, 0, R\}^k$ is the transition function,
- $q_0 \in Q$ is the initial state, and
- $q_1 \in Q$ is the halting state.

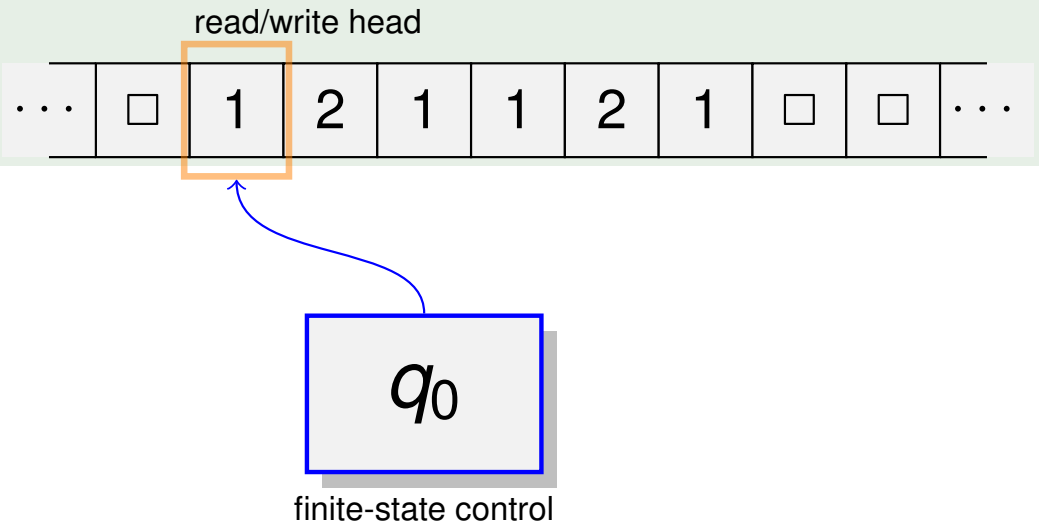
A TM works in discrete steps, where each step consists of 4 parts:

1. read current symbols under the heads,
2. write new symbols using the heads,
3. move the heads,
4. change the state.

Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

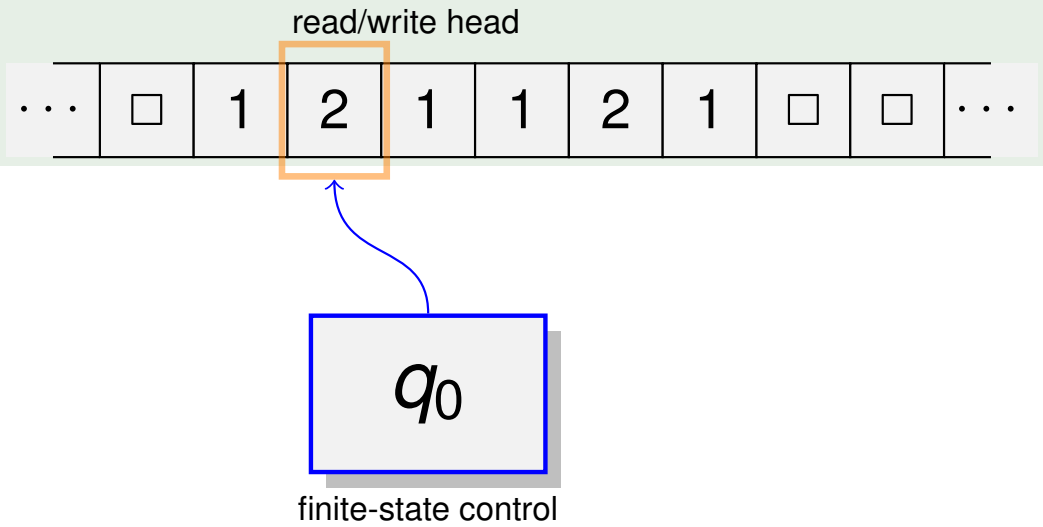
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

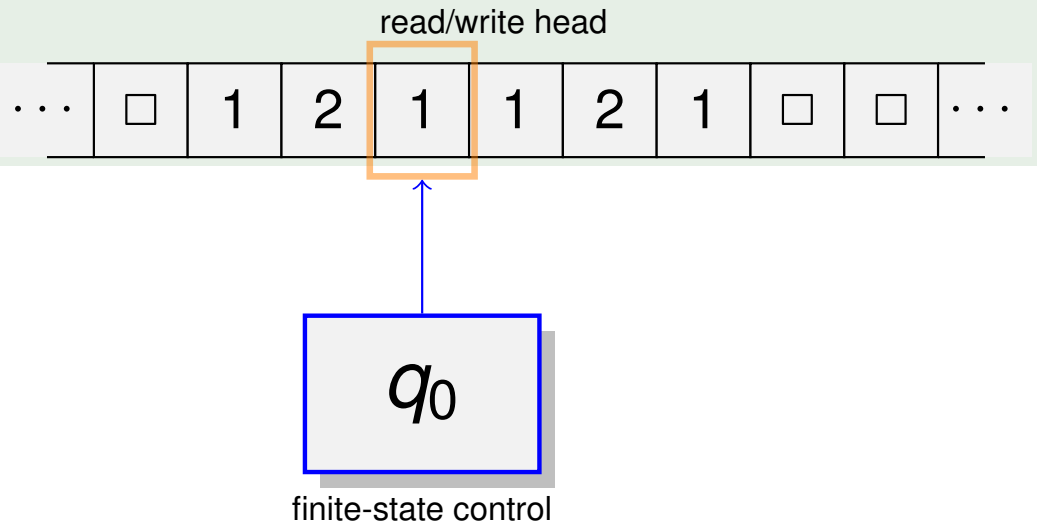
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

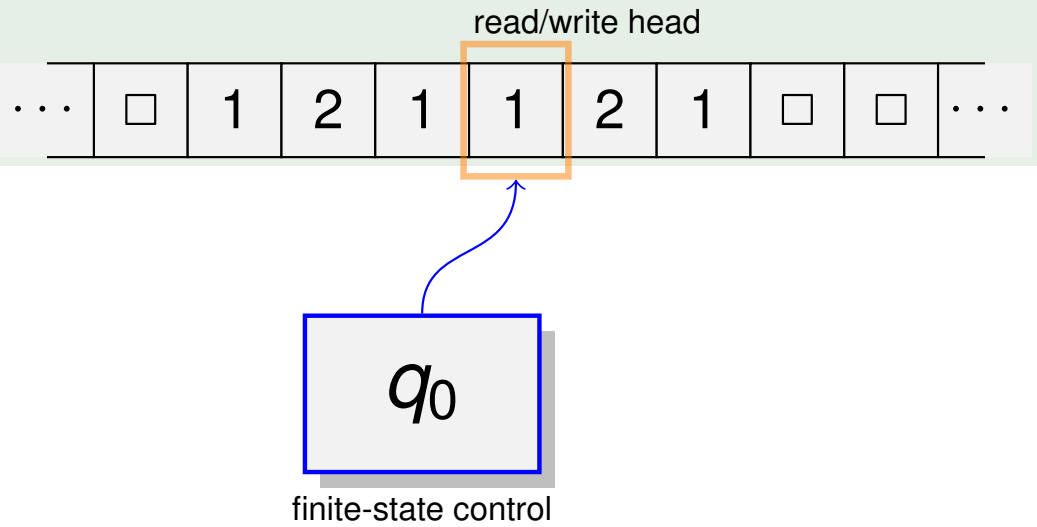
$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

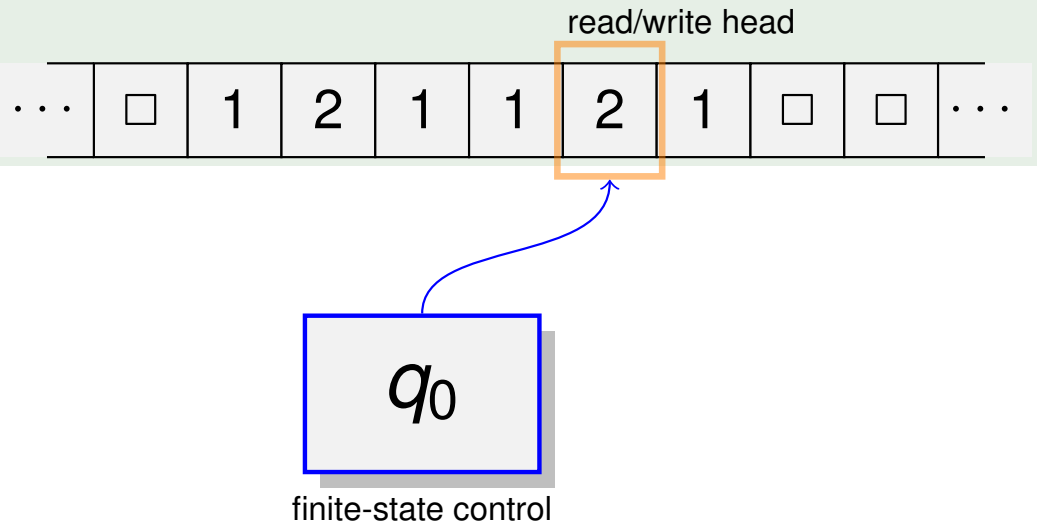
$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

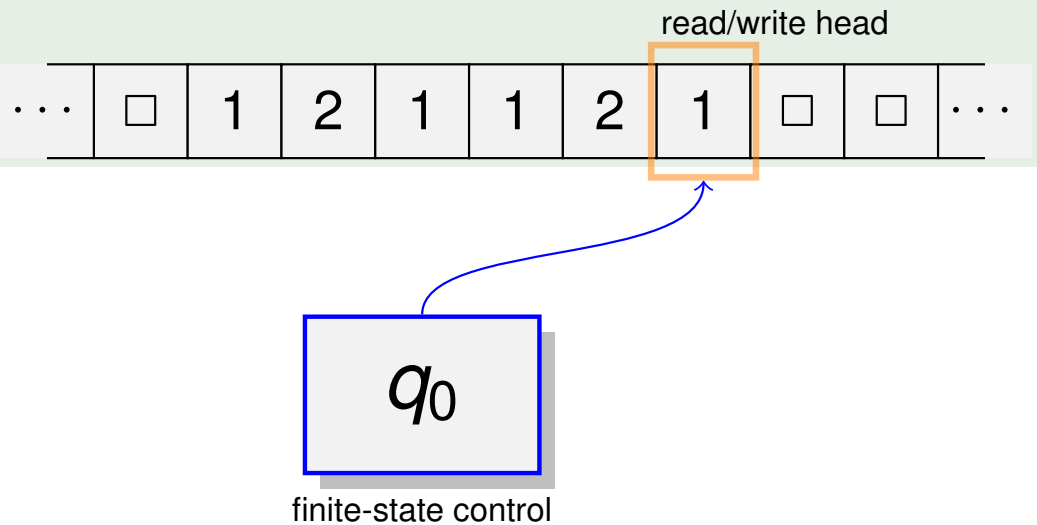
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

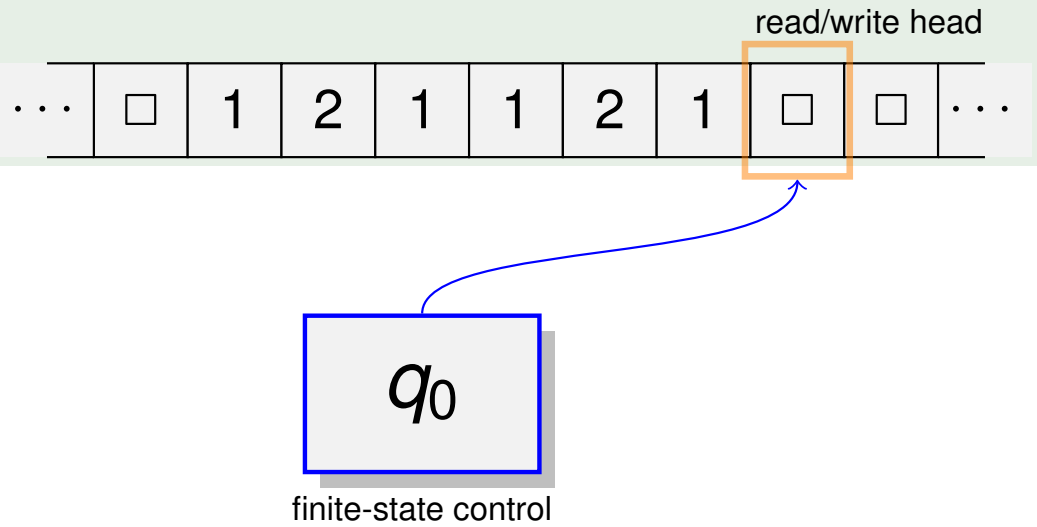
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

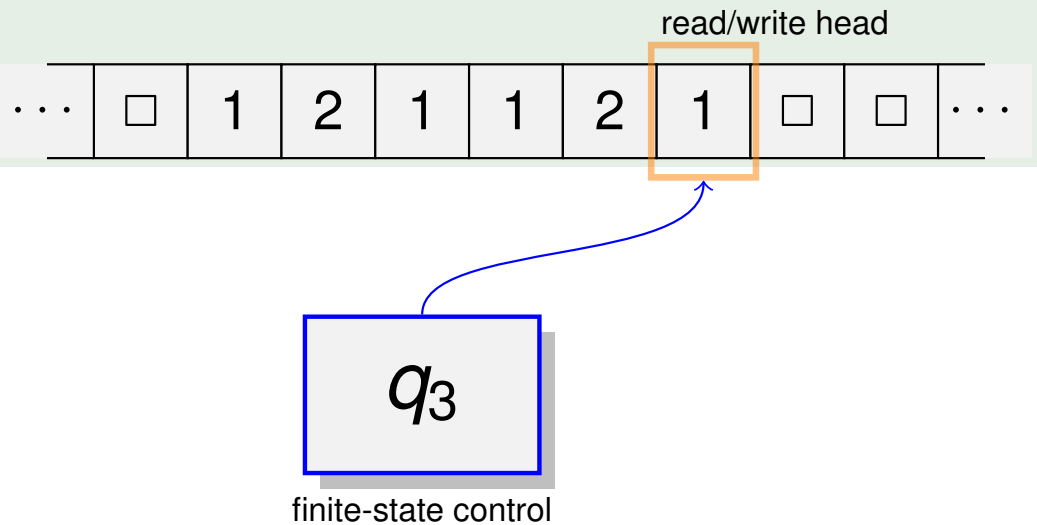
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

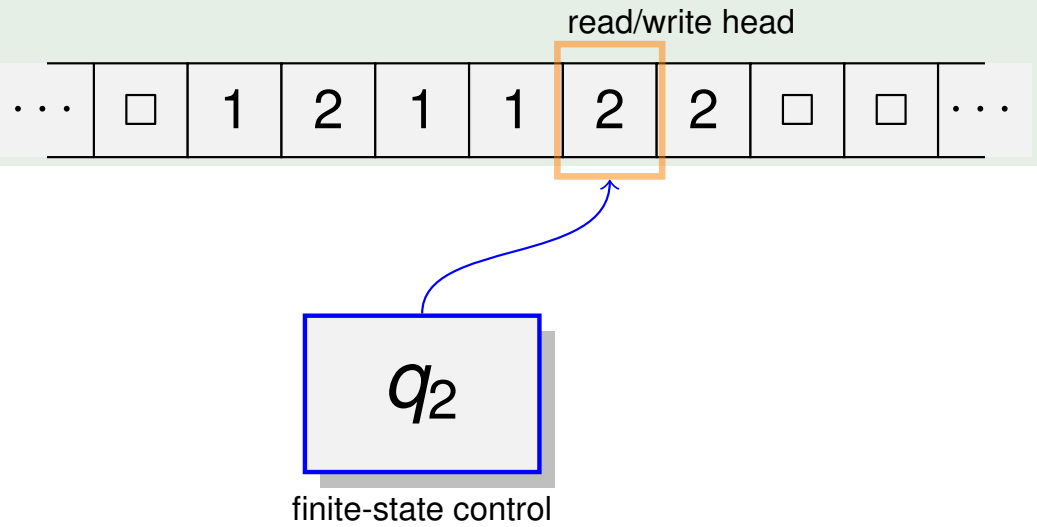
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

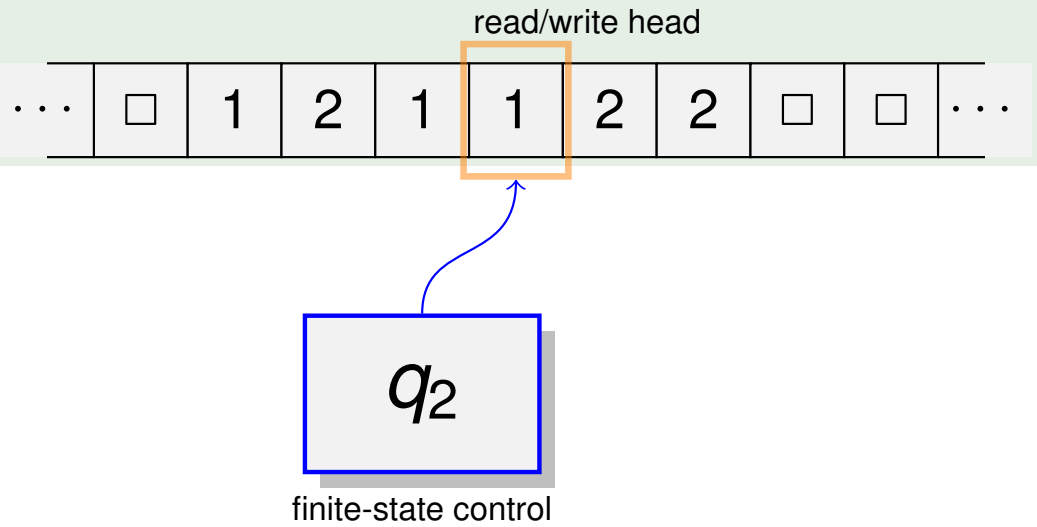
$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

