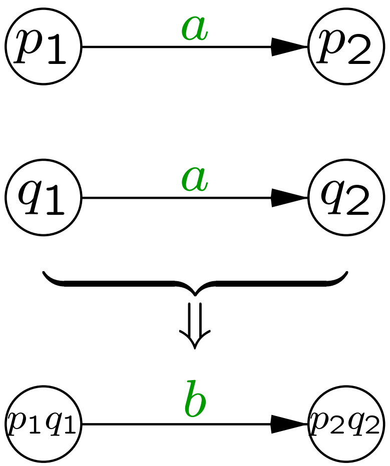
Closure and Determinism

Stack Languages and Predicting Machines



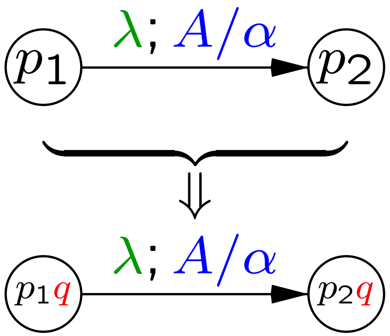
L, R deterministic $\Rightarrow L \cup R$ deterministic ?

- $L, R \in$ REG $\Rightarrow L \cup R \in$ REG

  simulate deterministic automata in parallel
  (automata should not block)

