

Deep Pushdown Automata

Alexander Meduna



meduna@fit.vutbr.cz

Brno University of Technology, the Czech Republic

Presented on November 16, 2009

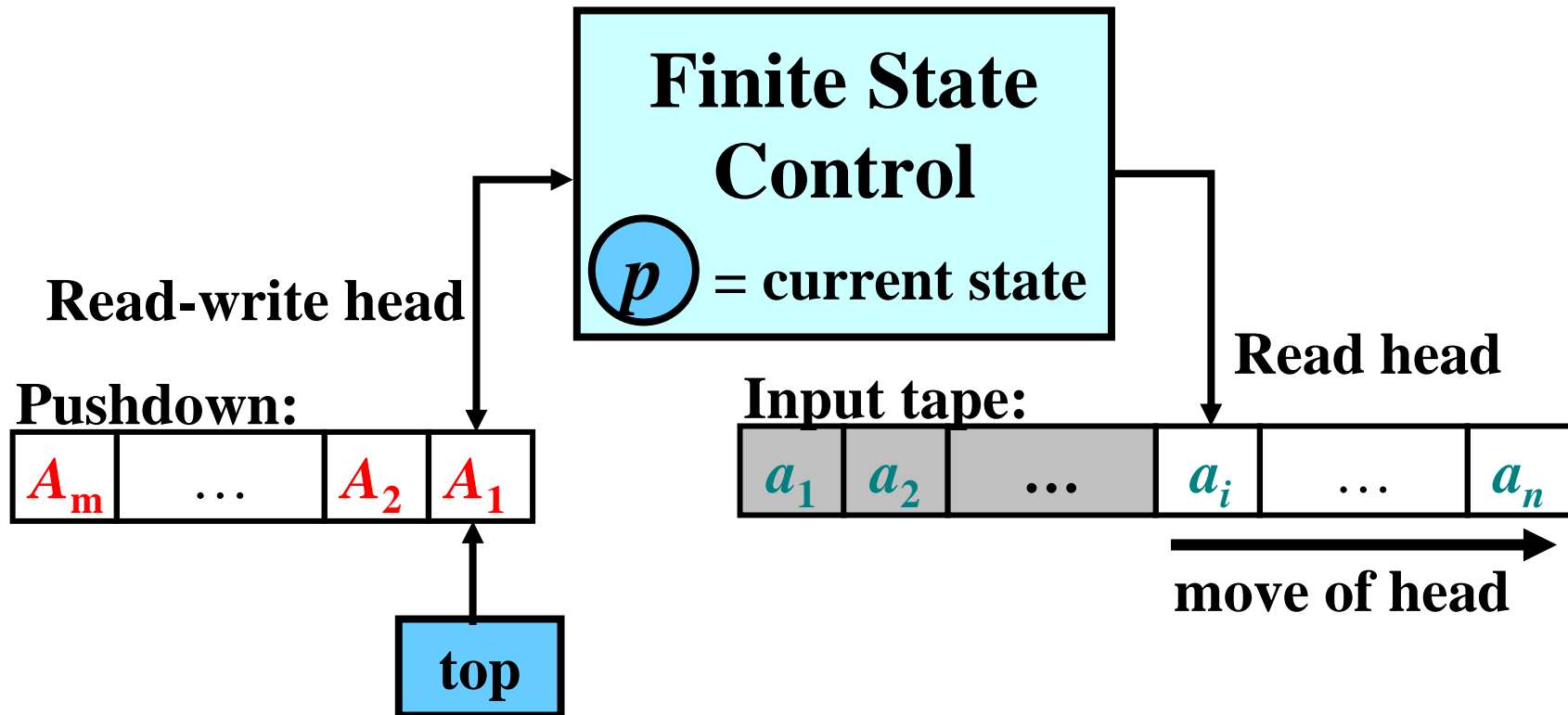
Liverpool Hope University

Based on

Meduna, A.: Deep Pushdown Automata,

Acta Informatica, 2006

Pushdown Automaton (PDA)



Inspiration

- conversion of CFG to PDA that acts as a general top-down parser
 - if pd top = **input** symbol, **pop**
 - if pd top = **non-input** symbol, **expand**