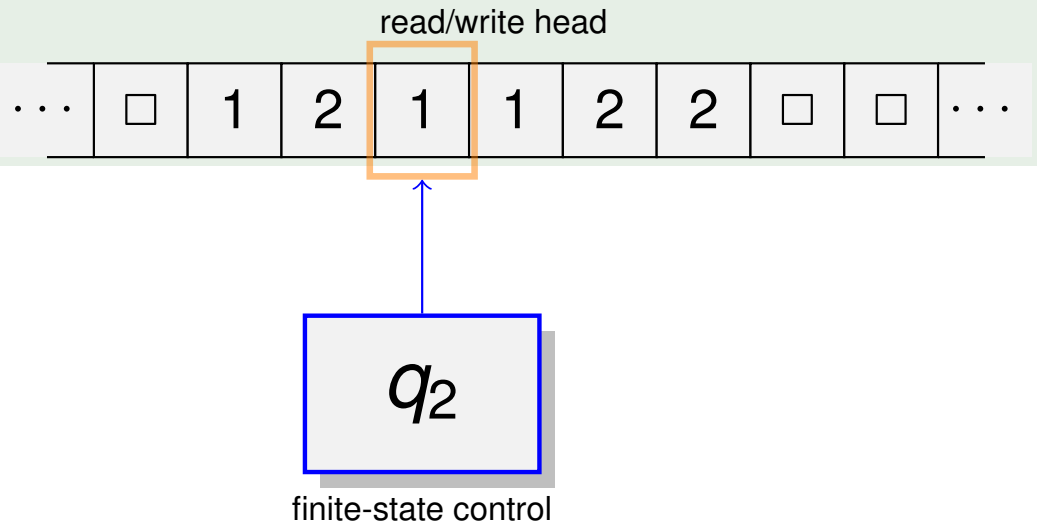
$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

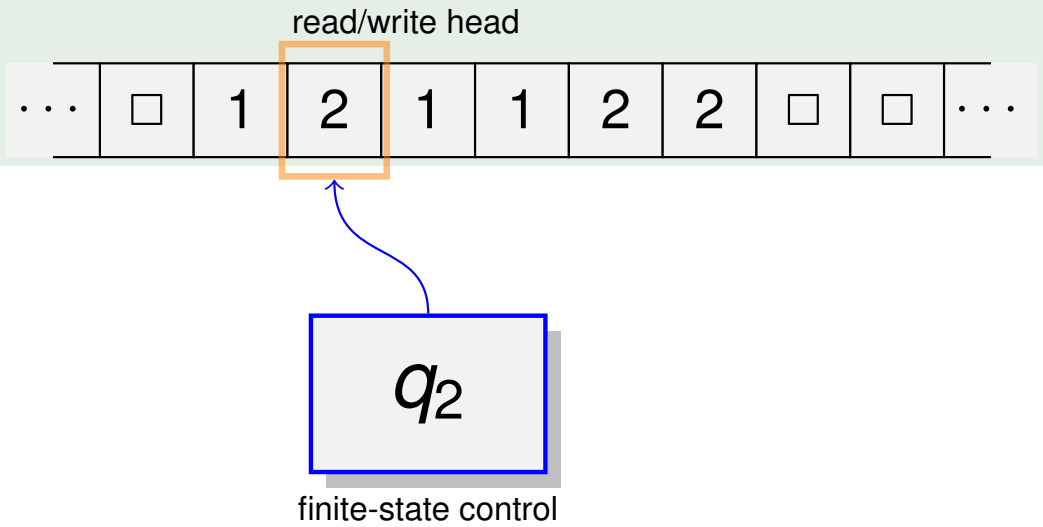
$$
\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

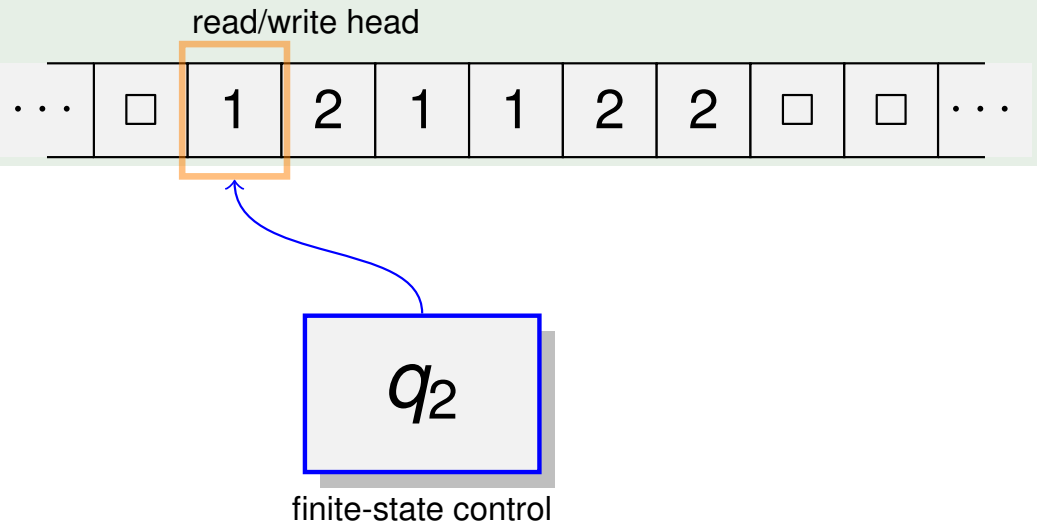
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

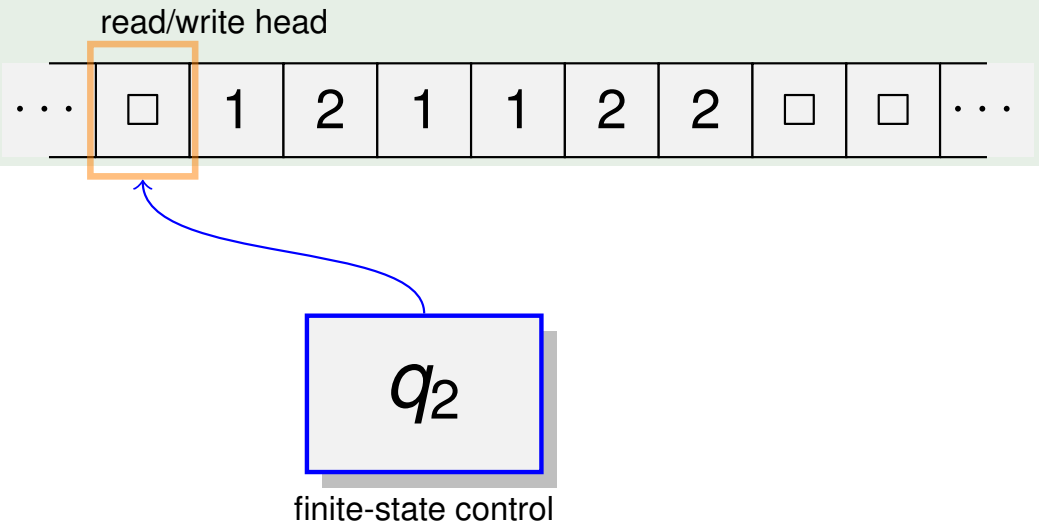
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

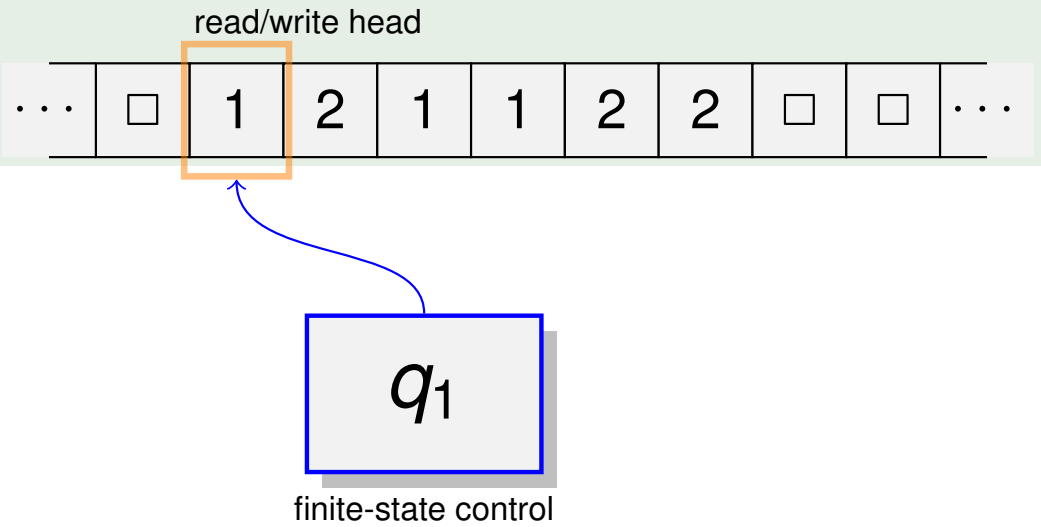
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM $M = (\{q_0, \ldots, q_3\}, \{1, 2\}, \{1, 2, \Box\}, \Box, \delta, q_0, q_1)$ for computing the function $+1$ on dyadic presentations:

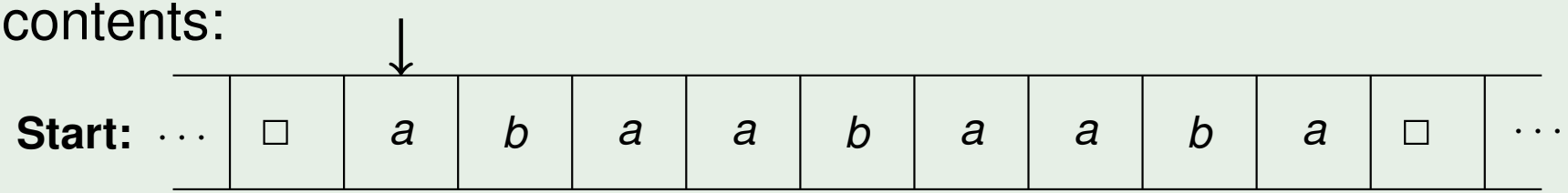
$$
\begin{array}{lcllcllcl}
q_0 1 & \to & q_0 1R & q_2 1 & \to & q_2 1L & q_3 1 & \to & q_2 2L \\
q_0 2 & \to & q_0 2R & q_2 2 & \to & q_2 2L & q_3 2 & \to & q_3 1L \\
q_0 \Box & \to & q_3 \Box L & q_2 \Box & \to & q_1 \Box R & q_3 \Box & \to & q_1 10
\end{array}
$$



Example:

A TM M for deciding membership in $\{ w \in \{a,b\}^* \mid w = w^R \}$.

On input $w = abaabaaba$, the TM M starts with the following tape contents:



Let $\Gamma = \{\Box, a, b\}$ and $Q = \{q_a, q_b, q'_a, q'_b, q_0, q_1, q_2, q_3\}$, where these states are to be used as follows:

q_a (q_b): Remember a (b) and move right.

q'_a (q'_b): Make one step to the left and test for a (b).

q_0 : Start and remember the first letter.

q_1 : Halt.

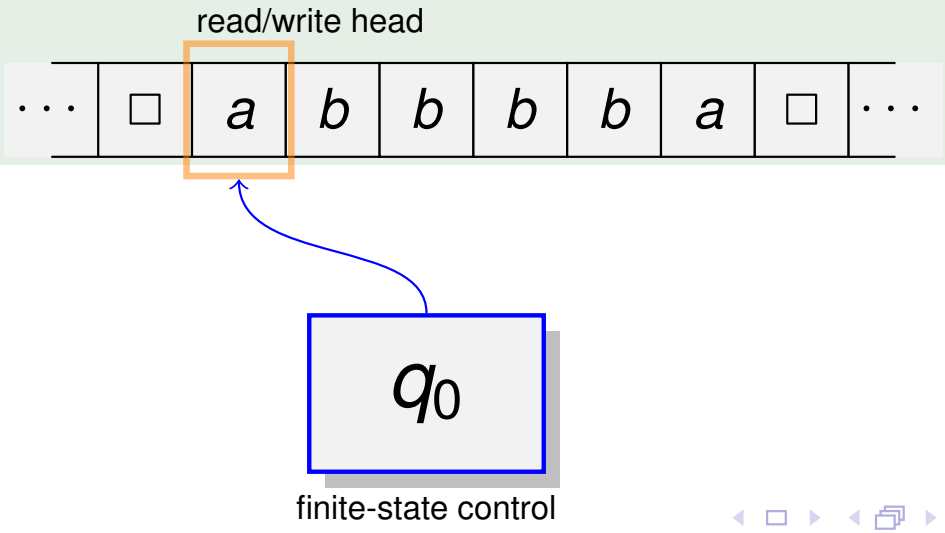
q_2 : Test was positive, return to the left.

q_3 : Test was negative, move left, erase the tape, and halt.

Example (cont.):

The transition function δ is defined as follows:

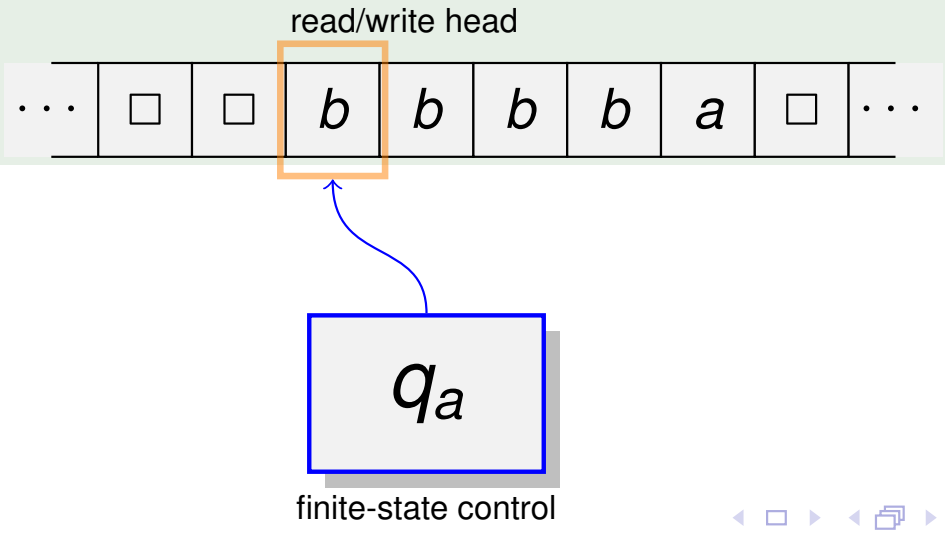
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

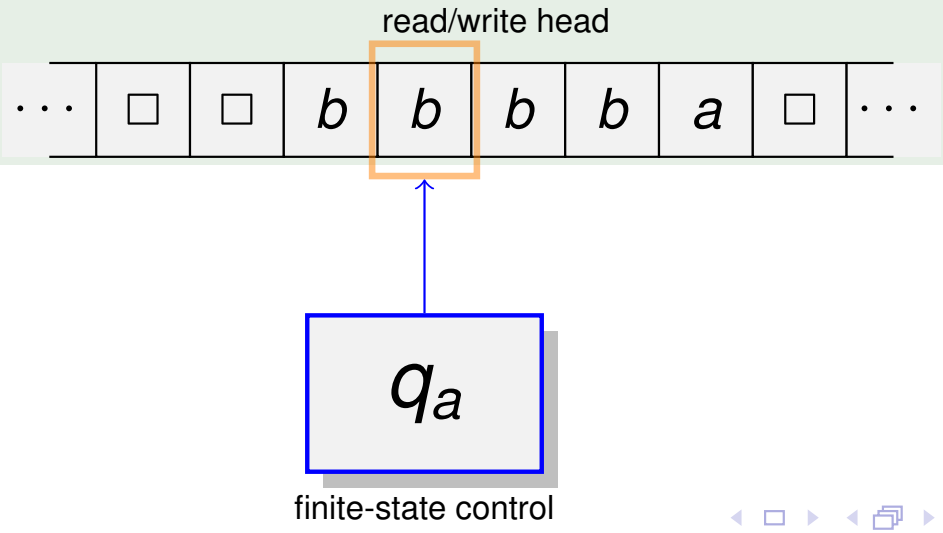
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

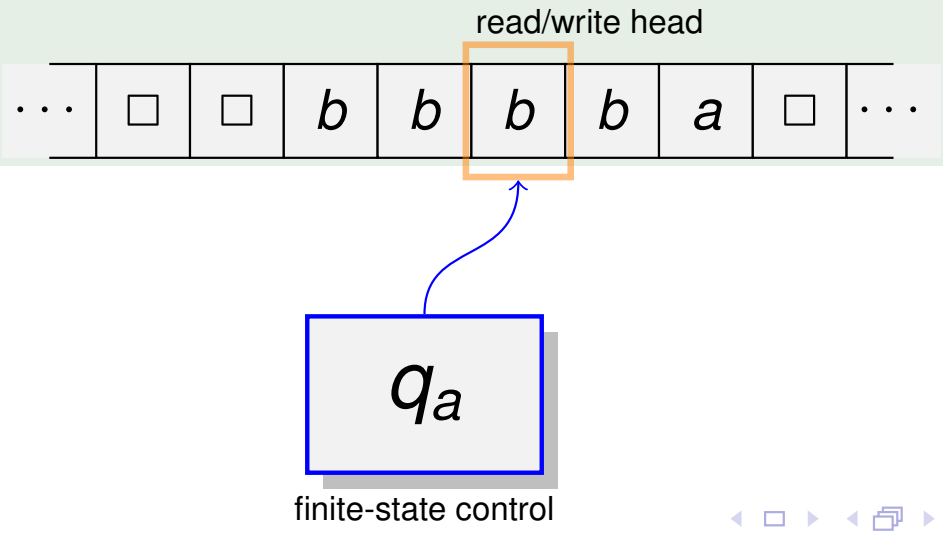
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