- $L, R \in$ DPDℓ $\Rightarrow L \cup R \notin$ DPDℓ

  $\{ a^n b^n \mid n \geq 1 \} \cup \{ a^n b^m c^n \mid m, n \geq 1 \}$



- $L \in$ DPDℓ, $R \in$ REG $\Rightarrow L \cup R \in$ DPDℓ

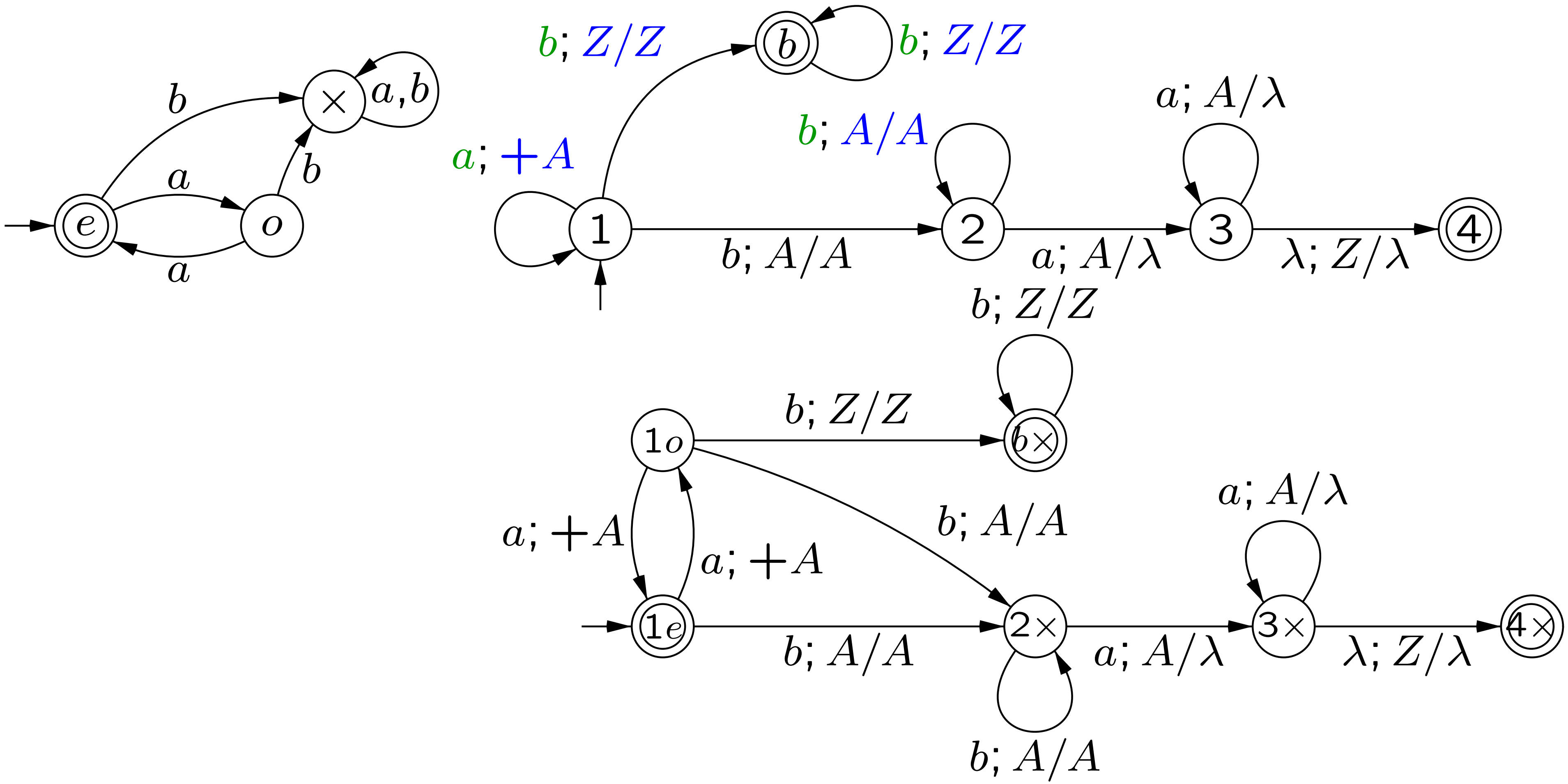
  parallel simulation with λ-moves

- avoid infinite λ-computations: !!

  PDA should read all possible input

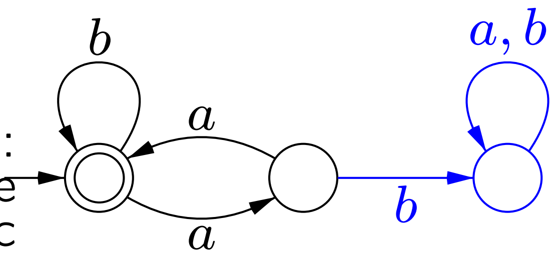
$L \in \mathsf{DPD}\ell,\ R \in \mathsf{REG}$

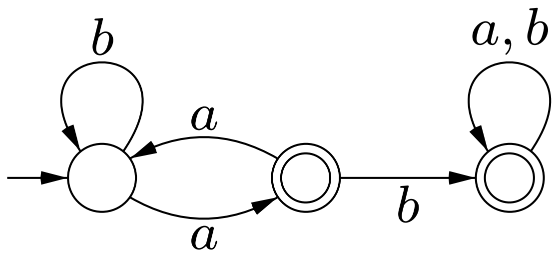
$\{\ a^n b^m a^n \mid m, n \in \mathbb{N}\ \} =$
$\{\ a^{2n} \mid n \in \mathbb{N}\ \} \cup \{\ a^n b^m a^n \mid m, n \in \mathbb{N}, m \geq 1\ \}$



- regular languages $\surd$

read all input:
complete
and deterministic





- context-free languages $\times$

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

complement $\{a, b, c\}^* - a^* b^* c^*$
$\cup \{ a^i b^j c^k \mid \ i \neq j \} \cup \{ a^i b^j c^k \mid \ j \neq k \}$

