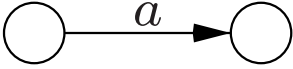


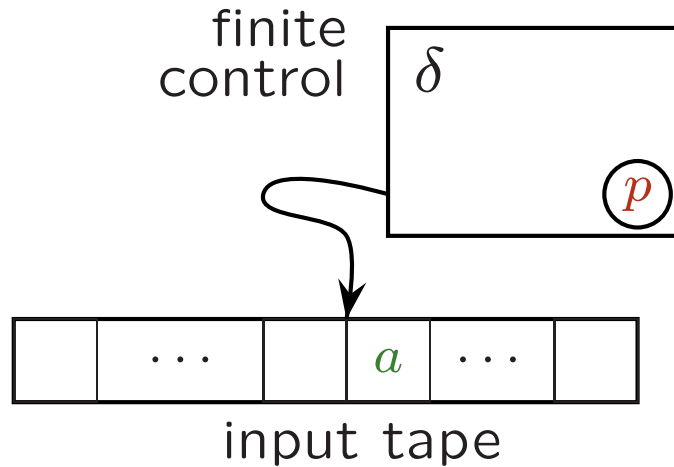
Chapter 1

Review of

Formal Languages and

Automata Theory

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



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

Q states p, q

$q_0 \in Q$ initial state

$F \subseteq Q$ final states

Σ input alphabet a, b w, x

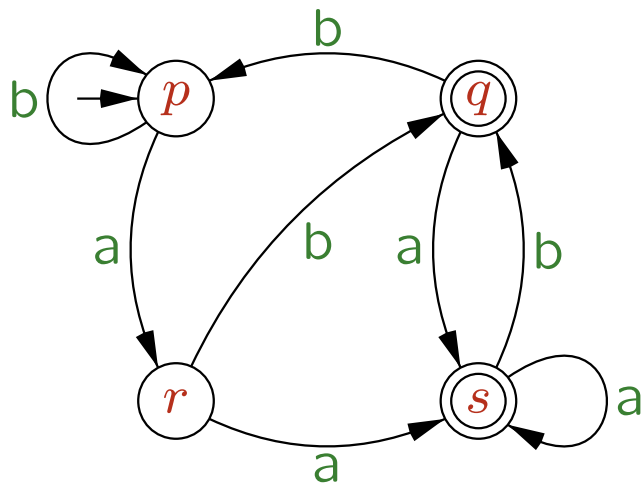
$\delta : Q \times \Sigma \rightarrow Q$ transition function

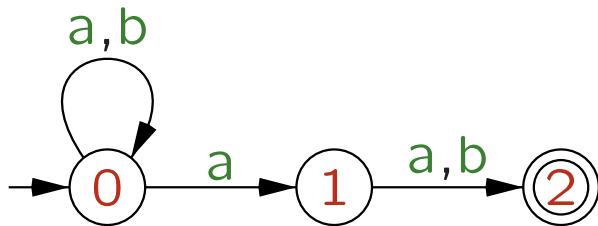
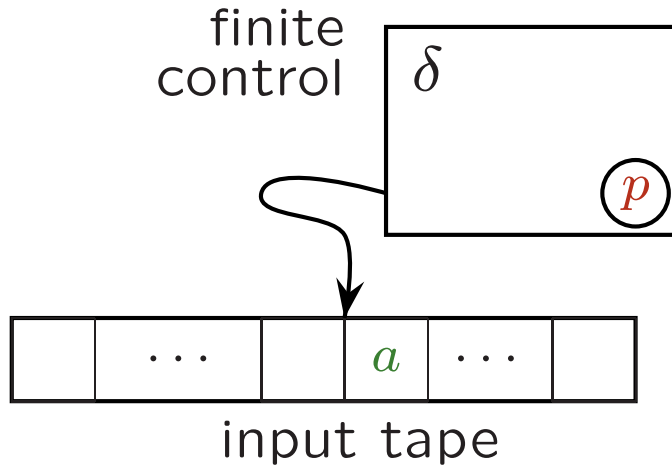
$$\delta^* : Q \times \Sigma^* \rightarrow Q$$

