transparencies made for

Second Course in Formal Languages and Automata Theory

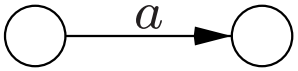
based on the book by Jeffrey Shallit
of the same title

Hendrik Jan Hoogeboom, Leiden

http://www.liacs.nl/~hoogeboo/second/

Chapter 1

Review of Formal Languages and Automata Theory

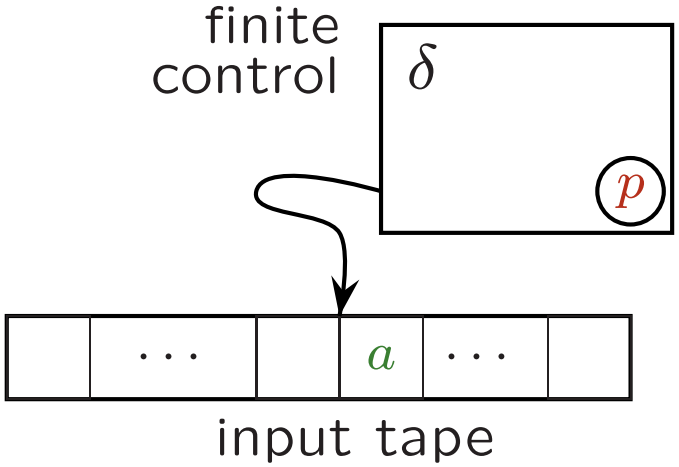
	grammar	**automaton**
3	regular	
	right-linear $A \to aB$	finite state 
2	context-free	
	$A \to \alpha$	pushdown (+lifo stack)
1	context-sensitive	
	$(\beta_\ell, A, \beta_r) \to \alpha$ $\alpha \to \beta \quad \lvert\beta\rvert \geq \lvert\alpha\rvert$ monotone	linear bounded
0	recursively enumerable	
	$\alpha \to \beta$	turing machine

Moved to respective chapters

Chapter 3

Finite Automata and Regular Languages

3.0 Review

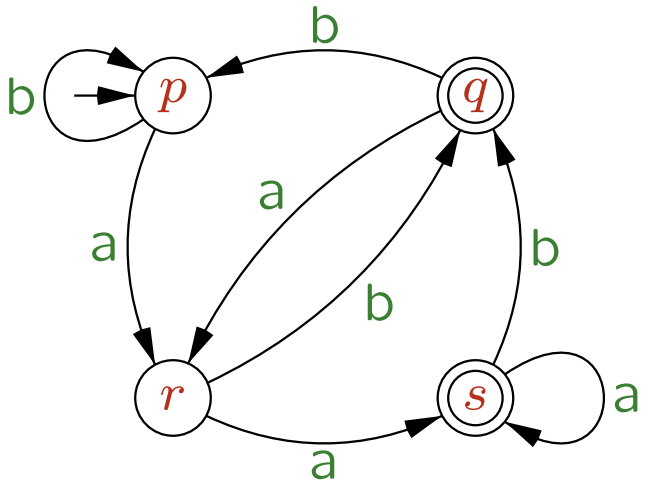


$$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$$

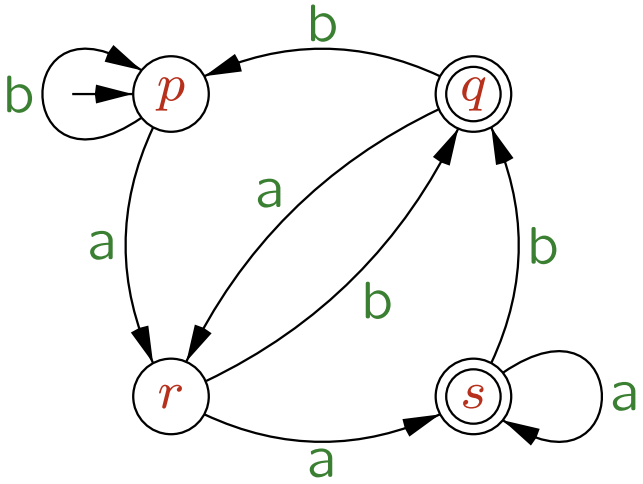
Q	*states*	p, q
$q_0 \in Q$	*initial state*	
$F \subseteq Q$	*final states*	
Σ	*input alphabet*	$a, b \quad w, x$
$\delta : Q \times \Sigma \to Q$	*transition function*	

$$\delta^* : Q \times \Sigma^* \to Q$$

$$\delta^*(q, \varepsilon) = q$$
$$\delta^*(q, xa) = \delta(\ \delta^*(q, x),\ a)$$



$$L(\mathcal{A}) = \{\ x \in \Sigma^* \mid \delta(q_0, x) \in F\ \}$$

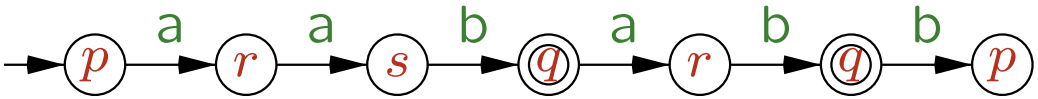


	a	b
p	r	p
q	r	p
r	s	q
s	s	q

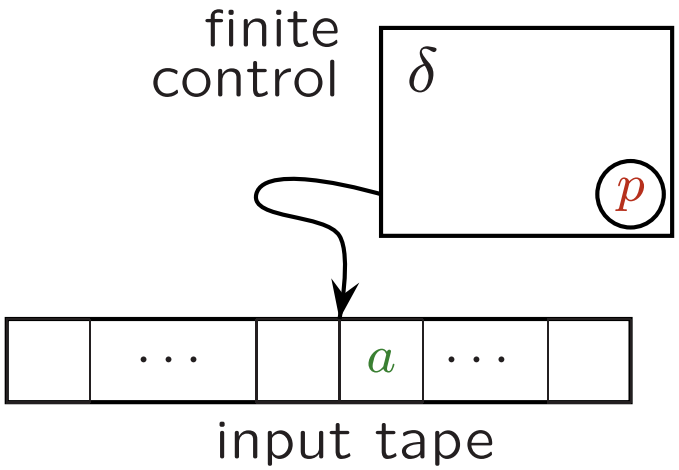
initial state p

final states q, s

run on $aababb$



does not accept



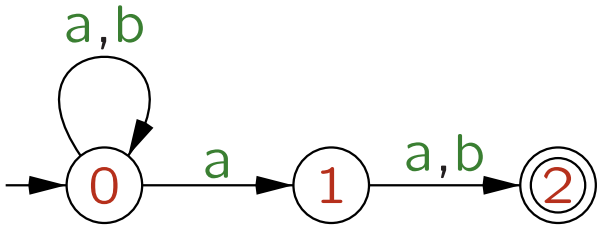
$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$

Q	*states*	p, q
$q_0 \in Q$	*initial state*	
$F \subseteq Q$	*final states*	
Σ	*input alphabet*	a, b w, x
$\delta : Q \times \Sigma \to 2^Q$	*transition function*	

$\delta^* : Q \times \Sigma^* \to 2^Q$

$\delta^*(q, \varepsilon) = \{q\}$

$\delta^*(q, xa) = \bigcup_{r \in \delta^*(q,x)} \delta(r, a)$



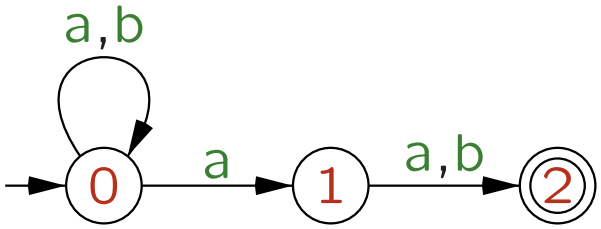
$\delta(0, \mathrm{a}) = \{0, 1\}$

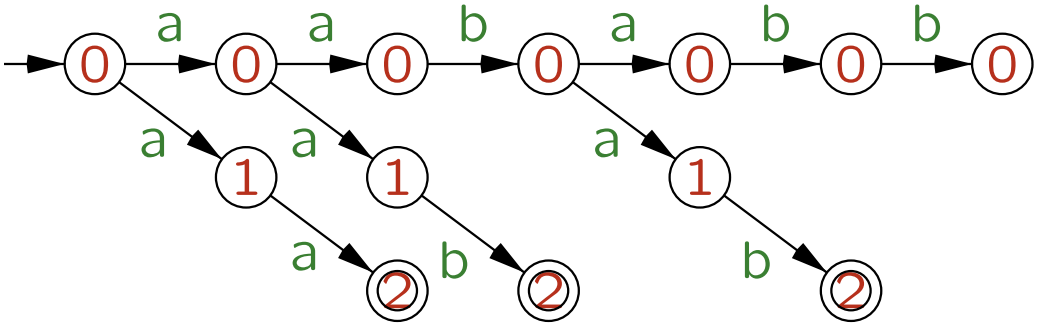
$\delta(2, \mathrm{a}) = \varnothing$

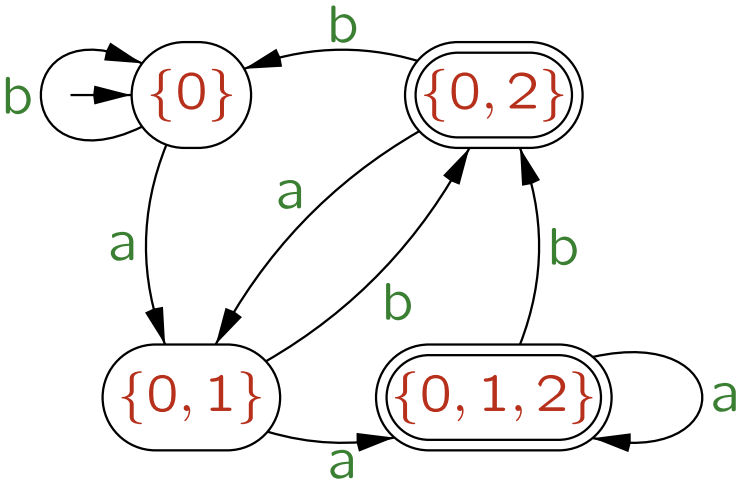
$L(\mathcal{A}) = \{ x \in \Sigma^* \mid \delta(q_0, x) \cap F \neq \varnothing \}$

$\delta' : 2^Q \times \Sigma \to 2^Q$ (deterministic)

$\delta'(U, a) = \bigcup_{p \in U} \delta(p, a)$



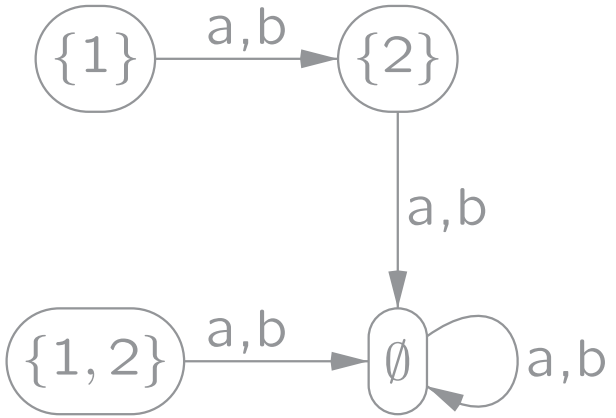


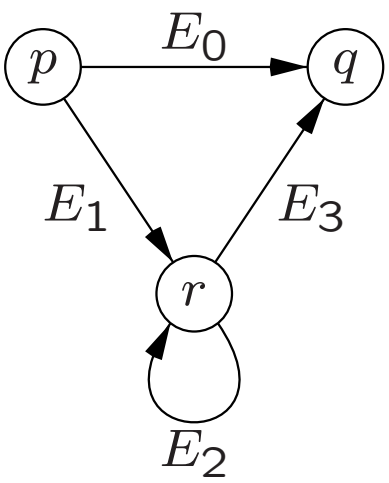


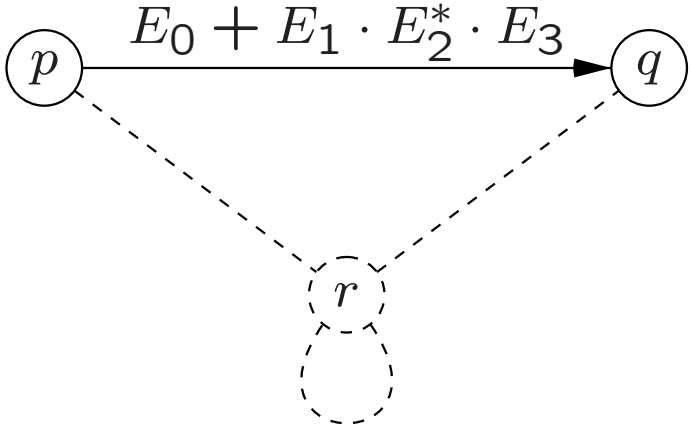
deterministic

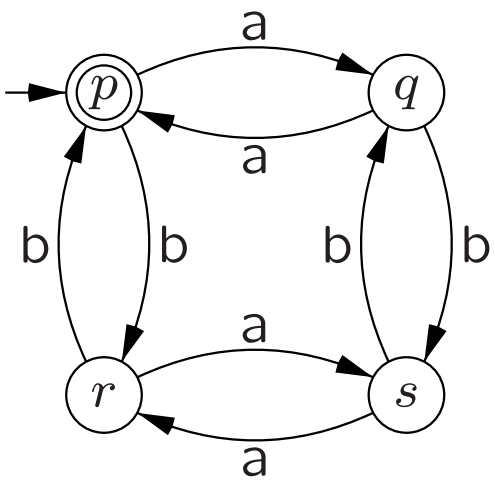
$\delta' : 2^Q \times \Sigma \to 2^Q$

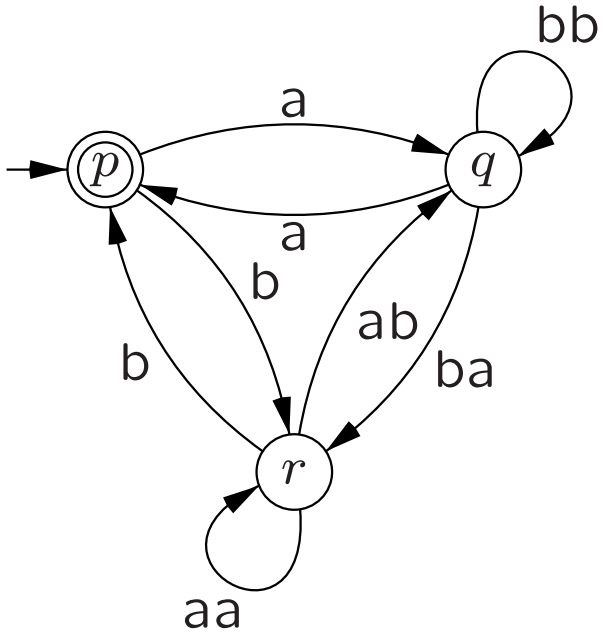
$\delta'(U, a) = \bigcup_{p \in U} \delta(p, a)$

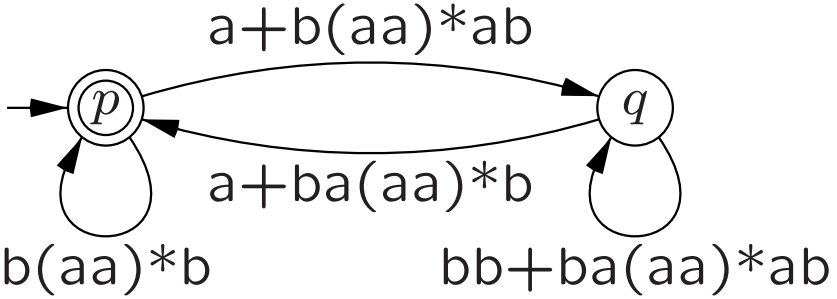



p
q
r
E_0
E_1
E_2
E_3


p
q
r
$E_0 + E_1 \cdot E_2^* \cdot E_3$


p
q
r
s
a
a
b
b
b
b
a
a


p
q
r
a
a
b
b
ab
ba
bb
aa


p
q
a+b(aa)*ab
a+ba(aa)*b
b(aa)*b
bb+ba(aa)*ab

long words 'loop' and can be pumped

$\forall$ for every regular language L

$\exists$ there exists a constant $n \geq 1$

such that

$\forall$ for every $z \in L$

with $|z| \geq n$

$\exists$ there exists a decomposition $z = uvw$

with $|uv| \leq n$, $|v| \geq 1$

such that

$\forall$ for all $i \geq 0$, $uv^i w \in L$

- clever idea, intuition

- formal construction, specification

- show it works, e.g., induction

once the idea is understood,
the other parts might be boring

but essential to test intuition

examples help to get the message

	RLIN REG	DPDA	CF PDAe	DLBA	MON LBA	REC	TYPE0 RE
intersection	+	−	−	+	+	+	+
complement	+	+	−	+	+	+	−
union	+	−	+	+	+	+	+
concatenation	+	−	+	+	+	+	+
star, plus	+	−	+	+	+	+	+
ε-free morphism	+	−	+	+	+	+	+
morphism	+	−	+	−	−	−	+
inverse morphism	+	+	+	+	+	+	+
intersect reg lang	+	+	+	+	+	+	+
mirror	+	−	+	+	+	+	+
	fAFL		fAFL	AFL	AFL	AFL	fAFL

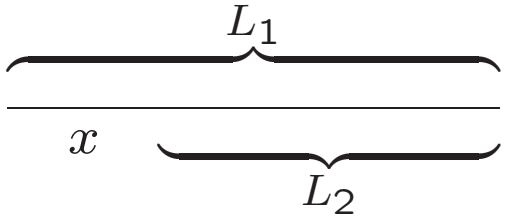
$\cap$, ^{c}, $\cup$ boolean operations

$\cup$, $\cdot$, * regular operations

h, h^{-1}, $\cap R$ (full) trio operations

3.1 Moore and Mealy machines

3.2 Quotients



$L_1, L_2 \subseteq \Sigma^*$

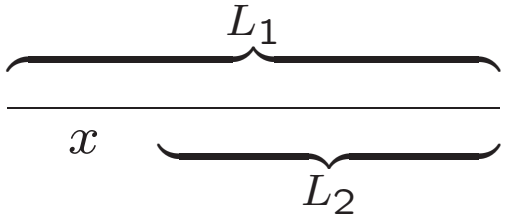
$L_1/L_2 = \{\, x \in \Sigma^* \mid xy \in L_1 \text{ for some } y \in L_2 \,\}$

Ex. $L_1 = a^+bc^+,\ L_2 = bc^+,\ L_3 = c^+$

$L_1/L_2 = a^+$

$L_1/L_3 = a^+bc^*$

Ex. $\text{Pref}(L) = L/\Sigma^*$



$L_1, L_2 \subseteq \Sigma^*$

$L_1/L_2 = \{\, x \in \Sigma^* \mid xy \in L_1 \text{ for some } y \in L_2 \,\}$

Ex. $L = \{\, a^{n^2} \mid n \geq 0 \,\}$

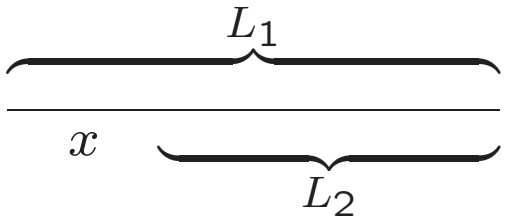
$L/L = \{\, a^{n^2-m^2} \mid n \geq m \geq 0 \,\} = a(aa)^* + (a^4)^*$

'$\subseteq$' $m^2 - n^2 = (m+n)(m-n)$

m	n	$m+n$	$m-n$	m^2-n^2
e	e	e	e	mult four
e	o	o	o	odd
o	e	o	o	odd
o	o	e	e	mult four

'$\supseteq$' $(k+1)^2 - k^2 = 2k+1$ odd

$(k+2)^2 - k^2 = 4k+4$ multiple of four



$L_1, L_2 \subseteq \Sigma^*$

$L_1/L_2 = \{\, x \in \Sigma^* \mid xy \in L_1 \text{ for some } y \in L_2 \,\}$

can 'hide' computations

Ex. $L_1 = \{\, a^{2n}cba^n \mid n \geq 1 \,\}\{\, ba^{2n}ba^n \mid n \geq 1 \,\}^*ba$

$L_2 = c \cdot \{\, ba^nba^n \mid n \geq 1 \,\}^*$

$L_1/L_2 = \{\, a^{2^n} \mid n \geq 1 \,\}$

$$a^{16}\, c\, ba^8\, ba^8\, ba^4\, ba^4\, ba^2\, ba^2\, ba\, ba$$

Thm. $L, R \subseteq \Sigma^*$ If R regular, then R/L regular.

$F' = \{\, q \in Q \mid \delta(q, y) \in F \text{ for some } y \in L \,\}$.

noncomputable ! (L arbitrary)

REG closed under quotient

REG / REG = REG

(see Ch.4) CF not closed, even

CF / CF = RE

CF / REG = CF

3.3 Morphisms and substitutions

'monoid'

$h : \Sigma \to \Delta^*$

$h : \Sigma^* \to \Delta^* \qquad h(xy) = h(x)h(y),\ h(\varepsilon) = \varepsilon$

$h : 2^{\Sigma^*} \to 2^{\Delta^*} \qquad h(L) = \bigcup_{x \in L} h(x)$

$0 \mapsto ab,\ 1 \mapsto ba,\ 2 \mapsto \varepsilon$

$00212 \mapsto ababba$

$\{\, 0^n 21^n \mid n \geq 0\} \mapsto \{(ab)^n (ba)^n \mid n \geq 0\,\}$

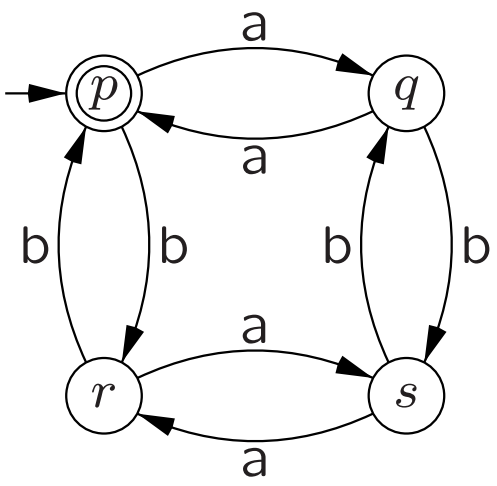
Thm. $h(K \cup L) = h(K) \cup h(L)$

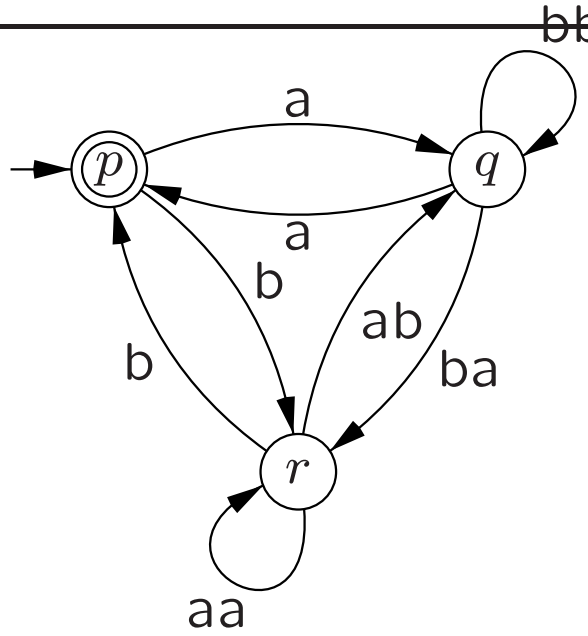
$h(K \cdot L) = h(K) \cdot h(L)$

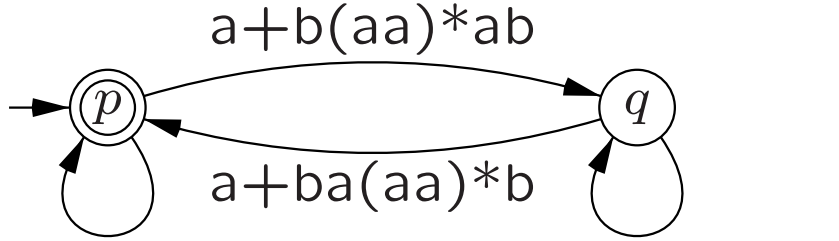
$h(K^*) = h(K)^*$

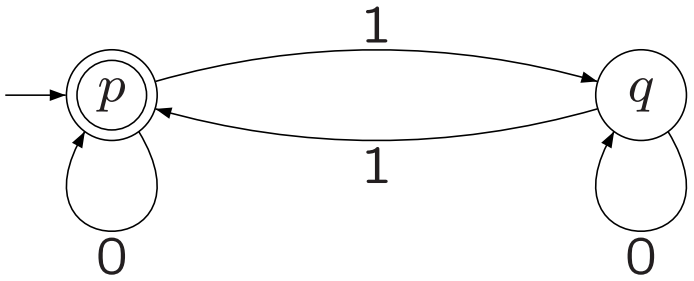
REG closed under morphisms

Theorem 3.3.1









$0 \mapsto$ b(aa)*b

$1 \mapsto$ a+b(aa)*ab

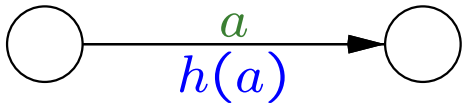
$K = \{x \in \{0,1\}^* \mid \#_1 x \text{ is even}\}$

$s(K) = \{x \in \{a,b\}^* \mid \#_a x, \#_b x \text{ are even}\}$

$h : \Sigma \to \Delta^*,\ K \subseteq \Delta^*$

$h^{-1}(K) = \{\, x \in \Sigma^* \mid h(x) \in K \,\}$

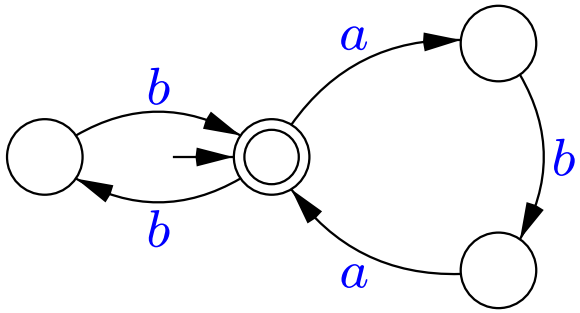
Thm. REG closed under inverse morphism

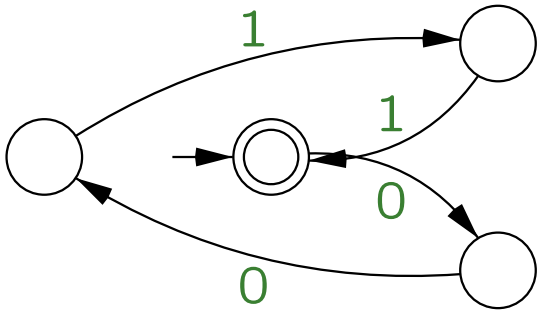


$\delta'(p, a) = \delta(p, h(a))$

$h : 0 \mapsto ab, 1 \mapsto ba$

$h^{-1}(\ \{bb, aba\}^*\ ) = \{0011\}^*$





$\mathsf{shuff}(K, L) = K \parallel L$ shuffle

$abb \parallel aca =$

$\{aabbca, aabcba, aabcab, aacabb, aacbab, aacbba, abbaca,$
$ababca, abacba, abacab, acabba, acabab, acaabb\}$

$x \parallel \varepsilon = \varepsilon \parallel x = \{x\}$

$ax \parallel by = a(x \parallel by) \cup b(ax \parallel y)$

$K \parallel L = \bigcup_{x \in K, y \in L} x \parallel y$

Thm. K, L regular, then $K \parallel L$ regular.

but where is that stated?