- deterministic cf languages $\surd$

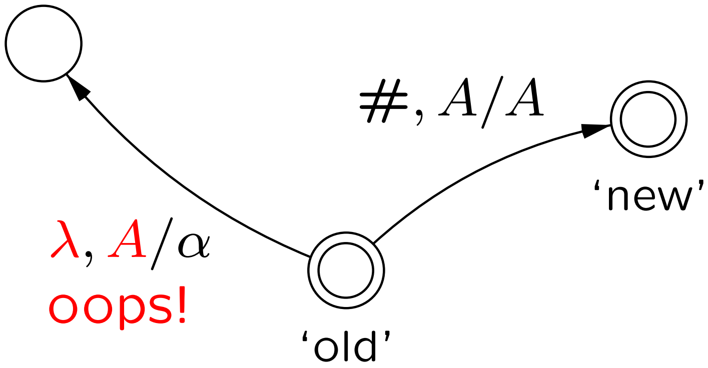
read all input $\mapsto$ as complete as possible
but $\mapsto$ infinite λ-computations

predicting machines

“can we reach a non-λ transition with present stack?”

$L \in \mathsf{DPD}\ell \Rightarrow L\# \in \mathsf{DPD}\ell$

'classic' construction



no λ-instructions leaving final state

(a normal form?)

how do we achieve that?

what about single letter quotient?

$$L/\{a\} = \{\, x \mid xa \in L \,\}$$

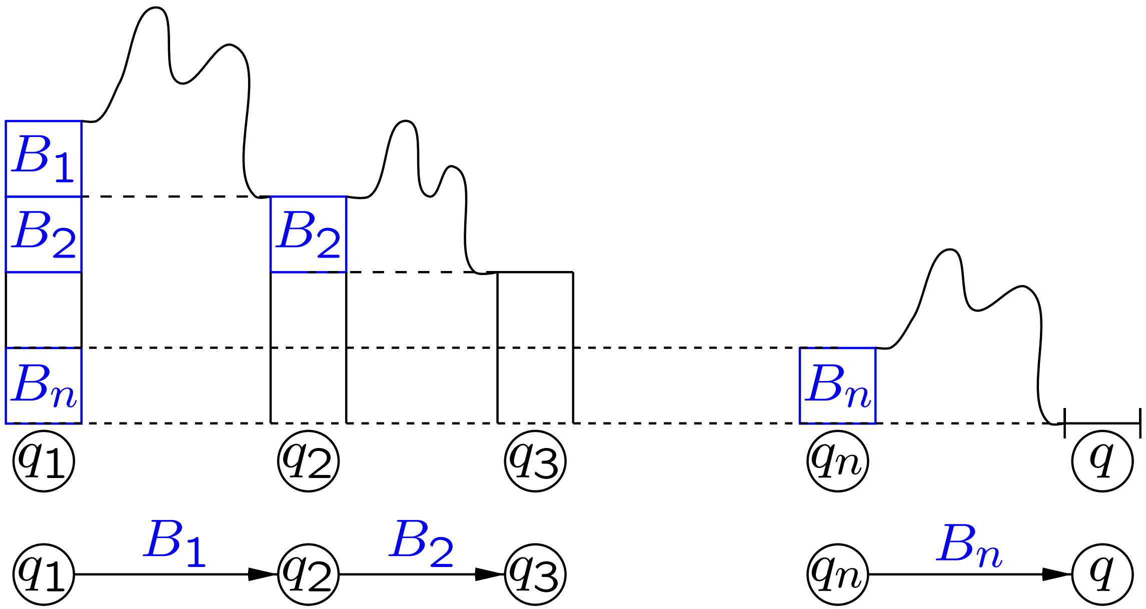
$\longrightarrow$ We study the language of stacks during computations of a PDA. This language is regular! The proof is a simple consequence of the $[\, p, A, q \,]$-construction! Exclamation mark! $\longleftarrow$

stack language

$$\mathsf{SN}(\mathcal{A}) = \{\, \alpha \in \Gamma^* \mid (\, p_{in}, w, \alpha \,) \vdash^* (\, q, \lambda, \lambda \,) \text{ for some } w \in \Sigma^*, \text{ some } q \in Q \,\}$$

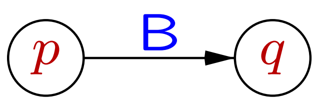
input w is irrelevant here

$B_1 B_2 \ldots B_n \in \mathsf{SN}(\mathcal{A})$



this is regular!