$$\delta^*(q, \epsilon) = q$$

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

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





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

$$\delta(2, a) = \emptyset$$

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

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

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

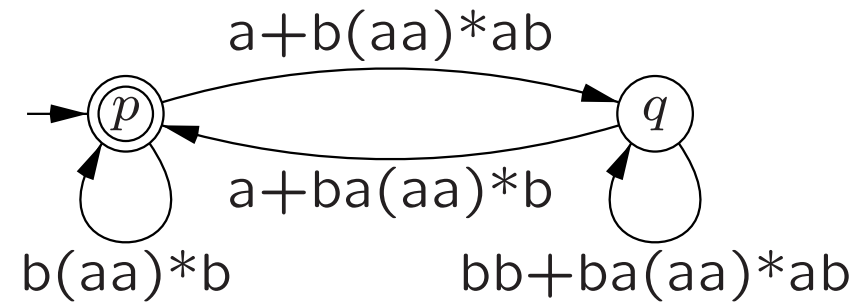
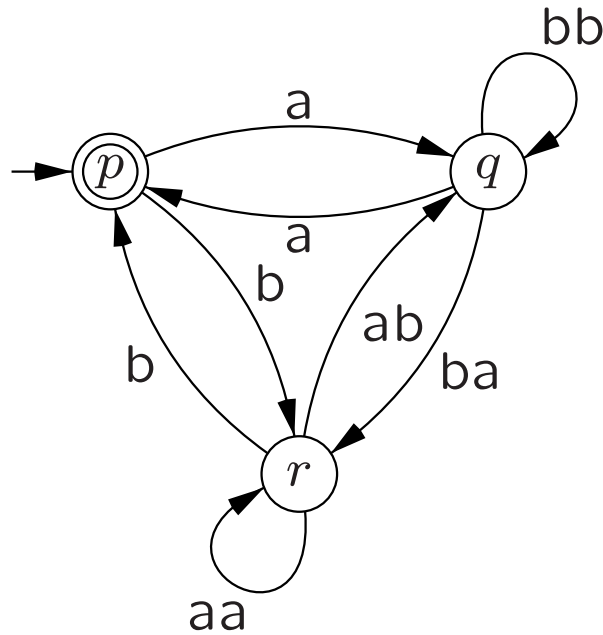
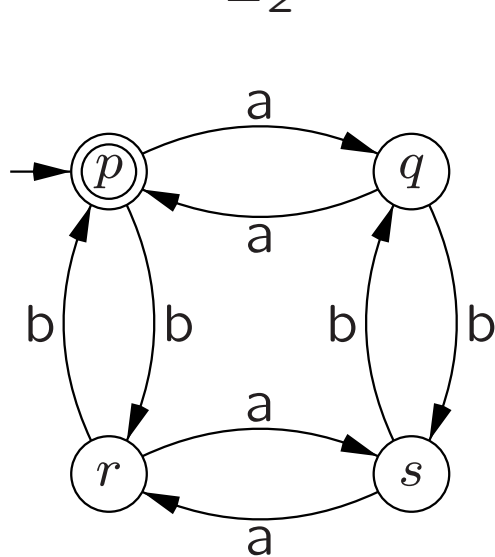
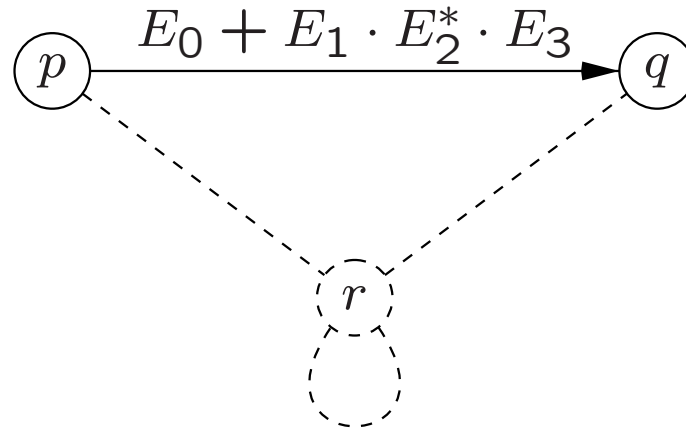
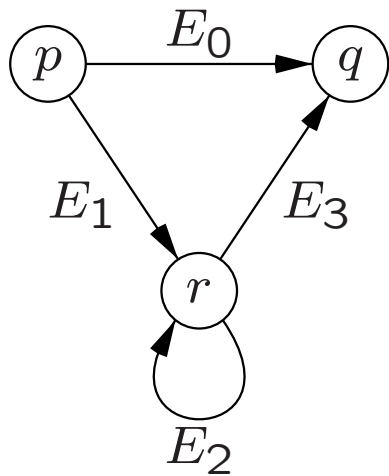
$$\delta^*(q, \epsilon) = \{q\}$$