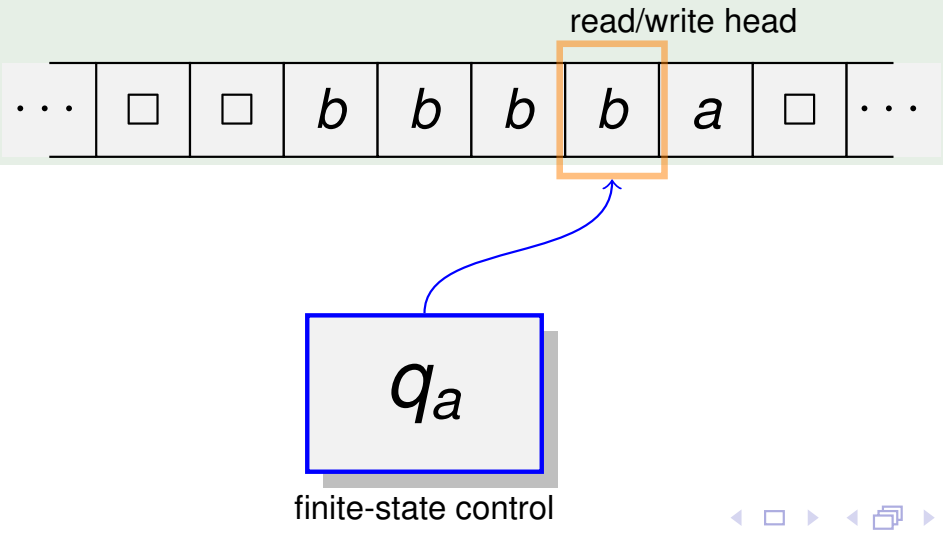
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

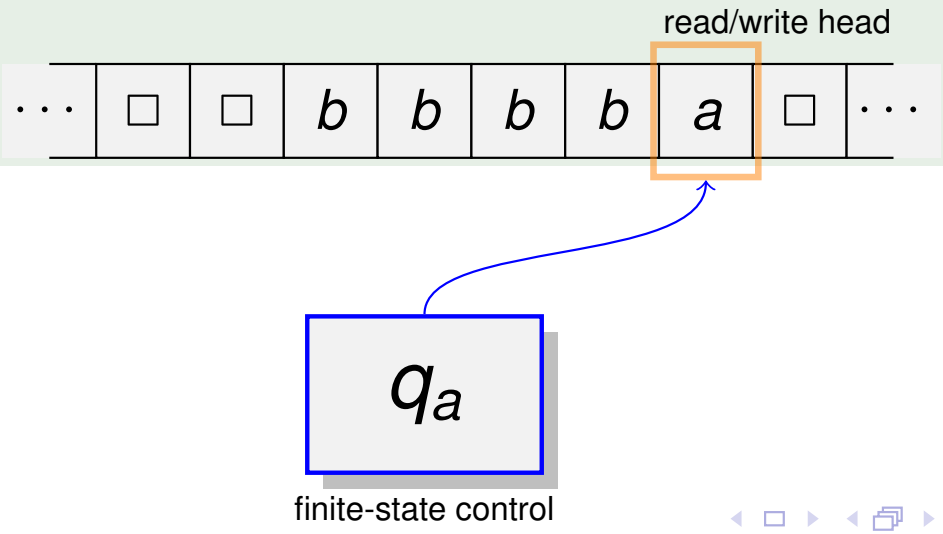
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

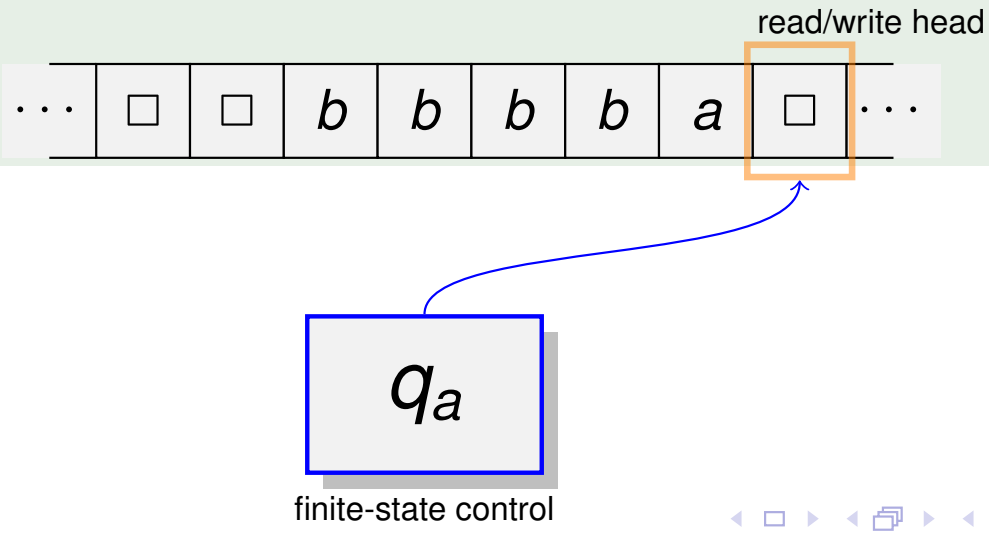
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

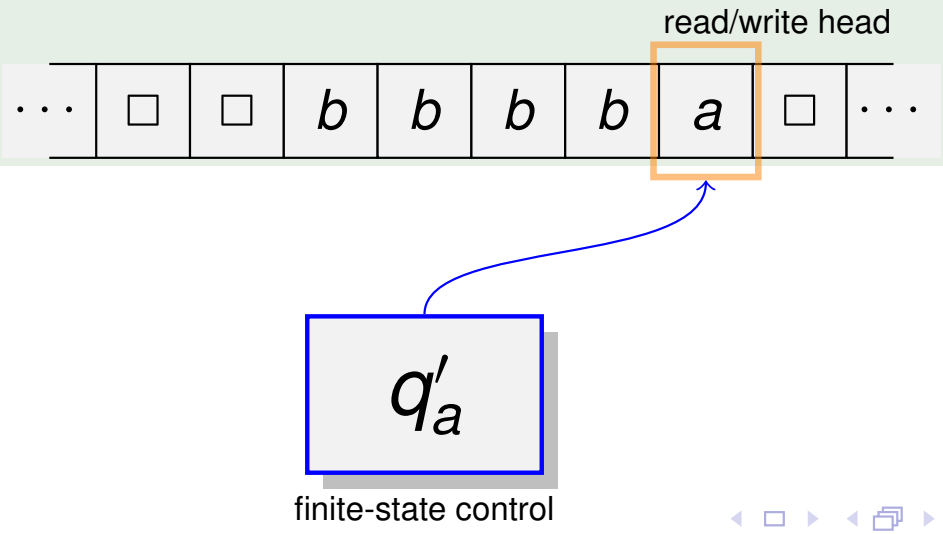
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a R \\
q_a b & \to & q_a b R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a 0 \\
q_b a & \to & q_b a R \\
q_b b & \to & q_b b R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a 0 \\
q_2 a & \to & q_2 a L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

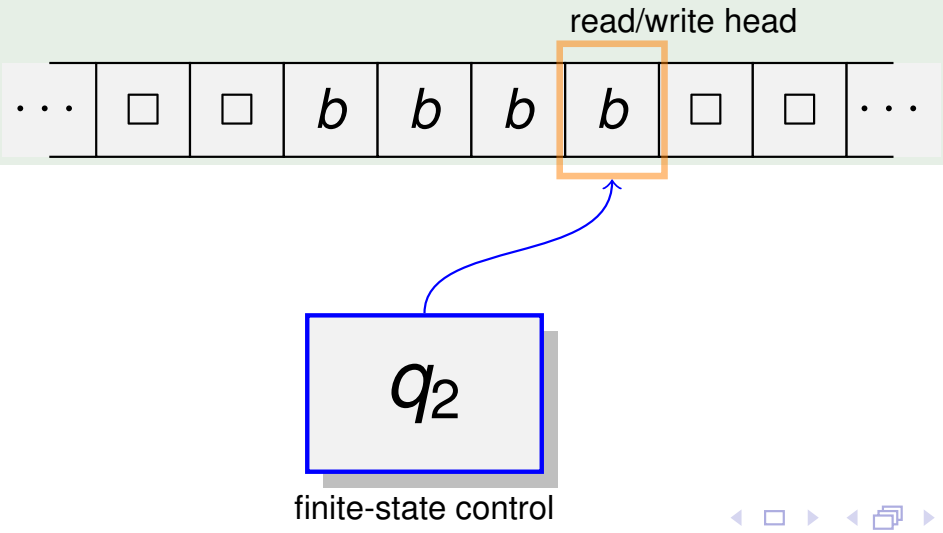
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

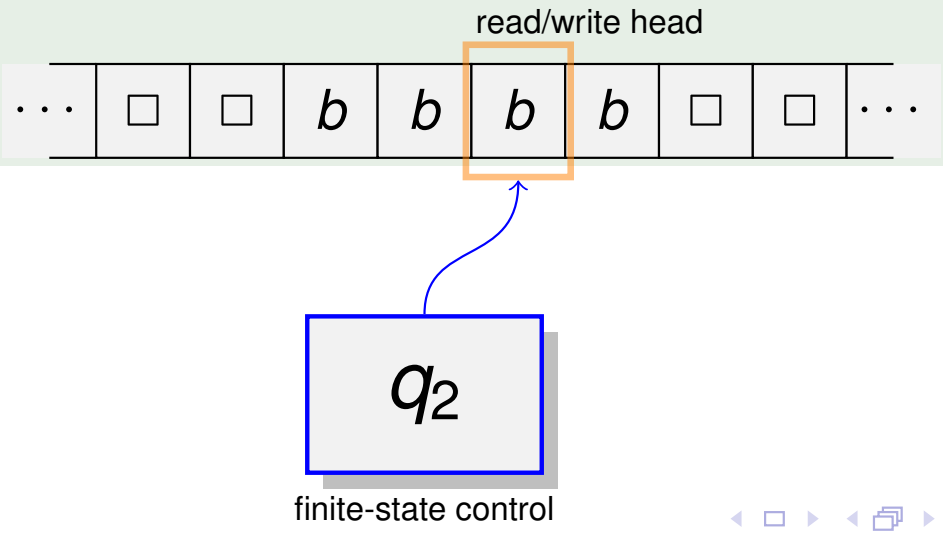
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

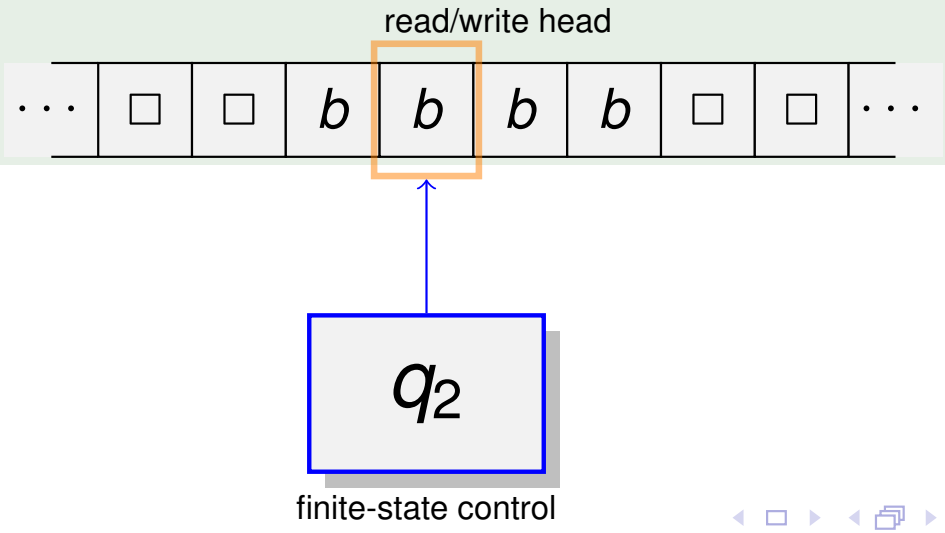
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

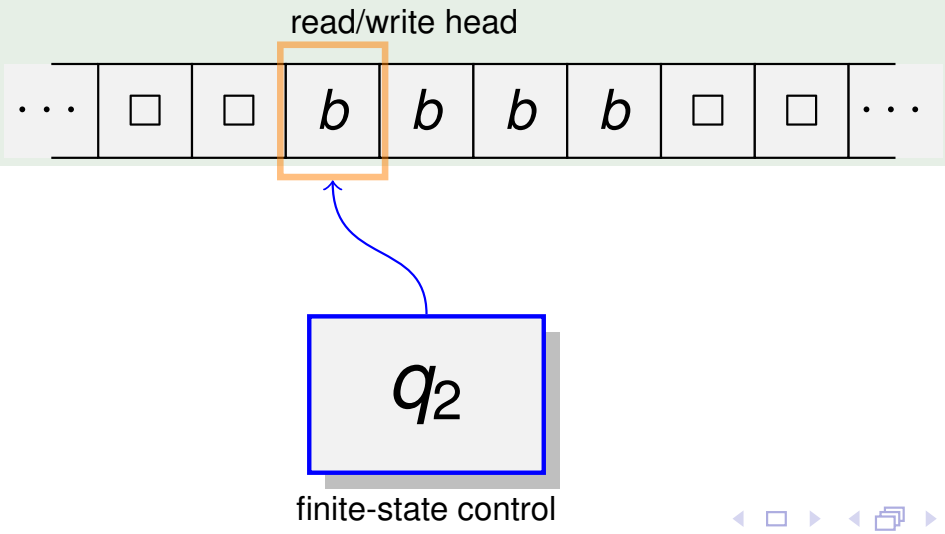
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

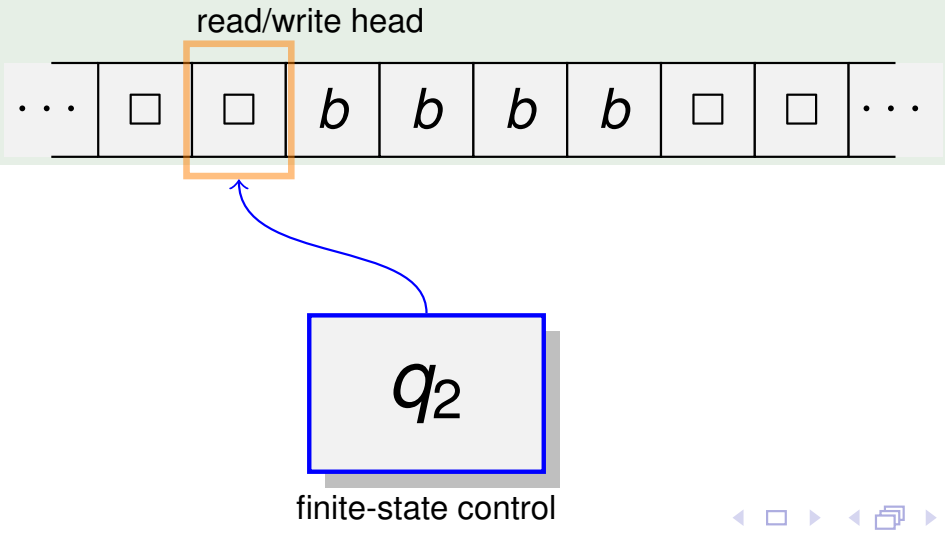
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a R \\
q_a b & \to & q_a b R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a 0 \\
q_b a & \to & q_b a R \\
q_b b & \to & q_b b R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a 0 \\
q_2 a & \to & q_2 a L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

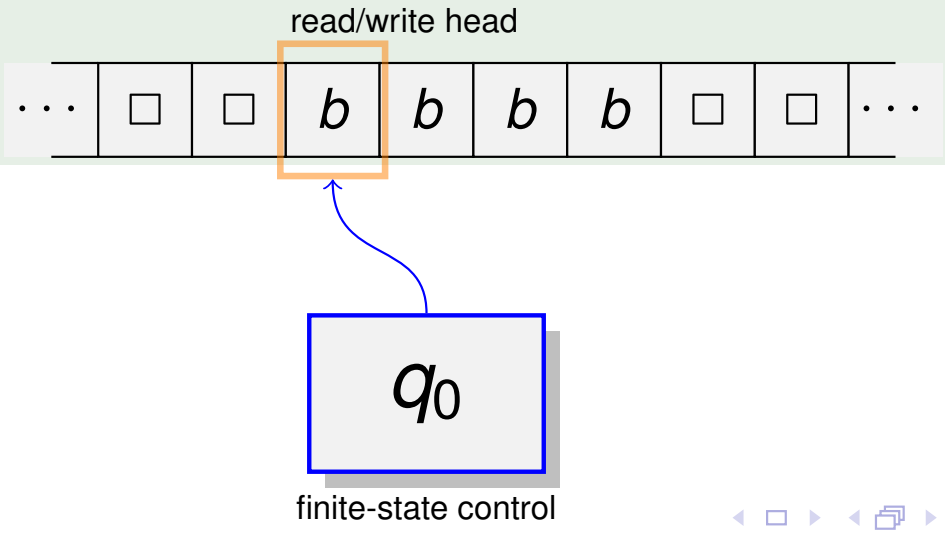
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

