7 Pushdown Automata

[Linz § 7, Kozen L. 23]

The Problem:

We have a **correspondence** between **regular languages** and **DFA's/NFA's**
But (so far) not with **CFL's** and — (What??)

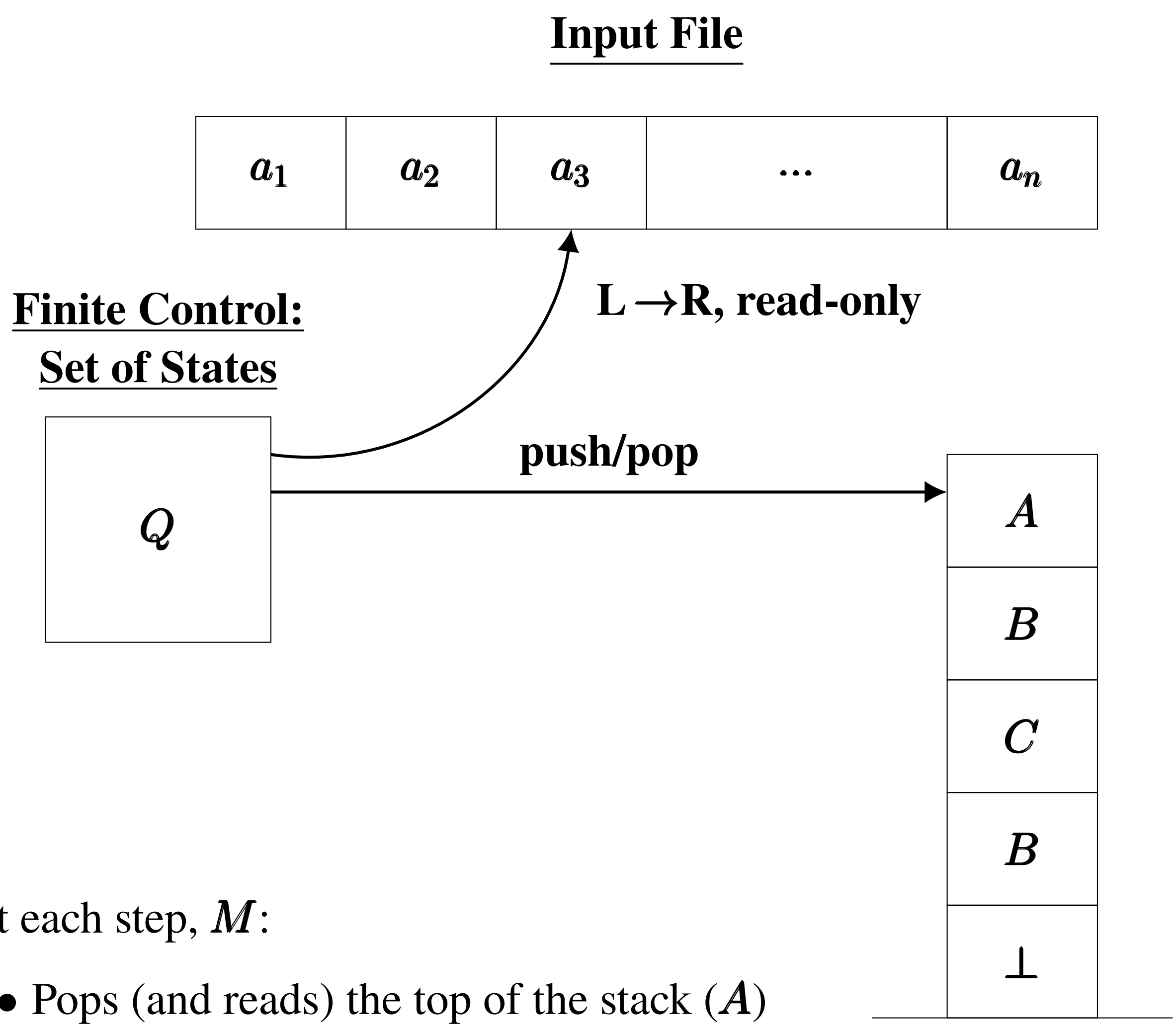
The **problem** is that for **CFL's** e.g. $\{a^n b^n \mid n \geq 0\}$
an "**automaton**" would require **unbounded memory**.

The Solution:

A finite automaton with **unbounded memory** in the form of a **stack**
i.e. a **pushdown automaton (PDA)**

It turns out we need **nondeterministic PDA's (NPDA's)**

An **NPDA** is like an **NFA** except that it has a **stack**.



At each step, M:

- Pops (and reads) the top of the stack (A)
- Depending on this (A), input (a_3) and **state**
  - **pushes** sequence of symbols on **stack**
  - **moves read head** 1 cell to the right
  - Enters **new state** (according to **transition rules**)

Also **λ-transitions**: just **pop** and **push** (**NOT** read cell or move head)

Formal definition: An **NPDA** is a 7-tuple:

$$M = (Q,\ \Sigma,\ \Gamma,\ \delta,\ s,\ \bot,\ F)$$

where

Q: set of **states**

Σ: **input** alphabet

Γ: **stack** alphabet

δ: **transition function**

$s \in Q$: **start** state

$\bot \in \Gamma$: **initial stack symbol**

$F \subseteq Q$: set of **final states**, and

$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \ \to\ \mathcal{P}_{\mathsf{fin}}(Q \times \Gamma^*)$

where $\mathcal{P}_{\mathsf{fin}}(X) \ =$ set of **finite subsets** of X.