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

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

$$\delta' : 2^Q \times \Sigma \rightarrow 2^Q \quad (\text{deterministic})$$

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



long words can be pumped

- ∀ for every regular language L
- ∃ there exists a constant $n \geq 1$
such that
- ∀ for every $z \in L$
with $|z| \geq n$
- ∃ there exists a decomposition $z = uvw$
with $|uv| \leq n$, $|v| \geq 1$
such that
- ∀ for all $i \geq 0$, $uv^i w \in L$

▷

The book uses a transition function

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma},$$

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

My personal favourite is a (finite) transition relation

$$\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma.$$

In the former one writes

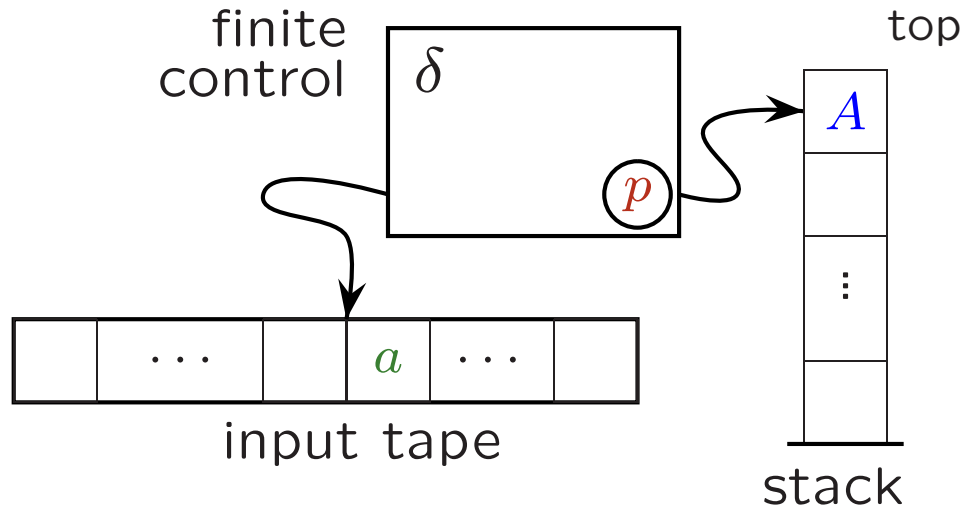
$$\delta(p, a, A) \ni (q, \alpha)$$

and in the latter

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

The meaning is the same.

◁



7-tuple

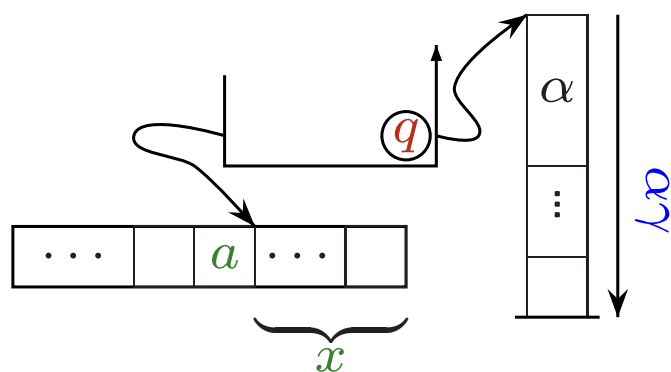
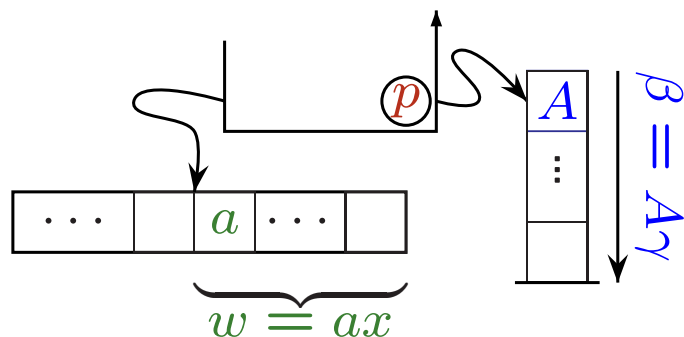
$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Q	<i>states</i>	p, q
$q_0 \in Q$	<i>initial state</i>	
$F \subseteq Q$	<i>final states</i>	
Σ	<i>input alphabet</i>	$a, b \quad w, x$
Γ	<i>stack alphabet</i>	$A, B \quad \alpha$
$Z_0 \in \Gamma$	<i>initial stack symbol</i>	