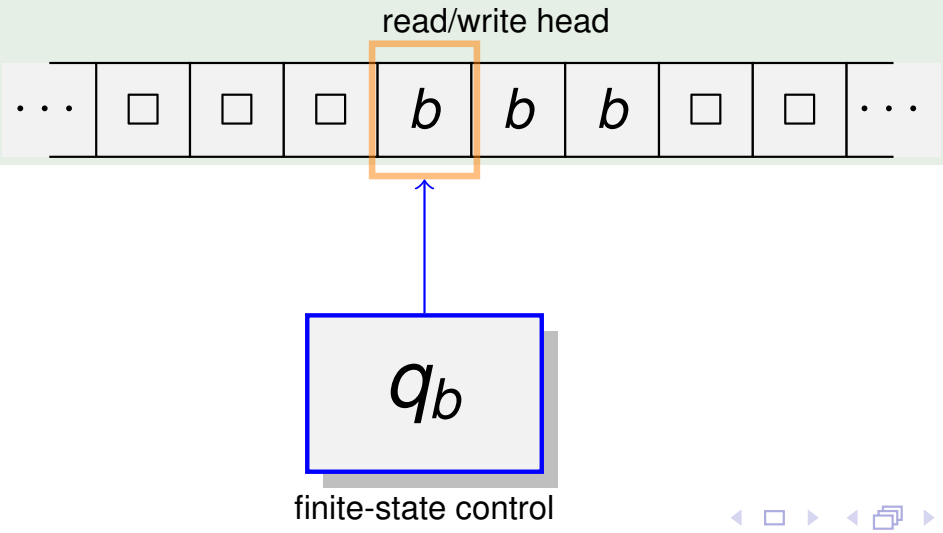
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

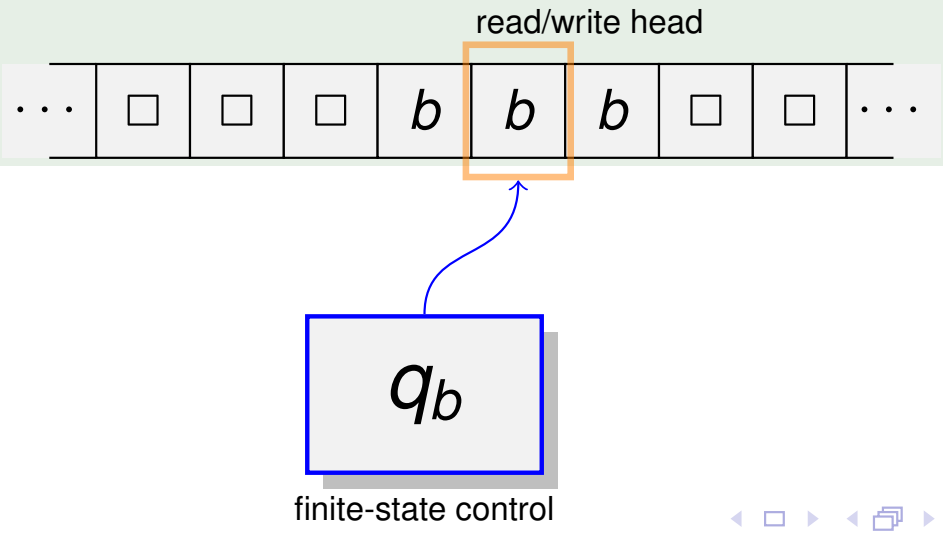
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

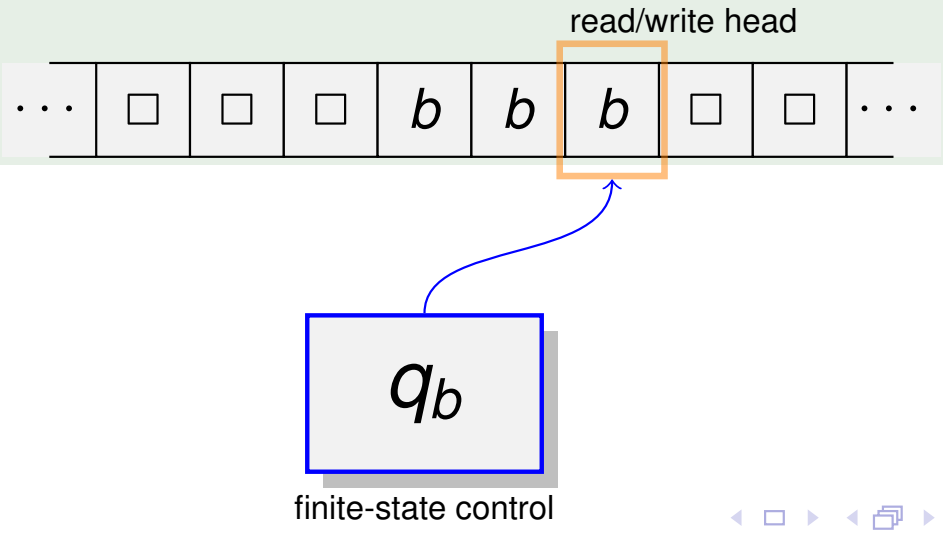
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

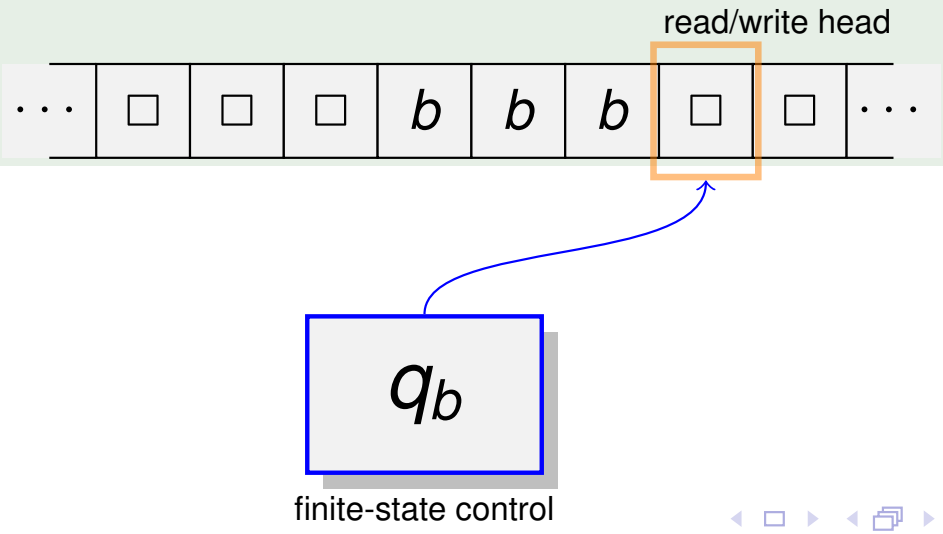
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

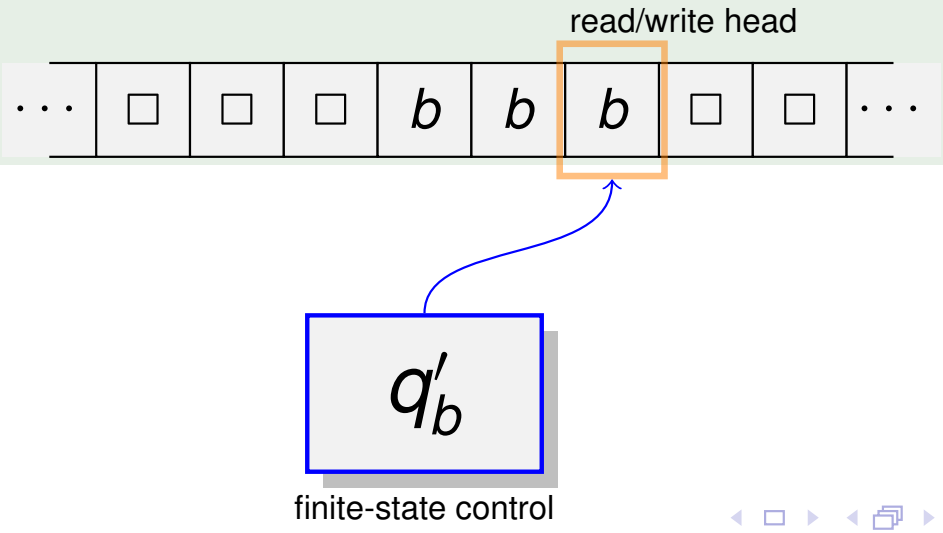
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

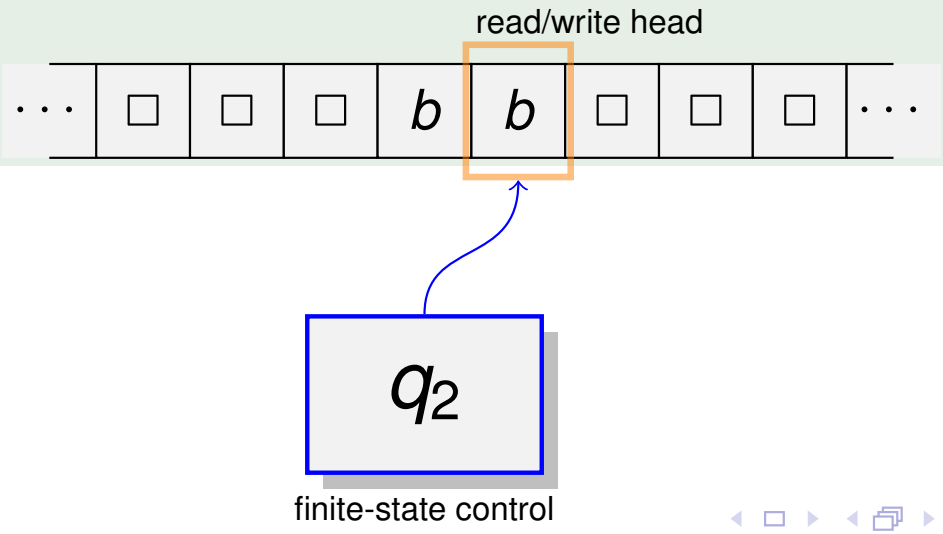
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

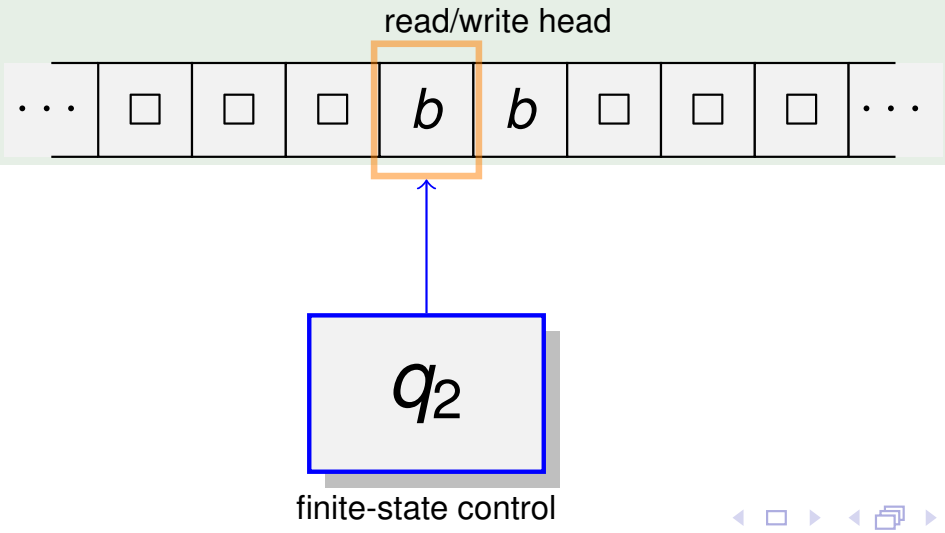
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

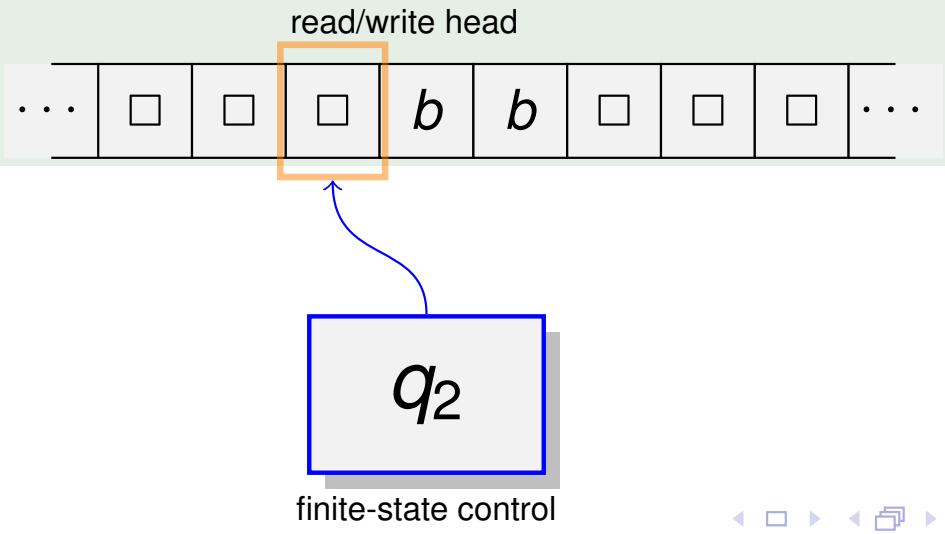
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

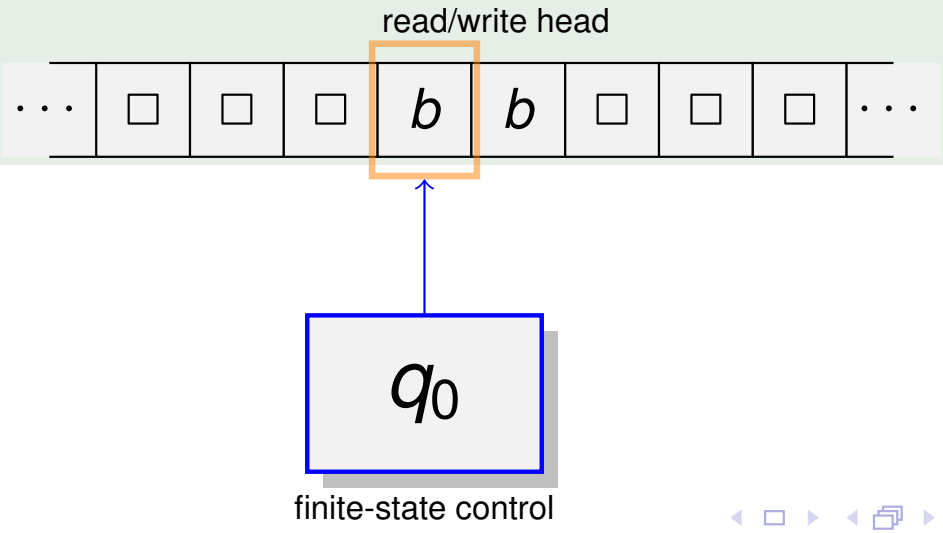
$$
\begin{array}{lcl}
q_0\Box \to q_1 a 0 & q_a'\Box \to q_1 a 0 & q_2 b \to q_2 b L \\
q_0 a \to q_a \Box R & q_b a \to q_b a R & q_2 \Box \to q_0 \Box R \\
q_0 b \to q_b \Box R & q_b b \to q_b b R & q_3 a \to q_3 \Box L \\
q_a a \to q_a a R & q_b \Box \to q_b' \Box L & q_3 b \to q_3 \Box L \\
q_a b \to q_a b R & q_b' a \to q_3 \Box L & q_3 \Box \to q_1 b 0 \\
q_a \Box \to q_a' \Box L & q_b' b \to q_2 \Box L & \\
q_a' a \to q_2 \Box L & q_b' \Box \to q_1 a 0 & \\
q_a' b \to q_3 \Box L & q_2 a \to q_2 a L &
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

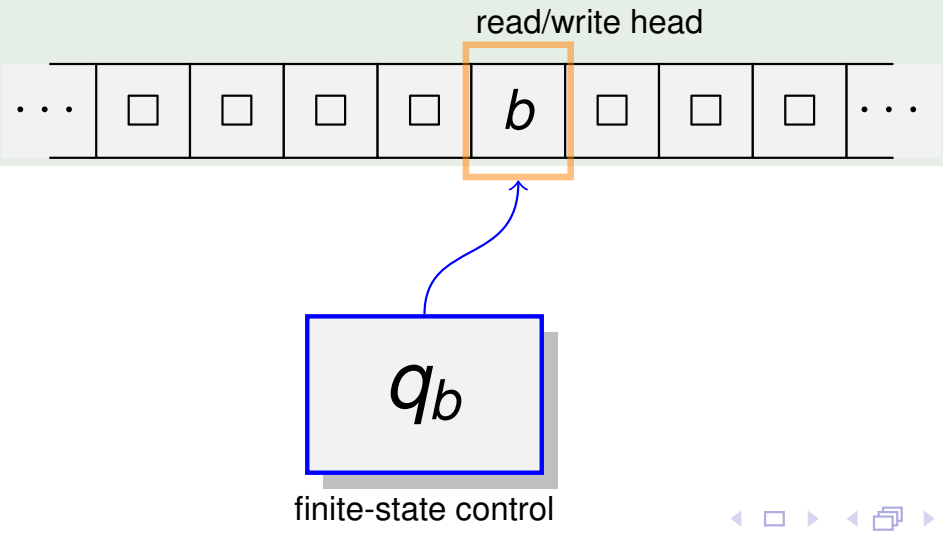
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