$abbba \parallel acac \ni abacbacba$

$K \parallel L$ using morphisms, intersection

copies of alphabet

Σ, $\Sigma_1 = \{ a_1 \mid a \in \Sigma \}$, $\Sigma_2 = \{ a_2 \mid a \in \Sigma \}$

$h_1 : \Sigma_1 \cup \Sigma_2 \to \Sigma^*$ $\quad a_1 \mapsto a \quad a_2 \mapsto \varepsilon$

$h_2 : \Sigma_1 \cup \Sigma_2 \to \Sigma^*$ $\quad a_1 \mapsto \varepsilon \quad a_2 \mapsto a$

$g : \Sigma_1 \cup \Sigma_2 \to \Sigma^*$ $\quad a_1 \mapsto a \quad a_2 \mapsto a$

$$\begin{array}{ccccc} abbba & \overset{h_1}{\leftarrow} & a_1b_1a_2c_2b_1a_2c_2b_1a_1 & \overset{h_2}{\rightarrow} & acac \\ \in K & & \downarrow g & & \in L \\ & & abacbacba & & \end{array}$$

$K \parallel L = g(\, h_1^{-1}(K) \cap h_2^{-1}(L) \,)$

3.4 Advanced closure properties of regular languages

$\frac{1}{2}L = \{\, x \in \Sigma^* \mid xy \in L \text{ for } y \text{ with } |y| = |x| \,\}.$

Thm. L regular, then $\frac{1}{2}L$ regular

guess middle state, simulate halves in parallel

$Q' = \{q_0'\} \cup Q \times Q \times Q$ middle, 1st, 2nd

$\delta'(q_0', \varepsilon) = \{[q, q_0, q] \mid q \in Q\}$ ε-move

$\delta'([q, p, r], a) = \{[q, \delta(p, a), \delta(r, b)] \mid b \in \Sigma\}$

$F' = \{[q, q, p] \mid q \in Q, p \in F\}$

$\sqrt{L} = \{\, x \in \Sigma^* \mid xx \in L \,\}.$

$\text{cut}_f L = \{\, x \mid xy \in L \text{ for } y \text{ with } |y| = f(|x|) \,\}.$

$f(n) = n$ $\quad \frac{1}{2}L$

$f(n) = 2^n$ $\quad \log L$ $\quad$ p.76

$f(n) = n^2$

which f ?

see: *transition matrix* (Ch. 3.8)

$$\text{cyc}(L) = \{\, x_1x_2 \mid x_2x_1 \in L \,\}.$$

Thm. If L is regular, then so is $\text{cyc}(L)$

guess middle,

simulate halves in opposite order

$$Q' = \{q_0'\} \cup Q \times Q \times \{0,1\}$$

middle, state, phase

$$\delta'(q_0', \varepsilon) = \{\, [q,q,0] \mid q \in Q\}$$ ε-move

$$\delta'([q,p,i],a) = \{\, [q,\delta(p,a),i] \,\}$$

$$\delta'([q,q_f,0],\varepsilon) = \{\, [q,q_0,1] \mid q_f \in Q \,\}$$

$$F' = \{\, [q,q,1] \mid q \in Q \,\}$$

▷

(1) Note that the construction introduces ε-moves.

(2) Is this a proof?

The slide gives the intuition ('guess middle') and the formal construction ($\delta'([q, q_f, 0], \varepsilon) = \{\ [q, q_0, 1] \mid q_f \in Q\ \}$).

What is missing is the (formal) argument that the construction works, the correctness proof, i.e., that starting with automaton $\mathcal{A}$ for L the constructed automaton $\mathcal{A}'$ actually accepts $\mathrm{cyc}(L)$.

Thus, if there is a computation for xy on $\mathcal{A}$, then there is a computation for yx on $\mathcal{A}'$ (and vice versa).

In informal notation,

$q_0 \overset{x}{\rightsquigarrow} p \overset{y}{\rightsquigarrow} q_f$ in $\mathcal{A}$, then
$q'_0 \overset{\varepsilon}{\rightsquigarrow} [p, p, 0] \overset{y}{\rightsquigarrow} [p, p_f, 0] \overset{\varepsilon}{\rightsquigarrow}$
$[p, q_0, 1] \overset{x}{\rightsquigarrow} [p, p, 1]$ in $\mathcal{A}'$.

For the reverse implication we need that indeed all computations in $\mathcal{A}'$ are of this form.

◁

3.5 Transducers

FST $\sim$ finite state automaton with output

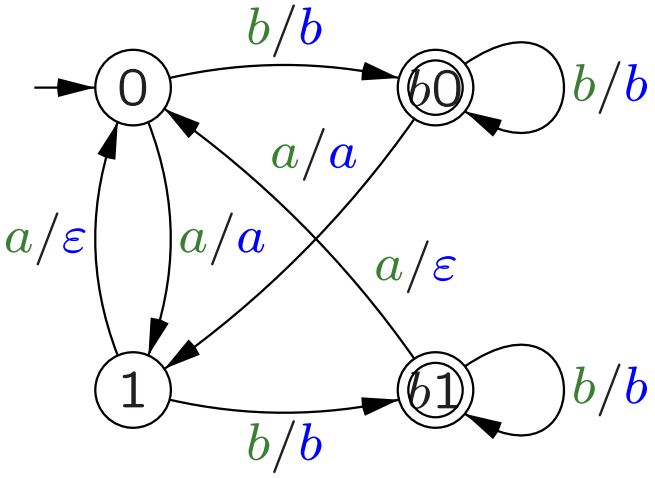
$\mathcal{A} = (Q, \Sigma, \Delta, S, q_{in}, F)$

$S \subseteq Q \times \Sigma^* \times \Delta^* \times Q$

$\ldots (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \ldots$

$T(\mathcal{A}) \subseteq \Sigma^* \times \Delta^*$ transduction (translation)

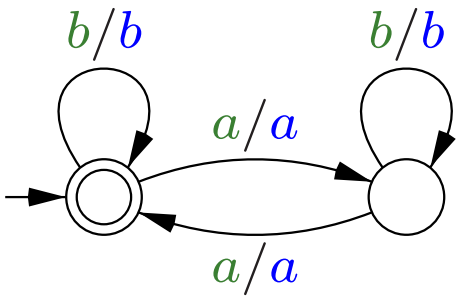
$x \to_{\mathcal{A}} y$ rational relation



$K \subseteq \Sigma^*$

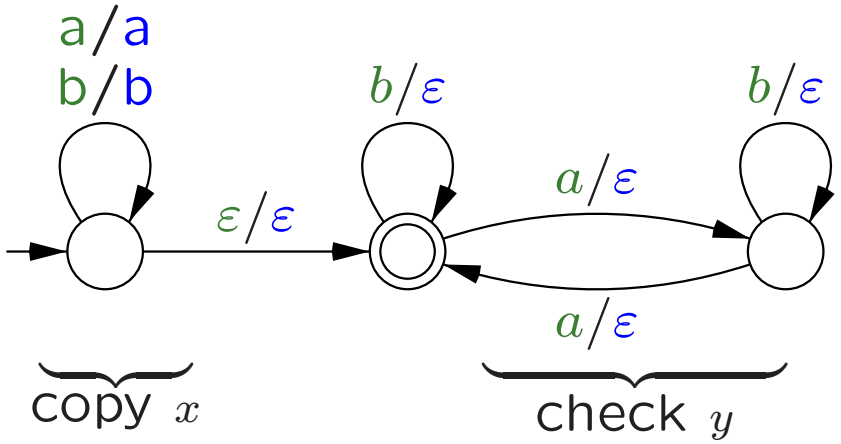
$T(K) = \{\, y \in \Delta^* \mid (x, y) \in T(\mathcal{A}), x \in K \,\}$

erase every 2nd a (keeping words ending in b)



* intersection, quotient, concatenation with regular languages
* morphism, inverse morphism
* prefix, suffix
* . . . erasing every second a

$T(K) = K \cap \{\, x \mid \#_a x \text{ even} \,\}$



$T(K) = \{\, x \mid xy \in K \text{ and } \#_a y \text{ even} \,\}$

$K, L \subseteq \Sigma^*$

$\Sigma' = \{a' \mid a \in \Sigma\} \qquad \Sigma \cap \Sigma' = \varnothing$

$f : \Sigma \cup \Sigma' \to \Sigma \qquad f(a) = f(a') = a$

f^{-1} non-det colouring

$h : \Sigma \to \Sigma' \qquad h(a) = a'$

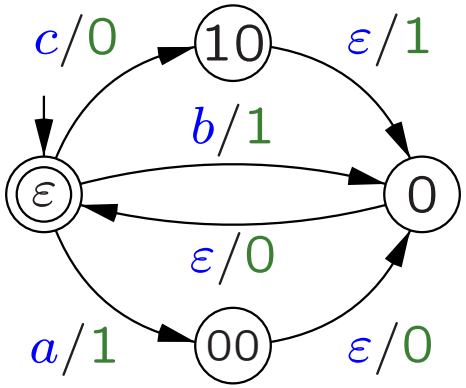
$g : \Sigma \cup \Sigma' \to \Sigma \qquad g(a) = a,\ g(a') = \varepsilon$

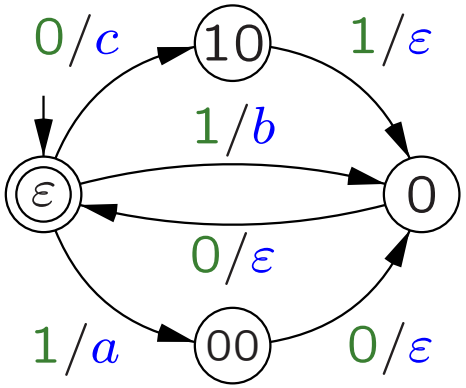
$K/L = g(\ f^{-1}(K) \cap \Sigma^* \cdot h(L)\ )$

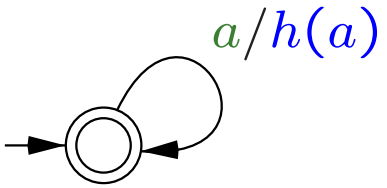
basic *full trio* operations:

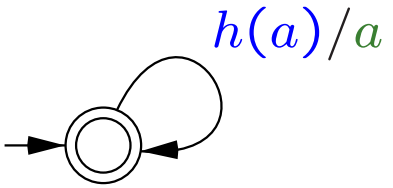
- morphism
- inverse morphism
- intersection regular

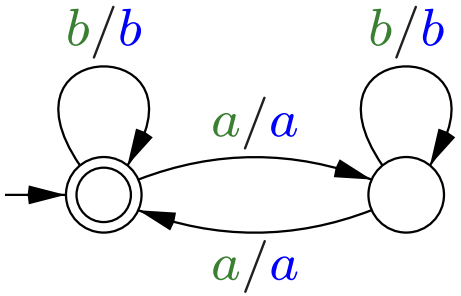
$$h : \begin{cases} a & \mapsto & 100 \\ b & \mapsto & 10 \\ c & \mapsto & 010 \end{cases}$$











- every 'basic' full trio operation is FST
- FST's are closed under composition

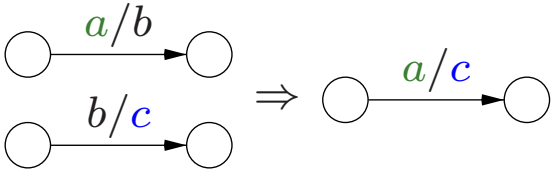
$\Rightarrow$ sequence of full trio op's is FST

FST $\mathcal{A}_i = (Q, \Sigma_i, \Sigma_{i+1}, S_i, q_{io}, F_i)$

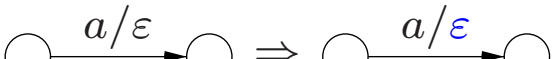
$T(\mathcal{A}_1)T(\mathcal{A}_2) \Rightarrow$ FST $\mathcal{A}' = (Q', \Sigma_1, \Sigma_3, S', q'_0, F')$

formally – $Q' = Q_1 \times Q_2$
– $q'_{in} = \langle q_{10}, q_{20} \rangle$
– $F' = F_1 \times F_2$, and
– S' is defined by

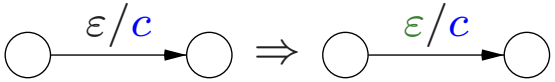
if $(p_1, a, b, q_1) \in S_1$, and $(p_2, b, c, q_2) \in S_2$
(with $b \neq \varepsilon$)
then $(\langle p_1, p_2 \rangle, a, c, \langle q_1, q_2 \rangle) \in S'$



if $(p_1, a, \varepsilon, q_1) \in S_1$ and $p \in Q_2$,
then $(\langle p_1, p \rangle, a, \varepsilon, \langle q_1, p \rangle,) \in S'$



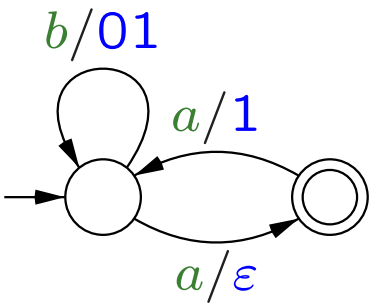
if $p \in Q_1$ and $(p_2, \varepsilon, c, q_2) \in S_2$,
then $(\langle p, p_2 \rangle, \varepsilon, c, \langle p, q_2 \rangle) \in S'$



'implicit $(p, \varepsilon, \varepsilon, p)$'

every full trio operation is a fs transduction

Thm. every FST is composition of full trio op's



$R_{\mathcal{M}}$ regular language over 'transitions'

$\{\ a{:}\varepsilon,\ a{:}1,\ b{:}01\ \}$

h and g select input and output

K	$\ni$	b	b	a	a	b	a
			$\uparrow h$				
$R_{\mathcal{M}}$	$\ni$	$b{:}01$	$b{:}01$	$a{:}\varepsilon$	$a{:}1$	$b{:}01$	$a{:}\varepsilon$
			$\downarrow g$				
$T_{\mathcal{M}}(K)$	$\ni$	01	01	ε	1	01	ε

$$\underset{\text{input}}{x} \xleftarrow{h} \underset{\text{computation}}{R} \xrightarrow{g} \underset{\text{output}}{y}$$

$$\boxed{T_{\mathcal{M}}(K) = g(\ h^{-1}(K)\ \cap\ R_{\mathcal{M}}\ )}$$

closure properties

trio: (≡ faithful cone)
morphism, ε-free morphism, intersection regular

full trio: (≡ cone)
. . . , (arbitrary) morphism, . . .

[full] semi-AFL: [full] trio & union

[full] AFL: [full] semi-AFL & concatenation, Kleene plus

3.6 Two-way finite automata

like TM may move in both directions,
no writing, tape bounded

$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$

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

▷tape markers◁ (!)

$\delta(\cdot, \triangleright) = (\cdot, R), \quad \delta(\cdot, \triangleleft) = (\cdot, L)$

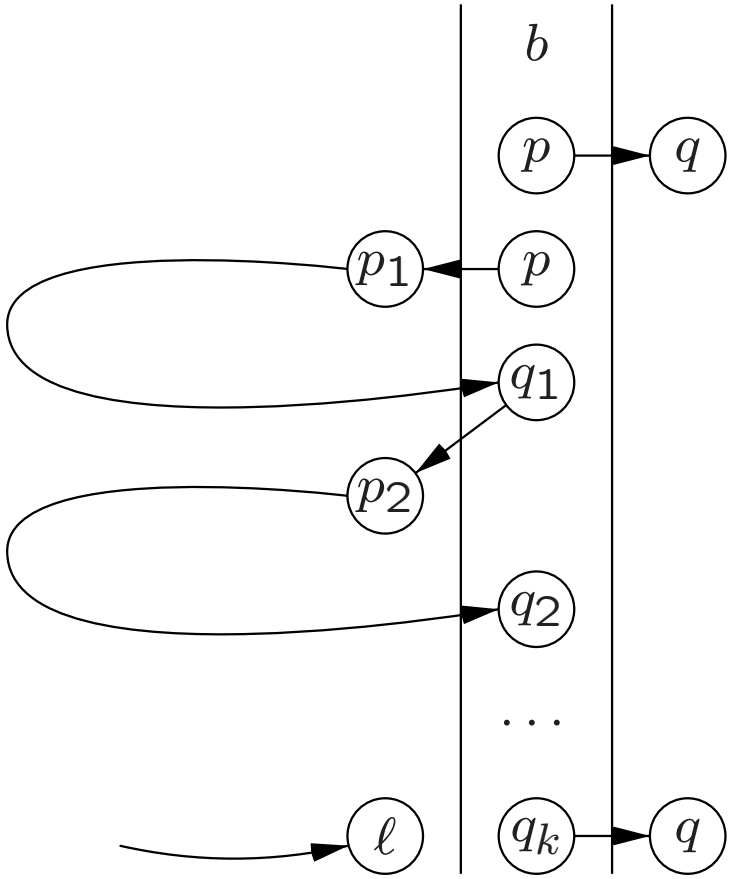
configuration $\triangleright\Sigma^* Q \Sigma^* \triangleleft \cup Q \triangleright \Sigma^* \triangleleft$

$wqax \vdash wapx$ when $\delta(q, a) = (p, R)$ move
$waqx \vdash wpax$ $\delta(q, a) = (p, L)$

infinite loops possible !

$L(\mathcal{M}) = \{ w \in \Sigma^* \mid q_0 \triangleright w \triangleleft \vdash^* \triangleright wp \triangleleft, p \in F \}$

Shefferdson [1959] 2DFA $\subseteq$ DFA



keep track of 'excursions' to the left

$\tau : Q \cup \{\bar{q}\} \to Q \cup \{\ell\}$ $\bar{q}$ final, ℓ for loop

updating τ τ_x to τ_{xb}

$\delta(p, b) = (q, R)$ then $\tau_{xb}(p) = q$

$\delta(p, b) = (p_1, L)$, $\tau_x(p_1) = q_1$,
$\delta(q_1, b) = (p_2, L)$,. . . , $\tau_x(p_k) = q_k$

until one of the following occurs

if $q_k = \ell$ then $\tau_{xb}(p) = \ell$

if $\delta(q_k, b) = (q, R)$ then $\tau_{xb}(p) = q$

if $q_k = q_i$ or $q_k = p$ then $\tau_{xb}(p) = \ell$

$\tau_x(\bar{q}) = \delta(q_0, x)$

Theorem 3.6.1

$\text{root}(L) = \{\, w \in \Sigma^* \mid w^n \in L \text{ for some } n \geq 1 \,\}$

Thm. root(L) is regular (for regular L)

simulate $\mathcal{M}$ for L on $\triangleright w \triangleleft$
accept right when $\mathcal{M}$ accepts
otherwise continue left at state reached

also $\frac{1}{2}(L)$ can be solved this way

and $\{\, w \in \Sigma^* \mid w^n \in L$ for *all* $n \geq 1 \,\}$?

3.7 The transformation automaton
3.8 Automata, graphs, and Boolean matrices

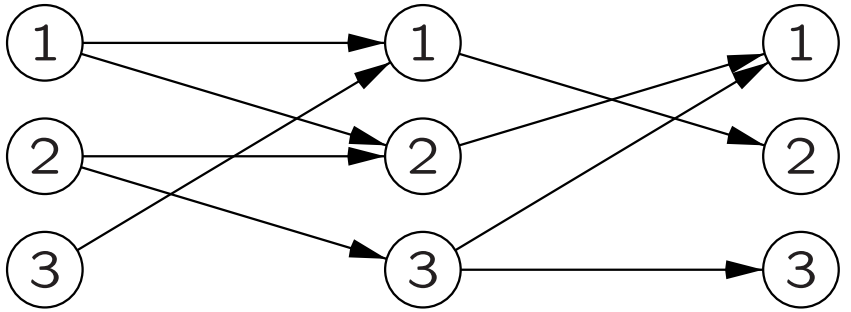
$C = AB$

$C_{ij} = \sum_{i=1}^{n} A_{ik} B_{kj}$

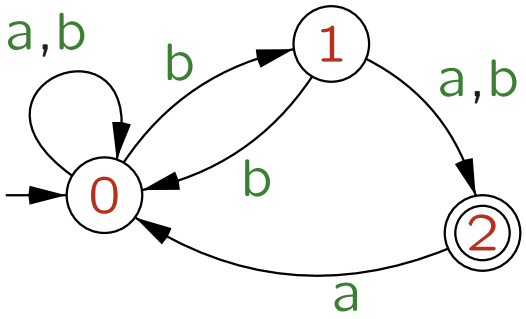
number of connections

Boolean

$C_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}$ exists connection



$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



$Q = \{q_0, q_1, \ldots, q_{n-1}\}$ (ordered)

M_a Boolean matrix

$(M_a)_{ij} = 1$ iff $\delta(q_i, a) \ni q_j$

$$M_{\mathsf{a}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad M_{\mathsf{b}} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

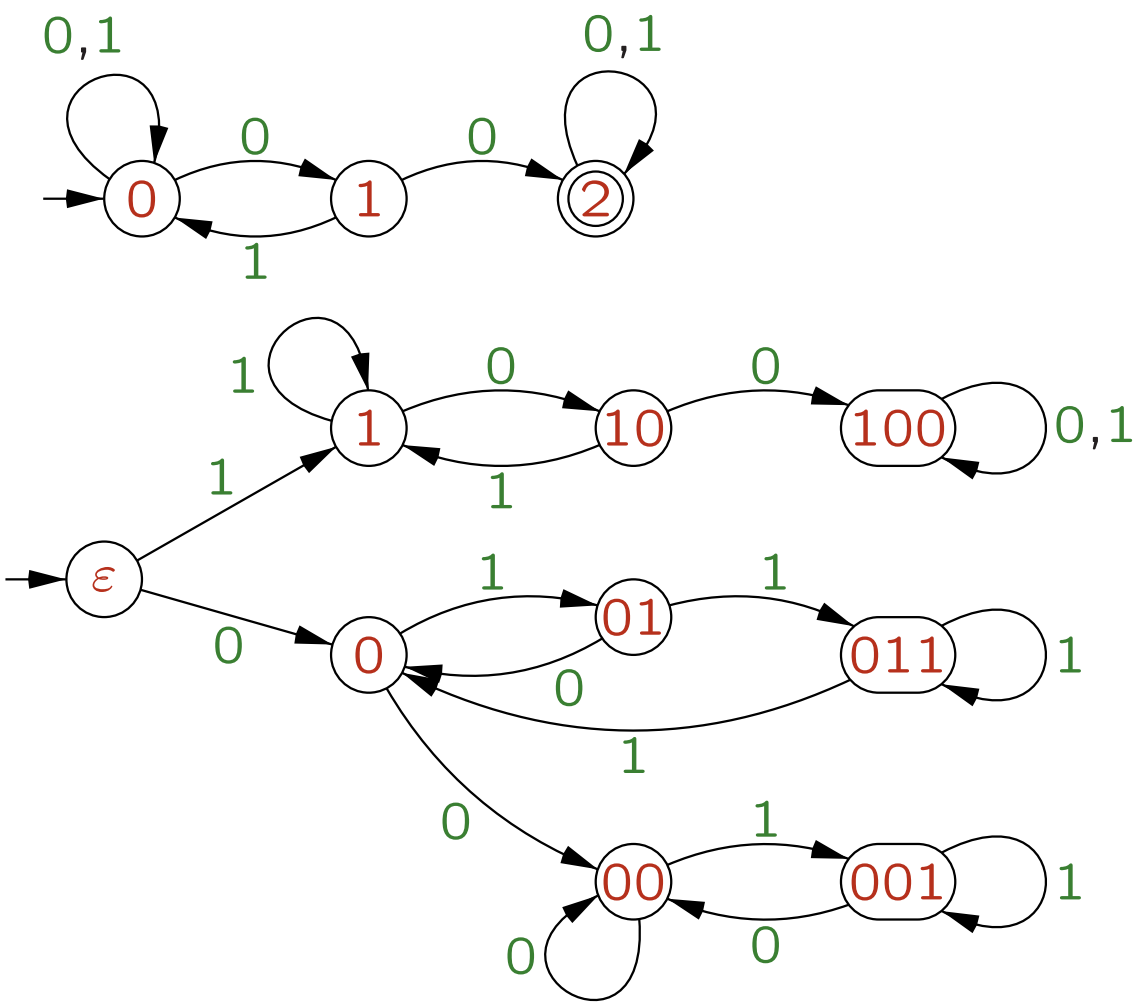
Prop. $(M_w)_{ij} = 1$ iff $\delta(q_i, w) \ni q_j$

$$M_{\mathsf{abb}} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Thm. for $w = a_1 a_2 \ldots a_t$, $a_i \in \Sigma$

$$M_w = M_{a_1} M_{a_2} \cdot \ldots \cdot M_{a_t}$$

Cor. $M_{xy} = M_x M_y$



From diagram $M_{000} = M_{00}$, etc.

transformation automaton

$$M_\varepsilon = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_0 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{00} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{01} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{10} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{001} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{011} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_{100} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

characteristic vectors

$u_0 = [1, 0, 0, \ldots, 0]$ (row)

$(u_F)_i = 1$ iff $q_i \in F$ (column)

$(M_w)_{ij} = 1$ iff $\delta(q_i, w) \ni q_j$

Thm. $x \in L(\mathcal{A})$ iff $u_0 \, M_x \, u_F = 1$

matrix represents computation

nondeterministic case

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$

$(M_a)_{ij} = 1$ iff $\delta(q_i, a) \ni q_j$

$w = a_1 a_2 \ldots a_t$

$M_w = M_{a_1} M_{a_2} \cdot \ldots \cdot M_{a_t}$

$\mathcal{A} = (Q', \Sigma, \delta', q_0', -)$

transformation automaton

$Q' = \{0,1\}^{Q \times Q}$ 0, 1-matrices

$q_0' = I$ identity matrix

$\delta(M, a) = M \cdot M_a$

no final states specified

Thm. $\delta'(I, w) = M$, then $M = M_w$

i.e., $(M)_{ij} = 1$ iff $\delta(q_i, w) \ni q_j$

$\sqrt{L} = \{\, x \in \Sigma^* \mid xx \in L \,\}.$

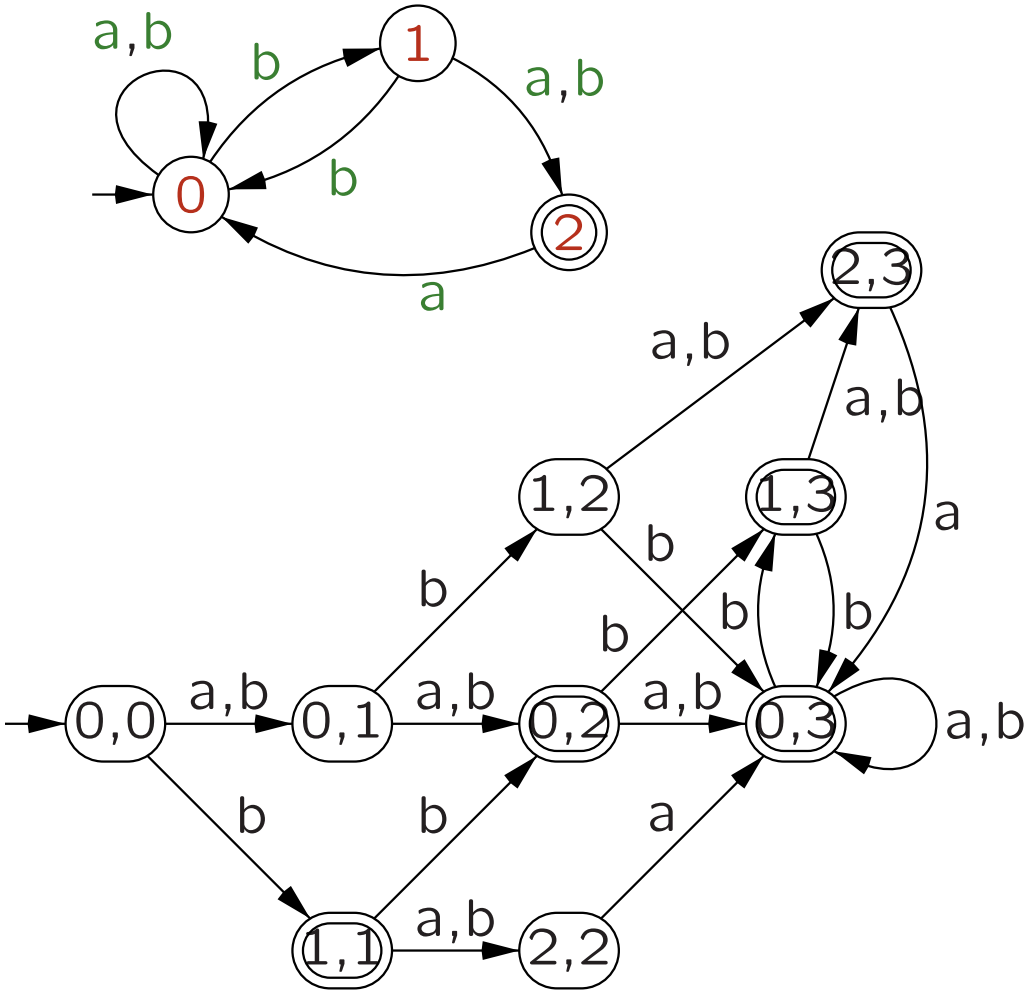
states $Q' = \{0,1\}^{Q \times Q}$ (the M_x's)