PDA as a general Top-Down Parser

- Configuration:

$$(state, input, pd)$$

- Pop:

$$(q, ax, a\alpha) \xrightarrow{p} (q, x, \alpha)$$

- Expansion:

$$(q, x, A\alpha) \xrightarrow{e} (p, x, \beta\alpha)$$

by rule $qA \rightarrow p\beta$

- Acceptance: $(s, x, S) \Rightarrow^* (f, \varepsilon, \varepsilon)$

Final state

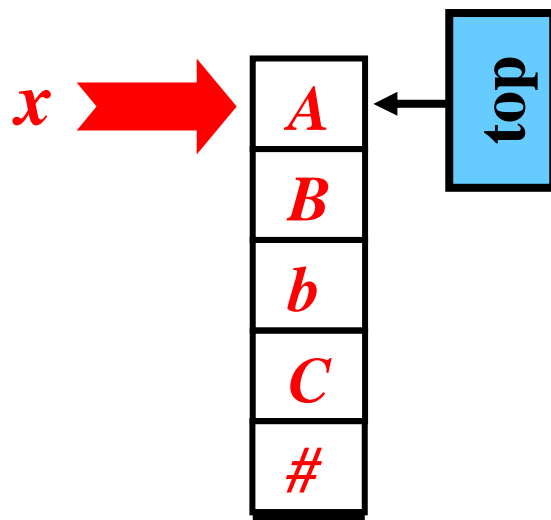


Deep Pushdown Automata: Fundamental Modification

- expansion may be performed **deeper in pd**

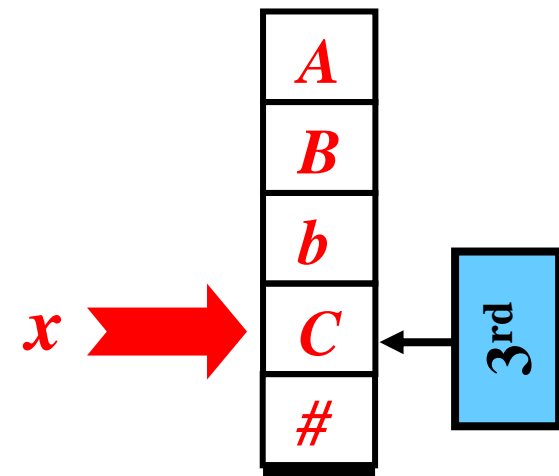
Standard Expansion