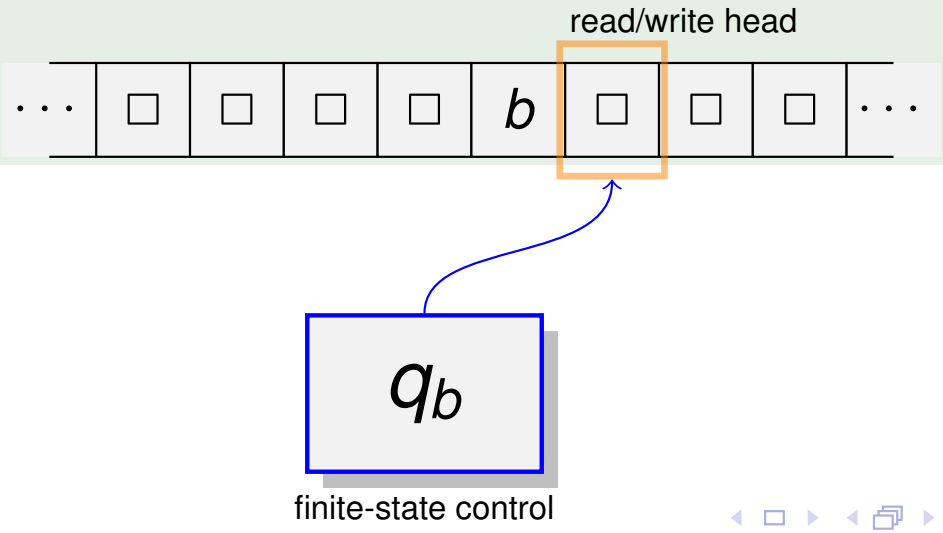
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

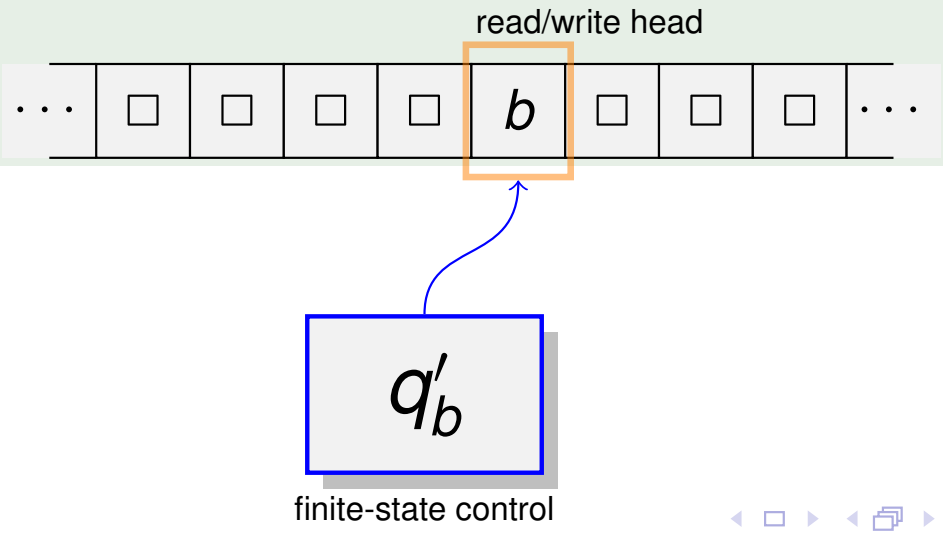
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

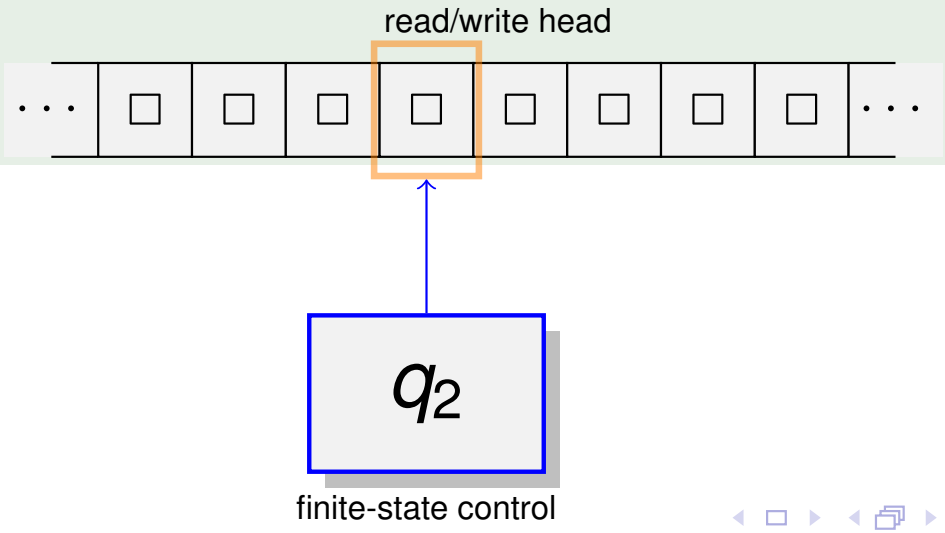
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

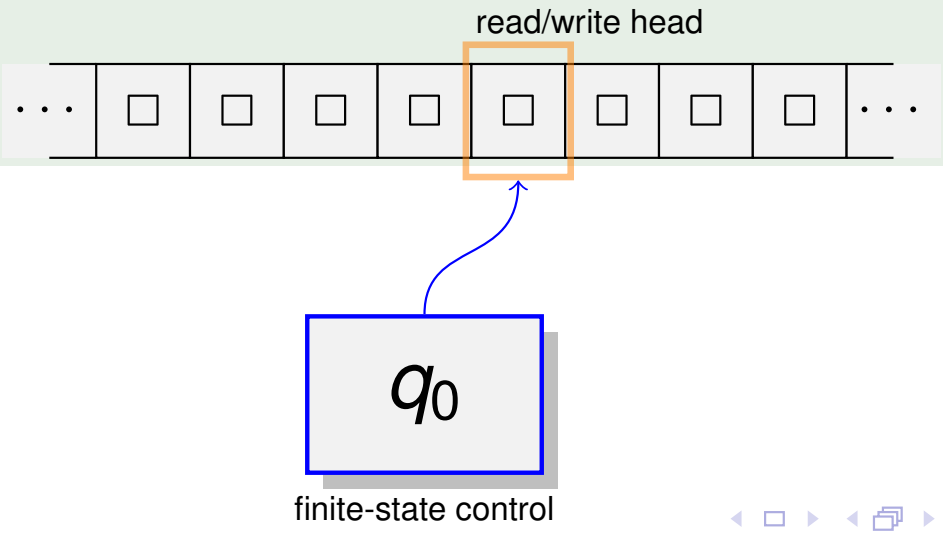
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

The transition function δ is defined as follows:

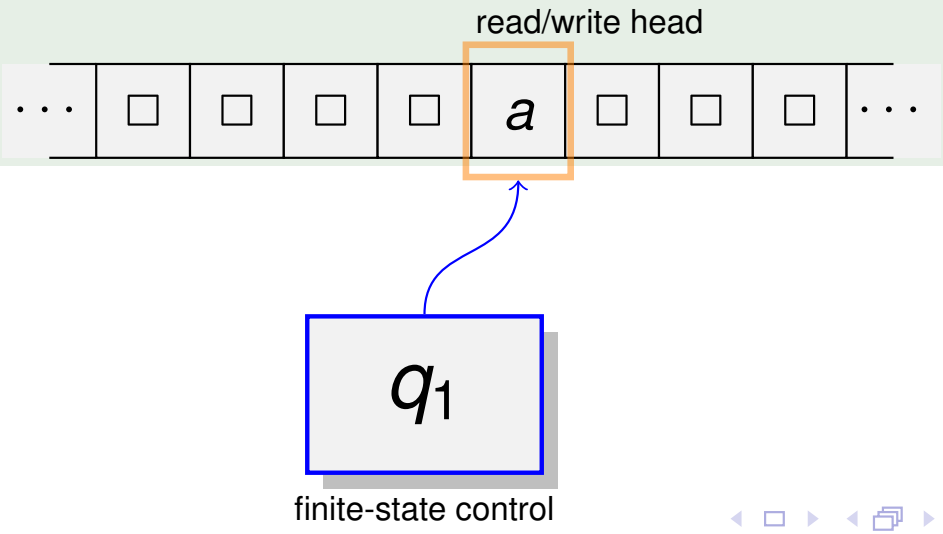
$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Example (cont.):

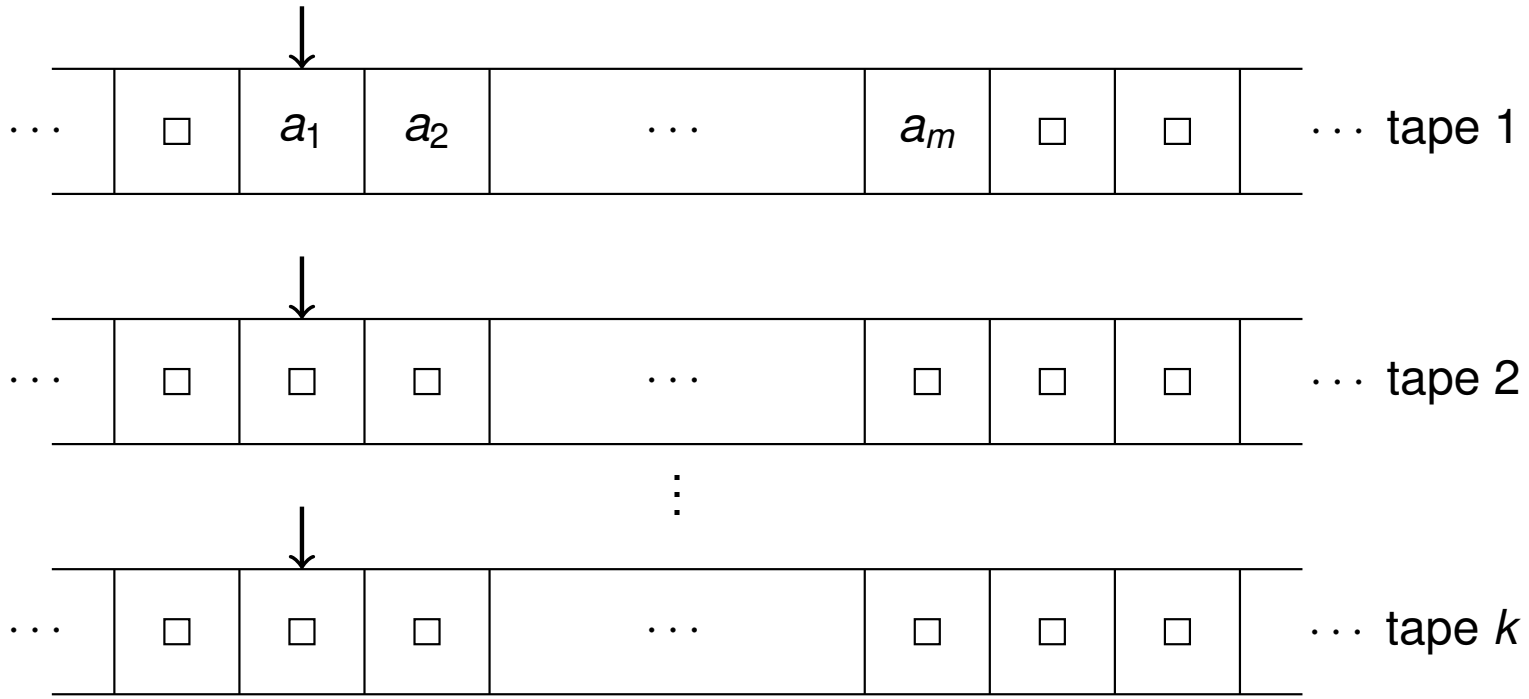
The transition function δ is defined as follows:

$$
\begin{array}{lcl}
q_0\Box & \to & q_1 a\, 0 \\
q_0 a & \to & q_a \Box R \\
q_0 b & \to & q_b \Box R \\
q_a a & \to & q_a a\, R \\
q_a b & \to & q_a b\, R \\
q_a \Box & \to & q'_a \Box L \\
q'_a a & \to & q_2 \Box L \\
q'_a b & \to & q_3 \Box L
\end{array}
\qquad
\begin{array}{lcl}
q'_a \Box & \to & q_1 a\, 0 \\
q_b a & \to & q_b a\, R \\
q_b b & \to & q_b b\, R \\
q_b \Box & \to & q'_b \Box L \\
q'_b a & \to & q_3 \Box L \\
q'_b b & \to & q_2 \Box L \\
q'_b \Box & \to & q_1 a\, 0 \\
q_2 a & \to & q_2 a\, L
\end{array}
\qquad
\begin{array}{lcl}
q_2 b & \to & q_2 b\, L \\
q_2 \Box & \to & q_0 \Box R \\
q_3 a & \to & q_3 \Box L \\
q_3 b & \to & q_3 \Box L \\
q_3 \Box & \to & q_1 b\, 0
\end{array}
$$



Let $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ be a k-tape TM, let $q \in Q$, and let $a_1, a_2, \ldots, a_m \in \Sigma$ be input symbols.

By $S(q, a_1 a_2 \cdots a_m)$ we denote the configuration of M, in which M is in state q and in which the contents of the tapes and the head positions are as follows:



Definition 4.1

(a) *Let* $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ *be a TM,* $\Sigma_1 \subseteq \Sigma \setminus \{*\}$, *where* $*$ *is a symbol from* Σ, $\Sigma_2 \subseteq \Sigma$, *and* $n \geq 0$.

A function $f : (\Sigma_1^*)^n \rightsquigarrow \Sigma_2^*$ *is computed by* M, *if the following holds for all* $x_1, x_2, \ldots, x_n \in \Sigma_1^*$:

(i) *If* $f(x_1, x_2, \ldots, x_n)$ *is defined, then starting from the configuration* $S(q_0, x_1 * x_2 * \cdots * x_n)$, M *will eventually reach the configuration* $S(q_1, f(x_1, x_2, \ldots, x_n))$.

(ii) *If* $f(x_1, x_2, \ldots, x_n)$ *is undefined, then starting from the configuration* $S(q_0, x_1 * x_2 * \cdots * x_n)$, M *will not halt at all, or it will halt in a configuration that is not of the form* $S(q_1, y)$ *for any* $y \in \Sigma^*$.

(b) *A function* $f : (\Sigma_1^*)^n \rightsquigarrow \Sigma_2^*$ *is called (Turing) computable, if there exists a TM that computes* f.

Definition 4.1 (cont.)

(c) *A function* $f : \mathbb{N}^n \rightsquigarrow \mathbb{N}$ *is called (Turing) computable, if the following dyadic encoding* $\psi : (\{1,2\}^*)^n \rightsquigarrow \{1,2\}^*$ *is (Turing) computable:*

$$\psi(x_1, x_2, \ldots, x_n) := \mathrm{dya}(f(\mathrm{dya}^{-1}(x_1), \mathrm{dya}^{-1}(x_2), \ldots, \mathrm{dya}^{-1}(x_n))).$$

(d) $\mathbb{TM}$ *denotes the class of functions that are (Turing) computable.*

Instead of the dyadic encoding one could choose an r-adic encoding for any $r > 2$. As the functions

$$\mathrm{dya} \circ \mathrm{ad}_r^{-1} : \{1, 2, \ldots, r\}^* \to \{1,2\}^*$$

and

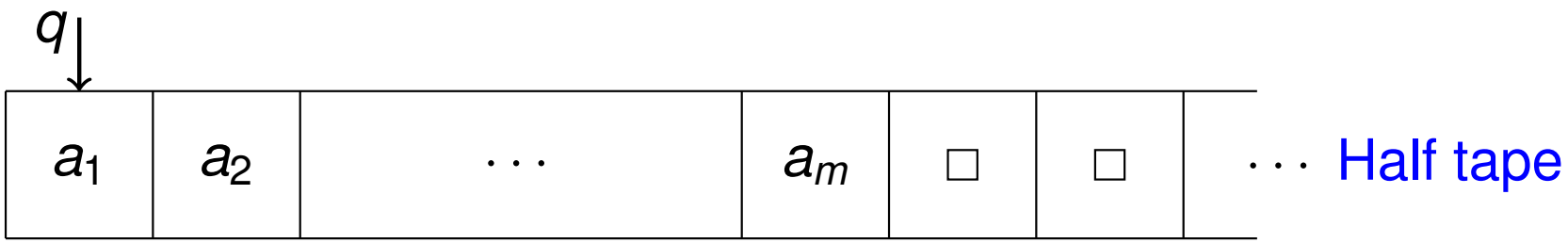
$$\mathrm{ad}_r \circ \mathrm{dya}^{-1} : \{1,2\}^* \to \{1, 2, \ldots, r\}^*$$

are (Turing) computable, the same class $\mathbb{TM}$ would be obtained.

Example (cont.):

The function $S(x) := x + 1$ is (Turing) computable.

A 1-Turingmachine (1-TM) is a TM with a single tape that is unbounded to the right only:



A transition $qa_1 \to q'bL$ has the same effect as $qa_1 \to 1'b\,0$.

The global situation above is denoted by $S(q, a_1 a_2 \cdots a_m)$.