M_x after reading x
final: $(M_x)_{q_0 p} = 1$ for some $p \in Q$ [unique]
and $(M_x)_{pq} = 1$ for some $q \in F$

$\frac{1}{2}L = \{\, x \in \Sigma^* \mid xy \in L \text{ for } y \text{ with } |y| = |x| \,\}.$

$(p,q) \in M^k$ iff $\delta(p,u) = q$ for some u, $|u| = k$.
$M^{k+1} = M^k M$

Prop. $\log L = \{\, x \mid xy \in L \text{ for } y \text{ with } |y| = 2^{|x|} \,\}.$
$M^{2^k} = (M^{2^{k-1}})^2$



product: aut $\times$

transformation aut

$(p, M^k) \xrightarrow{a} (\delta(p,a), M^{k+1})$

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$M^2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$$M^k = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (k \geq 3)$$

$M^k[i,j] = 1$ *iff*

length k path from i to j

monoid $(M, \circ, 1)$

- closed $a \circ b \in M$
- associative $(a \circ b) \circ c = a \circ (b \circ c)$
- identity $a \circ 1 = 1 \circ a = a$

$(\Sigma^*, \cdot, \varepsilon)$ strings

$(\mathbb{Z}^{n \times n}, \circ, I)$ $n \times n$-matrices
$(\{0,1\}^{n \times n}, \circ, I)$ Boolean matrices:
finite monoid

monoid morphism $h : (M, \circ, 1) \to (M', \circ', 1')$
$h : M \to M'$

- $h(a) \circ' h(b) = h(a \circ b)$
- $h(1) = 1'$

*

Def. $L \subseteq \Sigma^*$ recognizable iff
finite monoid $(M, \circ, 1)$,
monoid morphism $h : \Sigma^* \to M$
$S \subseteq M$ such that $L = h^{-1}(S)$

Cor. $M_{xy} = M_x M_y$

automaton as monoid

$\mu : \Sigma^* \to \{0,1\}^{Q \times Q}$
$x \mapsto M_x$ is a monoid morphism

Thm. REC = REG (for strings)

monoid as automaton

$\mathcal{A}_M = (M, \Sigma, \delta, 1, S)$
$\delta(m, a) = m \circ h(a) \qquad m \in M,\ a \in \Sigma$

$x \in L(\mathcal{A}_M)$ iff $\delta(1, x) \in S$ iff $h(x) = 1 \circ h(x) \in S$
iff $x \in h^{-1}(S)$

*

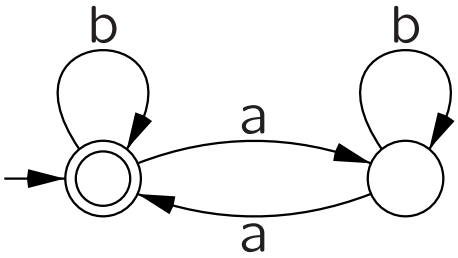
3.9 The Myhill-Nerode theorem

equivalence relation

- reflexive xRx for all x
- symmetric xRy implies yRx
- transitive xRy and yRz imply xRz

equivalence class $E = \{y \in S \mid xRy\}$

index of R



DFA $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$

ending in the same state

$xR_{\mathcal{M}}y$ iff $\delta(q_0, x) = \delta(q_0, y)$

baabb $R_{\mathcal{M}}$ ε

baabba $R_{\mathcal{M}}$ a

equivalence relation on Σ^*

- finite index $|Q|$
- right invariant $xR_{\mathcal{M}}y$ implies $xzR_{\mathcal{M}}yz$

right congruence

- $L(M)$ union of equivalence classes

R_M *saturates* L

$L \subseteq \Sigma^*$

xR_Ly when, for all u, ($xu \in L$ iff $yu \in L$)

equivalence relation on Σ^*

- index may be infinite
- right invariant xR_Ly implies xzR_Lyz
- L union of equivalence classes

R_1, R_2 equivalence relations

R_1 refinement of R_2: $R_1 \subseteq R_2$

Lem. L union of some classes of right-invariant equivalence relation E.
Then E refinement of R_L

Prf. xEy (right-invariant) $\Rightarrow xzEyz$ for all z (union of classes) $\Rightarrow xz \in L$ iff $yz \in L$ for all $z \Rightarrow xR_Ly$

$L \subseteq \Sigma^*$

xR_Ly when, for all u, ($xu \in L$ iff $yu \in L$)

xR_Ly iff $x^{-1}L = y^{-1}L$

$x^{-1}L = \{\, u \mid xu \in L \,\}$

$x^{-1}L$ may contain

ε

$\{a,b\}^*a$

even b's (≥ 2)

odd b's

	ε	a	b	bb
ε	$\surd$	$\surd$	$-$	$\surd$
b	$-$	$\surd$	$\surd$	$-$
ba	$\surd$	$\surd$	$\surd$	$-$

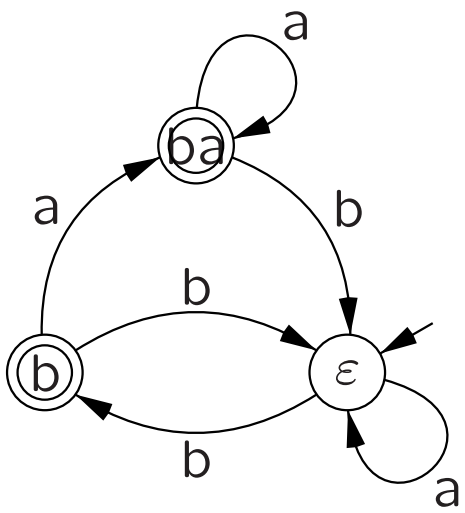
(x,y): $xy \in L$

$L = \{\, x \in \{a,b\}^* \mid x$ ends in a or even b's $\,\}$

[ε] even number b's [a]=[ε], [b]

[b] odd b's, ending in b [ba], [bb]=[ε]

[ba] odd b's, ending in a [baa]=[ba], [bab]=[ε]



Thm. $L \subseteq \Sigma^*$. equivalent:

a. L regular

b. L is union of equivalence classes of right-invariant equivalence relation E of finite index

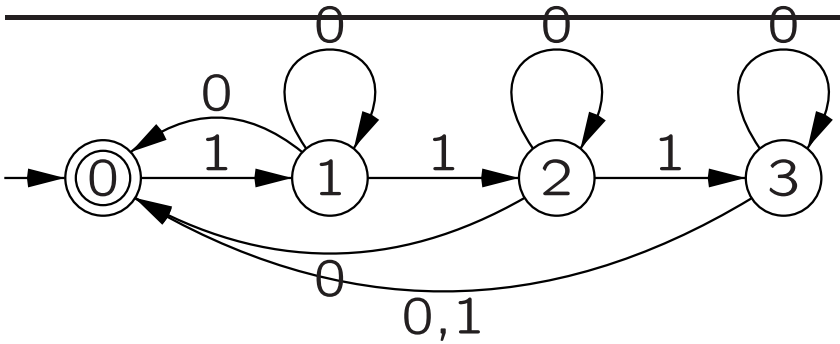
c. R_L has finite index

a.⇒b. $R_{\mathcal{M}}$ for automaton $\mathcal{M}$

b.⇒c. E is a refinement of R_L. index $R_L \leq$ index E

c.⇒a. use equivalence classes as states
$\delta([x], a) = [xa]$

automaton is 'inside' the language

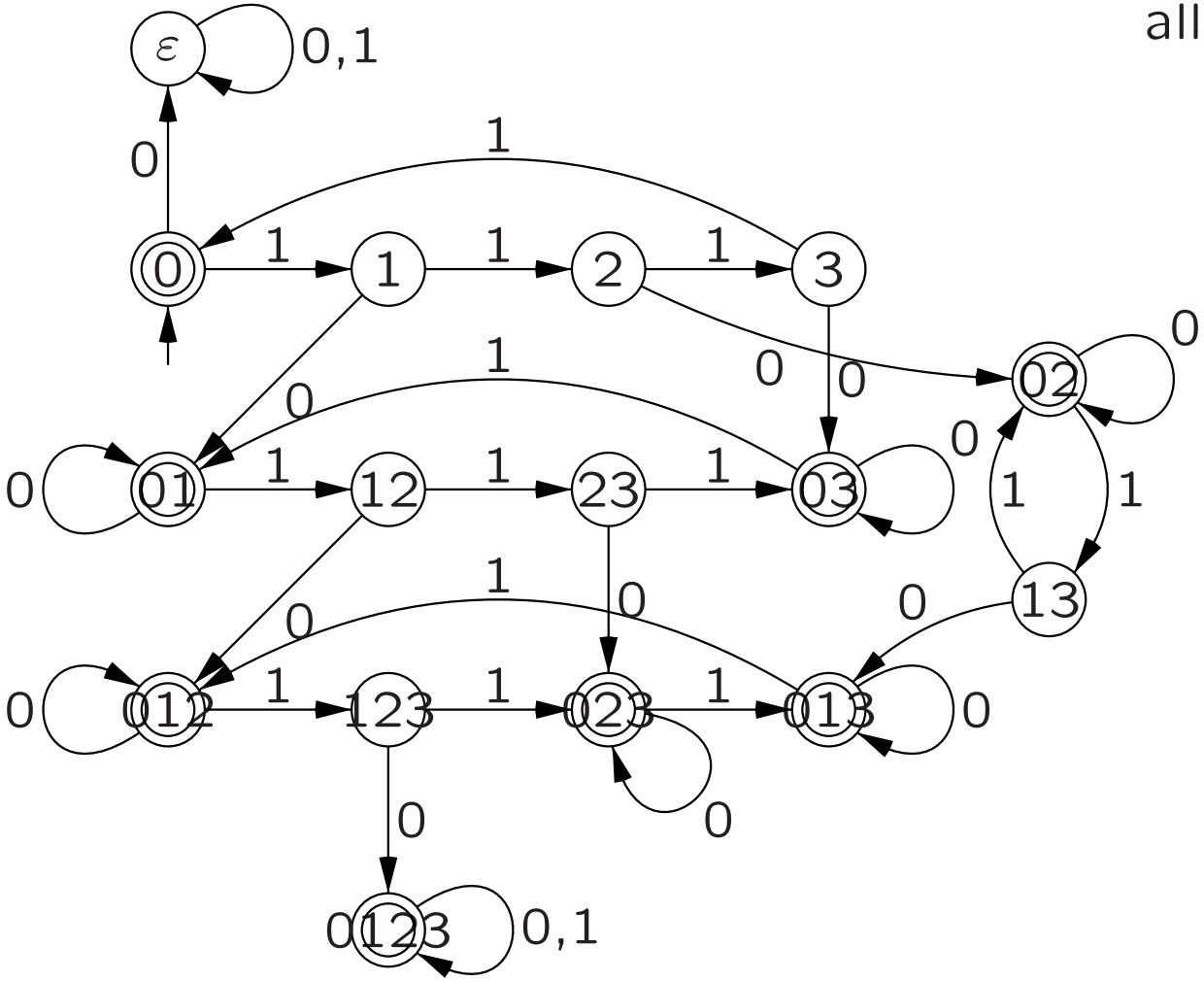


n state nfa

2^n state dfa

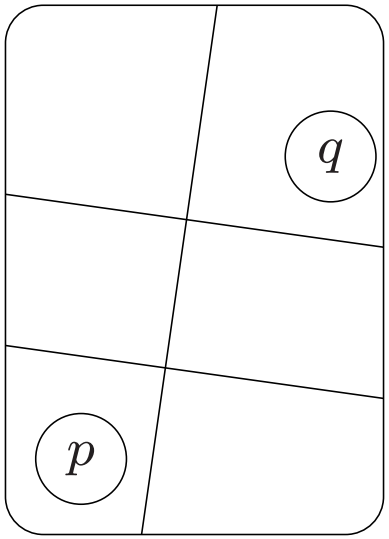
all reachable

all nonequivalent



Theorem 3.9.6

3.10 Minimization of finite automata



Σ^*/R_L

$L \subseteq \Sigma^*$

xR_Ly when, for all u, ($xu \in L$ iff $yu \in L$)

$xR_{\mathcal{M}}y$ when $\delta(q_0, x) = \delta(q_0, y)$

$xR_{\mathcal{M}}y$ then xR_Ly

Myhill-Nerode: R_L-classes $\rightsquigarrow$ automaton

$\delta([x], a) = [xa]$

Thm. unique minimal (det) automaton for L

$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$

$\mu : Q \to \Sigma^*/R_L$

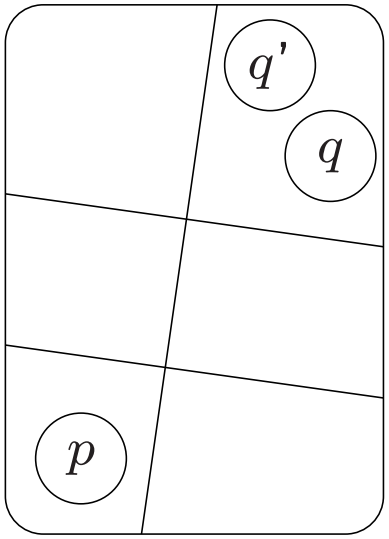
$q \mapsto [x]$, such that $\delta(q_0, x) = q$

well-defined ($R_{\mathcal{M}}$ refines R_L)

surjective ($q = \delta(q_0, x) \mapsto [x]$)

injective (surjective, same number states)

respects transitions (right invariant)



Σ^*/R_L

$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ dfa for L

xR_Ly when, for all u, ($xu \in L$ iff $yu \in L$)

$xR_{\mathcal{M}}y$ when $\delta(q_0, x) = \delta(q_0, y)$

$xR_{\mathcal{M}}y$ then xR_Ly

$p \equiv q$ indistinguishable

$\delta(p, z) \in F$ iff $\delta(q, z) \in F$

$\mu : Q \to \Sigma^*/R_L$

$q \mapsto [x]$, such that $\delta(q_0, x) = q$

well-defined ($R_{\mathcal{M}}$ refines R_L)

surjective ($p = \delta(q_0, x) \mapsto [x]$)

may not be injective

respects transitions (right invariant)

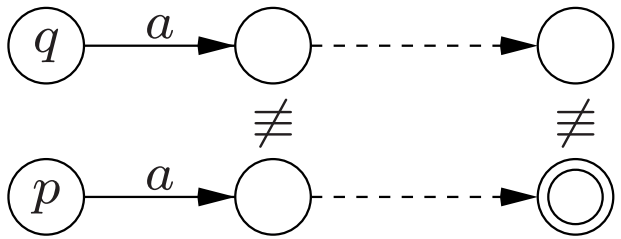
$p = \delta(q_0, x)$, $q = \delta(q_0, y)$

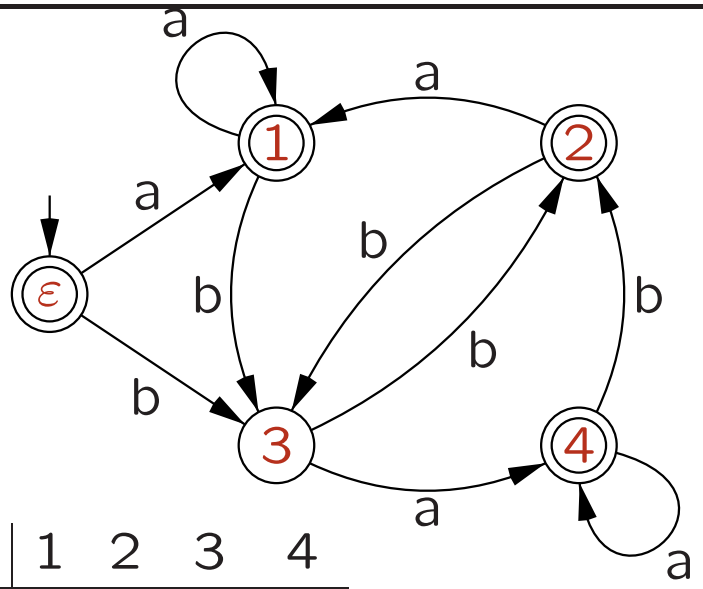
xR_Ly (or $[x] = [y]$) iff $p \equiv q$

▷ find indistinguishable states $\equiv$

0. $U\{p,q\} = 0$ for all $p, q \in Q$
1. $U\{p,q\} = 1$ for all $p \in F$, $q \in Q - F$
3. repeat
5. $\quad T = U$
8. $\quad$ if $T\{\delta(p,a), \delta(q,a)\} = 1$ then $U\{p,q\} = 1$
   for all $a \in \Sigma$, all p, q with $T\{p,q\} = 0$
. until no changes
9. return(U)

$U\{p,q\} = 1$ iff $p \not\equiv q$





	1	2	3	4
ε	.	.	X	.
1		.	X	.
2			X	.
3				X

$\delta(\varepsilon, b) = 3,\ \delta(4, b) = 2$

	1	2	3	4
ε	.	.	X	X
1		.	X	X
2			X	X
3				X

algorithm	worst-case	practice	implementation
NAIVE	$\mathcal{O}(n^3)$	reasonable	easy
MINIMIZE	$\mathcal{O}(n^2)$	good	moderate
FAST	$\mathcal{O}(n \log n)$	very good	difficult
BRZOZOWSKI	$\mathcal{O}(n2^{2n})$	often good	easy

Figure 3.18

Bulletin of the EATCS 37, 1989, p.130

MINIMIZATION BY REVERSAL IS NOT NEW

J.A. Brzozowski
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

I read with interest W. Brauer's note (Bulletin of EATCS, No. 35, June 1988, pp 113-116), about an algorithm, attributed to van de Snepscheut, for minimizing finite automata. I wholeheartedly agree with Dr. Brauer that the algorithm is simple and elegant; in fact, I considered it to be "rather surprising" when I discovered it in 1962. The key result is Theorem 13 in:

J.A. Brzozowski, "Canonical Regular Expressions and Minimal State Graphs for Definite Events", pp. 529-561 in *Mathematical Theory of Automata*, Vol. 12, of the MRI Symposia Series, Polytechnic Press of the Polytechnic Institute of Brooklyn; 1963.

The algorithm is also published in my Ph.D. Thesis:

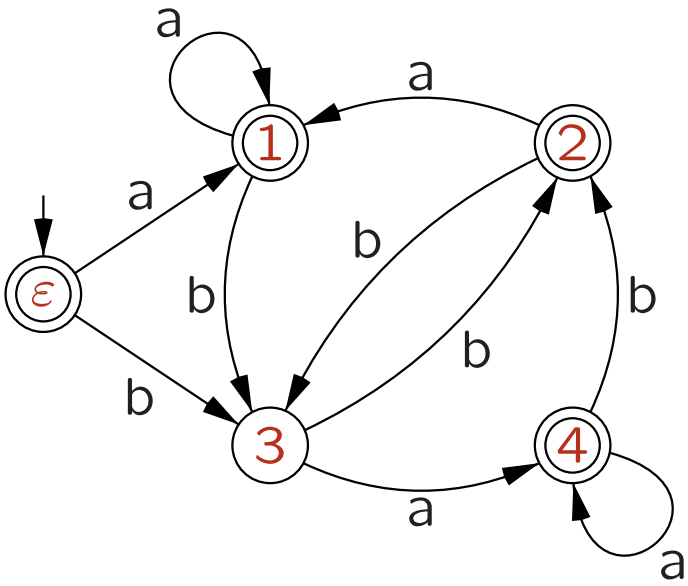
Regular Expression Techniques for Sequential Circuits, Ph.D. Dissertation, department of Electrical Engineering, Princeton University, Princeton, New Jersey; June 1962.

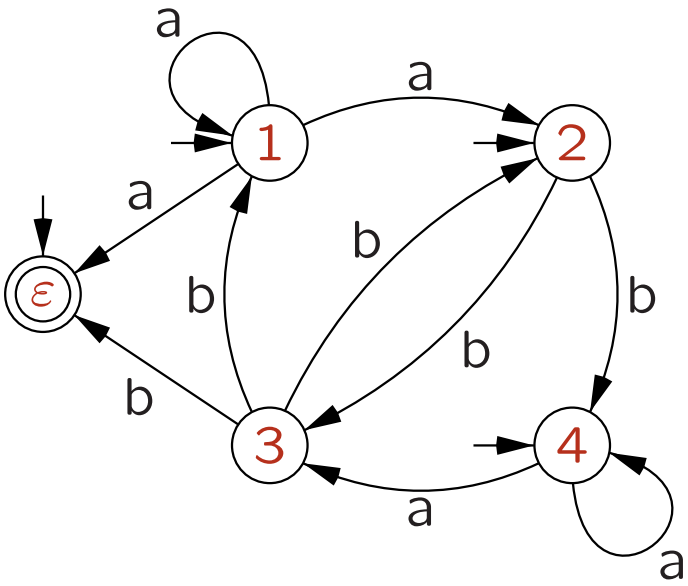
$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$

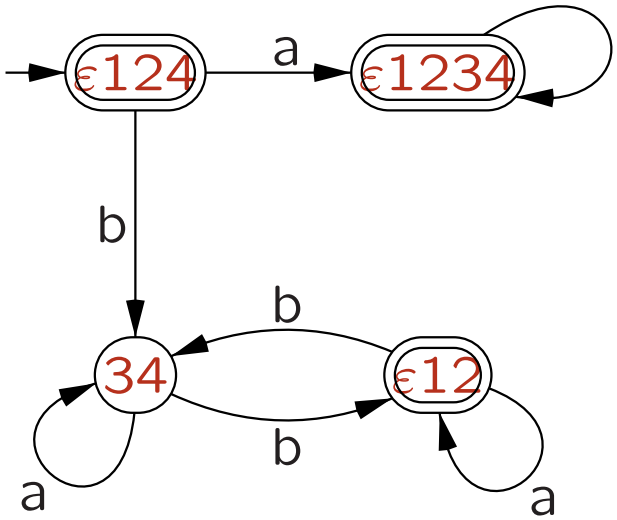
$R(\mathcal{M}) = (Q, \Sigma, \delta^R, F, q_0)$ reversing arrows
$q \in \delta(p, a)$ iff $p \in \delta^R(q, a)$
multiple initial states

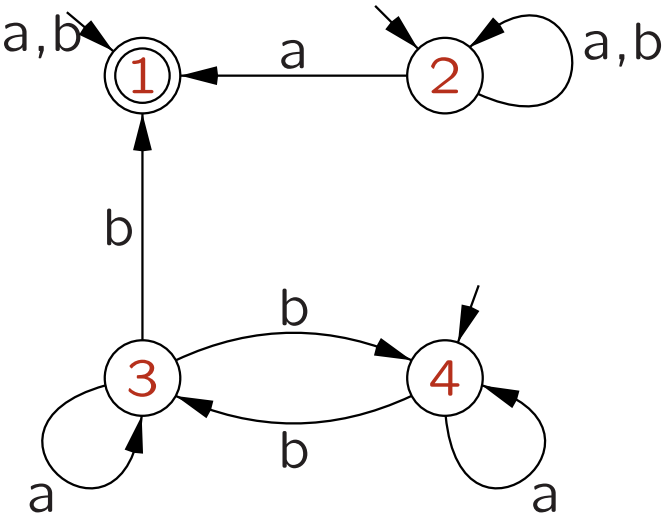
$S(\mathcal{M})$ subset, only *reachable* states

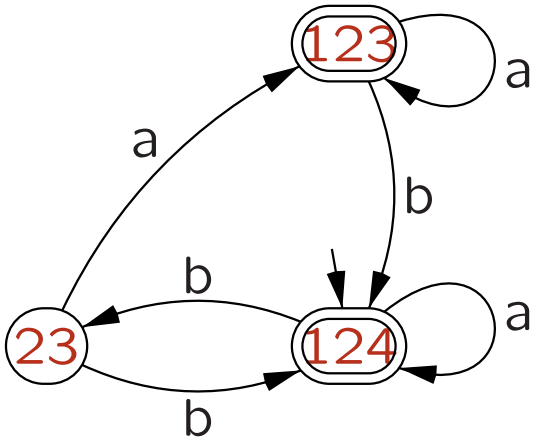
Thm. $S(R(S(R(\mathcal{M}))))$ minimal DFA equivalent $\mathcal{M}$











even number b's or ending in a

$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$
$R(\mathcal{M}) = (Q, \Sigma, \delta^R, F, q_0)$ reversing arrows
$S(R(\mathcal{M})) = (Q'', \Sigma, \delta'', q_0'', F'')$

$q \in \delta''(X, w^R)$ iff $\delta(q, w) \in X$

Lem. $\mathcal{M}$ DFA, only reachable states.
$S(R(\mathcal{M}))$ minimal DFA for L^R

$A, B \in Q''$: $A \equiv B$ then $A = B$

$p \in A$ then $\delta(q_0, w) = p$ some $w \in \Sigma^*$
so $\delta''(A, w^R) \ni q_0 \Leftrightarrow \delta''(A, w^R) \in F''$
$A \equiv B$ so $\delta''(B, w^R) \in F'' \Leftrightarrow \delta''(B, w^R) \ni q_0$
so $p = \delta(q_0, w) \in B$

hence $A \subseteq B$ (for all p)

hence $A = B$ (symmetric)

3.11 State complexity

3.12 Partial orders and regular languages

motivation:

for (*any*) language, consider the language of all its *subsequences*

surprise:

it will be regular

$\{ a^n b^{n^2} \mid n \geq 0 \} \mapsto a^* b^*$

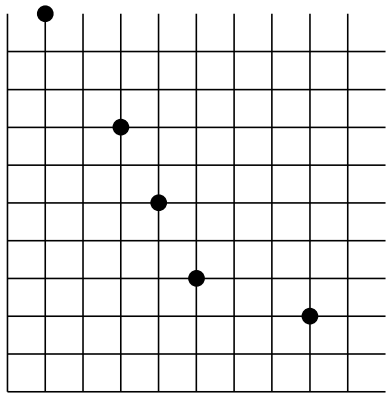
partial order

- reflexive $x \sqsubseteq x$ for all x
- *anti*symmetric $x \sqsubseteq y$ and $y \sqsubseteq x$ implies $x = y$
- transitive $x \sqsubseteq y$ and $y \sqsubseteq z$ imply $x \sqsubseteq z$

$\leq$ on $\mathbb{R}$, $\subseteq$ on $\mathcal{P}(V) = 2^V$, $\leq$ on $\mathbb{Z}^n$

incomparable neither $x \sqsubseteq y$ nor $y \sqsubseteq x$

subword ordering $x \leq y$ iff $y = uxv$



subsequence ordering $x|y$

$x = x_1 x_2 \dots x_n$ and $y = y_1 x_1 y_2 x_2 \dots y_n x_n y_{n+1}$

$ab^n a$ all comparable for $|$ (*chain*)

but all incomparable for $\leq$ (*antichain*)

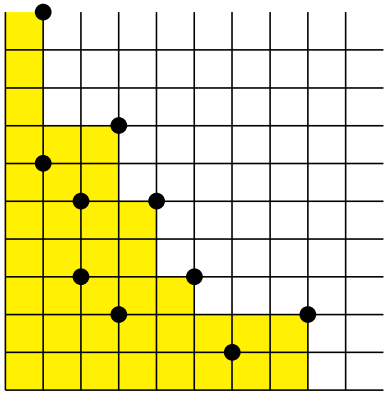
no infinite antichain for $\leq$ on $\mathbb{N}^n$

(Dickson's Lemma)

Thm. no infinite antichain for $|$ on Σ^*

($\sim$ Higman's Lemma)

Theorem 3.12.1

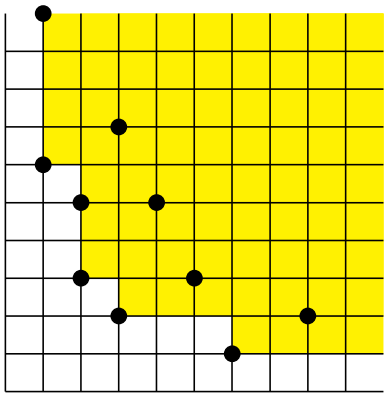


*sub*sequences

$$\mathsf{sub}(L) = \{x \in \Sigma^* \mid x|y \text{ where } y \in L\}$$

*super*sequences

$$\mathsf{sup}(L) = \{x \in \Sigma^* \mid y|x \text{ where } y \in L\}$$



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

$\mathsf{sub}(L) = a^* b^*$

$\mathsf{sup}(L) = \{a, b\}^* ab \{a, b\}^*$

3.12.6 $P_3 = \{ 2, 10, 12, 21, 102, 111, 122, 201, 212, 1002, \ldots \}$

$\mathsf{sub}(P_3) = \{0, 1, 2\}^*$

$\mathsf{sup}(P_3) =$

$$\Sigma^* 2 \Sigma^* \cup \Sigma^* 1 \Sigma^* 0 \Sigma^* \cup \Sigma^* 1 \Sigma^* 1 \Sigma^* 1 \Sigma^*$$

Thm. no infinite antichain for $|$ on Σ^*

Prf. *good* sequences $(w_1, w_2, \dots)$ st. $w_i \not| w_j$ $(i < j)$

order good sequences

$(w_1, w_2, w_3, \dots) < (v_1, v_2, v_3, \dots)$ iff

$|w_1| = |v_1|, \dots |w_k| = |v_k|$ but $|w_{k+1}| < |v_{k+1}|$

(1) every good sequence has a smaller one

$(w_1, w_2, w_3, \dots)$

has infinite subsequence starting with same a

$w_{i_1} = av_1$, $w_{i_2} = av_2$, ...