build automaton:



iff

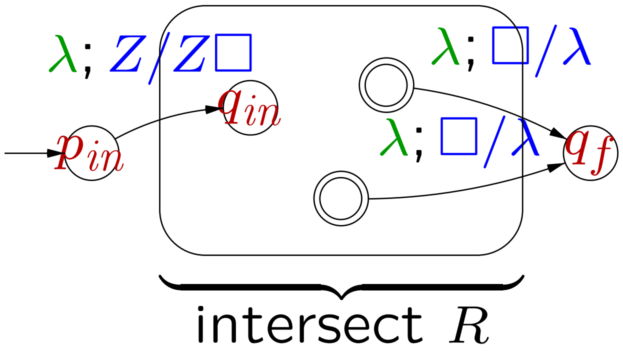
$(\, p, w, B \,) \vdash^* (\, q, \lambda, \lambda \,)$

iff

$[\, p, B, q \,] \Rightarrow^* w$

(for some $w \in \Sigma^*$)

every state initial & final



$$\mathrm{SN}(\mathcal{A}) = \{\, \alpha \in \Gamma^* \mid (\, p_{in}, w, \alpha \,) \vdash^* (\, q, \lambda, \lambda \,) \text{ for some } w \in \Sigma^*, \text{ some } q \in Q \,\}$$

variant [also regular]

$$\{ \ldots \mid \quad \ldots \text{ for some } w \in R, \text{ some } q \in F \,\}$$

$$\mathrm{SF}(\mathcal{A}) = \{\, \alpha \in \Gamma^* \mid (\, p_{in}, w, Z_{in} \,) \vdash^* (\, q, \lambda, \alpha \,) \text{ for some } w \in \Sigma^*, \text{ some } q \in F \,\}$$

$$\text{SN}(\mathcal{A}) = \{ \alpha \in \Gamma^* \mid (p_{in}, w, \alpha) \vdash^* (q, \lambda, \lambda) \text{ for some } w \in \Sigma^*, \text{ some } q \in F \}$$

Buchi: regular canonical systems

type-0 productions $\alpha \to \beta$

prefix rewriting $\begin{array}{|c|c|}\hline \alpha & \qquad\qquad \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|}\hline \beta & \qquad\qquad \\ \hline\end{array}$

$L(\text{rcs}) = \{ w \mid w \Rightarrow^* \lambda \}$

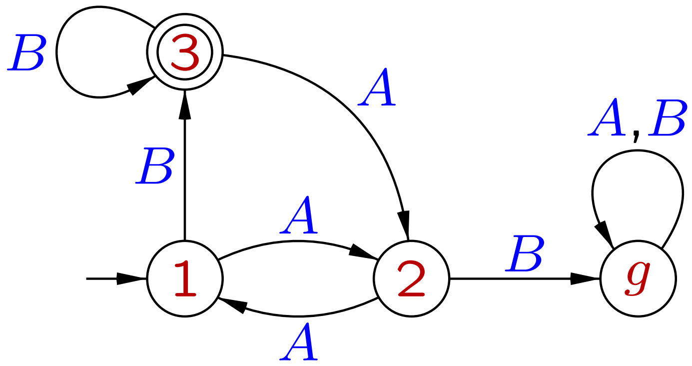
rcs defines regular language

simulate prefix $\alpha \to \beta$ by PDA [use F]

stack belongs to regular language R?