Using 1-TMs, we obtain the class 1-$\mathbb{TM}$ of functions on $\mathbb{N}$ that are computable by 1-TMs.

Lemma 4.2

Let f be an n-ary function that is (1-)Turing computable. Then there exists a (1-)TM $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ that computes f and that satisfies the following condition:

$\forall x_1, x_2, \ldots, x_n \in \Sigma_1^*$ *: M halts eventually starting from the configuration $S(q_0, x_1 * x_2 * \cdots * x_n)$ if and only if $f(x_1, x_2, \ldots, x_n)$ is defined.*

Theorem 4.3

$\mathbb{TM} \subseteq 1\text{-}\mathbb{TM}$.

Proof of Theorem 4.3.

Let $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ be a TM with k tapes that computes the function $f : (\Sigma_1^*)^n \rightsquigarrow \Sigma_2^*$, where $\Sigma_1 \subseteq \Sigma \setminus \{*\}$ and $\Sigma_2 \subseteq \Sigma$.

By Lemma 4.2 we can assume that, for all $x_1, x_2, \ldots, x_n \in \Sigma_1^*$, M halts eventually starting from $S(q_0, x_1 * x_2 * \cdots * x_n)$ iff $f(x_1, x_2, \ldots, x_n)$ is defined.

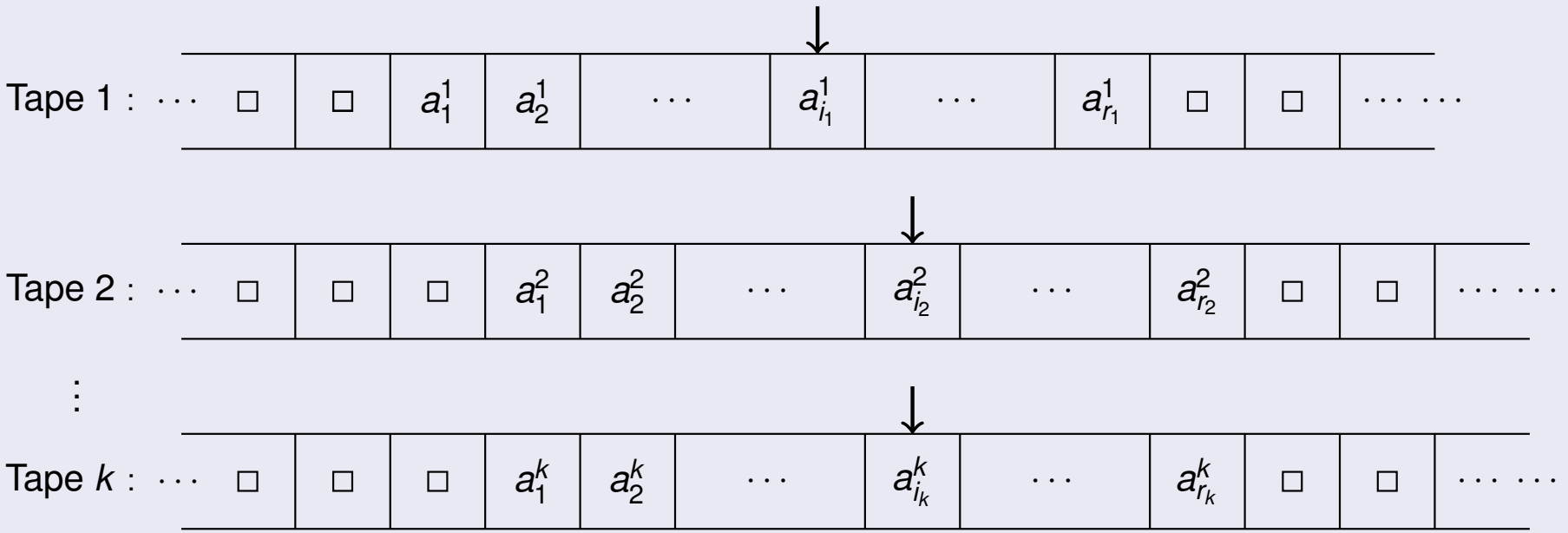
We describe a 1-TM $M' = (Q', \Sigma, \Gamma', \Box, \delta', q_0', q_1')$ that computes the function f by simulating M.

The alphabet Σ' includes Σ together with the following symbols: $\blacksquare, \nabla, \blacklozenge$.

On the tape of M', each configuration of M is encoded in a special way.

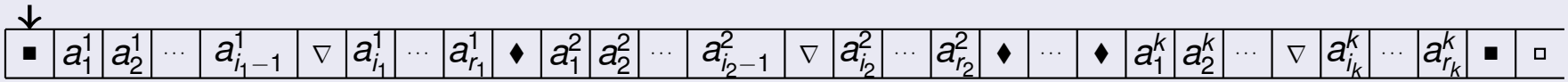
Proof of Theorem 4.3 (cont.)

Assume that M is in the configuration that is given by the state $q \in Q$ and the following tape situation:



Proof of Theorem 4.3 (cont.)

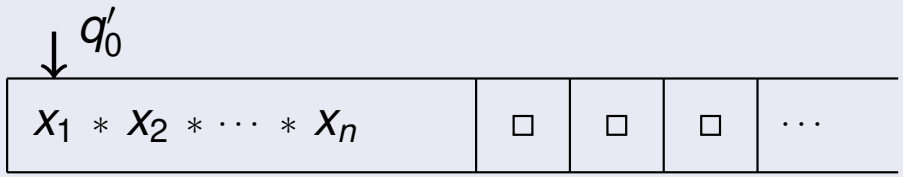
This configuration is encoded by the configuration of M' that is given by the state $(q, 1) \in Q'$ and the following tape situation:



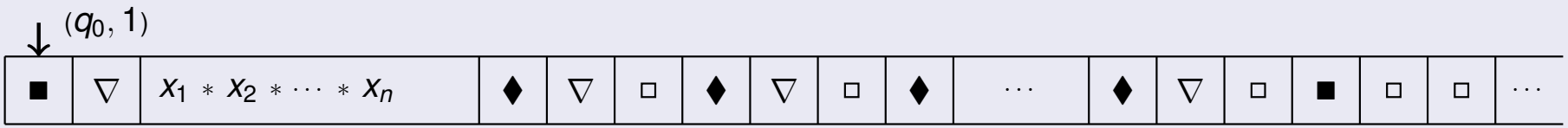
The 1-TM M' proceeds as follows.

Proof of Theorem 4.3 (cont.)

(1) At first the initial configuration



is transformed into the encoding of M's initial configuration:



Proof of Theorem 4.3 (cont.)

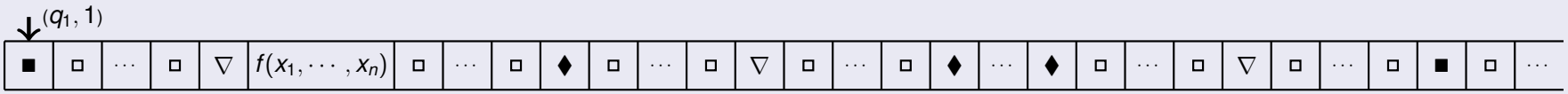
(2) Then M' simulates the TM M step by step. For simulating a single step of M, M''s head moves from left to right across the tape and stores the symbols currently read by M into M''s finite-state control.

As soon as all symbols have been read, M' moves its head back to the left, updating the symbols read and the positions of M's heads.

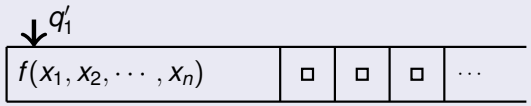
Observe that each transition of M may increase the length of the contents of some of its tapes.
Hence, M' must adjust the space provided for encoding these tapes by moving the corresponding suffix of its tape contents to the right.

Proof of Theorem 4.3 (cont.)

(3) If M does not halt on input $x_1 * x_2 * \cdots * x_n$, then neither does M'. If, however, M halts eventually, then M' halts with a tape contents of the following form:



This tape contents must now be reformated as follows:



It follows that M' computes the function f. □

A nondeterministic Turing machine (NTM) M with $k \geq 1$ tapes is given by a 7-tuple $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$, where $Q, \Sigma, \Gamma, \Box, q_0$ and q_1 are defined as for TMs, and

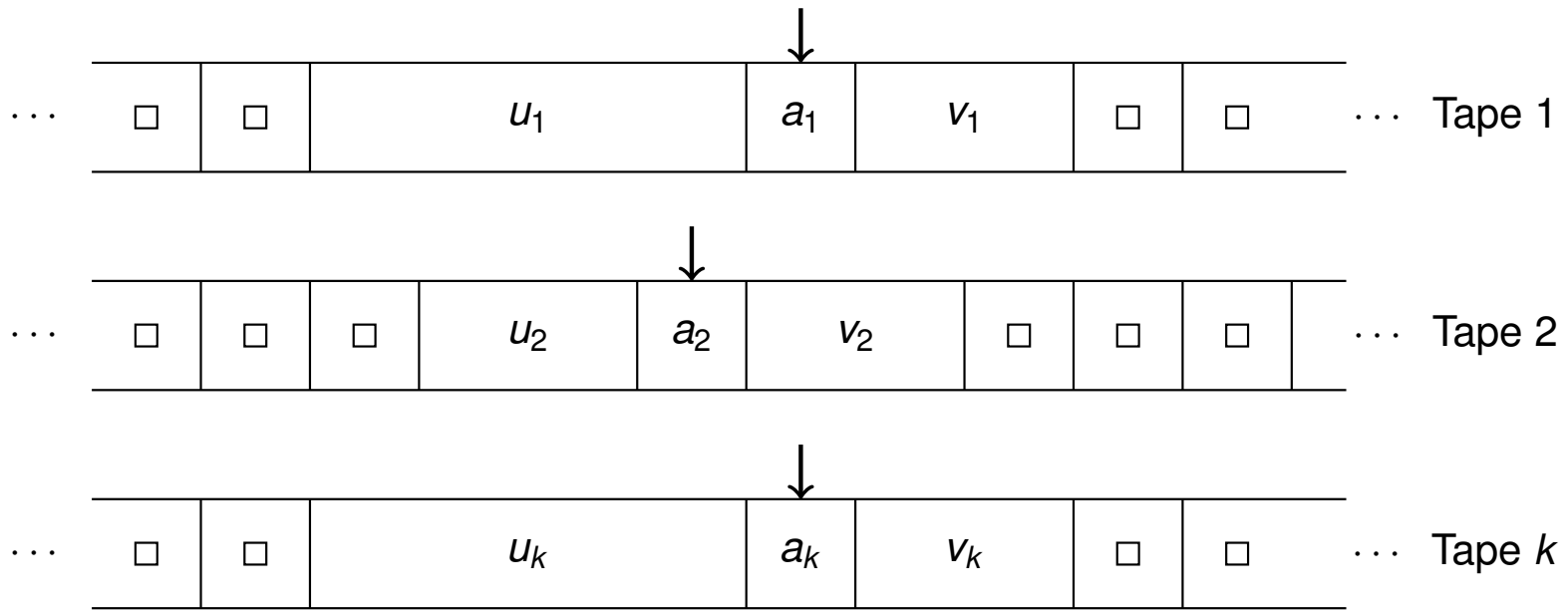
$$\delta : Q \times \Gamma^k \to 2^{Q \times \Gamma^k \times \{L,0,R\}^k}$$

is the transition relation. If

$$\delta(q, a_1, a_2, \ldots, a_k) = \{ (q_1^{(i)}, b_1^{(i)}, \ldots, b_k^{(i)}, m_1^{(i)}, \ldots, m_k^{(i)}) \mid i = 1, \ldots, r \},$$

then M has r options for its next step, if it is in state q reading symbol a_j on tape j $(1 \leq j \leq k)$.

Assume that M is in state $q \in Q$ and that the tape contents and head positions are as given in the following diagram:



This configuration is encoded as

$$(u_1 q a_1 v_1, u_2 q a_2 v_2, \ldots, u_k q a_k v_k).$$

Let CONF be the set of all encodings of all configurations of M.
Then M induces a binary relation $\vdash_M$ on CONF.

The language $L(M)$ accepted by M is defined as

$$L(M) := \{ w \in \Sigma^* \mid \exists K \in \mathrm{CONF} : (q_0 w, q_0, \ldots, q_0) \vdash_M^* K, \text{ and } K \text{ contains the halting state } q_1 \}.$$

Lemma 4.4

Each k-tape-(N)TM can be simulated by a 1-(N)TM.

Theorem 4.5

For each NTM M, there exists a TM M' such that $L(M) = L(M')$.

Proof of Theorem 4.5.

W.l.o.g. let M be a 1-NTM.
We construct a 2-tape-TM M' that simulates M.

Start of the simulation: Tape 1 contains the initial configuration of M, and tape 2 is empty.

A step of the simulation: Tape 1 (or 2) contains a list of all configurations that M can reach within $2i$ ($2i+1$) steps from the given initial configuration, and the other tape is empty.

For each configuration of M stored on tape 1 (2), all immediate successor configurations are written onto tape 2 (1).
As soon as a halting configuration is encountered, M' halts; otherwise, tape 1 (2) is erased.

Now tape 2 (1) contains all configurations that M can reach within $2i+1$ ($2i+2$) steps from the given initial configuration, and the next step of the simulation can be executed.

It follows that $L(M') = L(M)$. □

Let $\mathcal{L}(NTM)$ denote the class of languages that are accepted by NTMs, and let $\mathcal{L}(TM)$ be the class of languages that are accepted by TMs.

Corollary 4.6

$\mathcal{L}(NTM) = \mathcal{L}(TM)$.

A language $L \subseteq \Sigma^*$ is called recursively enumerable (r.e.), if there exists a TM M such that $L = L(M)$. By RE we denote the class of all recursively enumerable languages, that is, $\mathcal{L}(NTM) = \mathcal{L}(TM) = \text{RE}$.

A language is called $L \subseteq \Sigma^*$ recursive, if there exists a TM M satisfying the following conditions:

- $\forall w \in L \quad : \quad q_0 w \vdash_M^* q_1 1.$
- $\forall w \in \Sigma^* \setminus L \quad : \quad q_0 w \vdash_M^* q_1 0.$

By REC we denote the class of all recursive languages.

Theorem 4.7

(1) REC $\subseteq$ RE.

(2) *The class* REC *is closed under complementation.*

Theorem 4.8

A language L is recursive iff L and L^c are recursively enumerable.

Proof.

If L is recursive, then L and L^c are obviously r.e..

It remains to prove the converse implication.
Let M_1 be a TM such that $L(M_1) = L$, and let M_2 be a TM such that $L(M_2) = \Sigma^* \setminus L$. W.l.o.g. we can assume that M_1 and M_2 are 1-TMs.

Proof of Theorem 4.8 (cont.)

Let M' be the 2-tape-TM that proceeds as follows:

(1) At first M' copies its input from tape 1 onto tape 2.

(2) On tape 1, M' simulates the TM M_1, and on tape 2, it simulates the TM M_2. These simulations are executed in parallel, step by step.

If M_1 reaches its halting state q_1, then M' produces output 1 and halts, as then $w \in L(M_1) = L$,

if M_2 reaches its halting state q_1, then M' produces the output 0 and halts, as then $w \in L(M_2) = L^c$.

Since $L(M_1) \dot{\cup} L(M_2) = \Sigma^*$, exactly one of these two cases occurs for each input $w \in \Sigma^*$. Thus, M' decides correctly whether or not w belongs to L, which shows that L is recursive. □

4.2. Undecidability

A decision problem is called undecidable, if there does not exist a TM that answers each instance correctly after finitely many steps.

If $L \subseteq \Sigma^*$, then the Membership Problem for L is the following decision problem:

INSTANCE: A word $w \in \Sigma^*$.
QUESTION: Is $w \in L$?

Thus, this problem is decidable iff the language L is recursive.

In what follows we are interested in the Halting Problem for TMs:

INSTANCE: A TM M and an input word $w \in \Sigma^*$.
QUESTION: When starting with input w, will M halt eventually?

In order to study this problem we must encode the instance (M, w) in some way.

Let $M = (Q, \{0,1\}, \{0,1,\Box\}, \Box, \delta, q_0, q_n)$ be a 1-TM on $\Sigma = \{0,1\}$, and let $Q = \{q_0, q_1, \ldots, q_n\}$.

We will encode M through a word $c(M) \in \Sigma^+$.

Let $\delta = \{(q_{i_1}, a_{i_1}, q_{j_1}, a_{j_1}, m_{j_1}), \ldots, (q_{i_m}, a_{i_m}, q_{j_m}, a_{j_m}, m_{j_m})\}$, where $q_{i_e}, q_{j_e} \in Q$, $a_{i_e}, a_{j_e} \in \Sigma \cup \{\Box\}$, and $m_{j_e} \in \{L, 0, R\}$.

Each 5-tuple $(q_{i_e}, a_{i_e}, q_{j_e}, a_{j_e}, m_{j_e})$ is encoded as

$$c(q_{i_e}, a_{i_e}, q_{j_e}, a_{j_e}, m_{j_e}) := 0^{i_e+1}10^{e(a_{i_e})}10^{j_e+1}10^{e(a_{j_e})}10^{e(m_{j_e})},$$

$$\text{where } e(a_i) := \left\{ \begin{array}{ll} 1, & \text{if } a_i = 0, \\ 2, & \text{if } a_i = 1, \\ 3, & \text{if } a_i = \Box, \end{array} \right\} \text{ and } e(m) := \left\{ \begin{array}{ll} 1, & \text{if } m = L, \\ 2, & \text{if } m = 0, \\ 3, & \text{if } m = R. \end{array} \right\}$$

The function δ is interpreted as a sequence of 5-tuples.
Assuming that this sequence is sorted in lexicographical order, we take

$$\begin{aligned} c(M) \quad := \quad & 1110^{n+1}11111 \cdot c(q_{i_1}, a_{i_1}, q_{j_1}, a_{j_1}, m_{j_1}) \cdot 11 \cdot \\ & \ldots \cdot 11 \cdot c(q_{i_m}, a_{i_m}, q_{j_m}, a_{j_m}, m_{j_m}) \cdot 111. \end{aligned}$$

Lemma 4.9

The set

$$\{\, c(M) \mid M \text{ is a 1-TM on } \Sigma = \{0,1\} \text{ and } \Gamma = \{0,1,\Box\} \,\}$$

is recursive.