$(w_1, \dots, w_{i_1 - 1}, v_1, v_2, \dots) < (w_1, w_2, w_3, \dots)$

it is good $v_k | v_\ell$ then $av_k = w_{i_k} | w_{i_\ell} = av_\ell$

it is smaller

(2) there is a minimal good sequence

w_1 shortest word with good continuation

w_1, w_2 shortest word with good continuation

etcetera

($\Rightarrow$) contradiction

$\sqsubseteq$ partial order on S

reflexive, antisymmetric, transitive

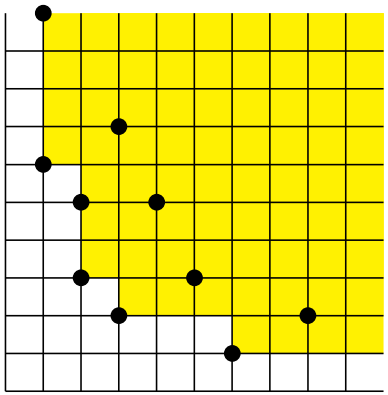
x minimal: $y \sqsubseteq x$ implies $y = x$

Lem. minimal elements are incomparable

$\min(L)$ minimal elements of L

if no infinite descending chain $x_1 \sqsupset x_2 \sqsupset x_3 \ldots$

well-founded



then for $y \in L$ some $y' \in \min(L)$ with $y' \sqsubseteq y$

$\sup(L) = \{x \in S \mid y \sqsubseteq x \text{ where } y \in L\}$

Lem. $\sup(L) = \sup(\min(L))$

special case: $|$ on Σ^*, $\leq$ on $\mathbb{N}^n$.
not $\leq$ on $\mathbb{Z}^n$.

$S - \mathrm{sub}(L) = \sup(\min(S - \mathrm{sub}(L)))$

because $\sup(S - \mathrm{sub}(L)) = S - \mathrm{sub}(L)$

- $\sup(L) = \sup(\min(L))$
- $\min(L)$ finite incomparable

Thm. $\sup(L)$ regular (for arbitrary L)

$w = a_1 a_2 \dots a_k \quad (a_i \in \Sigma)$

$\sup(\{w\}) = \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \dots \Sigma^* a_k \Sigma^*$

$\sup(L) = \sup(\min(L)) = \bigcup_{w \in \min(L)} \sup(\{w\})$

finite union

Thm. $\mathrm{sub}(L)$ regular (for arbitrary L)

$S - \mathrm{sub}(L) = \sup(\min(S - \mathrm{sub}(L)))$ regular

Chapter 4

Context-free Grammars and Languages

4.0 Review

▷

The book uses a transition funtion

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*},$$

i.e., a function into (finite) subsets of $Q \times \Gamma^*$.

My personal favourite is a (finite) transition relation

$$\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*.$$

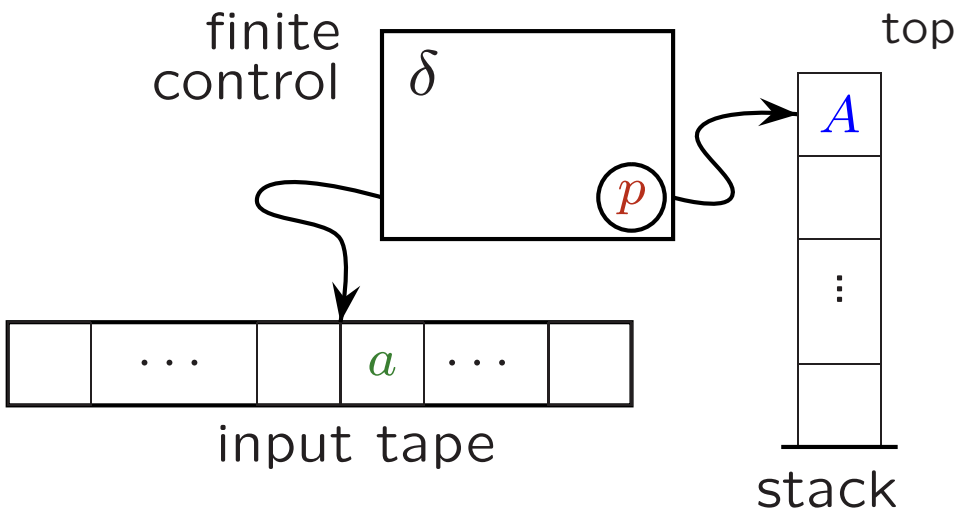
In the former one writes

$$\delta(p, a, A) \ni (q, \alpha)$$

and in the latter

$$(p, a, A, q, \alpha) \in \delta.$$

The meaning is the same. ◁



7-tuple

$$\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Q	*states*	p, q	
$q_0 \in Q$	*initial state*		
$F \subseteq Q$	*final states*		
Σ	*input alphabet*	a, b	w, x
Γ	*stack alphabet*	A, B	α
$Z_0 \in \Gamma$	*initial stack symbol*		

transition function (finite)

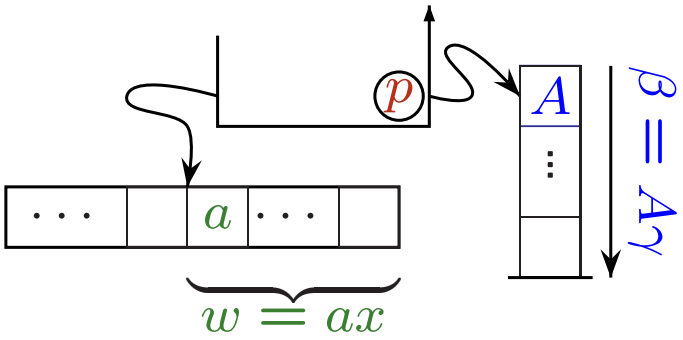
$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$$

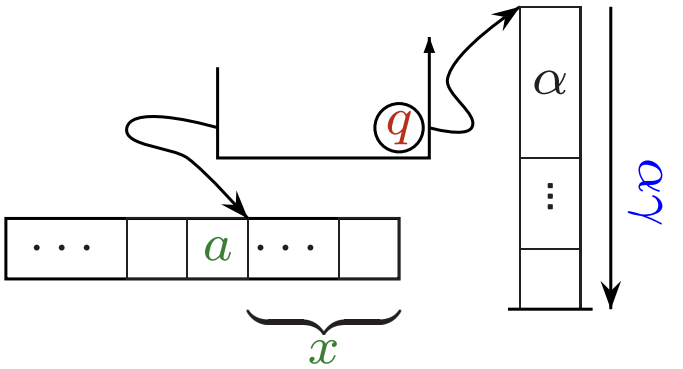
from to

$$(\quad p \quad a \quad A \quad) \ni (\quad q \quad \alpha \quad)$$

read pop push

before after





$Q \times \Sigma^* \times \Gamma^*$ configuration

(p, w, β) $\begin{cases} p & \text{state} \\ w & \text{input, unread part} \\ \beta & \text{stack, top-to-bottom} \end{cases}$

move (step) $\vdash_{\mathcal{A}}$

$(p, ax, A\gamma) \vdash_{\mathcal{A}} (q, x, \alpha\gamma)$ iff

$(p, a, A, q, \alpha) \in \delta$, $x \in \Sigma^*$ and $\gamma \in \Gamma^*$

computation $\vdash_{\mathcal{A}}^*$

$L(\mathcal{A})$ *final state language*

$\{ x \in \Sigma^* \mid (q_0, x, Z_0) \vdash_{\mathcal{A}}^* (q, \varepsilon, \gamma)$

for some $q \in F$ and $\gamma \in \Gamma^*$ $\}$

$L_e(\mathcal{A})$ *empty stack language*

$\{ x \in \Sigma^* \mid (q_0, x, Z_0) \vdash_{\mathcal{A}}^* (q, \varepsilon, \varepsilon)$

for some $q \in Q$ $\}$

▷

The basic theorem of context-free languages: Theorem 1.5.6. the equivalence of cfg and pda.

It is due to

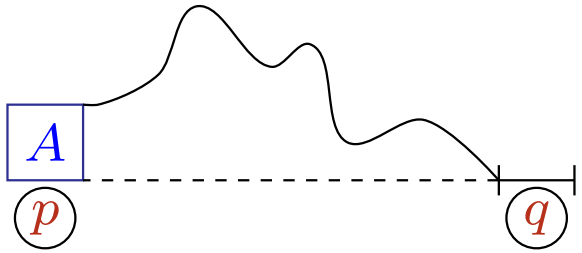
Chomsky 'Context Free Grammars and Pushdown Storage',

Evey 'Application of Pushdown Store Machines', and

Schützenberger 'On Context Free Languages and Pushdown Automata' all in 1962/3.

Starting with a pda under empty stack acceptance we construct an equivalent cfg. Its nonterminals are triplets $[p, A, q]$ representing computations of the pda. Productions result from recursively breaking down computations. A single instruction yields many productions, mainly because intermediate states of the computations have to be guessed.

◁

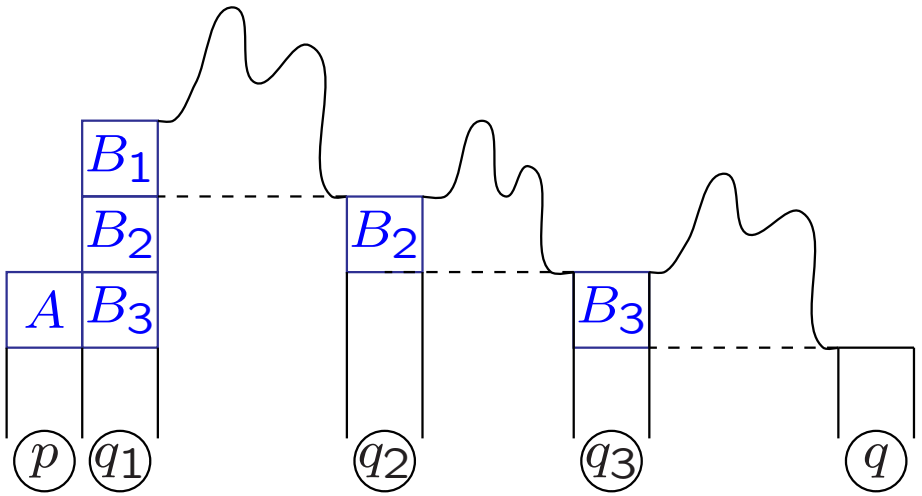


nonterminals $[\,p, A, q\,]$ $\qquad p, q \in Q,\ A \in \Gamma$

$[\,p, A, q\,] \Rightarrow_G^* w \iff (\,p, w, A\,) \vdash^* (\,q, \varepsilon, \varepsilon\,)$

productions

$S \to [\,q_{in}, Z_{in}, q\,]$ $\qquad$ for all $q \in Q$

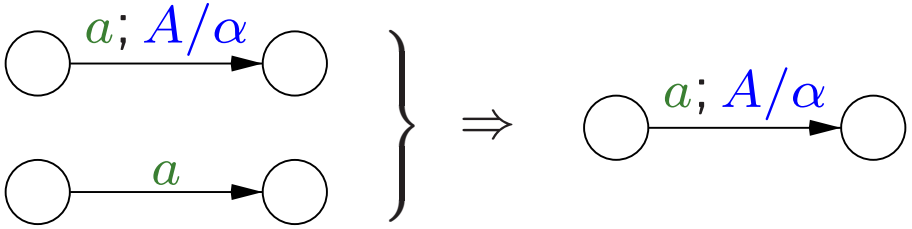


$[\,p, A, q\,] \to a\,[\,q_1, B_1, q_2\,]\,[\,q_2, B_2, q_3\,] \cdots [\,q_n, B_n, q\,]$

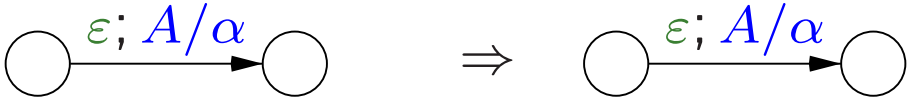
$\delta(p, a, A) \ni (q_1, B_1 \cdots B_n)$

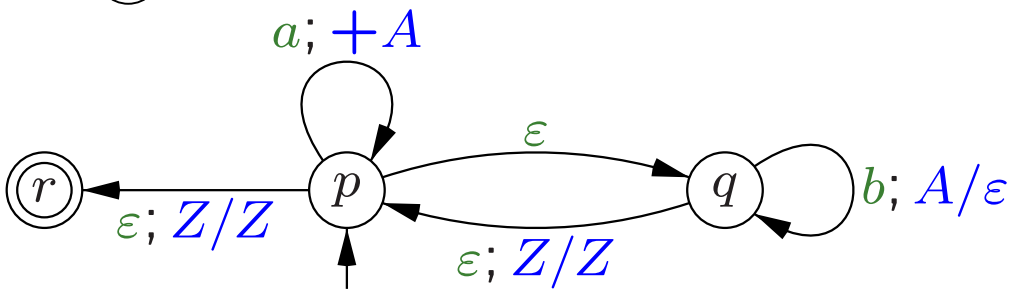
$q, q_2, \ldots, q_n \in Q$

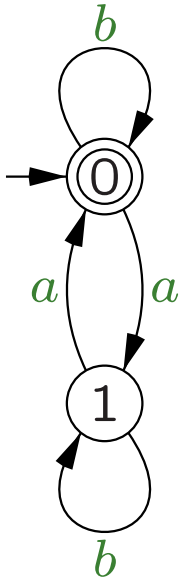
$[\,p, A, q\,] \to a \qquad \delta(p, a, A) \ni (q, \varepsilon)$

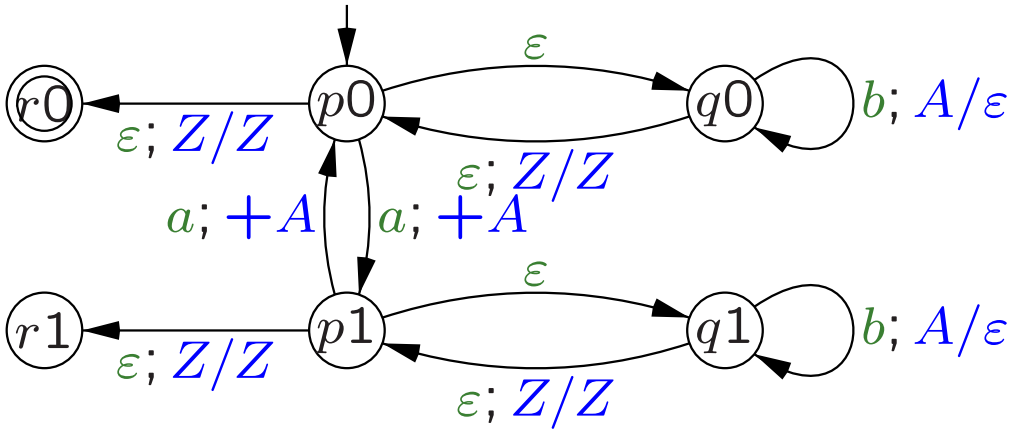


$\{\, a^n b^n \mid n \geq 1 \,\}^* \cap$
$\{\, w \in \{a,b\}^* \mid \#_a x \text{ even} \,\}$









4.1 Closure properties

closed under . . .
union, concatenation, star
(using grammars)

not under intersection, complement

$L = \{ a^n b^n c^n \mid n \geq 0 \}$ not in CF

$\{a^i b^i \mid i \geq 0\} c^* \cap a^* \{b^i c^i \mid i \geq 0\}$

$\{a, b, c\}^* - L$ *is* CF (exercise)

	RLIN REG	DPDA	CF PDAe	DLBA	MON LBA	REC	TYPE0 RE
intersection	+	−	−	+	+	+	+
complement	+	+	−	+	+	+	−
union	+	−	+	+	+	+	+
concatenation	+	−	+	+	+	+	+
star, plus	+	−	+	+	+	+	+
ε-free morphism	+	−	+	+	+	+	+
morphism	+	−	+	−	−	−	+
inverse morphism	+	+	+	+	+	+	+
intersect reg lang	+	+	+	+	+	+	+
mirror	+	−	+	+	+	+	+
	fAFL		fAFL	AFL	AFL	AFL	fAFL

$\cap\ ^c\ \cup$ boolean operations

$\cup\ \cdot\ ^*$ regular operations

$h\ h^{-1}\ \cap R$ (full) trio operations

▷

Next: An 'intuitive' pictorial representation of the direct product construction of a PDA and a FST, showing the image of a PDA language under a transduction is again accepted by a PDA. This proves closure of CF under several operations.

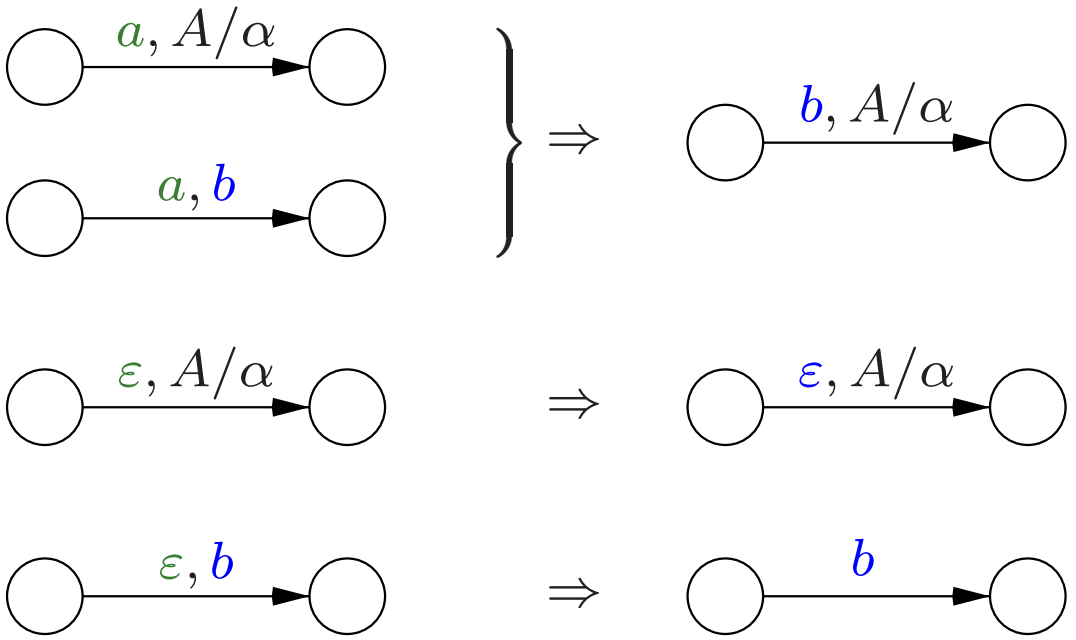
Same construction is given on the transparency after that one, but now in a more precise specification. No formal proof (induction on computations) is given.

Note! Shallit works the reverse way, from full trio operations to FST's. Recall that a family of languages is closed under FST's iff it is closed under morphisms, inverse morphisms and intersection with regular languages. The 'if'-part is Nivat's Theorem 3.5.3, the 'only-if' follows from the fact that these operations can all be performed by a suitable FST. ◁

Thm. CF is closed under fs transductions

$L \in$ CF (given by PDA) FST $\mathcal{A} : \Sigma^* \to \Delta^*$

$T(\mathcal{A})(L) = \{ v \in \Delta^* \mid u \in L, (u, v) \in T(\mathcal{A}) \}$



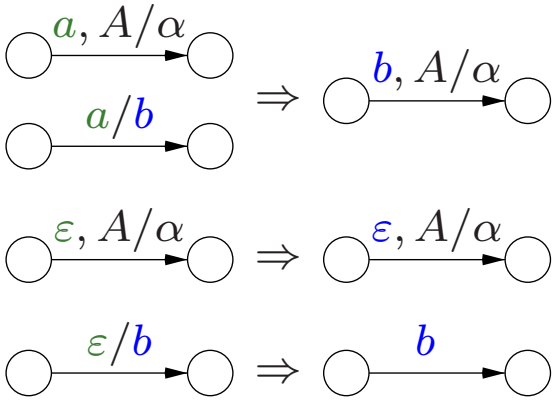
Cor. CF is closed under morphisms, inverse morphisms; intersection, quotient & concatenation with regular languages (x3); prefix, suffix ...

PDA $\mathcal{A} = (Q, \Delta, \Gamma, \delta, q_{in}, Z_{in}, F)$

FST $\mathcal{M} = (P, \Delta, \Sigma, \varepsilon, p_{in}, E)$

$T(\mathcal{M})(L(\mathcal{A})) \Rightarrow$ PDA $\mathcal{A}' = (Q', \Sigma, \Gamma, \delta', q'_{in}, Z_{in}, F')$

formally – $Q' = Q \times P$
– $q'_{in} = \langle q_{in}, p_{in} \rangle$
– $F' = F \times E$, and
– δ' is defined by



if $\delta(q_1, a, A) \ni (q_2, \alpha)$, and $(p_1, a, b, p_2) \in \varepsilon$ (with $a \neq \varepsilon$)
then $\delta'(\langle q_1, p_1 \rangle, b, A) \ni (\langle q_1, p_1 \rangle, \alpha)$

if $\delta(q_1, \varepsilon, A) \ni (q_2, \alpha)$ and $p \in P$,
then $\delta'(\langle q_1, p \rangle, \varepsilon, A) \ni (\langle q_1, p \rangle, \alpha)$

if $q \in Q$ and $(p_1, \varepsilon, b, p_2) \in \varepsilon$,
then $\delta'(\langle q, p_1 \rangle, b, A) \ni (\langle q, p_1 \rangle, \alpha)$

▷

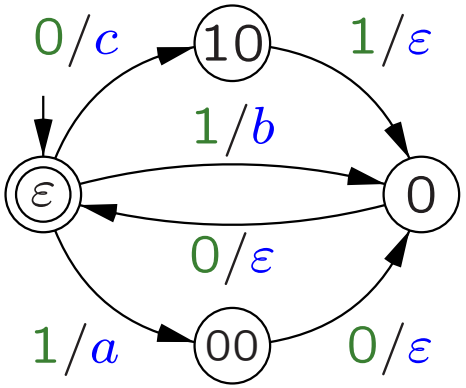
As an example of finite state transducers and the closure construction: the inverse morphism.

In Shallit this is Thm. 4.1.4, without explicit FST.

For a morphism h we construct a FST that realizes h^{-1}. Then for the context-free language $K = \{(100)^n(10)^n \mid n \geq 0\}$ we construct PDA for K and $h^{-1}(K)$.

◁

$$h : \begin{cases} a & \mapsto & 100 \\ b & \mapsto & 10 \\ c & \mapsto & 010 \end{cases}$$



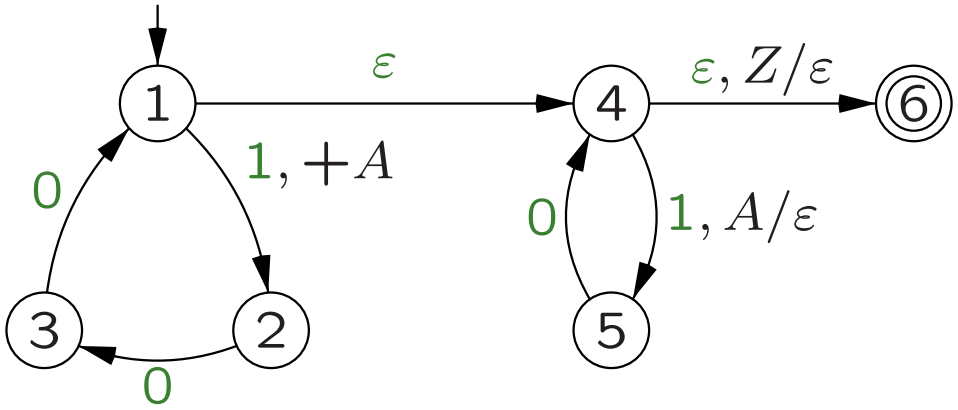
100 100 100 101 010

a a a b b b

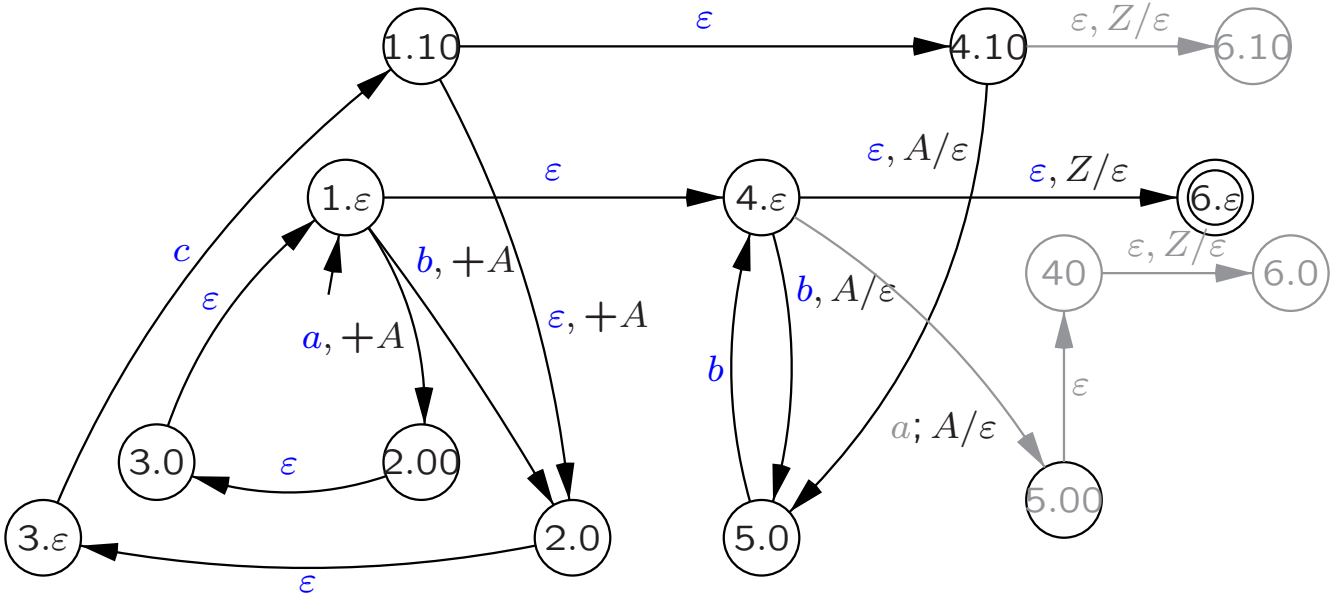
b c c c b b

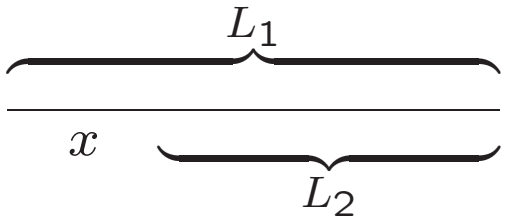
a b c c b b

$$K = \{\ (100)^n(10)^n \mid n \geq 0\ \}$$



$$h^{-1}(K) = \{\ w \in \{a,b,c\}^* \mid h(w) \in K\ \}$$





$L_1, L_2 \subseteq \Sigma^*$

$L_1/L_2 = \{\, x \in \Sigma^* \mid xy \in L_1 \text{ for some } y \in L_2 \,\}$

can 'hide' computations

Ex. $L_1 = \{\, a^{2n}cba^n \mid n \geq 1 \,\}\{\, ba^{2n}ba^n \mid n \geq 1 \,\}^*ba$

$L_2 = c \cdot \{\, ba^nba^n \mid n \geq 1 \,\}^*$

$L_1/L_2 = \{\, a^{2^n} \mid n \geq 1 \,\}$

Thm. CF not closed under quotient

▷

As promised, the CF languages are closed under right quotient with regular languages, since for every regular language R we can transform the FSA for R into a FST that performs the quotient by R as its function.