$qA \rightarrow px$



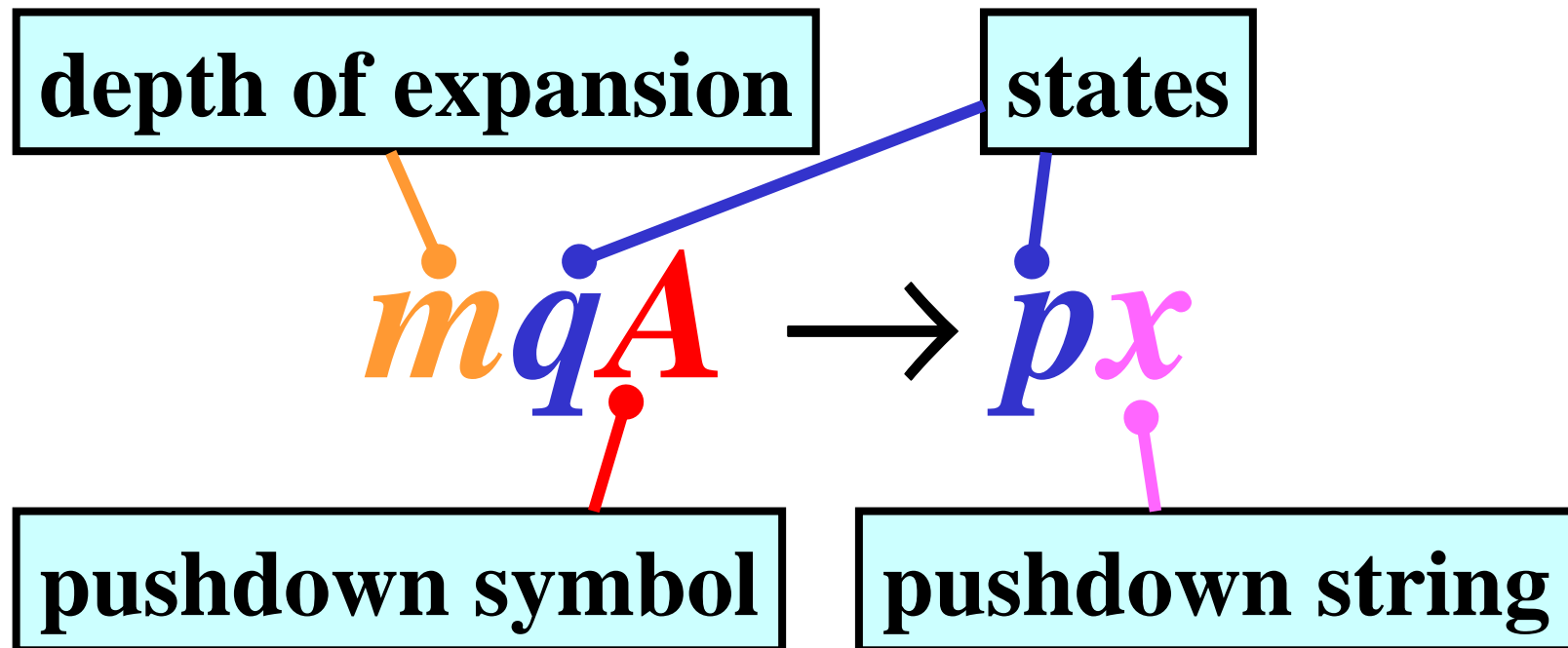
Deep Expansion

$3qC \rightarrow px$



Deep Pushdown Automata

- Same as Top-Down Parser except *deep expansions*
- *Expansion of depth m* :
 - the m th topmost non-input pd symbol is replaced with a string by rule



Expansion of Depth m

- *Expansion of depth m :*

$$(q, w, uAz) \xrightarrow{e} (p, w, uvz)$$

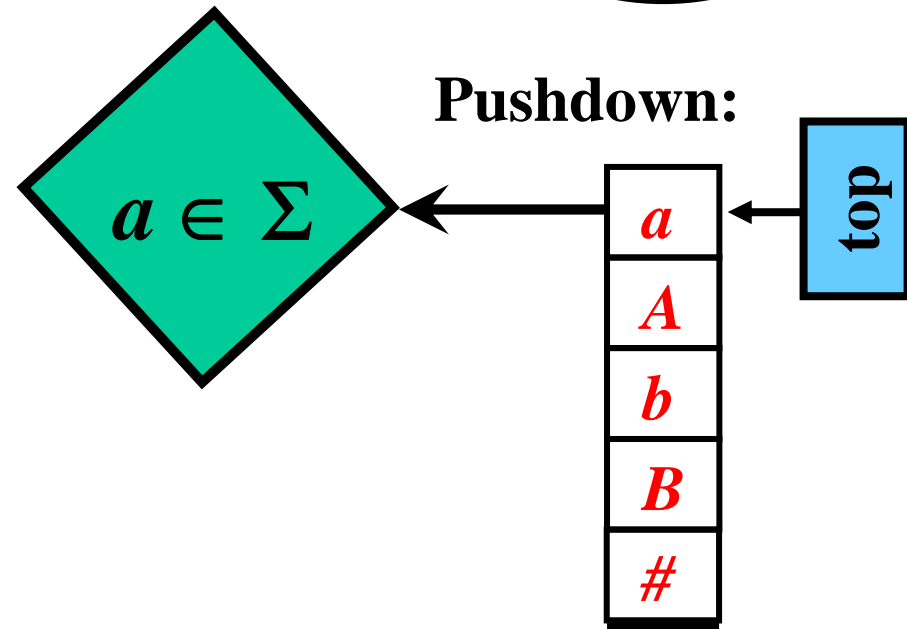
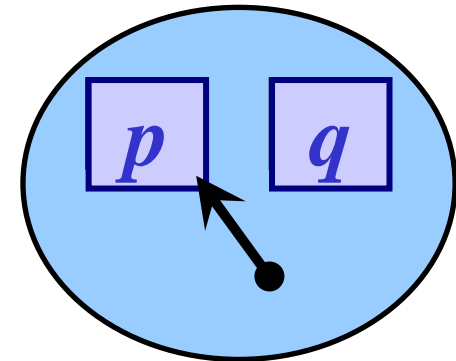
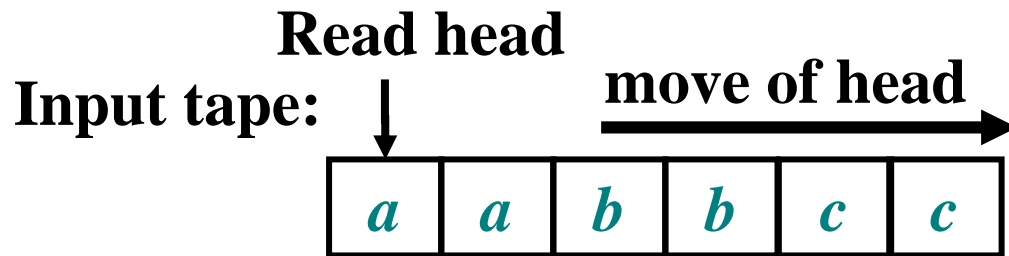
by *rule of depth m*