Proof of Lemma 4.9.

Let $w \in \{0,1\}^*$. If w is not an element of the regular language

$$1^3 \cdot 0^{n+1} \cdot 1^5 \cdot (0^{1 \le i \le n} \cdot 1 \cdot 0^{1 \le i \le 3} \cdot 1 \cdot 0^{1 \le i \le n+1} \cdot 1 \cdot 0^{1 \le i \le 3} \cdot 1 \cdot 0^{1 \le i \le 3} \cdot 11)^{\le 3 \cdot n} \cdot 1,$$

then w is not the encoding of a 1-TM.

If, however, w is an element of the above regular language, then one can try to reconstruct M from w. This reconstruction is successful iff w describes a function

$$\delta : \{q_0, \ldots, q_{n-1}\} \times \{0, 1, \Box\} \rightsquigarrow \{q_0, \ldots, q_n\} \times \{0, 1, \Box\} \times \{L, 0, R\}.$$

This function δ then yields the TM M satisfying $c(M) = w$. □

By M_∞ we denote the following TM, which does not halt on any input:

$$(\{q_0, q_1\}, \{0,1\}, \{0,1,\Box\}, \Box, \{(q_0, a, q_0, a, 0) \mid a \in \{0,1,\Box\}\}, q_0, q_1).$$

With each word $w \in \Sigma^*$, we now associate a TM M_w:

$$M_w := \begin{cases} M, & \text{if } c(M) = w, \\ M_\infty, & \text{if } w \text{ is not the encoding of any TM.} \end{cases}$$

By Lemma 4.9, the TM M_w can be reconstructed from w.

Now let $K \subseteq \{0,1\}^*$ be the following language:

$$K := \{\, w \in \{0,1\}^* \mid M_w \text{ halts on input } w \,\}.$$

Theorem 4.10

The language K is not recursive.

Proof of Theorem 4.10.

Assume to the contrary that K is recursive. Then there exists a 1-TM M_0 that decides membership in K, that is,

$$q_0^{(0)} w \vdash_{M_0}^* q_1^{(0)} 1, \text{ if } w \in K,$$

and

$$q_0^{(0)} w \vdash_{M_0}^* q_1^{(0)} 0, \text{ if } w \notin K.$$

By modifying M_0 we obtain a new TM M_1 that behaves as follows:

$$q_0^{(0)} w \vdash_{M_0}^* q_1^{(0)} a \vdash_{M_1} \begin{cases} q_2 a \vdash_{M_1} q_2 a \vdash_{M_1} \cdots, & \text{if } a = 1, \\ q_1 0, & \text{if } a = 0. \end{cases}$$

Proof of Theorem 4.10 (cont.)

Hence, for all $w \in \Sigma^*$: M_1 halts on input w iff $q_0^{(0)} w \vdash_{M_0}^* q_1^{(0)} 0$, that is, iff $w \notin K$.

Now let $u := c(M_1)$. Then $M_u = M_1$, and we have the following sequence of equivalent statements:

M_1 halts on input u iff $u \notin K$
iff M_u does not halt on input u
iff M_1 does not halt on input u, a contradiction!

This contradiction shows that the language K is not recursive. □

Corollary 4.11

$K \in \text{RE} \setminus \text{REC}$ *and* $K^c \notin \text{RE}$.

Corollary 4.12

The Halting Problem for TMs is undecidable.

Proof.

Let H be the following language:

$$H := \{ (w, u) \mid M_w \text{ halts on input } u \}.$$

Then $w \in K$ iff $(w, w) \in H$.

If H were recursive, then K would be recursive, too.

Thus, H is not recursive, that is,
the Halting Problem for (1-)TMs is undecidable. □

Let $\mathcal{S}$ be a set of recursively enumerable languages on $\{0,1\}$.
We interpret $\mathcal{S}$ as a property of recursively enumerable languages.

We say that a language L has property $\mathcal{S}$, if $L \in \mathcal{S}$.

The property $\mathcal{S}$ is called trivial, if $\mathcal{S} = \emptyset$ or $\mathcal{S} = \mathrm{RE}(\{0,1\})$.

Finally, let $L_{\mathcal{S}} := \{\, c(M) \mid L(M) \in \mathcal{S} \,\}$.

Theorem 4.13 (Rice 1953)

The language $L_{\mathcal{S}}$ is non-recursive for each non-trivial property $\mathcal{S}$ of recursively enumerable languages, that is, given a TM M, it is in general undecidable whether the language $L(M)$ has property $\mathcal{S}$.

Proof of Theorem 4.13.

W.l.o.g. we can assume that $\emptyset \notin \mathcal{S}$, as otherwise we could consider the set $\mathcal{S}^c := \mathrm{RE}(\{0,1\}) \setminus \mathcal{S}$ instead of $\mathcal{S}$.

As $\mathcal{S}$ is non-trivial, there exists a language $\emptyset \neq L \in \mathcal{S}$.

Let M_L be a TM such that $L(M_L) = L$.

Assume that the language $\mathcal{S}$ is decidable, that is, $L_{\mathcal{S}} \in \mathrm{REC}(\{0,1\})$.
Then there is a TM $M_{\mathcal{S}}$ for deciding $L_{\mathcal{S}}$.
From M_L and $M_{\mathcal{S}}$, we now construct a TM for the halting problem H.

Let M be a TM, and let $w \in \{0,1\}^*$ be an input word.

From M and w, we can construct a TM $M'_{M,w}$ that, on input $x \in \{0,1\}^*$, executes the following program:

(1) simulate M on input w;

(2) **if** M halts on input w **then** simulate M_L on input x.

Proof of Theorem 4.13 (cont.)

Then $L(M'_{M,w}) = \begin{cases} \emptyset, & \text{if } w \notin L(M), \\ L, & \text{if } w \in L(M). \end{cases}$

By our hypothesis, $\emptyset \notin \mathcal{S}$ and $L \in \mathcal{S}$.

Hence, $c(M'_{M,w}) \in L_{\mathcal{S}}$ iff $w \in L(M)$.

Thus, the TM $M_{\mathcal{S}}$ accepts on input $c(M'_{M,w})$ iff $w \in L(M)$,

and otherwise, $M_{\mathcal{S}}$ rejects this input.

It follows that the TM $M_{\mathcal{S}}$ decides membership in H.

As H is undecidable, this is a contradiction!

Hence, $L_{\mathcal{S}}$ is non-recursive. □

Corollary 4.14

The following properties are undecidable for recursively enumerable languages:

- *emptiness,*
- *finiteness,*
- *regularity,*
- *context-freeness.*

Let $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ be a 1-TM, and
let $\Delta := \Gamma \dot\cup Q \dot\cup \{\#\}$, where $\#$ is an additional symbol.

A valid computation of M is a word of the form

$$w = w_1 \# w_2^R \# w_3 \# w_4^R \cdots \# w_{2m}^R \# (w_{2m+1} \#)^\mu \in \Delta^+,$$

where $\mu \in \{0, 1\}$ and $n := \left\{ \begin{array}{ll} 2m, & \text{if } \mu = 0 \\ 2m+1, & \text{if } \mu = 1 \end{array} \right\}$,
that satisfies the following conditions:

(1) $\forall i = 1, 2, \ldots, n:\ \ w_i \in \Gamma^* \cdot Q \cdot \Gamma^*$, where w_i does not end with the symbol $\Box$;

(2) $w_1 = q_0 x$ for some $x \in \Sigma^*$, that is,
w_1 is an initial configuration of M;

(3) $w_n \in \Gamma^* \cdot q_1 \cdot \Gamma^*$, that is, w_n is a halting configuration of M;

(4) $\forall i = 1, 2, \ldots, n-1 : w_i \vdash_M w_{i+1}$.

By $\mathrm{GB}(M)$ we denote the language on Δ that consists of all valid computations of M.

Lemma 4.15

From a given 1-TM M, one can effectively construct two context-free grammars G_1 and G_2 such that $L(G_1) \cap L(G_2) = \mathrm{GB}(M)$.

Proof.

Let L_3 be the language

$$L_3 := \{ y\#z^R \mid y, z \in \Gamma^* \cdot Q \cdot \Gamma^* \text{ such that } y \vdash_M z \}.$$

From M one can easily construct a PDA that accepts L_3.

From L_3 we obtain the language L_1:

$$L_1 := (L_3 \cdot \#)^* \cdot (\{\varepsilon\} \cup (\Gamma^* \cdot q_1 \cdot \Gamma^* \cdot \#)).$$

From M we can construct a context-free grammar for the language L_1 (Theorem 3.20, Theorem 3.22).

Proof of Lemma 4.15 (cont.)

Further, let L_4 be the language

$$L_4 := \{ y^R \# z \mid y, z \in \Gamma^* \cdot Q \cdot \Gamma^* \text{ such that } y \vdash_M z \},$$

and let L_2 be obtained from L_4 as follows:

$$L_2 := q_0 \Sigma^* \cdot \# \cdot (L_4 \cdot \#)^* \cdot (\{\varepsilon\} \cup (\Gamma^* \cdot q_1 \cdot \Gamma^* \cdot \#)).$$

From M we can construct a context-free grammar for L_2.

Claim.

$L_1 \cap L_2 = \mathrm{GB}(M)$.

Proof of Lemma 4.15 (cont.)

Proof of Claim.

Let $w = w_1 \# w_2^R \# \cdots \# w_n \#$ such that $n \equiv 1 \bmod 2$.

If $w \in \mathrm{GB}(M)$, then properties (1) to (4) imply that $w \in L_1 \cap L_2$.

Conversely, if $w \in L_1 \cap L_2$, then we see from the definitions of L_1 and L_2 that w satisfies (1) and (4).

As $w \in L_2$, $w_1 = q_0 x$ for some $x \in \Sigma^*$, and

as $w \in L_1$, $w_n \in \Gamma^* \cdot q_1 \cdot \Gamma^*$, that is, $w \in \mathrm{GB}(M)$.

For $w = w_1 \# \cdots \# w_n^R \#$ such that $n \equiv 0 \bmod 2$, the proof is analogous.

Thus, $L_1 \cap L_2 = \mathrm{GB}(M)$. □

□

Let M be a 1-TM.

Then $L(M) \neq \emptyset$ iff $\mathrm{GB}(M) \neq \emptyset$.

Now let G_1 and G_2 be two context-free grammars such that $L(G_1) \cap L(G_2) = \mathrm{GB}(M)$.

Then $L(M) \neq \emptyset$ iff $L(G_1) \cap L(G_2) \neq \emptyset$.

As emptiness is undecidable for $L(M)$, this yields the following result.

Corollary 4.16

The following Intersection Emptiness Problem is undecidable:

INSTANCE: *Two context-free grammars G_1 and G_2.*
QUESTION: *Is $L(G_1) \cap L(G_2) = \emptyset$?*

The set $\Delta^* \setminus \mathrm{GB}(M) = \mathrm{GB}(M)^c$ is called the set of invalid computations of M.

Lemma 4.17

For each 1-TM M, $\mathrm{GB}(M)^c \in \mathrm{CFL}(\Delta)$.

As $L(M) = \emptyset$ iff $\mathrm{GB}(M)^c = \Delta^*$, we obtain the following undecidability result.

Corollary 4.18

The following Universality Problem is undecidable:

INSTANCE: *A context-free grammar G on Δ.*
QUESTION: *Is $L(G) = \Delta^*$?*

Theorem 4.19

The following problems are undecidable:

(1) INSTANCE: Two context-free grammars G_1 and G_2.

- *QUESTION: Is $L(G_1) = L(G_2)$?*
- *QUESTION: Is $L(G_1) \subseteq L(G_2)$?*
- *QUESTION: Is $L(G_1) \cap L(G_2)$ context-free?*
- *QUESTION: Is $L(G_1) \cap L(G_2)$ regular?*

(2) INSTANCE: A context-free grammar G and a regular set R.

- *QUESTION: Is $L(G) = R$?*
- *QUESTION: Is $R \subseteq L(G)$?*

(3) INSTANCE: A context-free grammar G.

- *QUESTION: Is $L(G)^c$ context-free?*
- *QUESTION: Is $L(G)^c$ regular?*

Proof.

Let G_1 be a context-free grammar s.t. $L(G_1) = R = \Sigma^*$.
Then the following holds for each context-free grammar G_2:

$$R = L(G_1) = L(G_2) \text{ iff } R = L(G_1) \subseteq L(G_2) \text{ iff } L(G_2) = \Sigma^*.$$

It follows from Corollary 4.18 that the first two problems of (1) and the two problems of (2) are undecidable.

The language $\mathrm{GB}(M)$ is finite and therewith regular, if $L(M)$ is finite; on the other hand, if $L(M)$ is infinite, then $\mathrm{GB}(M)$ is not even context-free, which can be shown by the Pumping Lemma (Theorem 3.14), if M makes at least 3 steps on each input.

Proof of Theorem 4.19 (cont.)

Let M be an arbitrary 1-TM. From M one can construct a 1-TM M' that accepts the same language as M, but that executes at least 3 steps on each input.

Now $L(M)$ is finite iff $\mathrm{GB}(M') = (\mathrm{GB}(M')^c)^c$ is context-free (regular).

Further, from M' we obtain two context-free grammars G_1 and G_2 such that $L(G_1) \cap L(G_2) = \mathrm{GB}(M')$.

As finiteness of $L(M)$ is undecidable, it follows that the questions of whether $(\mathrm{GB}(M')^c)^c$ or $L(G_1) \cap L(G_2)$ are context-free (regular) are undecidable, too. □

4.3. General Phrase-Structure Grammars

A phrase-structure grammar is a 4-tuple $G = (N, T, S, P)$, where $P \subseteq (N \cup T)^* \times (N \cup T)^*$ is a finite semi-Thue system on $N \cup T$ such that $|\ell|_N \geq 1$ for all $(\ell \to r) \in P$ (see Section 2).

Theorem 4.20

For each phrase-structure grammar G, the language $L(G)$ is recursively enumerable.

Proof.

For $G = (N, T, S, P)$ we describe a 2-tape-NTM M that accepts the language $L(G)$: The input w is stored on tape 1, and on tape 2, M guesses a derivation of G, starting from S.
As soon as a terminal word z has been generated on tape 2, M checks whether $z = w$ holds. In the affirmative, M halts, while in the negative, M enters an infinite loop. Hence, $L(M) = L(G)$. □

Theorem 4.21

If L is a recursively enumerable language, then there exists a phrase-structure grammar G such that $L(G) = L$.

Proof.

Let $L \subseteq \Sigma^*$ be r.e., and let $M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$ be a 1-TM s.t. $L(M) = L$. We construct a grammer $G = (N, \Sigma, A_1, P)$ by taking

$$N := ((\Sigma \cup \{\varepsilon\}) \times \Gamma) \cup Q \cup \{A_1, A_2, A_3\},$$

and defining P as follows:

(1) $A_1 \to q_0 A_2$,

(2) $A_2 \to [a, a]A_2$ for all $a \in \Sigma$,

(3) $A_2 \to A_3$,

(4) $A_3 \to [\varepsilon, \Box]A_3$,

(5) $A_3 \to \varepsilon$.

Using these productions G generates a sentential form

$$q_0[a_1, a_1][a_2, a_2] \cdots [a_n, a_n][\varepsilon, \Box] \cdots [\varepsilon, \Box]$$

for a word $w = a_1 a_2 \cdots a_n \in \Sigma^*$. This sentential form encodes the initial configuration of the TM M on input w.

Proof of Theorem 4.21 (cont.)

Further, P contains the following productions for simulating M:

(6) $q[a, X] \to [a, Y]p$, if $\delta(q, X) = (p, Y, R)$,

(7) $[b, Z]q[a, X] \to p[b, Z][a, Y]$ for all $b \in \Sigma \cup \{\varepsilon\}$ and $Z \in \Gamma$, if $\delta(q, X) = (p, Y, L)$,

(8) $q[a, X] \to p[a, Y]$, if $\delta(q, X) = (p, Y, 0)$.

If $q_0 w \Box^m \vdash_M^* upv$ holds for some $u, v \in \Gamma^*$, $p \in Q$, and $m \geq 0$, then

$$q_0[a_1, a_1][a_2, a_2] \cdots [a_n, a_n][\varepsilon, \Box]^m \to_P^*$$

$$[a_1, u_1] \cdots [a_k, u_k]p[a_{k+1}, v_1] \cdots [a_n, v_{n-k}][\varepsilon, v_{n-k+1}] \cdots [\varepsilon, v_{n+m-k}],$$

where $u = u_1 u_2 \ldots u_k$ and $v = v_1 v_2 \ldots v_{n+m-k}$.

Proof of Theorem 4.21 (cont.)