The next slide implements this construction. Given a PDA $\mathcal{A}$ and a FSA $\mathcal{M}$ it directly constructs the PDA for the quotient of the languages. It uses the general format for transductions from previous slides, as if the transducer for the quotient had been given. In fact, is has been implicitly derived from the FSA, by adding a single state $\imath$, see sketch to the left for a specific example. ◁

$L(\mathcal{A}) = L$ PDA $\mathcal{A} = (Q, \Delta, \Gamma, \delta, q_{in}, Z_{in}, F)$

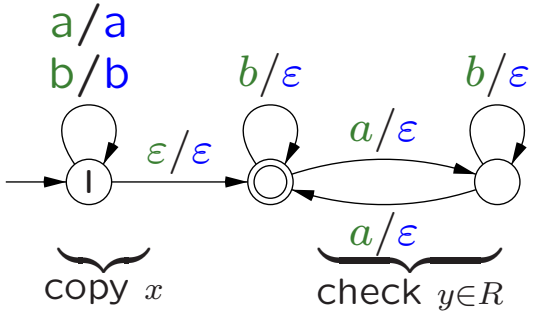
$L(\mathcal{M}) = R$ FSA $\mathcal{M} = (P, \Delta, \varepsilon, p_{in}, E)$

PDA for right quotient L/R

$\mathcal{A}' = (Q', \Delta, \Gamma, \delta', q'_{in}, Z_{in}, F')$

$Q' = Q \times (P \cup \{\text{ı}\})$

quotient transducer



$K/R =$

$\{\, x \mid xy \in K \text{ and } y \in R \,\}$

δ' contains

$(\, \langle q_1, \text{ı} \rangle, a, A, \langle q_2, \text{ı} \rangle, \alpha \,)$ for $\delta(q_1, a, A) \ni (q_2, \alpha)$

$(\, \langle p, \text{ı} \rangle, \varepsilon, A, \langle p, p_{in} \rangle \,)$ for $p \in P,\ A \in \Gamma$

$(\, \langle q_1, p \rangle, \varepsilon, A, \langle q_2, p \rangle, \alpha \,)$

for $\delta(q_1, \varepsilon, A) \ni (q_2, \alpha),\ p \in Q$

$(\, \langle q_1, p_1 \rangle, \varepsilon, A, \langle q_2, p_2 \rangle, \alpha \,)$

for $\delta(q_1, a, A) \ni (q_2, \alpha)$ & $(p_1, a, p_2) \in \varepsilon$

$q'_{in} = \langle q_{in}, \text{ı} \rangle$

$F' = F \times E$

family of languages $\mathcal{L}$ is a full trio (or cone)

iff $\mathcal{L}$ is closed under morphism h,
inverse morphism h^{-1}, and
intersection with regular languages $\cap R$

iff $\mathcal{L}$ is closed under finite state transductions T

Cor. full trio closed under prefix, quotient, . . .

Thm. REG and CF are full trio's.

4.2 Unary context-free languages

Lem. $L \subseteq \{1\}^*$, $w \in \{1\}^*$, then $L \cdot w^*$ is regular.

Prf. Let $n = |w|$. $0 \leq k < n$
consider all $x \in L$ such that $|x| \equiv k \bmod n$.
Either empty, or has minimal element.
Define $\min_n(L)$ as the set of these minimal elements for all $0 \leq k < n$. Finite set.
$L \cdot w^* = \min_n(L) \cdot w^*$.

Lem. L^* is regular for *any* unary L.

Prf. nonempty $w \in L$, then $L^* = L^* \cdot w^*$.

todo. rewrite next proof unary CFL

$L \subseteq \{0\}^*$ $L \in$ CF iff $L \in$ REG

pumping constant n
$z = uvwxy$, $1 \leq |vx| \leq n$
$z(0^{|vx|})^* \subseteq L$.

$L_0 = \{x \in L \mid |x| < n\}$ short
$L_k \subseteq L$: $L_k \cdot (0^k)^* \subseteq L$ pumping length k

$L_k = L_k \cdot (0^k)^*$

$L = \bigcup_{0 \leq k \leq n} L_k$ each L_k regular

4.6 Parikh's theorem

$h : \Sigma \to \{0\}, \quad x \mapsto 0$

CF $\rightsquigarrow$ REG same length sets

Parikh map *commutative image*

$\psi : \Sigma^* \to \mathbb{N}^k$

$w \mapsto (|w|_{a_1}, \ldots, |w|_{a_k})$

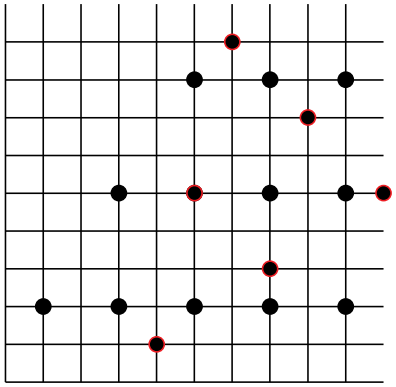
$aabaccbacca \mapsto (5, 2, 4)$

$c(ab)^*c(bc)^*c \mapsto \{ (k, k+\ell, 3+\ell) \mid k, \ell \in \mathbb{N} \} =$

$\{ (0,0,3) + k \cdot (1,1,0) + \ell \cdot (0,1,1) \mid k, \ell \in \mathbb{N} \}$

$(abc)^*$	REG
$\{ (ab)^n c^n \mid n \in \mathbb{N} \}$	LIN − REG
$\{ w \in \{ab, c\}^* \mid \#_a(w) = \#_b(w) \}$	CF − LIN
$\{ a^n b^n c^n \mid n \in \mathbb{N} \}$	CS − CF

$\mapsto \{ (n,n,n) \mid n \in \mathbb{N} \} = \{ n \cdot (1,1,1) \mid n \in \mathbb{N} \}$



linear set $\vec{u}_0, \vec{u}_1, \ldots \vec{u}_r \in \mathbb{N}^k$

$A = \{\vec{u}_0 + a_1\vec{u}_1 + \cdots + a_r\vec{u}_r \mid a_1, \ldots, a_r \in \mathbb{N}\}$

semilinear finite union

4.6.1 semilinear sets closed under
union, intersection and complement

4.6.3 X semilinear, then $X = \psi(L)$ for regular L

$\omega(\vec{u}_0) \cdot \{\, \omega(\vec{u}_1), \ldots, \omega(\vec{u}_r) \,\}^*$

$\omega : \mathbb{N}^k \to \{a_1, \ldots, a_k\}^* \qquad \psi(\omega(\vec{u})) = \vec{u}$

4.6.5 $\psi(L)$ semilinear for CFL L

Lemma 4.6.4

G Chomsky normal form

k variables $p = 2^{k+1}$

$z \in L(G)$, $|z| \geq p^j$

$S \Rightarrow^*$

$uAy \Rightarrow^*$

$uv_1Ax_1y \Rightarrow^*$

$uv_1v_2Ax_2x_1y \Rightarrow^*$

$\ldots \quad \Rightarrow^*$

$uv_1v_2 \ldots v_jAx_j \ldots x_2x_1y \Rightarrow^*$

$uv_1v_2 \ldots v_jwx_j \ldots x_2x_1y = z$

$v_ix_i \neq \varepsilon$

$|uv_1v_2 \ldots v_jx_j \ldots x_2x_1y| \leq p^j$

Theorem 4.6.5

$\psi(L)$ semilinear for CFL L

$L_U \subseteq L$

derivation with variables U

$L = \bigcup_{S \subseteq U \subseteq V} L_U$

$\ell = |U|$

$E = \{ w \in L_U \mid |w| < p^\ell \}$ $\quad S \Rightarrow^* w$

$F = \{ vx \mid 1 \leq |vx| \leq p^\ell,$

$A \Rightarrow^* vAx$ for some $A \in U \}$

$\psi(L_U) = \psi(EF^*)$

"$\subseteq$" induction on $|z|$, $z \in L_U$

"$\supseteq$" induction on t,

$z = e_0f_1 \ldots f_t \in EF^*$

Ex. $L = \{ a^i b^j \mid j \neq i^2 \}$ not in CF

$\psi(L) = \{ (i,j) \mid j \neq i^2 \}$ not semilinear
complement $\{ (i,i^2) \mid i \in \mathbb{N} \}$
corresponding regular language?
lengths $\{ i^2 + i \mid i \in \mathbb{N} \}$ cannot be pumped

4.3 Ogden's lemma
4.4 Applications of Ogden's lemma

long words can be pumped

$\forall$ for every CF language L

$\exists$ there exists a constant $n \geq 1$

such that

$\forall$ for every $z \in L$

with $|z| \geq n$

$\exists$ there exists a decomposition $z = uvwxy$

with $|vwx| \leq n$, $|vx| \geq 1$

such that

$\forall$ for all $i \geq 0$, $uv^iwx^iy \in L$

$\|x\|$ marked symbols in x

$\forall$ for every CF language L

$\exists$ there exists a constant $n \geq 1$

such that

$\forall$ for every $z \in L$

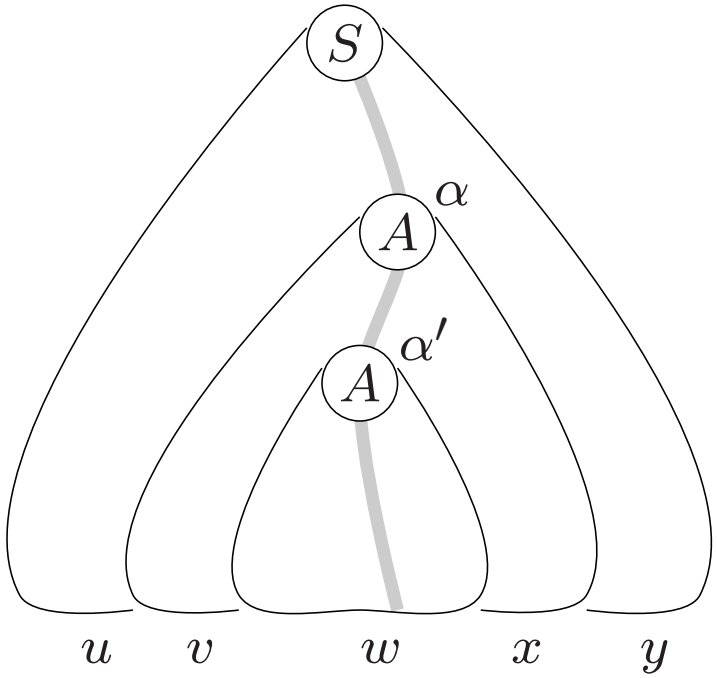
with $\|z\| \geq n$

$\exists$ there exists a decomposition $z = uvwxy$

with $\|vwx\| \leq n$, $\|vx\| \geq 1$

such that

$\forall$ for all $i \geq 0$, $uv^iwx^iy \in L$



$S \Rightarrow^* uAy$

$A \Rightarrow^* vAx$

$A \Rightarrow^* w$

$uv^iwx^iy \in L$

$G = (V, \Sigma, P, S)$

$k = |V|$ $d = \max\{|\alpha| \mid A \to \alpha \in P\}$

branch point: ≥ 2 children with marked descendants

if each path has $\leq \ell$ branch points, then $\leq d^\ell$ marked letters

pumping constant $n = d^{k+1} > d^k$

$\exists$ path with $> k$ branch points

take path with most branch points

α, α' same label A,
as low as possible

$\|vx\| \geq 1$ α branch point

$\|vwx\| \leq n$ no repetition below α

$\|w\| \geq 1$ α' branch point

$L = \{\ a^i b^j c^k \mid$
$\quad i = j$ or $j = k$ but not both $\}$
not context-free

n as Ogden, assume ≥ 3

$z = \underline{a^n}\ b^n\ c^{n+n!}$

$z = uvwxy$

v, x each cannot have different symbols else $uv^2wx^2y \notin a^*b^*c^*$

possibilities

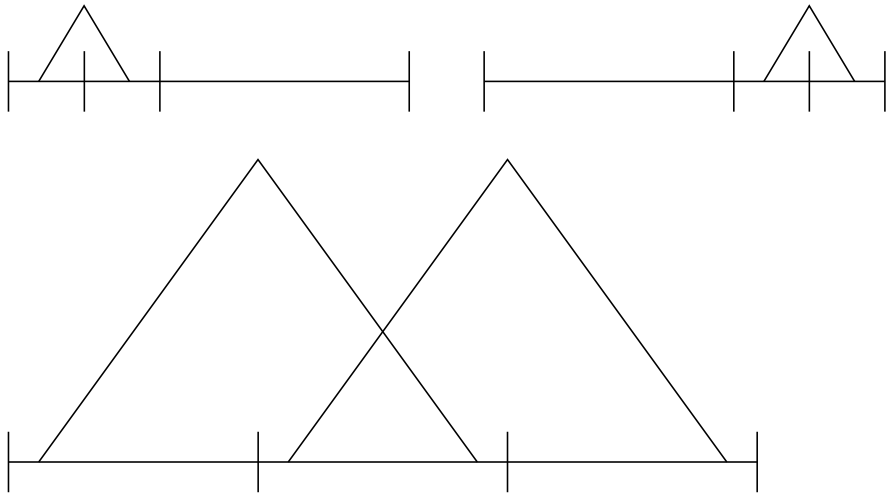
- $vx = a^k$
  $uv^0wx^0y = a^{n-k}b^nc^{n+n!} \notin L$
- $v = a^k$, $x = b^\ell$ $(k \neq \ell)$
  $uv^0wx^0y = a^{n-k}b^{n-\ell}c^{n+n!} \notin L$
- $v = a^k$, $x = b^\ell$ $(k = \ell)$
  consider $i = \frac{n!}{\ell} + 1$
  add $i - 1$ copies of ℓ a's
  $uv^iwx^iy = a^{n+n!}b^{n+n!}c^{n+n!} \notin L$
- $v = a^k$, $x = c^\ell$
  $uv^2wx^2y = a^{n+k}b^nc^{n+n!+\ell} \notin L$

grammar ambiguous
language inherently ambiguous

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

see example 4.3.2

$z = \underline{a^n}\, b^n\, c^{n+n!}$ $\quad$ $z' = a^{n+n!}\, b^n\, \underline{c^n}$



$$[p, A, q] \Rightarrow_G^* w \iff (p, w, A) \vdash_{\mathcal{M}}^* (q, \varepsilon, \varepsilon)$$

Thm. PDA $\mathcal{M}$ with n states and p stack symbols each CFG for $L_e(\mathcal{M})$ has at least $n^2 p$ variables

4.5 The interchange lemma

$\forall$ for every CF language L

$\exists$ there exists constant $c > 0$

$\forall$ such that for all $n \geq m \geq 2$,

all subsets $R \subseteq L \cap \Sigma^n$

$\exists$ there exists $Z = \{z_1, z_2, \ldots, z_k\} \subseteq R$,

with $k \geq \frac{|R|}{c(n+1)^2}$

and compositions $z_i = w_i x_i y_i$

such that

(a) $|w_1| = |w_2| = \cdots = |w_k|$

(b) $|y_1| = |y_2| = \cdots = |y_k|$

(c) $\frac{m}{2} < |x_1| = |x_2| = \cdots = |x_k| \leq m$

(d) $w_i x_j y_i \in L$ for all $1 \leq i, j \leq k$

Lem. G CFG in Chomsky normal form for L, $m \geq 2$
$z \in L$, $|z| \geq m$, then $S \Rightarrow^* wAy \Rightarrow^* wxy = z$
with $\frac{m}{2} < |x| \leq m$

$z \rightsquigarrow (n_1, A, n_2)$ where $n_1 = |w|$, $n_2 = |z|$

Chapter 2 Thue-Morse sequence

t_n number of 1's in base-2 expansion of n

or iterate $0 \mapsto 01$, $1 \mapsto 10$

$0 \cdot 1 \cdot 10 \cdot 1001 \cdot 10010110 \cdot 1001011001101001 \ldots$

overlapfree *no* $axaxa$ ($a \in \Sigma_2$, $x \in \Sigma_2^*$)

$00 \mapsto 1$, $01 \mapsto 2$, $10 \mapsto 0$, $11 \mapsto 1$ 'sliding'

$2102012101202102012021012102012 \ldots$

squarefree *no* xx ($x \in \Sigma_3^*$)

$\Sigma = \{0, 1, \ldots, i-1\}$ $\quad L_i = \{\, xyyz \mid x, y, z \in \Sigma^*, y \neq \varepsilon \,\}$

Thm. L_6 not in CF

[see Chapter 2] r squarefree string of length $\frac{n}{4}-1$ over $\{0, 1, 2\}$

$A_n = \{3r3r \amalg s \mid s \in \{4, 5\}^{n/2}\}$

$\amalg$ perfect shuffle (alternate strings)

$z \in A_n$ contains a square iff it *is* a square

$B_n = L_6 \cap A_n = \{3r3r \amalg ss \mid s \in \{4, 5\}^{n/4}\}$

$|B_n| = 2^{\frac{n}{4}}$ $\quad$ choose $m = n/2$

[take n large] $Z = \{z_1, z_2, \ldots, z_k\}$ $\quad k \geq \frac{2^{n/4}}{c(n+1)^2} > 2^{n/8}$

$z_i = w_i x_i y_i$, $\frac{m}{2} < |x_i| \leq m$ (etc.)

$w_i x_j y_i \in B_n$ hence $x_i = x_j$

(x_i fixed by other symbols in z_i)

hence $\frac{n}{4}$ symbols fixed for Z, $\frac{n}{8}$ in $\{4, 5\}$

at most $\frac{n}{8}$ free, $|Z| \leq 2^{n/8}$

contradiction

4.7 Deterministic context-free languages

▷

what we learn about

deterministic context-free
languages

- is an *automaton* notion

- less powerful than CF

- closed under complement
(nontrivial)

- see also Chapter 5 on parsing

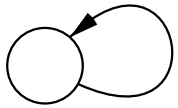
◁

	RLIN REG	DPDA	CF PDAe	DLBA	MON LBA	REC	TYPE0 RE
intersection	+	−	−	+	+	+	+
complement	+	+	−	+	+	+	−
union	+	−	+	+	+	+	+
concatenation	+	−	+	+	+	+	+
star, plus	+	−	+	+	+	+	+
ε-free morphism	+	−	+	+	+	+	+
morphism	+	−	+	−	−	−	+
inverse morphism	+	+	+	+	+	+	+
intersect reg lang	+	+	+	+	+	+	+
mirror	+	−	+	+	+	+	+
	fAFL		fAFL	AFL	AFL	AFL	fAFL

$\cap\ ^c\ \cup$ boolean operations

$\cup\ \cdot\ ^*$ regular operations

$h\ h^{-1}\ \cap R$ (full) trio operations



$a;\ Z/ZA$
$b;\ Z/ZB$
$\varepsilon;\ Z/\varepsilon$
$a;\ A/\varepsilon$
$b;\ B/\varepsilon$

$Z \to aZA$
$Z \to bZB$
$Z \to \varepsilon$
$A \to a$
$B \to b$

$P = \{\ ww^R \mid w \in \{a,b\}^*\ \}$ guessing the middle

$(aabbaa, Z) \vdash (aabbaa, \varepsilon) \nvdash$

⊤

$(abbaa, ZA) \vdash (abbaa, A) \vdash (bbaa, \varepsilon) \nvdash$

⊤

$(bbaa, ZAA) \vdash (bbaa, AA) \nvdash$

⊤

$(baa, ZBAA) \vdash (baa, BAA) \vdash (aa, AA) \vdash (a, A) \vdash (\varepsilon, \varepsilon)$ ok.

⊤

$(aa, ZBBAA) \vdash (aa, BBAA) \nvdash$

⊤

$(a, ZABBAA) \vdash (a, ABBAA) \vdash (\varepsilon, BBAA) \nvdash$

⊤

$(\varepsilon, ZAABBAA) \vdash (\varepsilon, ABBAA) \nvdash$

also $\{\ a^n b^n \mid n \in \mathbb{N}\ \} \cup \{\ a^n b^\ell c^n \mid \ell, n \in \mathbb{N}\ \}$

▷

Determinism means the automaton has no choice: at each moment it can take at most one step to continue its computation. To translate this intuition to a restriction on the instructions for PDA is nontrivial, as the next step is determined both by input letter and by topmost stack symbol. Additionally this is complicated by the choice between reading an input letter and following a λ-instruction.

We quote from our chapter:

The PDA $\mathcal{A} = (Q, \Delta, \Gamma, \delta, q_{in}, A_{in}, F)$ is *deterministic* if

- for each $p \in Q$, each $a \in \Delta$, and each $A \in \Gamma$, δ does not contain both an instruction $(p, \lambda, A, q, \alpha)$ and an instruction (p, a, A, q', α').

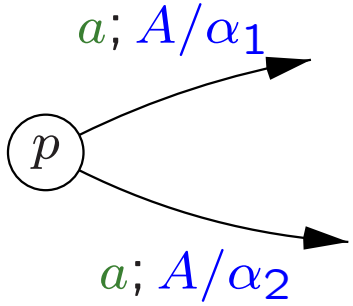
- for each $p \in Q$, each $a \in \Delta \cup \{\lambda\}$, and each $A \in \Gamma$, there is at most one instruction (p, a, A, q, α) in δ.

◁

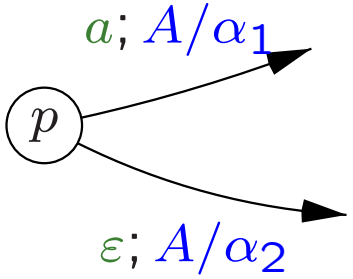
determinism means 'no choice'

- . . . where to start (ok)
- . . . between two actions
  with same *tape & stack* symbols
- . . . between letter or ε

not allowed



$(p, a, A) \ni (q_1, \alpha_1)$
$(p, a, A) \ni (q_2, \alpha_2)$

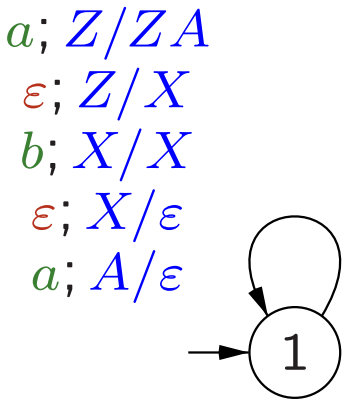


$(p, a, A) \ni (q_1, \alpha_1)$
$(p, \varepsilon, A) \ni (q_2, \alpha_2)$

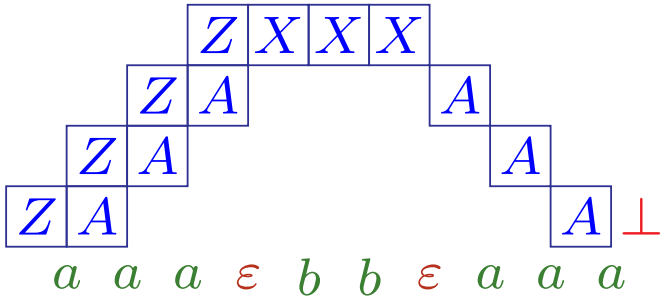
FSA = DFSA = RLIN
PDAe = PDA = CF
DPDAe $\subset$ DPDA $\subset$ CF

final state: deterministic CF languages

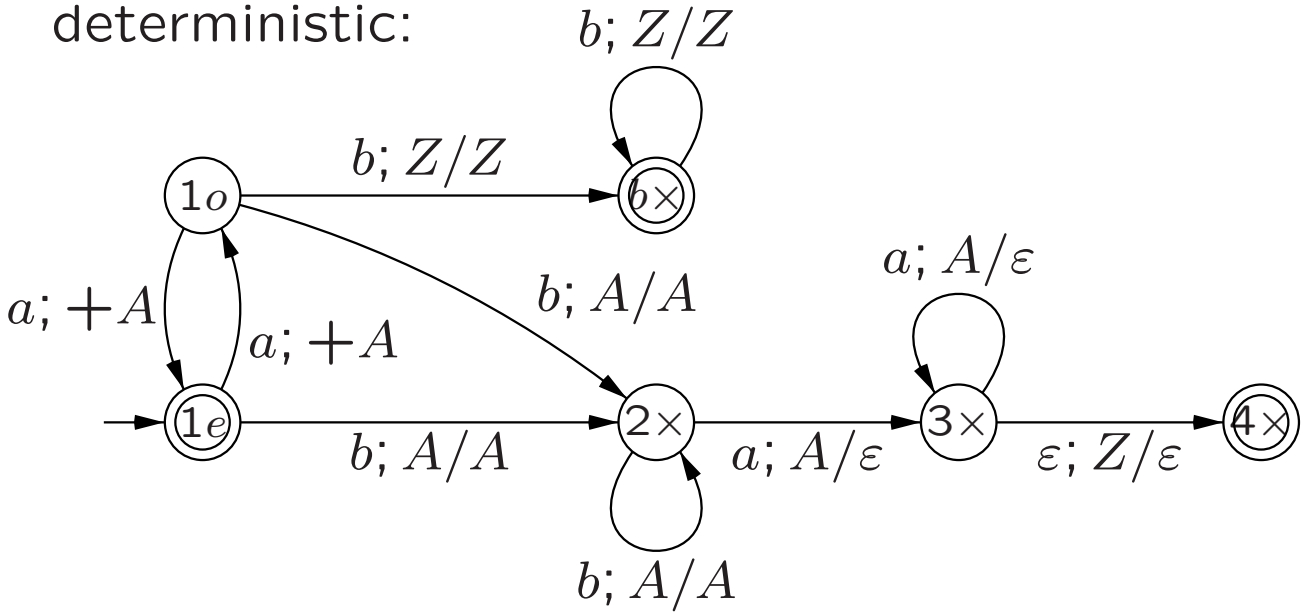
'context-free' but uses automata



$\{ a^n b^m a^n \mid m, n \in \mathbb{N} \}$



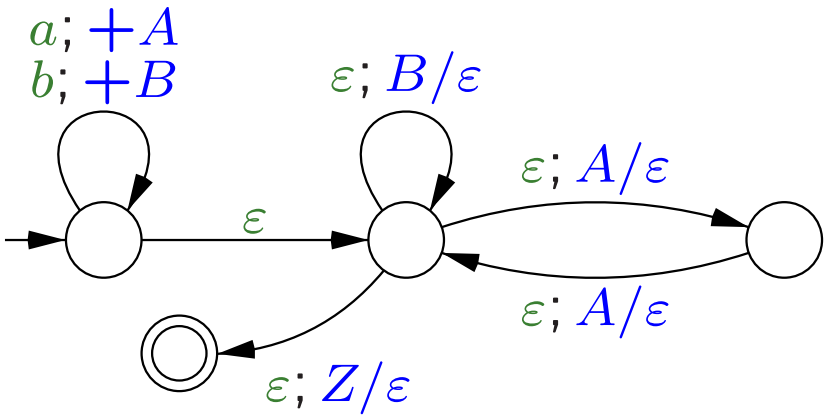
deterministic:



closure under complement $F \leftrightarrow Q - F$

⋆ completely read input
  - ∗ input+stack may block
  - ∗ infinite ε-computations!