transition function (finite)

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

$$\underbrace{\left(\begin{array}{ccc} \text{from} & & \\ p & a & A \\ & \text{read} & \text{pop} \end{array} \right)}_{\text{before}} \ni \underbrace{\left(\begin{array}{ccc} \text{to} & & \\ q & \alpha & \\ & \text{push} & \end{array} \right)}_{\text{after}}$$



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

(p, w, β) $\left\{ \begin{array}{l} p \text{ state} \\ w \text{ input, unread part} \\ \beta \text{ stack, top-to-bottom} \end{array} \right.$

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

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

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

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

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

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’,

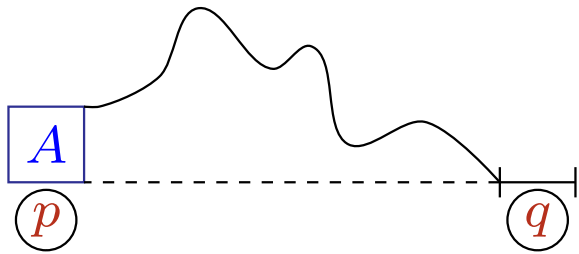
Evey ‘Application of Pushdown Store Machines’, and

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.

◁

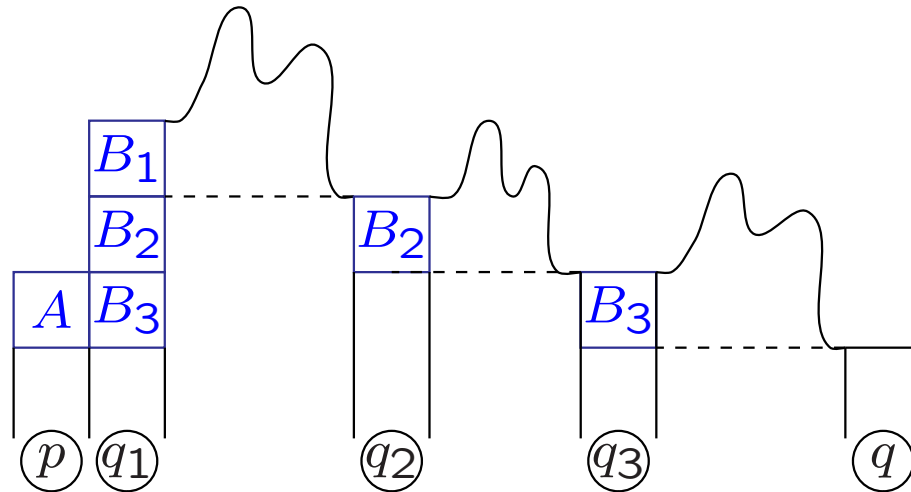


nonterminals $[p, A, q]$ $p, q \in Q, A \in \Gamma$

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

productions

$$S \rightarrow [q_{in}, Z_{in}, q] \quad \text{for all } q \in Q$$



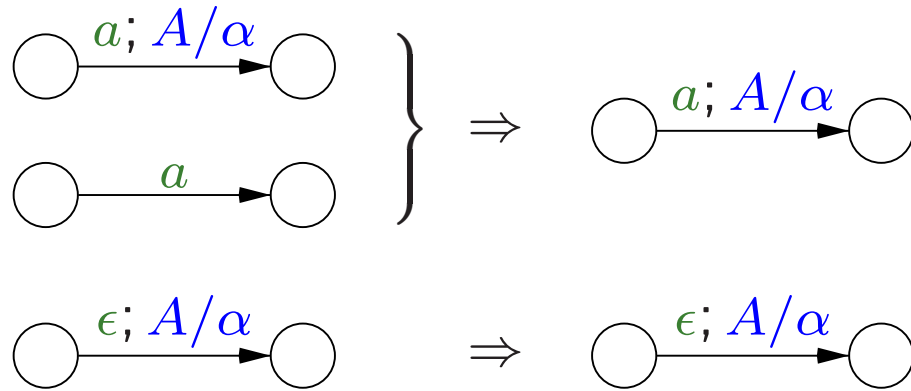
$$[p, A, q] \rightarrow a [q_1, B_1, q_2] [q_2, B_2, q_3] \cdots [q_n, B_n, q]$$

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

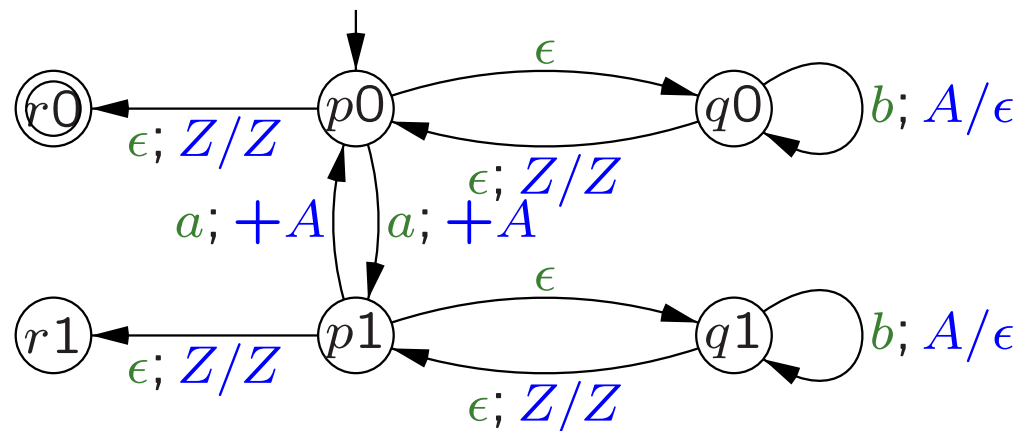
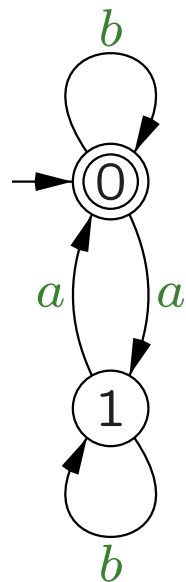
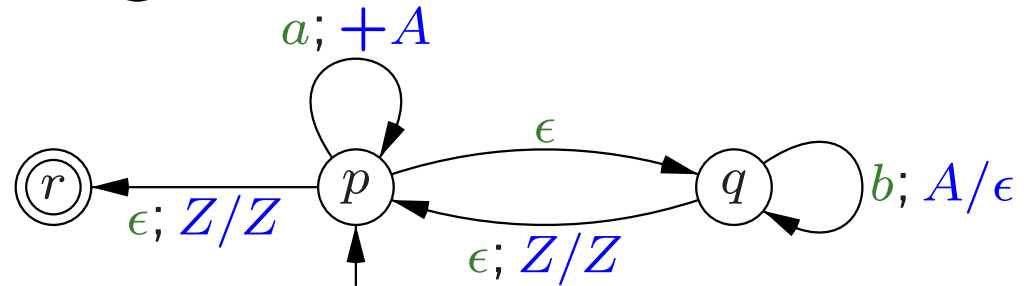
$$q, q_2, \dots, q_n \in Q$$