Finally, P also contains some productions for treating halting configurations:

$$(9)\qquad \left.\begin{array}{lcl} [a,X]q_1 & \to & q_1 a q_1 \\ q_1[a,X] & \to & q_1 a q_1 \\ q_1 & \to & \varepsilon \end{array}\right\} \text{ for all } a \in \Sigma \cup \{\varepsilon\} \text{ and } X \in \Gamma.$$

If $q_0 w \vdash_M^* u q_1 v$, that is, if $w \in L(M)$, then:

$$\begin{array}{lll} A_1 & \to^* & q_0[a_1,a_1][a_2,a_2]\cdots[a_n,a_n][\varepsilon,\Box]^m \\ & \to^* & [a_1,u_1]\cdots[a_k,u_k]q_1[a_{k+1},v_1]\cdots[a_n,v_{n-k}]\cdots[\varepsilon,v_{n+m-k}] \\ & \to^* & a_1 a_2 \cdots a_n = w, \end{array}$$

which shows that $L(M) \subseteq L(G)$.

Conversely, if $w \in L(G)$, then we see from the construction above that M halts on input w, which implies that $L(G) = L(M)$. □

Corollary 4.22

A language L is recursively enumerable if and only if it is generated by a phrase-structure grammar.

From the various characterizations of the class of r.e. languages, we obtain the following closure properties.

Corollary 4.23

(a) *The class RE is closed under union, intersection, product, Kleene star, reversal, and inverse morphisms.*

(b) *The class RE is not closed under complementation.*

4.4. Context-Sensitive Languages

A grammar $G = (N, T, S, P)$ is context-sensitive if each production $(\ell \to r) \in P$ has the form $\alpha X \beta \to \alpha u \beta$, where $X \in N$, $\alpha, \beta, u \in (N \cup T)^*$, and $u \neq \varepsilon$. In addition, G may contain the production $(S \to \varepsilon)$, if S does not occur on the right-hand side of any production.

A language L is called context-sensitive if there exists a context-sensitive grammar G such that $L = L(G)$.

By CSL(Σ) we denote the set of all context-sensitive languages on Σ, and CSL denotes the class of all context-sensitive languages.

A grammar $G = (N, T, S, P)$ is called monotone if $|\ell| \leq |r|$ holds for each production $(\ell \to r) \in P$. Also a monotone grammar may contain the production $(S \to \varepsilon)$ if S does not occur on the right-hand side of any production.

Theorem 4.24

(a) *For each context-sensitive grammar G, there is a monotone grammar G' such that $L(G') = L(G)$*

(b) *For each monotone grammar G', there is a context-sensitive grammar G such that $L(G) = L(G')$.*

Proof.

(a) $\Rightarrow$ (b): By definition each context-sensitive grammar is monotone.

(b) $\Rightarrow$ (a): Let $G' = (N', T, S', P')$ be a monotone grammar.
From G' we construct $G'' = (N'', T, S', P'')$ by taking

$$N'' := N' \cup \{ A_a \mid a \in T \} \text{ and } P'' := h(P') \cup \{ A_a \to a \mid a \in T \},$$

where h is defined through $A \mapsto A$ $(A \in N')$ and $a \mapsto A_a$ $(a \in T)$.
G'' is monotone, and all the new productions are context-sensitive.
It remains to replace the productions in $h(P')$ by context-sensitive productions.

Proof of Theorem 4.24 (cont.)

Let $A_1 \cdots A_m \to B_1 \cdots B_n$ $(2 \leq m \leq n)$ be a production from $h(P')$, where $A_i, B_j \in N''$. We introduce new nonterminals $Z_1, Z_2, \ldots, Z_m$ and replace the above production by the following ones:

$$\begin{array}{rcl}
A_1 \cdots A_m & \to & Z_1 A_2 \cdots A_m \\
Z_1 A_2 \cdots A_m & \to & Z_1 Z_2 A_3 \cdots A_m \\
 & \vdots & \\
Z_1 Z_2 \cdots Z_{m-1} A_m & \to & Z_1 \cdots Z_m B_{m+1} \cdots B_n \\
Z_1 \cdots Z_m B_{m+1} \cdots B_n & \to & B_1 Z_2 \cdots Z_m B_{m+1} \cdots B_n \\
 & \vdots & \\
B_1 \cdots B_{m-1} Z_m B_{m+1} \cdots B_n & \to & B_1 \cdots B_{m-1} B_m B_{m+1} \cdots B_n.
\end{array}$$

The new productions are context-sensitive. It's obvious that they simulate the old production. On the other hand, the new productions can only be used for this purpose, as the new nonterminals $Z_1, Z_2, \ldots, Z_m$ do not occur in any other production. By repeating the process above for each production from $h(P')$, we obtain a context-sensitive grammar G such that $L(G) = L(G')$ □

Example.

Let G be the following monotone grammar:

$$G = (\{S, B\}, \{a, b, c\}, S, \{S \to aSBc, S \to abc, cB \to Bc, bB \to bb\}).$$

We claim that $L(G) = L := \{ a^n b^n c^n \mid n \geq 1 \}$.

Claim 1:

$L \subseteq L(G)$.

Proof of Claim 1.

We show that $a^n b^n c^n \in L(G)$ by induction on n.
For $n = 1$, we have $S \to abc$.
Assume that $S \to^* a^n b^n c^n$ has been shown for some $n \geq 1$.
We consider the following derivation:

$$S \to aSBc \to^* a \cdot a^n b^n c^n \cdot Bc \to^* a^{n+1} b^n Bc^{n+1} \to a^{n+1} b^{n+1} c^{n+1}.$$

□

Example (cont.)

Claim 2:

$L(G) \subseteq L$.

Proof of Claim 2.

By applying production 1 repeatedly, we obtain a sentential form $a^n S(Bc)^n$, which is rewritten into a sentential form $a^n S\alpha$ by production 3, where $\alpha \in \{B, c\}^+$ and $|\alpha|_B = |\alpha|_c = n$.
In order to get rid of S, production 2 must be applied, that is, we obtain $a^n S\alpha \to a^{n+1} bc\alpha$.
To get rid of all nonterminals B in α, all occurrences of c must be moved to the right and then all B are rewritten into b by production 4, that is, $a^{n+1} bc\alpha \to^* a^{n+1} bB^n c^{n+1} \to^* a^{n+1} b^{n+1} c^{n+1}$.
Hence, $L(G) \subseteq L$. □

Together Claims 1 and 2 show that $L(G) = L$. □

Each context-free grammar in CNF (Theorem 3.9) is context-sensitive.

On the other hand, $\{ a^n b^n c^n \mid n \geq 1 \} \notin$ CFL (see the first example after Theorem 3.14).

Corollary 4.25

CFL $\subsetneq$ CSL.

A grammar $G = (N, T, S, P)$ is in Kuroda Normal Form, if it only contains productions of the following forms:

$$(A \to a), (A \to BC), (AB \to CD), \text{ where } a \in T \text{ and } A, B, C, D \in N.$$

With respect to the production $(S \to \varepsilon)$, we have the same restriction as before.

Theorem 4.26

Given a context-sensitive grammar G, one can effectively construct an equivalent context-sensitive grammar G′ that is in Kuroda Normal Form.

Proof of Theorem 4.26.

As in the proof of Theorem 3.9 we can first revise G in such a way that terminals only occur on the right-hand side of productions of the form $(A \to a)$.

Next we replace productions of the form $(A \to B_1 B_2 \cdots B_n)$ $(n > 2)$ by new productions

$$(A \to B_1 Z_2), (Z_2 \to B_2 Z_3), \ldots, (Z_{n-1} \to B_{n-1} B_n),$$

where $Z_2, Z_3, \ldots, Z_{n-1}$ are new nonterminals.

Finally, let $(A_1 A_2 \cdots A_m \to B_1 \cdots B_n)$ be a productions s.t. $m > 1$ and $m + n > 4$. We choose new nonterminals $Z_2, Z_3, \ldots, Z_{n-1}$ and replace the production above by the following productions in Kuroda form:

Proof of Theorem 4.26 (cont.)

$$
\begin{array}{rcl}
(A_1A_2 & \to & B_1Z_2), \\
(Z_2A_3 & \to & B_2Z_3), \\
 & \vdots & \\
(Z_{m-1}A_m & \to & B_{m-1}Z_m), \\
(Z_m & \to & B_mZ_{m+1}), \\
(Z_{m+1} & \to & B_{m+1}Z_{m+2}), \\
 & \vdots & \\
(Z_{n-1} & \to & B_{n-1}B_n).
\end{array}
$$

The new productions can simulate the old one. On the other hand, they cannot be used in any other way, as the new nonterminals do not occur in any other productions.

By repeating this process for all productions of the form above, we obtain a grammar G' in Kuroda Normal Form s.t. $L(G') = L(G)$. □

A linear-bounded automaton (LBA) M is a 1-NTM

$$M = (Q, \Sigma, \Gamma, \Box, \delta, q_0, q_1)$$

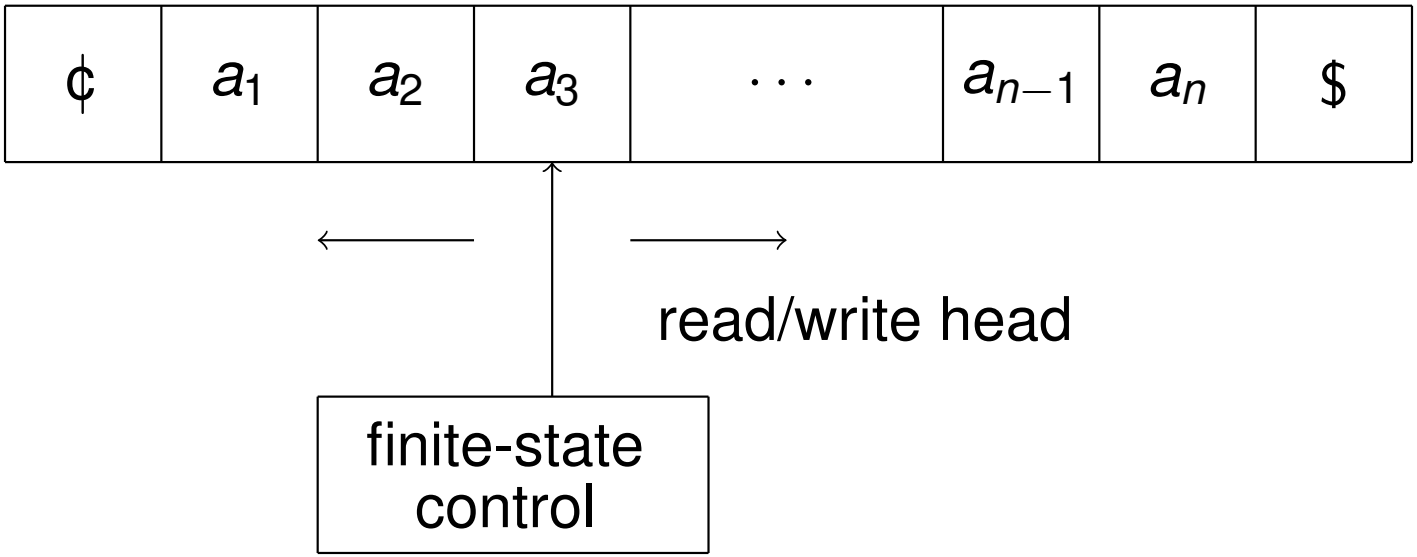
with two special symbols $¢, \$ \in \Gamma$.

For $w \in \Sigma^*$, $q_0 ¢ w \$$ is the initial configuration of M on input w, and

$$L(M) := \{ w \in \Sigma^* \mid \exists \alpha, \beta \in \Gamma^* : q_0 ¢ w \$ \vdash_M^* ¢ \alpha q_1 \beta \$ \}$$

is the language accepted by M.

An LBA can be depicted as follows:



Theorem 4.27

For each $L \in$ CSL, there exists an LBA M such that $L = L(M)$.

Proof.

We proceed as in the proof of Theorem 4.20, only that here our NTM M has a tape with two tracks instead of two tapes:

On track 1, the input word w is stored, and on track 2, a derivation of the context-sensitive grammar G is simulated nondeterministically. For doing so, we can assume that G is in Kuroda Normal Form.

If the simulated derivation generates the word w, then M accepts. If another terminal word is obtained, or if the space provided by the length of the input w does not allow for another step, then M enters an infinite loop.

Thus, M is an LBA such that $L(M) = L$. □

Theorem 4.28

For each LBA M, there exists a context-sensitive grammar G such that $L(G) = L(M)$.

Proof.

Here we proceed as in the proof of Theorem 4.21, that is, from a given LBA M, we construct a grammar G that simulates the computations of M.

Given a word $w = a_1 a_2 \cdots a_n \in \Sigma^*$ as input, the corresponding initial configuration $q_0 ¢ w \$$ of M is encoded by the word

$$\binom{¢\, a_1}{¢\, q_0 a_1}\binom{a_2}{a_2}\cdots\binom{a_{n-1}}{a_{n-1}}\binom{a_n \$}{a_n \$},$$

which is derived from the start symbol S of G by applying some context-free productions.

Proof of Theorem 4.28 (cont.)

Then a computation of the LBA M is simulated on the lower track only using productions $(\ell \to r)$ satisfying $|\ell| = |r|$.

If $w \in L(M)$, then a word of the following form can be derived:

$$\binom{¢\,a_1}{¢\,b_1}\binom{a_2}{b_2}\cdots\binom{a_i}{q_1\,b_i}\cdots\binom{a_{n-1}}{b_{n-1}}\binom{a_n\,\$}{b_n\,\$},$$

which can be rewritten into the word $a_1 a_2 \cdots a_{n-1} a_n = w$ by using monotone productions.

Hence, G is a context-sensitive grammar satisfying $L(G) = L(M)$. □

Corollary 4.29

A language L is context-sensitive iff there exists an LBA M such that $L(M) = L$.

A language L is called deterministic context-sensitive if it is accepted by a deterministic LBA. We denote the corresponding class of languages by DCSL.

Obviously, CFL $\subsetneq$ DCSL $\subseteq$ CSL, but it is still open whether the inclusion DCSL $\subseteq$ CSL is proper.
This is the famous LBA Problem (see [Hartmanis, Hunt 1974]).

Corollary 4.30

CSL $\subseteq$ REC, *that is, each context-sensitive language has decidable membership problem.*

Proof.

Let G be a context-sensitive grammar. In order to decide whether $w \in L(G)$ it suffices to generate all sentential forms α of G that satisfy the condition $|\alpha| \leq |w|$ and that can be derived from the start symbol S. If w is found in this way, then $w \in L$; otherwise, $w \notin L$. □

Corollary 4.31

For each recursively enumerable language L on Σ, there exists a context-sensitive language L' on $\Gamma \supsetneq \Sigma$ such that $\Pi_\Sigma(L') = L$. Here $\Pi_\Sigma : \Gamma^* \to \Sigma^*$ is the morphism that is defined by

$$a \mapsto a \ (a \in \Sigma) \text{ and } b \mapsto \varepsilon \ (b \in \Gamma \setminus \Sigma).$$

Proof.

Let L be a r.e. language on Σ. Then there exists a 1-TM $M = (Q, \Sigma, \Delta, \square, \delta, q_0, q_1)$ s.t. $L(M) = L$.

We take $\Gamma := \Sigma \cup \{\$\}$ and

$$L' := \{ w\$^n \mid w \in L \text{ and } M \text{ accepts } w \text{ in space } |w| + n \}.$$

Proof of Corollary 4.31 (cont.)

For $w \in \Sigma^*$ and $m \in \mathbb{N}$, M is given the input $w\$^m$.
Now M runs until it either accepts, which implies that $w\$^m \in L'$,
or until it needs more space than $|w| + m$, or until it gets into a loop.
In the latter two cases, $w\$^m \notin L'$.

From M we easily obtain an LBA that accepts L'.
Hence, $L' \in \text{CSL}$ and $L = \Pi_\Sigma(L')$. □

Because of Corollary 4.31, CSL is called a basis for the class RE.

Corollary 4.32

The class CSL *is not closed under morphisms.*

Proof.

This follows from the inclusion $\text{CSL} \subseteq \text{REC} \subsetneq \text{RE}$ and from Corollary 4.31. □

Theorem 4.33

The class CSL *is closed under union, product, Kleene star, and* ε*-free morphisms.*

Theorem 4.34

The class CSL *is closed under intersection and inverse morphisms.*

Theorem 4.35 (Immerman 1987, Szelepczenyi 1987)

The class CSL *is closed under complementation.*

Corollary 4.36

CSL $\subsetneq$ REC.

Actually, the membership problem for a context-sensitive language is decidable in exponential time.

On the other hand, Corollaries 4.14 and 4.31 imply that the following problems are undecidable for CSL:

- finiteness,
- emptiness,
- regularity,
- context-freeness,
- inclusion, and
- equality.

Chapter 5:

Summary

Summary on Characterizations:

Language classes	Grammars	Automata
Typ 3 (regular)	regular	DFA NFA
det. context-free		DPDA
Typ 2 (context-free)	context-free	PDA
Typ 1 (context-sensitive)	monotone	LBA
Typ 0 (recursively enumerable)	general	TM NTM

Summary on Closure Properties:

Operation	REG	DCFL	CFL	CSL	RE
Union	+	-	+	+	+
Intersection	+	-	-	+	+
Intersection with REG	+	+	+	+	+
Complementation	+	+	-	+	-
Product	+	-	+	+	+
Kleene star	+	-	+	+	+
Morphism	+	-	+	-	+
Inverse Morphism	+	+	+	+	+

Summary on Decision Problems:

Decision problem	REG	DCFL	CFL	CSL	RE
Membership	+	+	+	+	−
Emptiness	+	+	+	−	−
Finiteness	+	+	+	−	−
Equality	+	+	−	−	−
Inclusion	+	−	−	−	−
Regularity	+	+	−	−	−

\+ decidable
− undecidable