⋆ computations without reading
  - ∗ accept afterwards



$\{A, B, Z\}$, initial Z

Lem. equivalent PDA that always scans entire input

$(q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha) \quad q \in Q,\ \alpha \in \Gamma^*$

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
$Q' = Q \cup \{d, f\},\ \Gamma' = \Gamma \cup \{X_0\},\ F' = F \cup \{f\},$

'dead' states
$\delta'(d, a, X) = \{(d, X)\}$
$\delta'(f, a, X) = \{(d, X)\}$ for $a \in \Sigma$ and $X \in \Gamma'$

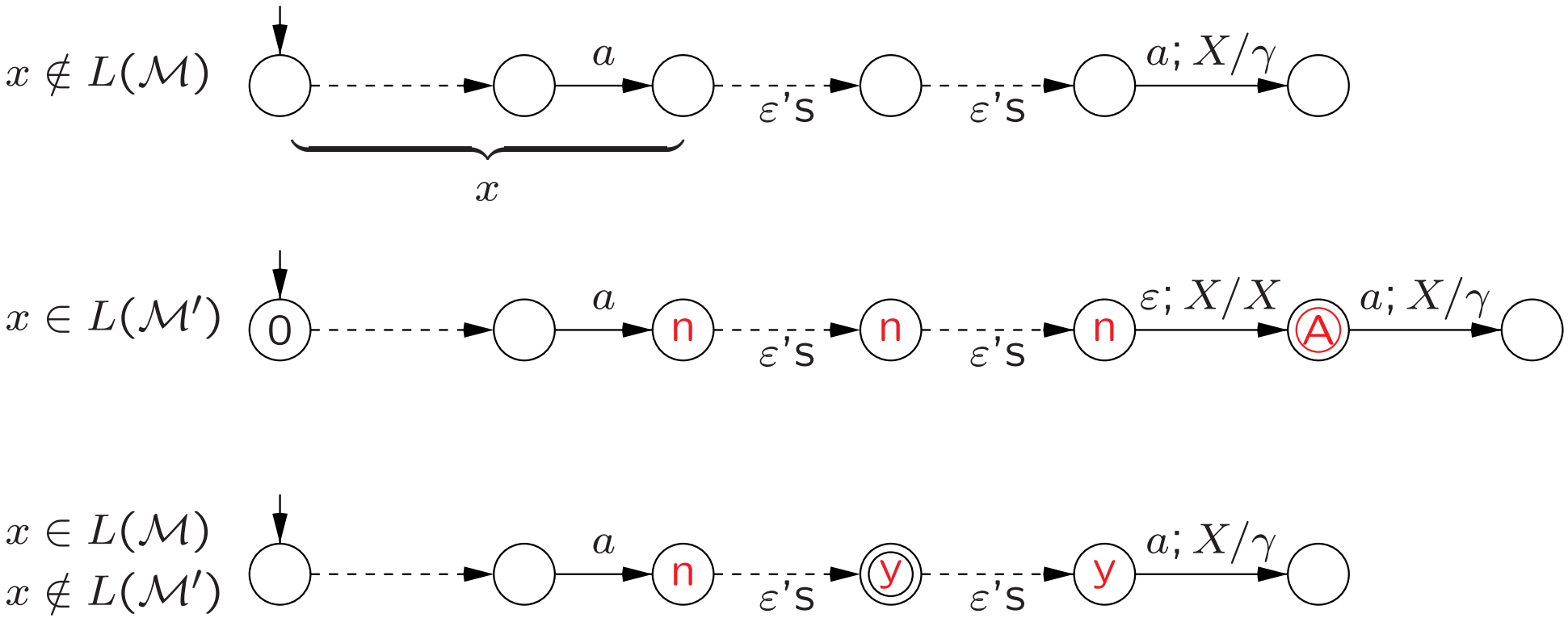
avoid empty stack
$\delta'(q_0', \varepsilon, X_0) = \{(q_0, Z_0 X_0)\}$

add 'bottom' X_0
$\delta'(q, a, X_0) = \{(d, X_0)\}$ for $q \in Q$ and $a \in \Sigma$

undefined transitions
$\delta'(q, a, X_0) = \{(d, X_0)\}$
when $\delta(q, a, X) = \varnothing$ and $\delta(q, \varepsilon, X) = \varnothing$

infinite loops*
when $\mathcal{M}$ enters infinite ε-loop on (q, ε, X)
$\delta'(q, \varepsilon, X) = \{(d, X)\}$ without final states
$\delta'(q, \varepsilon, X) = \{(f, X)\}$ with final state

* "The actual implementation is a bit complex"


$x \notin L(\mathcal{M})$
a
ε's
ε's
$a; X/\gamma$
x
$x \in L(\mathcal{M}')$
0
a
n
ε's
n
ε's
n
$\varepsilon; X/X$
A
$a; X/\gamma$
$x \in L(\mathcal{M})$
$x \notin L(\mathcal{M}')$
a
n
ε's
y
ε's
y
$a; X/\gamma$

Thm. DCFL (= DPDA) closed under complement

$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
$Q' = Q \times \{n, y, A\},\ F' = Q \times \{A\}$
$q_0' = [q_0, y]$ if $q_0 \in F$, $q_0' = [q_0, n]$ otherwise

$\delta(q, a, X) = (p, \gamma)$	$\delta'([q, y], a, X) = ([p, y], \gamma)$	$p \in F$
$(a \in \Sigma)$	$\delta'([q, y], a, X) = ([p, n], \gamma)$	$p \notin F$
	$\delta'([q, n], \varepsilon, X) = ([q, A], X)$	
	$\delta'([q, A], a, X) = ([p, y], \gamma)$	$p \in F$
	$\delta'([q, A], a, X) = ([p, n], \gamma)$	$p \notin F$
$\delta(q, \varepsilon, X) = (p, \gamma)$	$\delta'([q, y], \varepsilon, X) = ([p, y], \gamma)$	
	$\delta'([q, n], \varepsilon, X) = ([p, y], \gamma)$	$p \in F$
	$\delta'([q, n], \varepsilon, X) = ([p, n], \gamma)$	$p \notin F$

Ex. $\{ w \in \{a, b\}^* \mid w \neq xx \}$ not in DCFL

Thm. L DCFL
at least one Myhill-Nerode class is infinite

$x \in \Sigma^* \rightsquigarrow x', q, A\alpha$
after processing xx' stack height $|A\alpha|$ minimal
$(q_0, xx', Z_0) \vdash^* (q, \varepsilon, A\alpha)$

any continuation independent of α
infinitely many xx' end in same minimal q, A
infinitely many xx' all in L or all in $\Sigma^* - L$
have the same 'extensions'

$(q_0, xx'z, Z_0) \vdash^* (q, z, A\alpha) \vdash^* (p, \varepsilon, \gamma\alpha)$ $(p \in F)$
iff $(q_0, x_1x_1'z, Z_0) \vdash^* (q, z, A\alpha_1) \vdash^* (p, \varepsilon, \gamma\alpha_1)$

Cor. PAL $= \{ x \in \{a, b\}^* \mid x = x^R \}$ not in DCFL

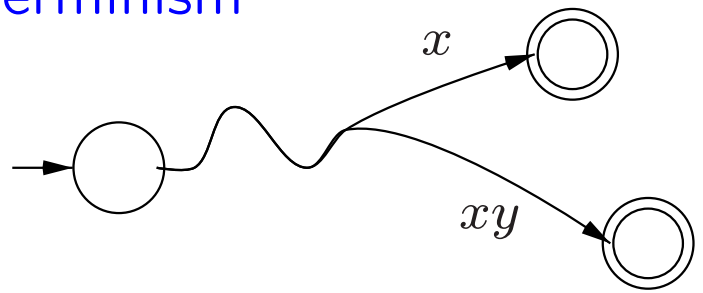
(exercise) no strings equivalent

▷

Consider a language that both includes string x and an extension xy of it. Nondeterministic automata may have quite different accepting computations on both strings. For deterministic automata we know that the computation that accepts xy must start with the accepting computation on x. ◁

language L $\quad x \in L,\ xy \in L$

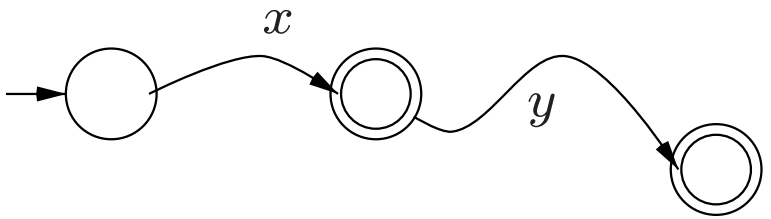
* nondeterminism



$\underline{a^n}\ \underline{b^n}$

$\underline{a^n}\ b^m\ \underline{c^n}$ $\quad$ different behaviour on b's

* determinism



computation on xy and on x must coincide!

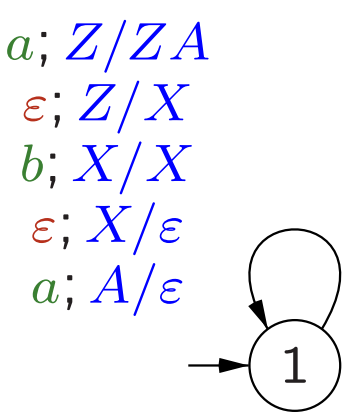
apply this to:

$$\text{haspref}(L) = \{\ xy \mid \underline{x} \in L, \underline{xy} \in L, y \neq \varepsilon\ \}$$

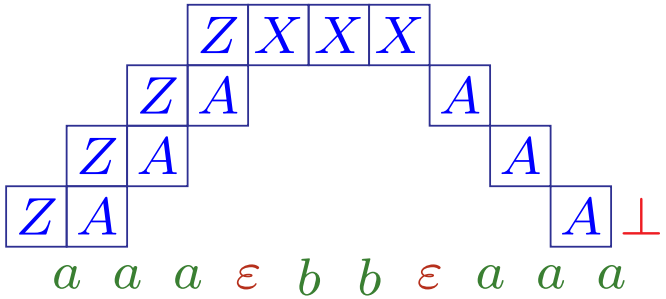
For nondeterministic PDA, i.e. for CFG, equivalence is undecidable.

Geraud Senizergues (2001) proved that the equivalence problem for deterministic PDA (i.e. given two deterministic PDA A and B, is $L(A) = L(B)$?) is decidable.

4.8 Linear languages



$\{ a^n b^m a^n \mid m, n \in \mathbb{N} \}$



$Z \to aZa$

$Z \to X$

$X \to bX$

$X \to \varepsilon$

linear grammar: rhs at most one variable

$A \to \alpha B \beta,\ X \to \alpha$

$A, B \in V,\ \alpha.\beta \in \Sigma^*$

$\{ a^n b^n \mid n \in \mathbb{N} \}$

$\{ a^n b^n c^m \mid m, n \in \mathbb{N} \}$

$\{ a^n b^n a^m b^m \mid m, n \in \mathbb{N} \}$ not LIN, *why?*

long words can be pumped

$\forall$ for every LIN language L

$\exists$ there exists a constant $n \geq 1$
such that

$\forall$ for every $z \in L$
with $|z| \geq n$

$\exists$ there exists a decomposition $z = uvwxy$
with $|uvxy| \leq n$, $|vx| \geq 1$
such that

$\forall$ for all $i \geq 0$, $uv^iwx^iy \in L$

context-free	((((())())())))(())
linear	((((((()))))))
example	((()))((((()))))

$L = \{ a^i b^i c^j d^j \mid i, j \geq 0 \}$ in CFL − LIN

$z = a^n b^n c^n d^n$

$|uvxy| \leq n$

v and x each consist of a's or d's

$v = a^k$, $x = d^\ell$, $k + \ell \geq 1$

$uv^0wx^0y = a^{n-k} b^n c^n d^{n-\ell} \notin L$

and two other possibilities

$\{ x \in \{a,b\}^* \mid x = x^R \}$ in LIN - DCF

$\{ a^i b^i c^j d^j \mid i,j \geq 0 \}$ in DCF - LIN

- $\{ a^i b^i c^j d^j \mid i, j \geq 0 \}$ in CFL − LIN

  $= \{ a^i b^i \mid i \geq 0\} \cdot \{c^j d^j \mid j \geq 0 \}$ in LIN·LIN
- not closed under concatenation

  $= \{ a^i b^i \mid i \geq 0\} \cdot c^* d^* \cap a^* b^* \cdot \{c^j d^j \mid j \geq 0 \}$
- not closed under intersection
- closed under finite state transductions:
  (inverse) morphism, intersection regular
  use machine model $\longrightarrow$
- not closed under star

  $T(\{ a^i b^i \mid i \geq 0\}^*) = \{ a^i b^i c^j d^j \mid i, j \geq 0 \}$

▷

As we have seen, both the context-free and the reguar languages have characterizations using grammars as well as using automata.

Here we show the same holds for the linear languages, they are accepted by one-turn push-down automata, where the stack behaviour consists of two phases, the first one adding to the stack, the second one popping.

This cannot be directly derived from the classical PDA to CFG triplet construction, as this will not generally yield a linear grammar when one starts with a one-turn pushdown.

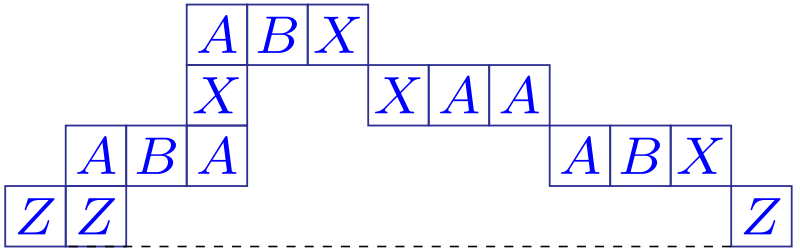
◁

RLIN = FSA

LIN = 1tPD

CF = PD

one-turn pushdown automata



$Q = Q^+ \cup Q^-$, $q_{in} \in Q^+$

$(\, p, a, A, q, \alpha \,) \in \delta$ then $\begin{cases} p, q \in Q^+ \text{ and } |\alpha| \geq 1, \text{ or} \\ p \in Q,\ q \in Q^- \text{ and } |\alpha| \leq 1 \end{cases}$

standard construction:

$(\, p, a, A, q, BC \,) \in \delta$ then

$$[\, p, A, r \,] \to a [\, q, B, s \,] [\, s, C, r \,]$$

not linear

$$[\, p, A, q \,] \Rightarrow_G^* w \iff (\, p, w, A \,) \vdash^* (\, q, \varepsilon, \varepsilon \,)$$

here $q \in Q^-$

*

$$\delta(p, a, A) \ni (q_1, B_1 \cdots B_n)$$
$$[\, p, A, q \,] \to a\, [\, q_1, B_1, q_2 \,]\, \underbrace{[\, q_2, B_2, q_3 \,] \cdots [\, q_n, B_n, q \,]}_{\text{generate regular languages}}$$

$$p, q_1 \in Q, q, q_2, \ldots, q_r \in Q^-$$
$$B_1, \ldots, B_r \in \Gamma \ (1 \le r \le \text{max-rhs})$$

$p \in Q^-$ if $(q, \alpha) \in \delta(p, a, A)$ then $q \in Q^-$, $|\alpha| \le 1$

$[\, p, A, r \,] \to a[\, q, B, r \,]$ $\qquad \delta(p, a, A) \ni (q, B)$

$[\, p, A, q \,] \to a$ $\qquad \delta(p, a, A) \ni (q, \varepsilon)$

include this information in $[\, q_1, B_1, q_2 \,]$

generate regular language(s) to the right

backwards! (left-linear grammar)

then next step pushdown

LIN / LIN = RE

later perhaps, Chapter 6

LIN *not* closed under quotient

extra exercise

*

7. Is the class of CFLs closed under the shuffle operation shuff $\|$ (introduced in Section 3.3)? How about perfect shuffle $\amalg$?

   not context-free
   $\{ ww \mid w \in \Sigma^* \}$
   $\{ a^n b^n c^n \mid n \geq 0 \}$
   $\{ a^n b^m a^n b^m \mid n, m \geq 0 \}$

   intersect shuffle with regular language

15. Let $G = (V, \Sigma, P, S)$ be a context-free grammar.

(a) Prove that the language of all sentential forms derivable from S is context-free.

(b) Prove that the language consisting of all sentential forms derivable by a leftmost derivation from S is context-free.

variables V become terminals
simulated by 'new' variables

leftmost derivations are precisely simulated when constructing PDA for CFG

Chapter 5

Parsing and recognition

expand-match

expand $(q, \varepsilon, A, q, \alpha)$ for each production $A \to \alpha$
match $(q, a, a, q, \varepsilon)$ for each terminal a

leftmost derivations
pre-order

shift-reduce

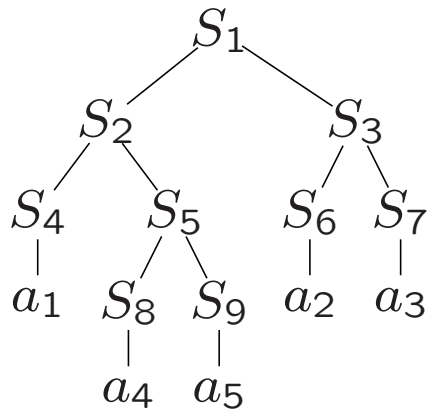
'extended' pda, multiple pop
top-of-stack right

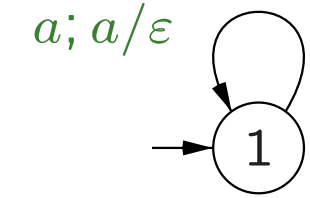
shift $(q, a, \varepsilon, q, a)$ for each terminal a
reduce $(q, \varepsilon, \alpha, q, A)$ for each production $A \to \alpha$

rightmost derivations, backwards
post-order

$L = a^+$

$S \to SS \mid a$





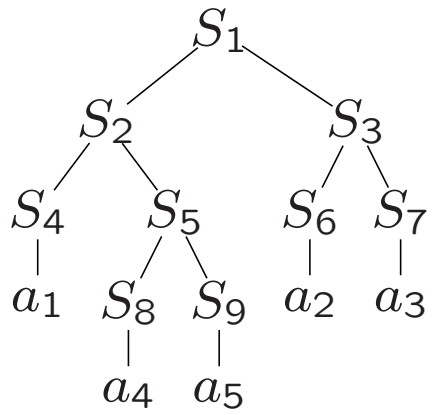
$$
\begin{array}{llll}
(1, & aaaaa, & S_1) & \vdash \\
(1, & aaaaa, & S_2S_3) & \vdash \\
(1, & aaaaa, & S_4S_5S_3) & \vdash \\
(1, & aaaaa, & aS_5S_3) & \vdash_m \\
(1, & aaaa, & S_5S_3) & \vdash \\
(1, & aaaa, & S_8S_9S_3) & \vdash \\
(1, & aaaa, & aS_9S_3) & \vdash_m \\
(1, & aaa, & S_9S_3) & \vdash \\
(1, & aaa, & aS_3) & \vdash_m \\
(1, & aa, & S_3) & \vdash \\
(1, & aa, & S_6S_7) & \vdash \\
(1, & aa, & aS_7) & \vdash_m \\
(1, & a, & S_7) & \vdash \\
(1, & a, & a) & \vdash_m \\
(1, & \varepsilon, & \varepsilon) &
\end{array}
$$

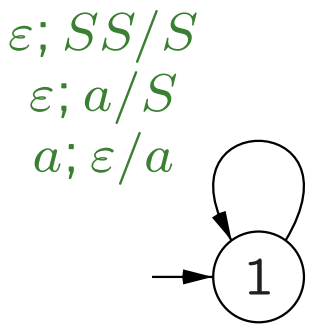
pre-order $\equiv$ leftmost

expand
$(q, \varepsilon, A, q, \alpha)$ for $A \to \alpha$
match
$(q, a, a, q, \varepsilon)$ for $a \in \Sigma$

$L = a^+$

$S \to SS \mid a$





$$
\begin{array}{lrll}
(1, & aaaaa, & Z &) \vdash_s \\
(1, & aaaa, & a &) \vdash \\
(1, & aaaa, & S_4 &) \vdash_s \\
(1, & aaa, & S_4a &) \vdash \\
(1, & aaa, & S_4S_8 &) \vdash_s \\
(1, & aa, & S_4S_8a &) \vdash \\
(1, & aa, & S_4S_8S_9 &) \vdash \\
(1, & aa, & S_4S_5 &) \vdash \\
(1, & aa, & S_2 &) \vdash_s \\
(1, & a, & S_2a &) \vdash \\
(1, & a, & S_2S_6 &) \vdash_s \\
(1, & a, & S_2S_6a &) \vdash \\
(1, & \varepsilon, & S_2S_6S_7 &) \vdash \\
(1, & \varepsilon, & S_2S_3 &) \vdash \\
(1, & \varepsilon, & S_1 &)
\end{array}
$$

shift
$(q, a, \varepsilon, q, a)$ for $a \in \Sigma$
reduce
$(q, \varepsilon, \alpha, q, A)$ for $A \to \alpha$

'extended' pda, multiple pop
top-of-stack right

post-order $\equiv$ rightmost in reverse

5.1 Recognition and parsing in general grammars

Cocke, Younger, and Kasami

$w \in L(G)$?

$w = a_1 a_2 \dots a_n$ and $w[i..j] = a_i \dots a_j$

A in $U[i,j]$ iff $A \Rightarrow^* w[i..j]$

initialization

A in $U[i,i]$ iff $A \to a_i$

recursion

A in $U[i,j]$ iff $A \to BC$ and
B in $U[i,k]$, C in $U[k+1,j]$ $(i \leq k < j)$

Thm. $w \in L(G)$ for given CFG G in $\mathcal{O}(n^3)$ $n = |w|$

A in $U[i,i]$ iff

$A \to a_i$

A in $U[i,j]$ iff

$A \to BC$ and

B in $U[i,k]$,

C in $U[k+1,j]$

$(i \leq k < j)$

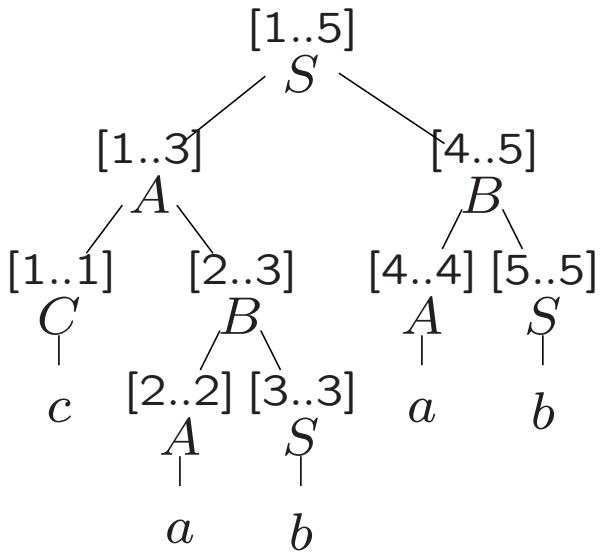
$$
\begin{array}{lcl}
S & \to & AB \mid b \\
A & \to & CB \mid AA \mid a \\
B & \to & AS \mid b \\
C & \to & BS \mid c
\end{array}
$$

dynamic programming

$w = cabab$

i/j	1	2	3	4	5
1	C	$-$	A	A	S, B
2		A	S, B	$-$	C
3			S, B	$-$	C
4				A	S, B
5					S, B

$w \in L(G)$ as $S \in U[1,5]$



$$
\begin{aligned}
S &\to AB \mid b\\
A &\to CB \mid AA \mid a\\
B &\to AS \mid b\\
C &\to BS \mid c
\end{aligned}
$$

$w = cabab$

i/j	1	2	3	4	5
1	C	–	$A : (C, B, 1)$	$A : (A, A, 3)$	$S : (A, B, 3), (A, B, 4)$ $B : (A, S, 3), (A, S, 4)$
2		A	$S : (A, B, 2)$ $B : (A, S, 2)$	–	$C : (B, S, 3)$
3			S, B	–	$C : (B, S, 3)$
4				A	$S : (A, B, 4)$ $B : (A, S, 4)$
5					S, B

Thm. parse tree in $\mathcal{O}(n^3)$ steps, $n = |w|$

5.2 Earley's method

item $A \to \alpha \bullet \beta$ for $A \to \alpha\beta$
complete item $A \to \alpha\bullet$

$w = a_1 a_2 \ldots a_n$
`make-early-table` (M_{ij}) $A \to \alpha \bullet \beta$ in M_{ij}
$S \Rightarrow^* w[1..i]A\delta$ and $\alpha \Rightarrow^* w[i+1, j]$

A. for every $S \to \gamma$ in P
put $S \to \bullet\gamma$ in M_{00}

'scanning' B. for $A \to \alpha \bullet a_k\beta$ in M_{ik-1}
put $A \to \alpha a_k \bullet \beta$ in M_{ik}

'completion' C. if $A \to \alpha \bullet B\beta$ in M_{ij} and $B \to \gamma\bullet$ in M_{jk}
put $A \to \alpha B \bullet \beta$ in M_{ik}

'prediction' D. if $A \to \alpha \bullet B\beta$ in M_{ik} then
for every $B \to \gamma$ in P put $B \to \bullet\gamma$ in M_{kk}

Thm. $w \in L$ iff $S \to \alpha\bullet \in M_{0,n}$ for some α

$S \to T + S \mid T$
$T \to F * T \mid F$
$F \to (S) \mid a$

$w = (a + a) * a$

initialization & prediction

$$M_{0,0} \left| \begin{array}{llll} S \to \bullet T + S & T \to \bullet F * T & F \to \bullet(S) \\ S \to \bullet T & T \to \bullet F & F \to \bullet a \end{array} \right.$$

scanning $(_1$

$$M_{0,1} \left| \begin{array}{l} F \to (\bullet S) \end{array} \right.$$

[completion &] prediction

$$M_{1,1} \left| \begin{array}{llll} S \to \bullet T + S & T \to \bullet F * T & F \to \bullet(S) \\ S \to \bullet T & T \to \bullet F & F \to \bullet a \end{array} \right.$$

scanning a_2

$$M_{1,2} \left| \begin{array}{l} F \to a\bullet \end{array} \right.$$

completion [& prediction]

$$M_{1,2} \left| \begin{array}{ll} T \to F \bullet * T & S \to T \bullet + S \\ T \to F\bullet & S \to T\bullet \end{array} \right.$$
$$M_{0,2} \left| \begin{array}{l} F \to (S\bullet) \end{array} \right.$$

Thm. $C \to \eta \bullet \gamma$ in $M_{u,v}$ iff $C \to \eta\gamma$ in P, $\eta \Rightarrow^* w[u+1..v]$ and $S \Rightarrow^* w[1..u]C\delta$ for some δ.