This is where **nondeterminism** comes in!

Notation: $\alpha, \beta, \ldots$ for elements of Γ^*, i.e. sequence of stack items.

Notation for Transition Function:

$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \longrightarrow \mathcal{P}_{\mathsf{fin}}(Q \times \Sigma^*)$

If $(q, \alpha) \in \delta(p, a, A)$
we write

$\delta : (p, a, A) \longrightarrow (q, \alpha)$

This is a **transition** of M.

Notation: $s, p, q, (f) \in Q$

$a, b, \ldots \in \Sigma$

$\perp, A, B, \ldots \in \Gamma$

$\alpha, \beta, \ldots \in \Gamma^*$

s : **start state**

So the transition

$$\delta : (p, a, A) \longrightarrow (q, B_1 \ldots B_k) \quad (k \geq 0)$$

means:

pop A,
push $B_1, \ldots, B_k$ (B_k **first,** B_1 **last**),
move head right 1 cell,
enter q.

and

$$\delta : (p, \lambda, A) \longrightarrow (q, B_1 \ldots B_k) \quad (k \geq 0)$$

means:

> **pop** A,
> **push** $B_1, \ldots, B_k$ (B_k **first,** B_1 **last**),
> **enter** q.

Configurations

[Linz: "Instantaneous descriptions"]

A **configuration** of M is a triple

$$(p, u, \alpha) \in Q \times \Sigma^* \times \Gamma^*$$

$p =$ current state,

$u =$ unread part of input ($a_3 \ldots a_n$ in the diagram)

$\alpha =$ current stack contents ($ABCB\perp$ in the diagram)

A **configuration** gives **complete info** about the **current state** of M during computation.

With input u, the **start configuration** is:

$$(s,\ u,\ \perp)$$

Next configuration relation: $\vdash$ (or $\vdash_M$)

(a) Suppose $\delta : (p, a, A) \longrightarrow (q, \alpha)$

Then $(p, av, A\beta) \vdash \underline{(q, v, \alpha\beta)}$

(b) Suppose $\delta : (p, \lambda, A) \longrightarrow (q, \alpha)$

Then $(p, v, A\beta) \vdash \underline{(q, v, \alpha\beta)}$

Configuration changes over a number of steps:

For **configurations** $C, C', \ldots$

Define $C \overset{n}{\vdash} C'$ for $n \geq 0$, by **induction** on n:

$$C \overset{0}{\vdash} C' \iff C = C'$$

$$C \overset{n+1}{\vdash} C' \iff C \overset{n}{\vdash} C'' \overset{1}{\vdash} C'$$

for some C''.

Then, $C \overset{*}{\vdash} C' \iff C \overset{n}{\vdash} C'$ for **some** $n = 0, 1, \ldots$

Note:

$\overset{*}{\vdash}$ is the **transitive-reflexive closure** of $\vdash$.

A **configuration sequence** for M is a sequence

$$C = C_1 \underset{M}{\vdash} C_2 \underset{M}{\vdash} \ldots \underset{M}{\vdash} C_n = D$$

$$\textbf{i.e.,} \quad C \overset{*}{\underset{M}{\vdash}} D$$

where C is the **initial configuration** $(s, u, \perp)$
and D is the **final configuration** (q, λ, α).

So the **language accepted by** M is:

$$L(M) = \{u \in \Sigma^* \mid (s, u, \perp) \overset{*}{\vdash} (q, \lambda, \alpha) \ \textbf{where} \ q \in F\}$$

i.e., the set of all strings over Σ that can put M into a **final state** at the end of the string.

<u>Notes</u>:

(1) α at the end is irrelevant.

(2) If the **stack** is empty at any stage, M **must stop there**!

(3) We write **PDA** for **NPDA**.

Example 1: Give a PDA which accepts

$$L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w)\}$$

Define M :

$$Q = \{s, f\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{\perp, A, B\}$$
$$F = \{f\}$$

δ :

(1) $(s,\ a,\ \perp) \longrightarrow (s,\ A\perp)$

(2) $(s,\ a,\ A) \longrightarrow (s,\ A\,A)$ — 'A' pushed on stack

(3) $(s,\ a,\ B) \longrightarrow (s,\ \lambda)$ — 'a' **cancels** 'B' on top of stack!

(4) $(s,\ b,\ \perp) \longrightarrow (s,\ B\perp)$

(5) $(s,\ b,\ B) \longrightarrow (s,\ BB)$ — 'B' pushed on stack

(6) $(s,\ b,\ A) \longrightarrow (s,\ \lambda)$ — 'b' cancels 'A'!

(7) $(s,\ \lambda,\ \perp) \longrightarrow (f,\ \lambda)$ — Successful! Get '$\perp$' at end of word!

Notes:

(1) At any stage, stack contains:

$\perp$ plus **either** all A's: excess of 'a' so far in w
or all B's: excess of 'b' so far in w

(2) At the end (i.e. reading 'λ'): want the stack to have only '$\perp$'

(3) There is **no transition** from (s, λ, A) or (s, λ, B)

These mean **excess** of 'a' or 'b' at end of word!

(4) Check: does this work for the empty word?

Yes! Just use (7).