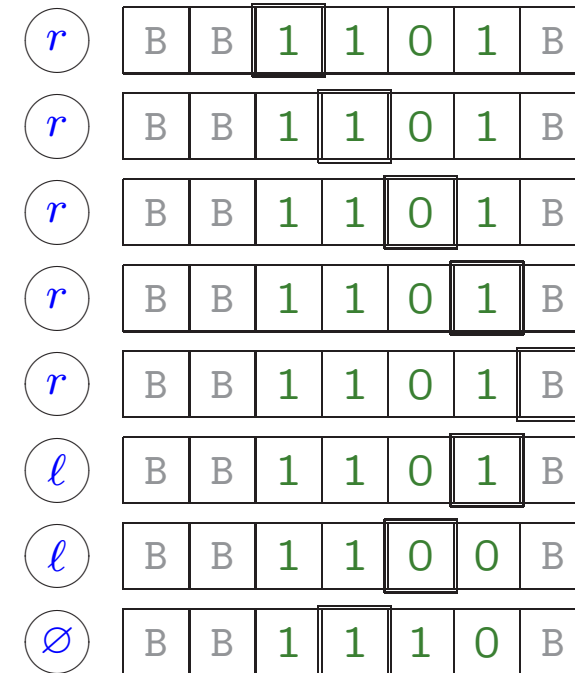
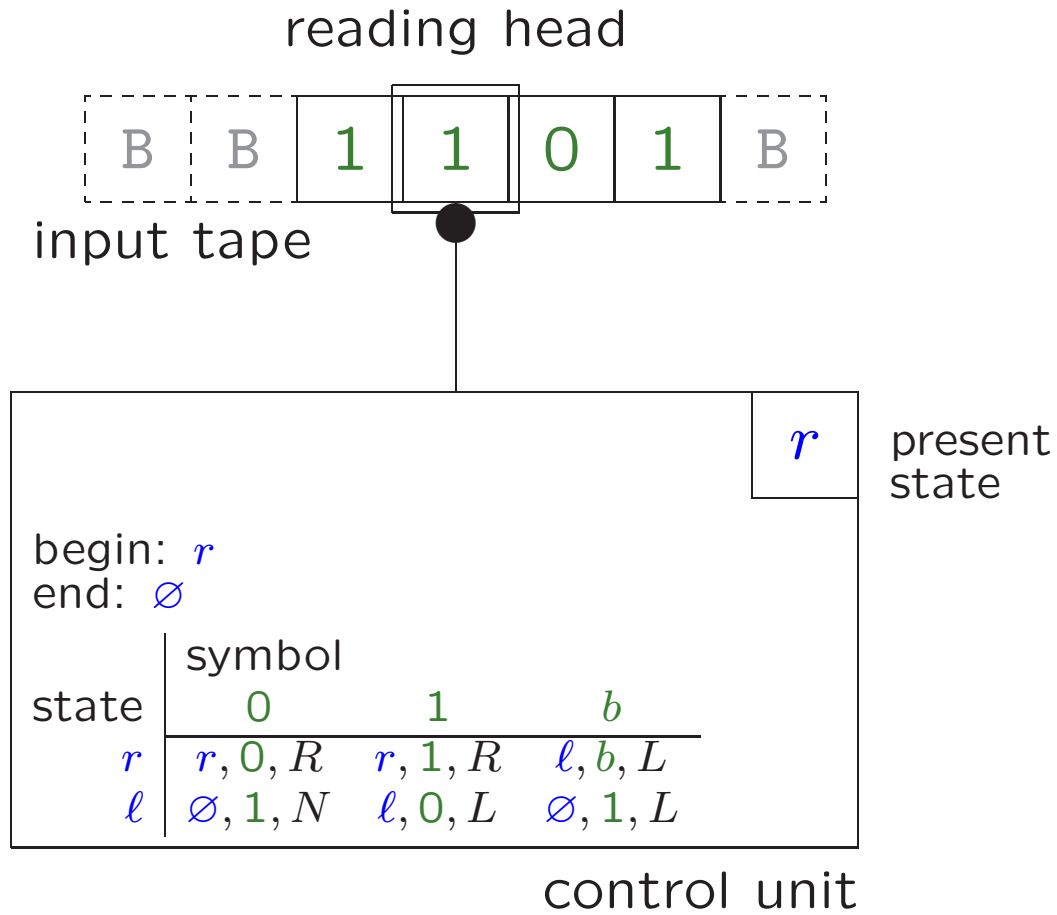
$$[p, A, q] \rightarrow a$$