e.g. $R = B(AA + B)^*$

deterministic automaton for *reverse*



update stack

B/AB

$\langle B, 1\rangle/\langle A, 1\rangle\langle B, 2\rangle$

$\langle B, 2\rangle/\langle A, 2\rangle\langle B, 1\rangle$

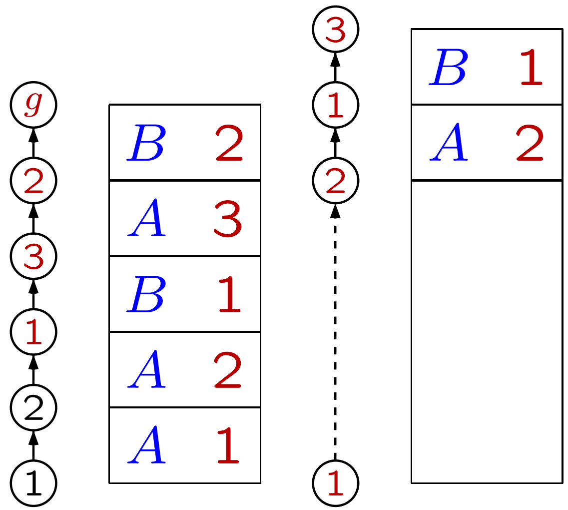
$\langle B, 3\rangle/\langle A, 3\rangle\langle B, 2\rangle$

$\langle B, g\rangle/\langle A, g\rangle\langle B, g\rangle$

success $\in R$

$\langle B, 1\rangle$, $\langle B, 3\rangle$ on top

add state info to stack



- stack language SN($\mathcal{A}$) is regular
- a (deterministic) PDA can keep regular info on its stack

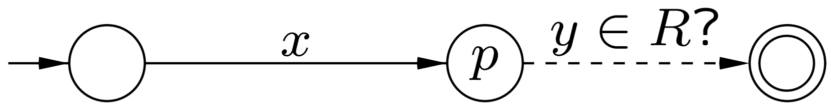
$\Rightarrow$ a (deterministic) PDA can predict future behaviour using present stack by inspecting top

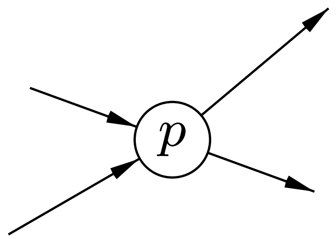
predicting machines

"reaches $\mathcal{A}$ the empty stack from *my* stack?"

B	yes	no	yes
A	yes	yes	no
B	no	yes	yes
A	no	no	yes
A	yes	no	no
	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$

DPDℓ closed quotient with REG



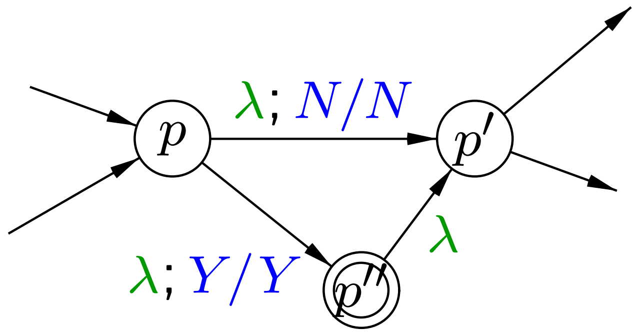


can we accept extending with $y \in R$?

stack α satisfies:

$$(\, p, y, \alpha \,) \vdash_{\mathcal{A}}^{*} (\, q, \lambda, \beta \,),\ q \in F,\ y \in R, \text{ some } \beta$$

quotient automaton



$$\mathrm{SN}(\mathcal{A}_{p,R}) = \{\, \alpha \in \Gamma^* \mid (\, p, w, \alpha \,) \vdash_{\mathcal{A}_{p,R}}^{*} (\, q, \lambda, \lambda \,) \text{ for some } w \in \Sigma^* \,\}$$

Y: $\alpha \in \mathrm{SN}(\mathcal{A}_{p,R})$

N: $\alpha \notin \mathrm{SN}(\mathcal{A}_{p,R})$

$\mathcal{A}_{p,R}$ constructed from $\mathcal{A}$

- initial state p
- intersection with R (product construction)
- change to empty stack acceptance

as promised:

▶ deterministic cf languages $\surd$

read all input $\mapsto$ as complete as possible
but $\mapsto$ infinite λ-computations

predicting machines

"can we reach a non-λ transition with present stack?"