Thm. $O(n^3)$ where $n = |w|$

Thm. G unambiguous, no useless symbols, no unit productions, no ε-productions, then $O(n^2)$ where $n = |w|$

5.3 Top-down parsing

expressions, terms

$\Sigma = \{\ a, +, [,]\ \}$
$V = \{\ E, Z, X, T\ \}$

$E \to TZ$
$Z \to X \mid \varepsilon$
$X \to +TZ$
$T \to a \mid [E]$

	E	Z	X	T
a	$E \to TZ$			$T \to a$
$+$		$Z \to X$	$X \to +TZ$	
$[$	$E \to TZ$			$T \to [E]$
$]$		$Z \to \varepsilon$		
$\$$		$Z \to \varepsilon$		

$z = a\ +\ [\ a\ +\ a\ ]$

$E \overset{E,a}{\Rightarrow} TZ \overset{T,a}{\Rightarrow} aX \overset{X,+}{\Rightarrow} a{+}TZ \overset{T,[}{\Rightarrow}$

$a{+}[E]Z \overset{E,a}{\Rightarrow} a{+}[TZ]Z \overset{T,a}{\Rightarrow} a{+}[aZ]Z \overset{Z,+}{\Rightarrow}$

$a{+}[aX]Z \overset{X,+}{\Rightarrow} a{+}[a{+}TZ]Z \overset{T,a}{\Rightarrow} a{+}[a{+}aZ]Z \overset{Z,]}{\Rightarrow}$

$a{+}[a+a]Z \overset{Z,\$}{\Rightarrow} a{+}[a+aZ]$

grammar is *strong* LL(k)

$S \Rightarrow_\ell^* u_1 A v_1 \Rightarrow_\ell u_1 \alpha_1 v \Rightarrow_\ell^* u_1 x_1 \qquad A \to \alpha_1$

$S \Rightarrow_\ell^* u_2 A v_2 \Rightarrow_\ell u_2 \alpha_2 v \Rightarrow_\ell^* u_2 x_2 \qquad A \to \alpha_2$

$u_i, x_i \in \Sigma^*,\ A \in V,\ \alpha_i \in (\Sigma \cup V)^*$

if $\text{first}_k(x_1) = \text{first}_k(x_2)$ then $\alpha_1 = \alpha_2$

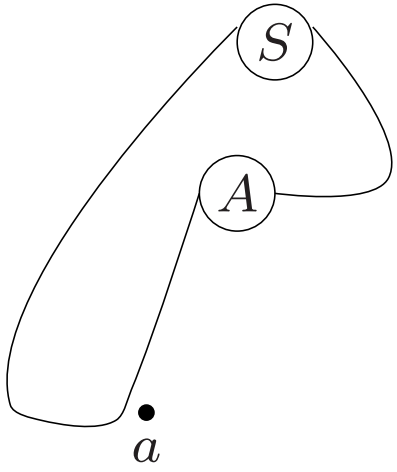
grammar is LL(k) $\quad u_1 = u_2$

$A \to aBaa \mid aBba \qquad B \to b \mid \varepsilon$

$A \Rightarrow aBaa \Rightarrow abaa \mid aaa$

$A \Rightarrow bBba \Rightarrow bbba \mid bba$

is LL(2), not 'strong'



$S \Rightarrow_\ell^* uAv \Rightarrow_\ell^* uax = z$

$(u \in \Sigma^*,\ a \in \Sigma,\ A \in V,\ v \in \Sigma^*)$

A and a determine production $A \to \alpha$

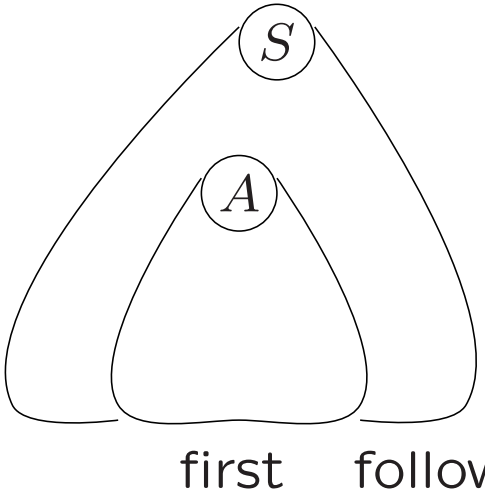
$\text{first}(\alpha) = \{x \in \Sigma^* \mid \alpha \Rightarrow^* x\}$

$\text{follow}(A) = \{x \in \Sigma^* \mid S \Rightarrow_\ell^* uAv \text{ and } v \Rightarrow^* x\}$

$a \in \text{init}_1(\ \text{first}(\alpha)\ \text{follow}(A)\ )$ for some $A \to \alpha$

LL(k): first k symbols here $k = 1$

complication when ε in first



$\text{FIRST}(u) = \text{init}_1(\text{first}(u))$

$\text{FOLLOW}(A) = \text{init}_1(\text{follow}(A))$

$\text{init}_1(\varepsilon) = \varepsilon,\ \text{init}_1(ax) = a$

$E \to TZ$
$Z \to X \mid \varepsilon$
$X \to +TZ$
$T \to a \mid [E]$

i	E	Z	X	T
0	$\varnothing$	ε	$+$	$a, [$
1	$a, [$	$\varepsilon, +$	$+$	$a, [$
2	$a, [$	$\varepsilon, +$	$+$	$a, [$

$F_i(a) = \{a\}$

$F_0(A) = \{\, a \in \Sigma \mid A \to ay \in P \,\}$
$\quad \cup \{\varepsilon \mid A \to \varepsilon \in P\}$

$F_{i+1}(A) = F_i(A) \cup$

$$\bigcup_{A \to Y_1 \cdots Y_m \in P} \mathsf{init}_1(F_i(Y_1) \cdot \cdots \cdot F_i(Y_m))$$

repeat until $F_{i+1} = F_i$

Thm. $a \in \mathsf{FIRST}(A)$ iff $a \in \bigcup_{i \in \mathbb{N}} F_i(A)$

Prf. if $A \Rightarrow^n ax$ $(a \in \Sigma)$ then $a \in F_n(A)$

$\mathsf{FIRST}(A) = \mathsf{init}_1\{x \in \Sigma^* \mid A \Rightarrow^* x\}$

if $A \Rightarrow^n ax$ $(a \in \Sigma)$ then $a \in F_n(A)$

$n = 1$, $A \Rightarrow ax$, hence $a = \mathsf{init}_1(F_0(a)F_0(x))$ for the production $A \to ax$

may be empty?

$E \to TZ$
$Z \to X \mid \varepsilon$
$X \to +TZ$
$T \to a \mid [E]$

i	E	Z	X	T
0	], \$	$\varnothing$	$\varnothing$	+
1	], \$	], \$	$\varnothing$	+,], \$
2	], \$	], \$	], \$	+,], \$
3	], \$	], \$	], \$	+,], \$

$$FL_0(A) = \{\, a \in \Sigma \mid B \to uAv, a \in \mathsf{FIRST}(v) \,\} \cup \{\, \$ \mid A = S \,\}$$

$$FL_{i+1}(A) = FL_i(A) \cup \bigcup_{\substack{B \to uAv \in P \\ \varepsilon \in \mathsf{FIRST}(v)}} FL_i(B)$$

Thm. $a \in \mathsf{FOLLOW}(A)$ iff $a \in \bigcup_{i \in \mathbb{N}} FL_i(A)$

5.4 Removing LL(1) conflicts

construct equivalent grammar in LL(1) form
do not always work: e.g.,

$L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$

$S \to A \mid B$
$A \to aAb \mid \varepsilon$
$B \to aBbb \mid \varepsilon$
cannot be made into a LL(k) form for any k

immediate left recursion $E \to E\alpha$

$$E \to E\alpha_1 \mid \cdots \mid E\alpha_k \mid \underbrace{\beta_1 \mid \cdots \mid \beta_j}_{\text{do not start with E}}$$

$$E \Rightarrow^* \beta_.\alpha_. \ldots \alpha_.$$

$$E \to \beta_1 E' \mid \cdots \mid \beta_j E'$$
$$E' \to \alpha_1 E' \mid \cdots \mid \alpha_k E' \mid \varepsilon$$

(statement) ::= if (expression)
then (statement) [else (statement)]

$S \to \mathsf{i}E\mathsf{t}S\mathsf{e}S \mid \mathsf{i}E\mathsf{t}S \mid \mathsf{x}$
$E \to \mathsf{y}$

common prefix $\mathsf{i}E\mathsf{t}S$

$S \to \mathsf{i}E\mathsf{t}SS' \mid \mathsf{x}$
$S' \to \mathsf{e}S \mid \varepsilon$
$E \to \mathsf{y}$

factoring

5.5 Bottom-up Parsing

LR(0)
left-to-right scan of input
rightmost derivation (in reverse)
item $A \to \alpha \bullet \beta$
right sentential form $\Rightarrow_r^*$
α handle of $\delta\alpha w$ $S \Rightarrow_r^* \delta A w \Rightarrow_r \delta\alpha w$
viable prefix
valid item $S \Rightarrow_r^* \delta A w \Rightarrow_r \delta\alpha\beta w$, where $\gamma = \delta\alpha$

Ex. $S \to T$

$T \to$ aTa | bTb | c

rightmost derivation [duh: linear grammar]

$S \Rightarrow T \Rightarrow aTa \Rightarrow abTba \Rightarrow abcba$

$\gamma = abTba$ right sentential form

bTb handle of γ

$T \to bT \bullet b$ is valid for viable abT: $\Rightarrow^* aTa \Rightarrow_r abTba$

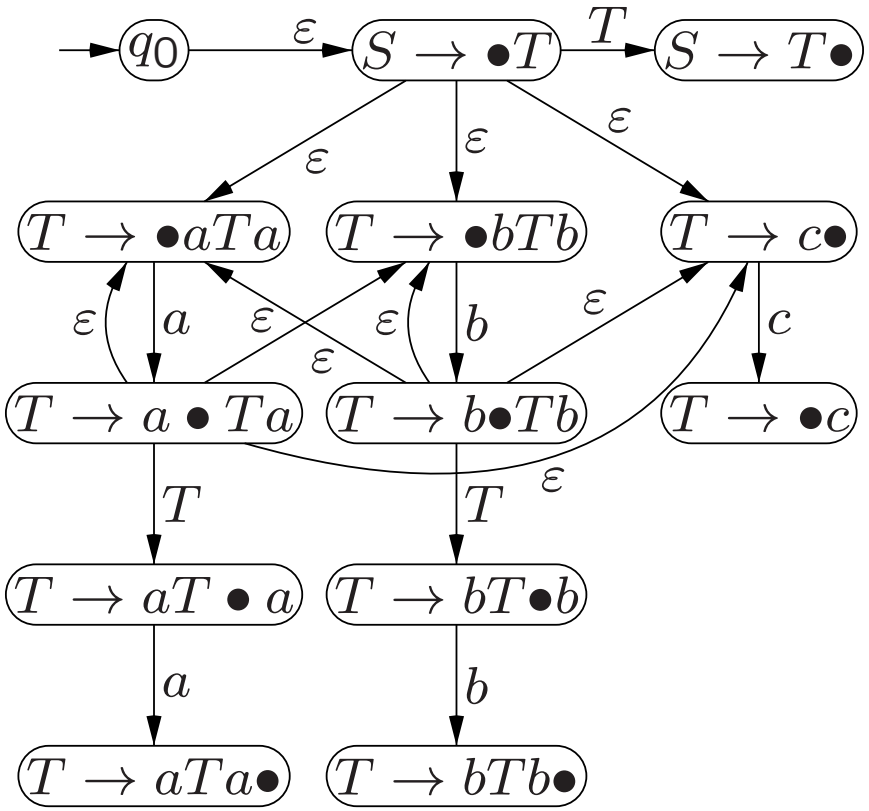
Knuth automaton for grammar G

accepts viable prefixes

state: corresponding valid items

Thm. G without useless symbols
Knuth NFA-ε for G
$A \to \alpha \bullet \gamma$ in $\delta(q_0, \gamma)$ iff $A \to \alpha \bullet \gamma$ valid for γ

Def. G has no useless symbols
axiom S not at right-hand side
if $A \to \alpha\bullet$ is valid for viable γ, then no other complete item



$S \to T$

$T \to$ aTa | bTb | c

$Q = items(G) \cup \{q_0\}$

$\Delta = \Sigma \cup V$

$F = Q$

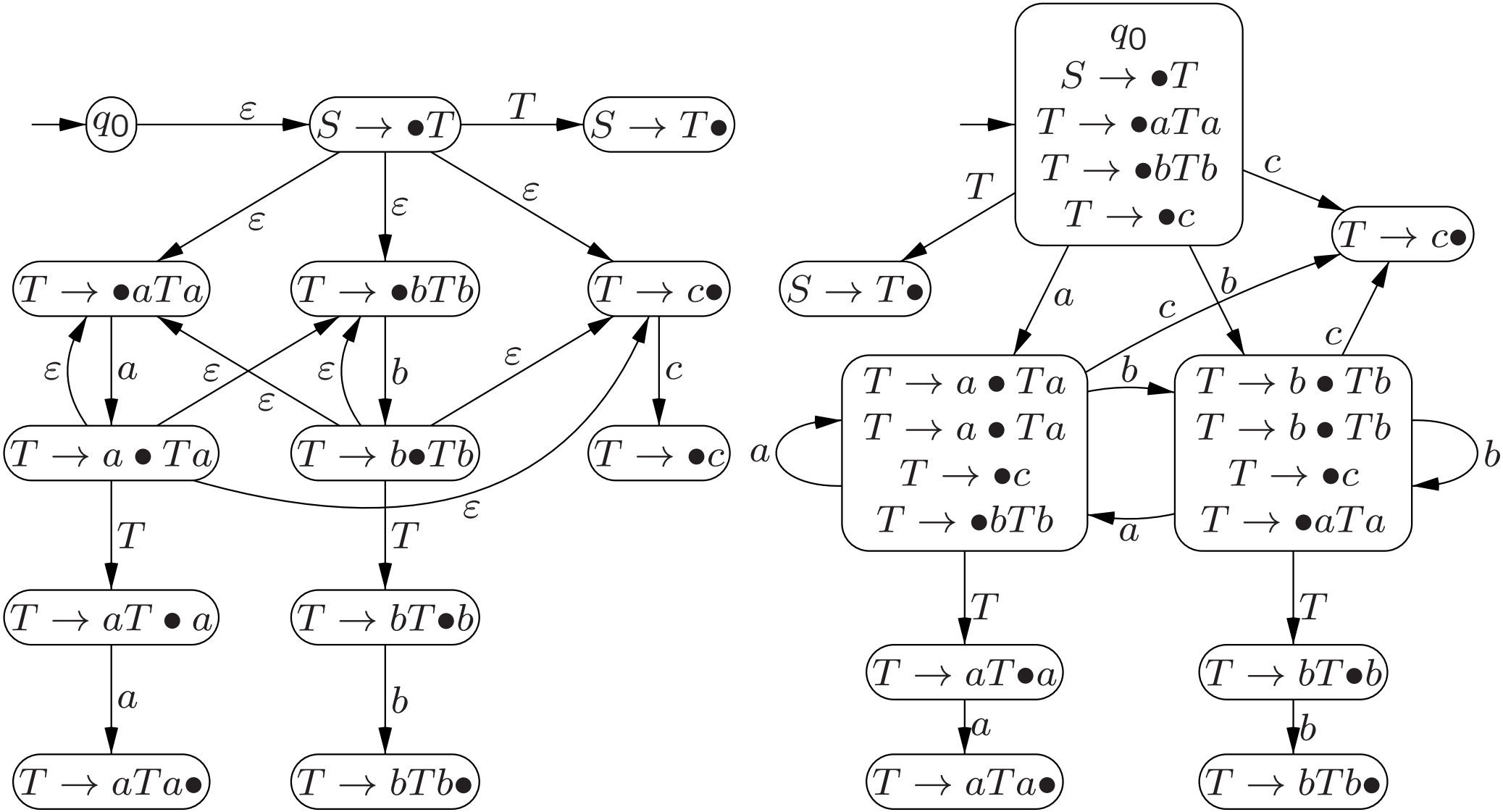
$\delta(q_0, \varepsilon) = \quad \{S \to \bullet\alpha \mid S \to A \text{ in } G\}$

$\delta(A \to \alpha \bullet B\beta, \varepsilon) =$

$\{B \to \bullet\rho \mid B \to \rho \text{ in } G\}$

$\delta(A \to \alpha \bullet X\beta, X) = \quad \{A \to \alpha X \bullet \beta\}$

Theorem

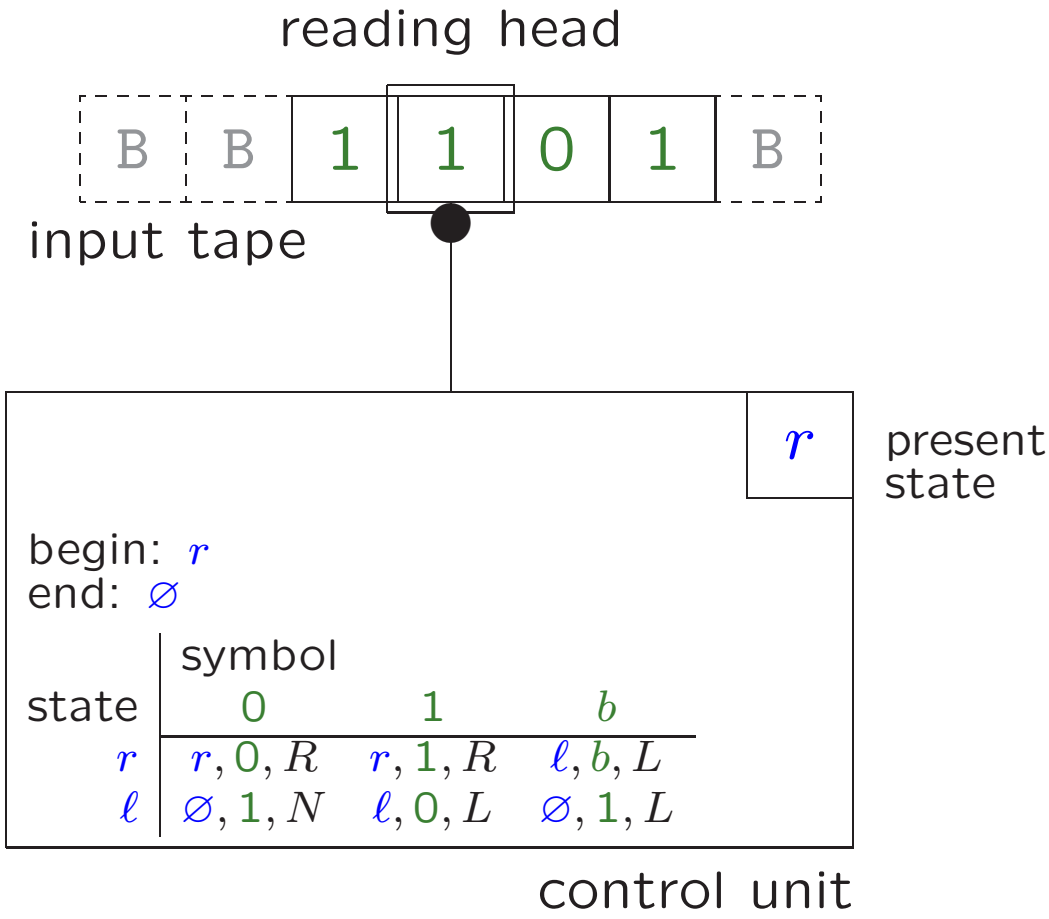

q_0
$S \to \bullet T$
$S \to T\bullet$
$T \to \bullet aTa$
$T \to \bullet bTb$
$T \to \bullet c$
$T \to c\bullet$
$T \to a \bullet Ta$
$T \to b\bullet Tb$
$T \to aT \bullet a$
$T \to bT\bullet b$
$T \to aTa\bullet$
$T \to bTb\bullet$
ε
a
b
c
T
q_0
$S \to \bullet T$
$T \to \bullet aTa$
$T \to \bullet bTb$
$T \to \bullet c$
$S \to T\bullet$
$T \to c\bullet$
$T \to a \bullet Ta$
$T \to a \bullet Ta$
$T \to \bullet c$
$T \to \bullet bTb$
$T \to b \bullet Tb$
$T \to b \bullet Tb$
$T \to \bullet c$
$T \to \bullet aTa$
$T \to aT\bullet a$
$T \to bT\bullet b$
$T \to aTa\bullet$
$T \to bTb\bullet$
a
b
c
T

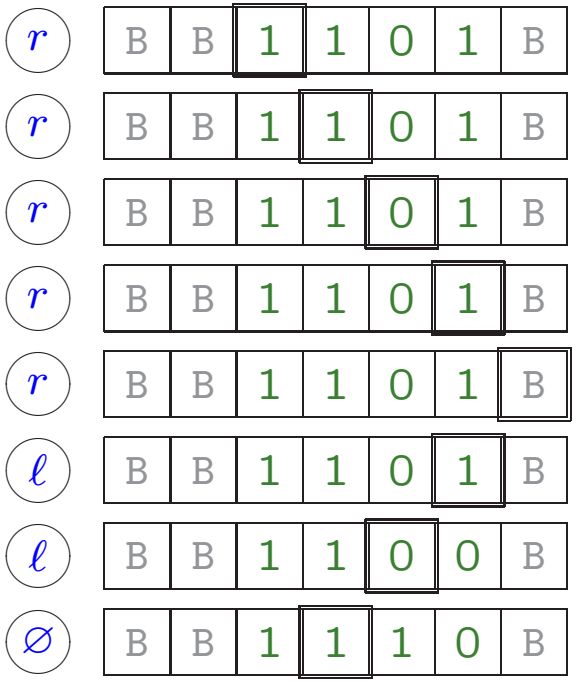
Theorem

Chapter 6

Turing machines

6.0 Review
6.1 Unrestricted grammars
6.2 Kolmogorov complexity
6.3 The incrompressibility method
...


reading head
B B 1 1 0 1 B
input tape
r
present state
begin: r
end: ∅
symbol
state 0 1 b
r r, 0, R r, 1, R ℓ, b, L
ℓ ∅, 1, N ℓ, 0, L ∅, 1, L
control unit


r B B 1 1 0 1 B
r B B 1 1 0 1 B
r B B 1 1 0 1 B
r B B 1 1 0 1 B
r B B 1 1 0 1 B
ℓ B B 1 1 0 1 B
ℓ B B 1 1 0 0 B
∅ B B 1 1 1 0 B

$$\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, h)$$

Q	states
$q_0 \in Q$	initial state
$h \in Q$	halting state
Γ	tape alphabet
$\Sigma \subseteq \Gamma$	input alphabet $\quad \mathtt{B} \in \Gamma - \Sigma$

transition function (partial)

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$$

(nondet) $$\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R, S\}$$

wqx configuration $w, x \in \Gamma^*,\ q \in Q$

$$\begin{array}{lll} \alpha Z p X \beta \vdash \alpha q Z Y \beta & \text{if} & \delta(p, X) = (q, Y, L) \\ \alpha p X \beta \vdash \alpha Y q \beta & \text{if} & \delta(p, X) = (q, Y, R) \\ \alpha p X \beta \vdash \alpha q Y \beta & \text{if} & \delta(p, X) = (q, Y, S) \end{array}$$

$L(\mathcal{M}) =$

$\{\ x \in \Sigma^* \mid q_0 \mathtt{B} x \vdash^* \alpha h \beta$ for some $\alpha, \beta \in \Gamma^*\ \}$

- blocks: no move defined		*'no'*
- move off tape (one-sided)		*'no'*
- infinite computation 'loop' cannot tell)		*'no'* (but we
- halt in state h		*'yes'*

RE	recursively enumerable	*enumerate*
REC	recursive — TM always stops	*decide*

REC closed complementation, RE is not

equivalent variants:
multiple tapes
one sided, two sided
two-dimensional tape
nondeterminism

> “ It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathcal{U}$ is supplied with a tape on the beginning of which is written the S.D. [=description] of some computing machine $\mathcal{M}$, then $\mathcal{U}$ will compute the same sequence as $\mathcal{M}$. ”

A.M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem.* Proc. London Math. Soc. Ser. 2 42, 230-265, 1937. doi:10.1112/plms/s2-42.1.230

universal TM $\mathcal{M}_U$:
on input $e(T)e(x)$, $\mathcal{M}_U$ simulates T on input x and halts if and only if T halts on x.

$\Gamma_U = \{\mathsf{B} = a_0, a_1, a_2, \dots\}$
$Q_U = \{q_0, q_1, q_2, \dots\}$
$e(a_i) = 0^{i+1}$ $e(q_i) = 0^{i+1}$

alphabet $\Sigma = \{b_1, \dots, b_r\}$
$e(\Sigma) = 111\, e(b_1)\, 1\, e(b_2)\, 1 \cdots 1\, e(b_r)\, 111$

$e(L) = 0$, $e(R) = 00$, $e(S) = 000$

instruction m: $\delta(q, a) = (p, b, D)$
$e(m) = 11\, e(q)\, 1\, e(a)\, 1\, e(p)\, 1\, e(b)\, 1\, e(D)\, 11$

TM $\mathcal{M}$ with instructions $m_1, \dots, m_t$
$e(\mathcal{M}) =$
$11111\, e(q_0)\, 1\, e(h)\, 1\, e(\Sigma)\, 1\, e(\Gamma)\, 1\, e(m_1)\, 1 \cdots 1\, e(m_t)\, 11111$

1.7 Unsolvability

problem $\equiv$ language

$L_P = \{$ code$(x) \mid P(x)$ holds $\}$ 'yes' instances

P solvable $\equiv$ L_P recursive

halting problem:
given Turing machine T and input w, does T halt on w?

unsolvable (undecidable, uncomputable)

halting language
$\{ e(T)e(w) \mid T$ halts on $w \}$
in RE $-$ REC

> “ We can show further that *there can be no machine* $\mathcal{E}$ *which, when supplied with the S.D of an arbitrary machine* $\mathcal{M}$*, will determine whether* $\mathcal{M}$ *ever prints a given symbol (*0 *say).* ”

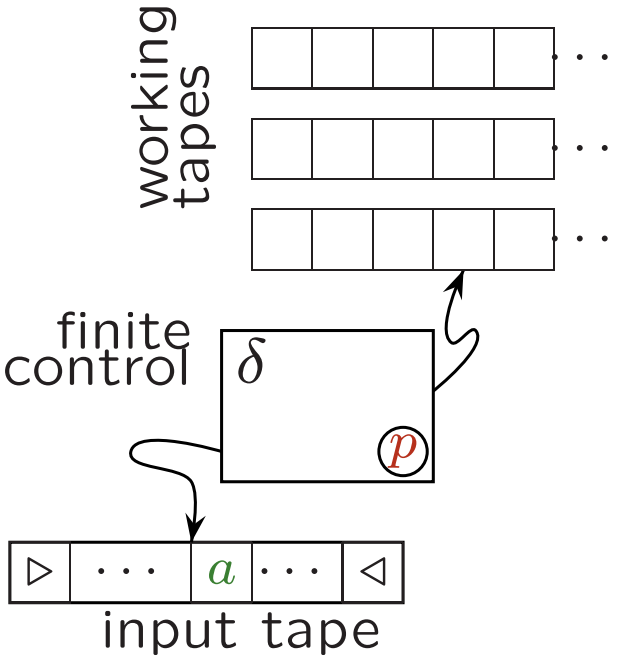
1.8 Complexity theory

▷

Walter J. Savitch Deterministic simulation of non-deterministic turing machines (Detailed Abstract)

STOC '69 Proceedings of the first annual ACM symposium on Theory of computing 247 - 248 (1969)

doi:10.1145/800169.805439 ◁



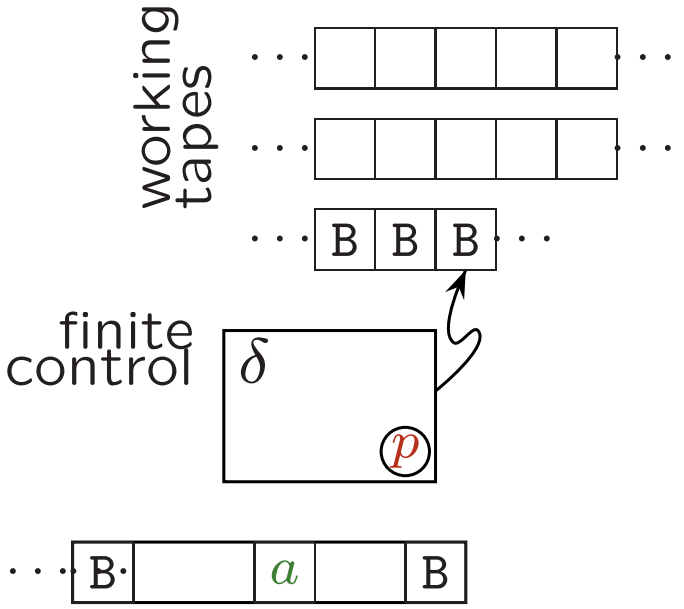
DSPACE(f) NSPACE(f)

space complexity

online Turing machine

multiple working tapes

single sided



DTIME(f) NTIME(f)

time complexity

input on tape

multiple working tapes

double sided

time complexity $T(n)$

P vs. NP

Karp reduction $L_1 \leq L_2$

$x \in L_1$ iff $f(x) \in L_2$

f computable in polynomial time

NP complete:

- $L \in$ NP
- $L' \leq L$ for every $L' \in$ NP

Savitch $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{DSPACE}(f(n)^2)$

$f(n) \geq \log n$

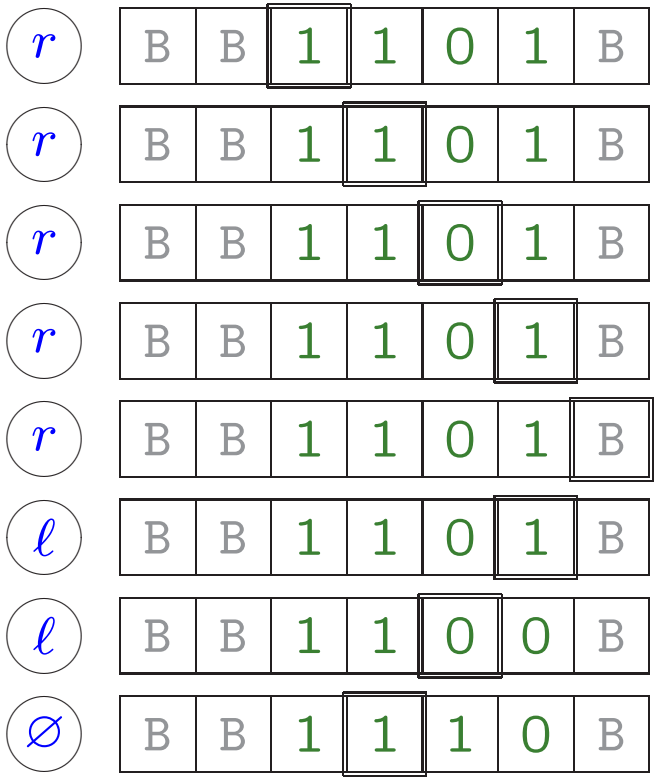
PSPACE = NPSPACE

QBF is NSPACE complete (Cook-Levin)

$(Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n)\, \phi(x_1, x_2, \ldots, x_n)$

$A = Q_2 x_2 \cdots Q_n x_n\, \phi(0, x_2, \ldots, x_n)$
$B = Q_2 x_2 \cdots Q_n x_n\, \phi(1, x_2, \ldots, x_n)$

$Q_1 = \exists\colon\ A \vee B$, $Q_1 = \forall\colon\ A \wedge B$
use recursion
stack depth max n
closed formula $\phi(\vec{x})$



at step k

T_{iak} tape cell i contains symbol a

H_{ik} read/write head is at tape cell i

Q_{qk} M is in state q

computation M on $x \iff \varphi$ satisfiable

▷

Cook, Stephen (1971). "The complexity of theorem proving procedures". Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151–158.