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



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





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

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

transition function (partial)

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

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

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

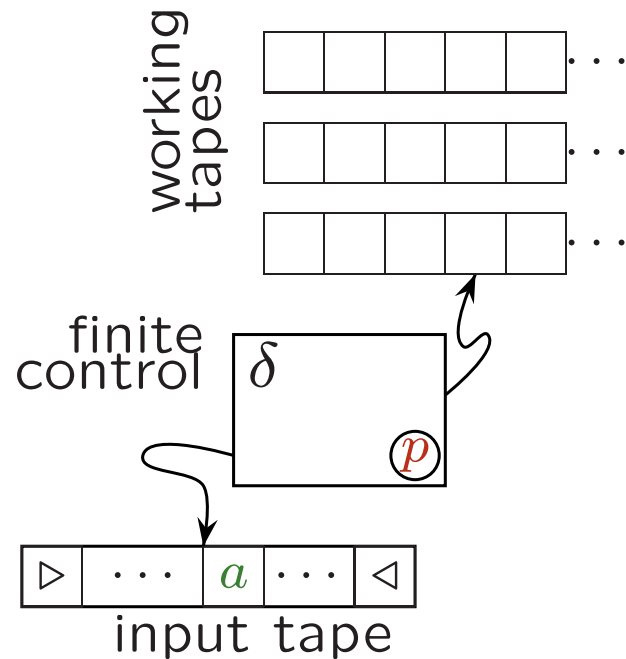
$$\alpha Z p X \beta \vdash \alpha q Z Y \beta \quad \text{if } \delta(p, X) = (q, Y, L)$$

$$\alpha p X \beta \vdash \alpha Y q \beta \quad \text{if } \delta(p, X) = (q, Y, R)$$

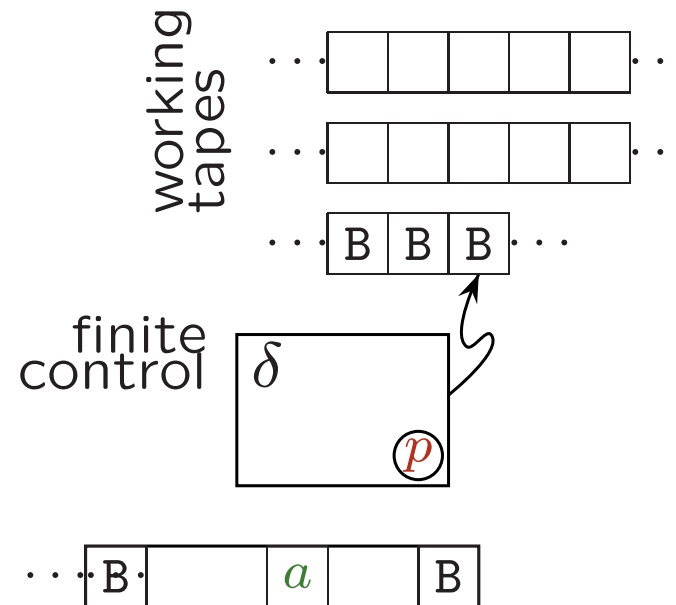
$$\alpha p X \beta \vdash \alpha q Y \beta \quad \text{if } \delta(p, X) = (q, Y, S)$$

$L(\mathcal{M}) =$

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



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

- blocks: no move defined *'no'*
- move off tape (one-sided) *'no'*
- infinite computation 'loop' *'no'* (but we
cannot tell)
- 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 .

$$\Gamma_U = \{B = a_0, a_1, a_2, \dots\}$$

$$Q_U = \{q_0, q_1, q_2, \dots\}$$

$$e(a_i) = 0^{i+1} \quad e(q_i) = 0^{i+1}$$

$$\text{alphabet } \Sigma = \{b_1, \dots, b_r\}$$

$$e(\Sigma) = 111 e(b_1) 1 e(b_2) 1 \dots 1 e(b_r) 111$$

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

$$\text{instruction } m: \delta(q, a) = (p, b, D)$$

$$e(m) = 11 e(q) 1 e(a) 1 e(p) 1 e(b) 1 e(D) 11$$

TM \mathcal{M} with instructions m_1, \dots, m_t

$$e(\mathcal{M}) =$$

$$11111 e(q_0) 1 e(h) 1 e(\Sigma) 1 e(\Gamma) 1 e(m_1) 1 \dots 1 e(m_t) 11111$$

problem \equiv language

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

$\{ e(T)e(w) \mid T \text{ halts on } w \}$

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).* ”

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)

10.1145/800169.805439

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:

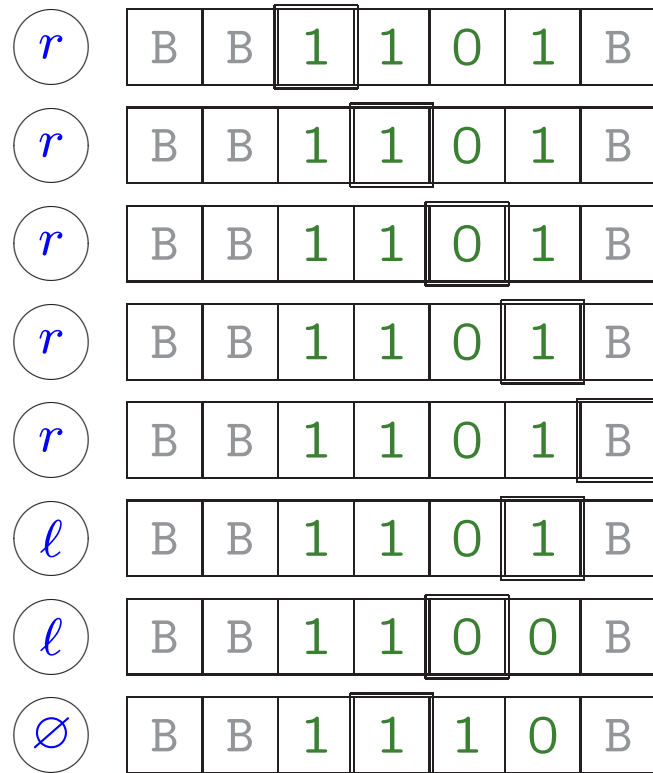
- $L \in \text{NP}$
- $L' \leq L$ for every $L' \in \text{NP}$

Savitch

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

$f(n) \geq \log n$

$\text{PSPACE} = \text{NPSPACE}$



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



$$(Q_1x_1)(Q_2x_2)\cdots(Q_nx_n)\phi(x_1, x_2, \dots, x_n)$$

$$A = Q_2x_2\cdots Q_nx_n\phi(0, x_2, \dots, x_n)$$

$$B = Q_2x_2\cdots Q_nx_n\phi(1, x_2, \dots, x_n)$$

use recursion

stack depth max n

closed formula

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 $x[i] = B$ Q_{q_00} initial state H_{00} initial position head $T_{iak} \rightarrow \neg T_{ibk}$ one symbol per tape cell $a \neq b$ $Q_{pk} \rightarrow \neg Q_{qk}$ one state at a time $p \neq q$ $H_{ik} \rightarrow \neg H_{jk}$ one head position at a time $i \neq j$ $T_{iak} \wedge T_{ib(k+1)} \rightarrow H_{ik}$ tape changed only if written $a \neq b$ $H_{ik} \wedge Q_{pk} \wedge T_{iak} \rightarrow \bigvee_{(p,a,q,b,d) \in \delta} H_{(i+d)(k+1)} \wedge Q_{q(k+1)} \wedge T_{ib(k+1)}$
possible transitions $Q_{hp(n)}$ finish in accepting state h

CNF conjunctive normal form

conjunction \wedge clauses (disjunction \vee literals)

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

$$a \rightarrow \neg b \iff \neg a \vee \neg b$$

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

and finally

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

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

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 $k \leq 2^{|x|}$

$k = 1$ 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

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

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/>