Levin, Leonid (1973). "Universal search problems (in Russian, Universal'nye perebornye zadachi)". Problems of Information Transmission (Russian: Problemy Peredachi Informatsii) 9 (3): 265–266. (Russian), translated into English by Trakhtenbrot, B. A. (1984). "A survey of Russian approaches to perebor (brute-force searches) algorithms". Annals of the History of Computing 6 (4): 384–400. doi:10.1109/MAHC.1984.10036

wikipedia:Cook-Levin Theorem

◁

SAT is NP-hard

variables

T_{iak} tape cell i contains symbol a at step k

H_{ik} read/write head is at tape cell i at step k

Q_{qk} M is in state q at step k

conjunction of

$T_{ix[i]0}$ initial tape $x[i] = x_i$ or $x[i] = \mathtt{B}$

$Q_{q_0 0}$ initial state

H_{00} initial position head

$T_{iak} \to \neg T_{ibk}$ one symbol per tape cell $a \neq b$

$Q_{pk} \to \neg Q_{qk}$ one state at a time $p \neq q$

$H_{ik} \to \neg H_{jk}$ one head position at a time $i \neq j$

$T_{iak} \wedge T_{ib(k+1)} \to H_{ik}$ tape changed only if written $a \neq b$

$H_{ik} \wedge Q_{pk} \wedge T_{iak} \to \bigvee_{(p,a,q,b,d)\in\delta} H_{(i+d)(k+1)} \wedge Q_{q(k+1)} \wedge T_{ib(k+1)}$
possible transitions

$Q_{h\,p(n)}$ finish in accepting state h

CNF conjunctive normal form

conjunction $\bigwedge$ clauses (disjunction $\bigvee$ literals)

$(x_1 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee x_3 \vee x_4 \vee x_5)$

$a \to \neg b \iff \neg a \vee \neg b$

$a \wedge b \to c \iff \neg a \vee \neg b \vee c$

and finally

$a \wedge b \wedge c \to \bigvee_{i \in I}(d_i \wedge e_i \wedge f_i) \iff *$

$(\neg a \vee \neg b \vee \neg c \vee \bigvee_{i \in I} z_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee d_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee e_i) \wedge \bigwedge_{i \in I}(\neg z_i \vee f_i)$

3SAT is NP-complete

$a \vee b \vee c \vee d \vee e \iff * \ (a \vee b \vee x_1) \wedge (\neg x_1 \vee c \vee x_2) \wedge (\neg x_2 \vee d \vee e)$

3SAT is NP-complete

$\exists x_1\, \exists x_2\, \exists x_3\, \exists x_4 : (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$

TQBF is PSPACE-complete

$\exists x_1\, \forall x_2\, \exists x_3\, \forall x_4 : (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$

configuration c (sequence of variables T_{ia}, H_i, Q_q)

$\Phi_k(c_1, c_2)$ from c_1 to c_2 in (at most) k steps $\quad k \leq 2^{|x|}$

$k = 1$ see SAT construction

$\Phi_{2k}(c_1, c_2) = (\exists c)(\ \Phi_k(c_1, c) \wedge \Phi_k(c, c_2)\ )$ idea OK, but doubles size

$\Phi_{2k}(c_1, c_2) = (\exists c)(\forall c)(\forall c')(\ [\ (c, c') = (c_1, c) \vee (c, c') = (c, c_2)\ ] \rightarrow \Phi_k(c_1, c)\ )$

▷

Independently from Alan Turing, Emil Post developed a very similar 'mechanic' approach to formalize computability. It was published only slightly later.

Emil Post. Finite Combinatory Processes— Formulation 1. The Journal of Symbolic Logic, Vol. 1 (1936) 103-105.

...and a fixed unalterable set of directions which will both direct operations in the symbol space and determine the order in which those directions are to be applied.

The worker is assumed to be capable of performing the following primitive acts:
(a) Marking the box he is in (assumed empty),
(b) Erasing the mark in the box he is in (assumed marked),
(c) Moving to the box on his right,
(d) Moving to the box on his left,
(e) Determining whether the box he is in, is or is not marked. ◁

6.6 Unsolvability and context-free languages

valid computation $w_1\#w_2^R\#w_3\#w_4^R\#\ldots\#w_{2k}^R\#$ or
$w_1\#w_2^R\#w_3\#w_4^R\#\ldots\#w_{2k-1}\#$

- w_i configuration $\Gamma^*Q\Gamma^*$
- w_1 initial $q_0\mathrm{B}x$, $x \in \Sigma^*$
- w_n accepting yhz where h halting state
- $w_i \vdash w_{i+1}$ for $1 \leq i < n$

valid(M)
invalid$(M) = (\Gamma \cup Q \cup \{\#\})^* - \text{valid}(M)$

Thm. valid$(M) = L(G_1) \cap L(G_2)$ for cfg G_i, effect.

TM steps are 'local'

$\alpha ZpX\beta \vdash \alpha qZY\beta$ if $\delta(p, X) = (q, Y, L)$
$\alpha pX\beta \vdash \alpha Yq\beta$ if $\delta(p, X) = (q, Y, R)$
$\alpha pX\beta \vdash \alpha qY\beta$ if $\delta(p, X) = (q, Y, S)$

mirror $\alpha ZpX\beta\#\beta^R YZq\alpha^R$, etc.
G_1 odd $w_i\#w_{i+1}^R$ steps
G_2 even $w_i^R\#w_{i+1}$ steps

$L(M) = \varnothing$ is undecidable

Thm. undecidable $L(G_1) \cap L(G_2) = \varnothing$ for cfgs G_i

invalid $x_1 \# x_2 \# \ldots x_m \#$

- x_i not a configuration
- x_1 not initial
- x_m not final
- not $x_i \vdash x_{i+1}^R$ for odd i
- not $x_i^R \vdash x_{i+1}$ for even i

Thm. invalid$(M) = L(G)$ for cfg G, effectively

Cor. undecidable $L(G) = \Sigma^*$ for cfg G over Σ

Cor. undecidable $L(G_1) = L(G_2)$ for cfgs G_i

Thm. undecidable '$L(G)$ is regular' for cfg G

$L_0 \subseteq \Sigma^*$ nonregular context-free
$L_1 = L_0 \# \Sigma^* \cup \Sigma^* \# L(G)$
L_1 regular iff $L(G) = \Sigma^*$
$w \notin L(G)$ then $L_1 \cap (\Sigma^* \# w) = L_0 \# w$
$L(G) = \Sigma^*$ then $L_1 = \Sigma^* \# \Sigma^*$

Thm. if $L(G)$ regular, then finding FSA A with $L(A) = L(G)$ is not effective

$\Delta = \Gamma \cup Q \cup \{\#\}$
regular $R_x = q_0 (\Sigma^* - \{y\}) \# \Delta^*$
$L_x = \text{invalid}(M) \cup R_x$ effectively context-free
if $x \in L(M)$ then $L_x = \Delta^* - \{w\}$,
w accepting computation x
if $x \notin L(M)$ then $L_x = \Delta^*$
regular in both cases!

linear languages: $\alpha p X \beta \# \cdots \# \beta^R Y q \alpha^R$

'step'

$$K = \{\, \gamma_1 \$ \gamma_2 \# \ldots \# \gamma_n \# \gamma_n'^R \# \ldots \gamma_2'^R \# \gamma_1'^R \mid \gamma_i \vdash \gamma_i' \,\}$$

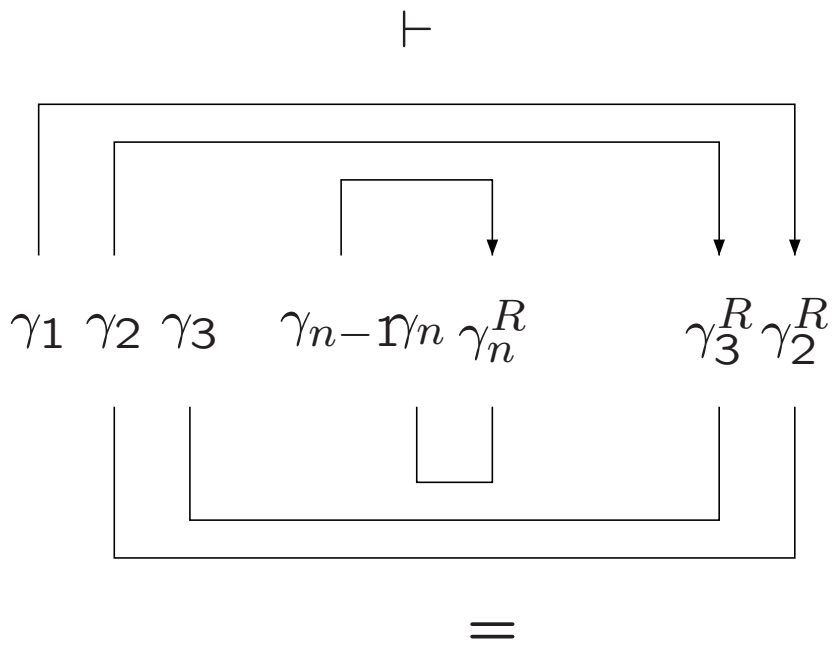
'copy'

$$L = \{\, \$\gamma_1 \# \gamma_2 \# \ldots \# \gamma_n \# \gamma_n \# \ldots \gamma_2 \# \gamma_1 \mid \gamma_i \text{ config} \,\}$$

also: γ_n final (halting)

quotients yields γ_1 with valid computation

detail: γ_1 has state



$$K = \{\ \gamma_1\$\gamma_2\#\ldots\#\gamma_n\#\gamma_n'^R\#\ldots\gamma_2'^R\#\gamma_1'^R \mid \gamma_i \vdash \gamma_i'\ \}$$

$$L = \{\ \$\gamma_1\#\gamma_2\#\ldots\#\gamma_n\#\gamma_n\#\ldots\gamma_2\#\gamma_1 \mid \gamma_i \text{ config}\ \}$$

Chapter 7

Other language classes

7.1 Context-sensitive languages
7.2 The Chomsky hierarchy
7.3 2DPDAs and Cook's theorem

7.1 Context-sensitive languages

CS

context-sensitive $G = (V, \Sigma, P, S)$

productions $\alpha B \gamma \to \alpha \beta \gamma$

$B \in V$, $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, $\beta \neq \varepsilon$

$(\alpha, B, \gamma) \to \beta$ $\quad$ $B \to \beta$ in *context* (α, γ)

rewriting $\xi_1 \alpha B \gamma \xi_2 \Rightarrow \xi_1 \alpha \beta \gamma \xi_2$

$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$

$S \to ABSc$
$S \to Abc$
$BA \to XA$
$XA \to XB$
$XB \to AB$
$Bb \to bb$
$A \to a$

$\{ a^n b^n c^n \mid n \geq 1 \}$ in CS - CF

$BA \Rightarrow XA \Rightarrow XB \Rightarrow AB$

$S \Rightarrow ABSc \Rightarrow ABABScc \Rightarrow^* (AB)^{n-1} Sc^{n-1} \Rightarrow$
$(AB)^{n-1} Abc^n = A(BA)^{n-1} bc^n \Rightarrow^*$
$A(AB)^{n-2} ABbc^n \Rightarrow$
$A(AB)^{n-2} Abbc^n = AA(BA)^{n-2} bbc^n \Rightarrow^*$
$AA(AB)^{n-2} bbc^n = AA(AB)^{n-3} ABbbc^n \Rightarrow$
$AA(AB)^{n-3} Abbbc^n = AAA(BA)^{n-3} bbbc^n \Rightarrow^*$
$A^n b^n c^n \Rightarrow^* a^n b^n c^n$

MON

length-increasing (monotone)

$G = (V, \Sigma, P, S)$

productions $\alpha \to \beta \in P$ $\quad |\alpha| \leq |\beta|$

rewriting $\gamma_1 \alpha \gamma_2 \Rightarrow \gamma_1 \beta \gamma_2$

$L(G) = \{\, w \in \Sigma^* \mid S \Rightarrow^* w \,\}$

Ex. $S \to aBSc$

$S \to abc$

$Ba \to aB$

$Bb \to bb$

$L(G) = \{\, a^n b^n c^n \mid n \geq 1 \,\}$

MON=CS

Thm. length-increasing iff context-sensitive

$CDE \to JKLMN$

$CDE \to X_1DE$
$X_1DE \to X_1X_2E$
$X_1X_2E \to X_1X_2X_3$
$X_1X_2X_3 \to JX_2X_3$
$JX_2X_3 \to JKX_3$
$JKX_3 \to JKLMN$

special symbols X_i used here only
only one production for X_i

reorder steps

$$\alpha_1 CDE\beta_1 \Rightarrow \alpha_1 X_1 DE\beta_1 \Rightarrow^* \alpha_2 X_1 DE\beta_2 \Rightarrow \alpha_2 X_1 X_2 E\beta_2$$

$$\alpha_1 \Rightarrow^* \alpha_2 \qquad DE\beta_1 \Rightarrow^* DE\beta_2$$

$$\alpha_1 CDE\beta_1 \Rightarrow^* \alpha_1 CDE\beta_2 \Rightarrow \alpha_1 X_1 DE\beta_2 \Rightarrow \alpha_1 X_1 X_2 E\beta_2 \Rightarrow^* \alpha_2 X_1 DE\beta_2$$

etcetera

LBA

linear-bounded automaton

one-tape nondeterministic Turing machine
with tape markers $\triangleright$ $\triangleleft$ ($\sharp$, $\flat$)
$\sim$ linear space TM

Thm. CS $\subseteq$ LBA

introduce 'tracks'
technically $\Sigma \rightsquigarrow \Sigma \times \Sigma_1 \times \Sigma_2$
new alphabets Σ_i

simulate derivation steps nondeterministically

Thm. LBA $\subseteq$ CS

basic simulation

$\triangleright\, a\, b\, X\, q\, a\, B\, a\, A \triangleleft$ in state q reading a

	LBA	MON
move right	$\delta(p, X) = (q, Y, R)$	$pX \to Yq$
stationary	$\delta(p, X) = (q, Y, S)$	$pX \to qY$
move left	$\delta(p, X) = (q, Y, L)$	$ZpX \to qZY$

state q and markers $\triangleright$, $\triangleleft$ have to disappear

need composite symbols

combine with neighbouring tape symbol

$[pX]$, $[\triangleright X]$, $[X \triangleleft]$, $[p \triangleright X]$, $[\triangleright pX]$, $[pX \triangleleft]$, $[Xp \triangleleft]$, $[p \triangleright X \triangleleft]$, $[\triangleright pX \triangleleft]$, $[\triangleright Xp \triangleleft]$

$\triangleright\, a\, b\, X\, q\, a\, B\, a\, A \triangleleft \rightsquigarrow [\triangleright a]\, b\, X\, [qa]\, B\, a\, [A \triangleleft]$

add *many* cases for the rules

cases

$..\,pXZ\,..\Rightarrow\,..\,YqZ\,..$

$\triangleright pXZ\,..\Rightarrow\triangleright YqZ\,..$

$..\,pXZ\triangleleft\Rightarrow\,..\,YqZ\triangleleft$

$\triangleright pXZ\triangleleft\Rightarrow\triangleright YqZ\triangleleft$

$..\,pX\triangleleft\Rightarrow\,..\,Yq\triangleleft$

$\triangleright pX\triangleleft\Rightarrow\triangleright Yq\triangleleft$

initialize tape $(a\in\Sigma)$

$S\to[q_0\triangleright a\triangleleft]\mid[q_0\triangleright a]\,S_1$

$S_1\to aS_1\mid[a\triangleleft]$

move right $\delta(p,X)=(q,Y,R)$ $(X\neq\triangleright)$

$[pX]\,Z\to Y\,[qZ]$ *+ variants*

$[\triangleright pX]\,Z\to[\triangleright Y]\,[qZ]$

$[pX]\,[Z\triangleleft]\to Y\,[qZ\triangleleft]$

$[\triangleright pX]\,[Z\triangleleft]\to[\triangleright Y]\,[qZ\triangleleft]$

$[pX\triangleleft]\to[Yq\triangleleft]$ (i.e., $Z=\triangleleft$)

$[\triangleright pX\triangleleft]\to[\triangleright Yq\triangleleft]$

at left marker $\delta(p,\triangleright)=(q,\triangleright,R)$

$[p\triangleright Z]\to[\triangleright qZ]$

$[p\triangleright Z\triangleleft]\to[\triangleright qZ\triangleleft]$

halt and clean

$[h\triangleright X]Y\to X[Y\$]$

$[X\$]Y\to X[Y\$]$

$[X\$][Y\triangleleft]\to XY$

$[h\triangleright X\triangleleft]\to X$

$[h\triangleright X][Y\triangleleft]\to XY$

(etcetera)

* stationary $\delta(p, X) = (q, Y, S)$

$[pX] \to [qY]$ + variants

$[\triangleright pX] \to [\triangleright qY]$ $[pX\triangleleft] \to [qY\triangleleft]$ $[\triangleright pX\triangleleft] \to [\triangleright qY\triangleleft]$

* move left $\delta(p, X) = (q, Y, L)$ $(X \neq \triangleleft)$

$Z[pX] \to [qZ]Y$ + variants

$[\triangleright Z][pX] \to [\triangleright qZ]Y$ $Z[pX\triangleleft] \to [qZ][Y\triangleleft]$ $[\triangleright Z][pX\triangleleft] \to [\triangleright qZ][Y\triangleleft]$

$[\triangleright pX] \to [q \triangleright Z]Y$ $[\triangleright pX\triangleleft] \to [q \triangleright Z][Y\triangleleft]$

$\delta(p, \triangleleft) = (q, \triangleleft, L)$

$[Zp\triangleleft] \to [qZ\triangleleft]$ $[\triangleright Zp\triangleleft] \to [\triangleright qZ\triangleleft]$

Thm. Every CSL is recursive

$x \in L$ is decidable
technically: by an always halting TM

generate *all* strings up to length $|x|$

Thm. CSL $\subset$ REC [strict]

diagonalization

fix alphabet $\{0,1\}$
(effectively) enumerate all CSG's G_i.
(effectively) enumerate all strings x_i.

define $L \subseteq \{0,1\}^*$ $\quad x_i \in L$ iff $x_i \notin L(G_i)$

$x_i \in L$ is decidable as $x_i \notin L(G_i)$ is decidable.

$L \neq L_i$ for all i

▷

N. Immerman, Nondeterministic space is closed under complementation, SIAM Journal on Computing 17 (1988) 935–938.

R. Szelepcsényi, The method of forcing for nondeterministic automata, Bulletin of the EATCS 33 (1987) 96–100. ◁

twenty boxes contain six bombs
opening a box with a bomb will detonate it

how can you 'prove' that box number one contains a bomb,
that is, without opening it

use a *nondeterministic* approach

(is not a practical method)

nondeterminism — Immerman and Szelepcsényi (1988)

Thm. CSL closed under complement

configuration:

state, tape contents, position head

on $|x| = n$, at most $C = |Q||\Gamma|^n(n+2)$ moves

accept with empty tape: $q_0 \triangleright x \triangleleft \vdash^* h \triangleright \mathrm{B}^n \triangleleft$

Prf. 1. count reachable configurations
2. complementation
3. implement on LBA

2. input: x string,

R number of reachable configurations

accept x when not accepted by $\mathcal{M}$,

when R different configurations γ, *not* $h \triangleright \mathrm{B}^n \triangleleft$,

can be guessed together with computation

$q_0 \triangleright x \triangleleft \vdash^* \gamma$

inductive counting

1. R_i configurations reachable in at most i steps (from $q_0 \triangleright x \triangleleft$)

$R_0 = 1$

$R_i \rightsquigarrow R_{i+1}$

$r = 0$ counter

for each configuration β
check all possible predecessors (as follows)
guess R_i different configurations γ together with computation in $\leq i$ moves
if $\gamma = \beta$ or $\gamma \vdash \beta$ then $r{+}{+}$

(when the guesses do not work, nothing is accepted)

linear bounded !

guessing a computation

$\gamma_0 \vdash \gamma_1 \vdash \cdots \vdash \gamma_i = \beta$

only two configurations at a time

$\gamma_k \vdash \gamma_{k+1}$

and β (makes three)

3. add 'track' to tape
linear bounded ($\sim$ linear space TM)

technically $\Sigma \rightsquigarrow \Sigma \times \Sigma_1 \times \Sigma_2$
new alphabets Σ_i

store number of configurations,
at most $C = |Q||\Gamma|^n(n+2)$

$n \lg |\Gamma| + \lg |Q| + \lg(n+2)$ bits linear

generate different configurations
(in lexicographic order)

7.2 The Chomsky hierarchy

7.3 2DPDAs and Cook's theorem

2DPDA push-down automaton

deterministic, two-way, endmarkers $\triangleright$ $\triangleleft$ ($\sharp$, $\flat$)

$\mathcal{M} = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, \delta, q_0, Z_0, F)$

Q	finite set states
Σ	input alphabet
Γ	tape alphabet
$\triangleright, \triangleleft$	tape endmarkers (not in Σ)
$q_0 \in Q$	initial state
$Z_0 \in \Gamma$	initial stack symbol
$F \subseteq Q$	final states
δ	(partial) transition function

$\delta : Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \times \Gamma \to Q \times \{-1, 0, 1\} \times \Gamma^*$

$\delta(q, a, X) = (p, j, \alpha)$

no ε-moves

Ex. $\{\, 0^i 1^i 2^i \mid i \geq 1 \,\}$ in 2DPDA - CFL

check input belongs to $\{\, 0^i 1^i \mid i \geq 1 \,\} 2^*$
as ordinary (deterministic) PDA

return to left tape marker

check input belongs to $0^* \{\, 1^i 2^i \mid i \geq 1 \,\}$

open: is CFL $\subseteq$ 2DPDA ?

Ex. $\{ ww \mid w \in \{0,1\}^* \}$ in 2DPDA - CFL

find middle
    move left: push length on stack
    move right, pop two symbols every step

copy second half on stack
    move right, from middle to end

find middle
    again, see above, on stack above w

compare 1st half (tape) with 2nd half (stack)
    note direction

aba$\underbrace{\text{abb}}$abab c abb

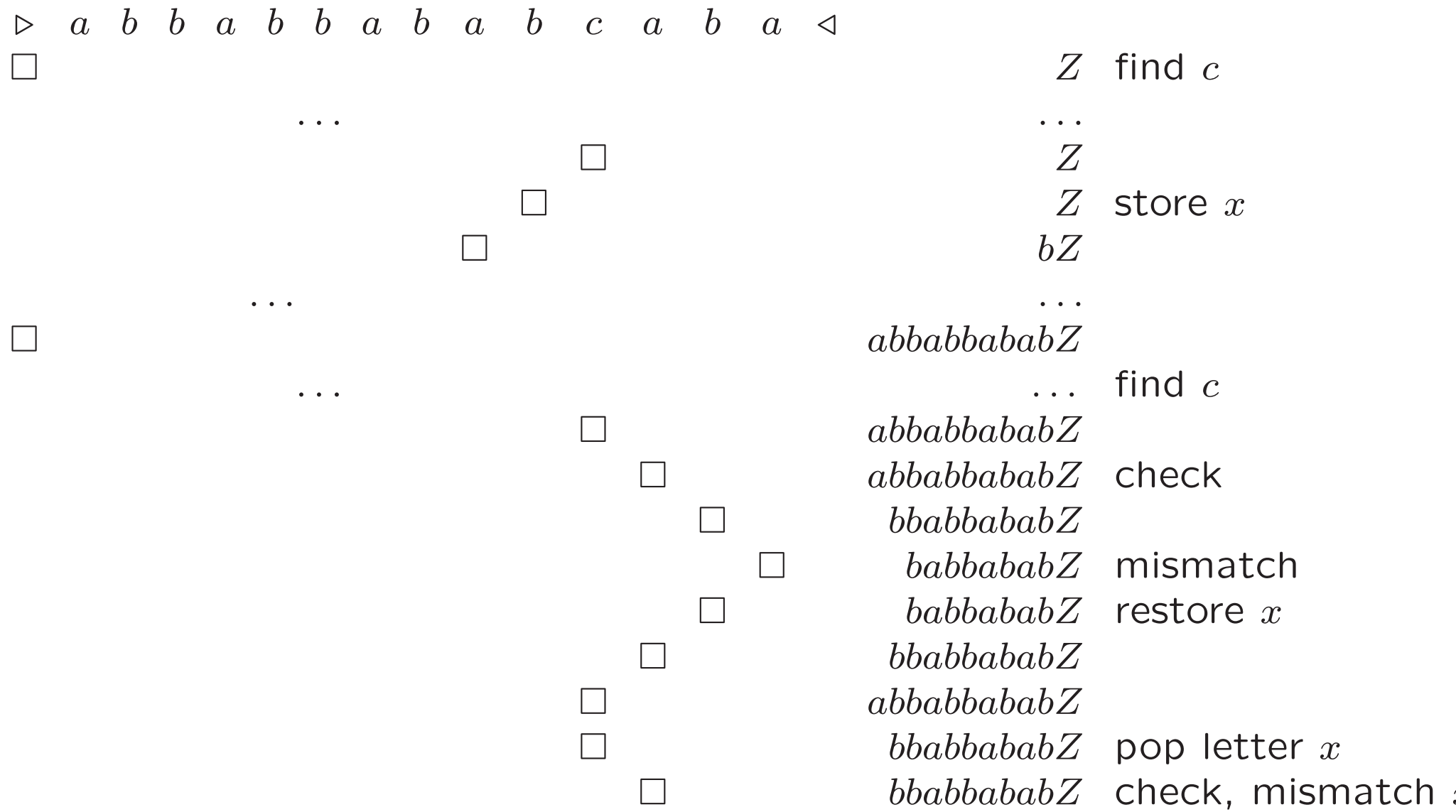
Ex. $\{\, xcy \mid x \in \{a,b\}^*,\ y$ subword of $x \,\}$ in 2DPDA

pattern matching: pattern y, text x

find match: move y along x

here:

move x over y


▷ a b b a b b a b a b c a b a ◁
□ Z find c
... ...
□ Z
□ Z store x
□ bZ
... ...
□ abbabbababZ
... ... find c
□ abbabbababZ
□ abbabbababZ check
□ bbabbababZ
□ babbababZ mismatch
□ babbababZ restore x
□ bbabbababZ
□ abbabbababZ
□ bbabbababZ pop letter x
□ bbabbababZ check, mismatch

A 2DPDA can be simulated in linear time on a RAM.

	RLIN REG	DPDA	CF PDAe	DLBA	MON LBA	REC	TYPE0 RE
intersection	+	−	−	+	+	+	+
complement	+	+	−	+	+	+	−
union	+	−	+	+	+	+	+
concatenation	+	−	+	+	+	+	+
star, plus	+	−	+	+	+	+	+
ε-free morphism	+	−	+	+	+	+	+
morphism	+	−	+	−	−	−	+
inverse morphism	+	+	+	+	+	+	+
intersect reg lang	+	+	+	+	+	+	+
mirror	+	−	+	+	+	+	+
	fAFL		fAFL	AFL	AFL	AFL	fAFL

$\cap\ ^c\ \cup$ boolean operations

$\cup\ \cdot\ ^*$ regular operations

$h\ h^{-1}\ \cap R$ (full) trio operations

Theorem