$$[mqA \rightarrow pv],$$

where u contains $m - 1$ non-input symbols

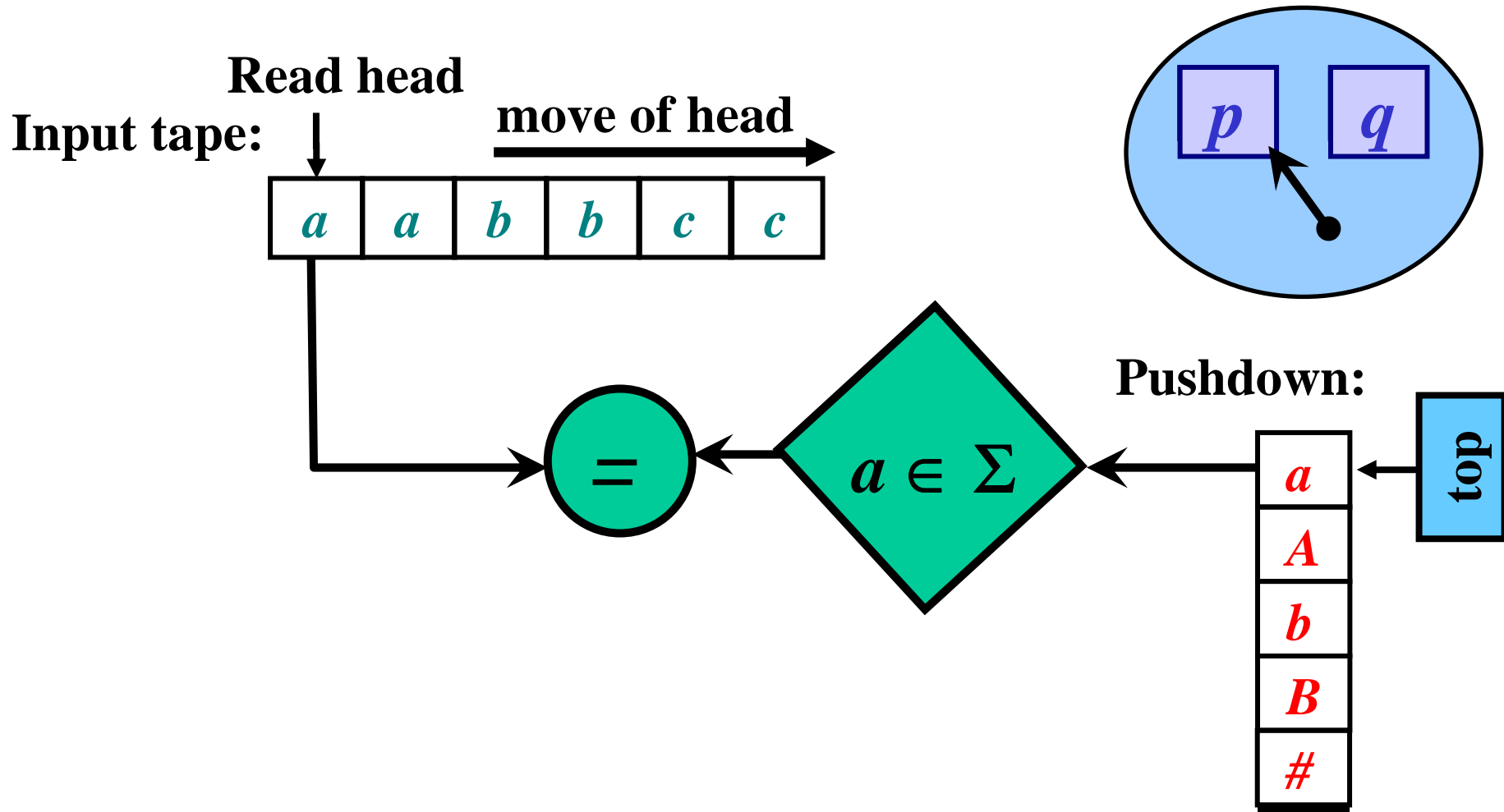
Pop: Illustration

Move: $(p, aabbcc, aAbB\#)_p \Rightarrow (p, abbcc, AbB\#)$



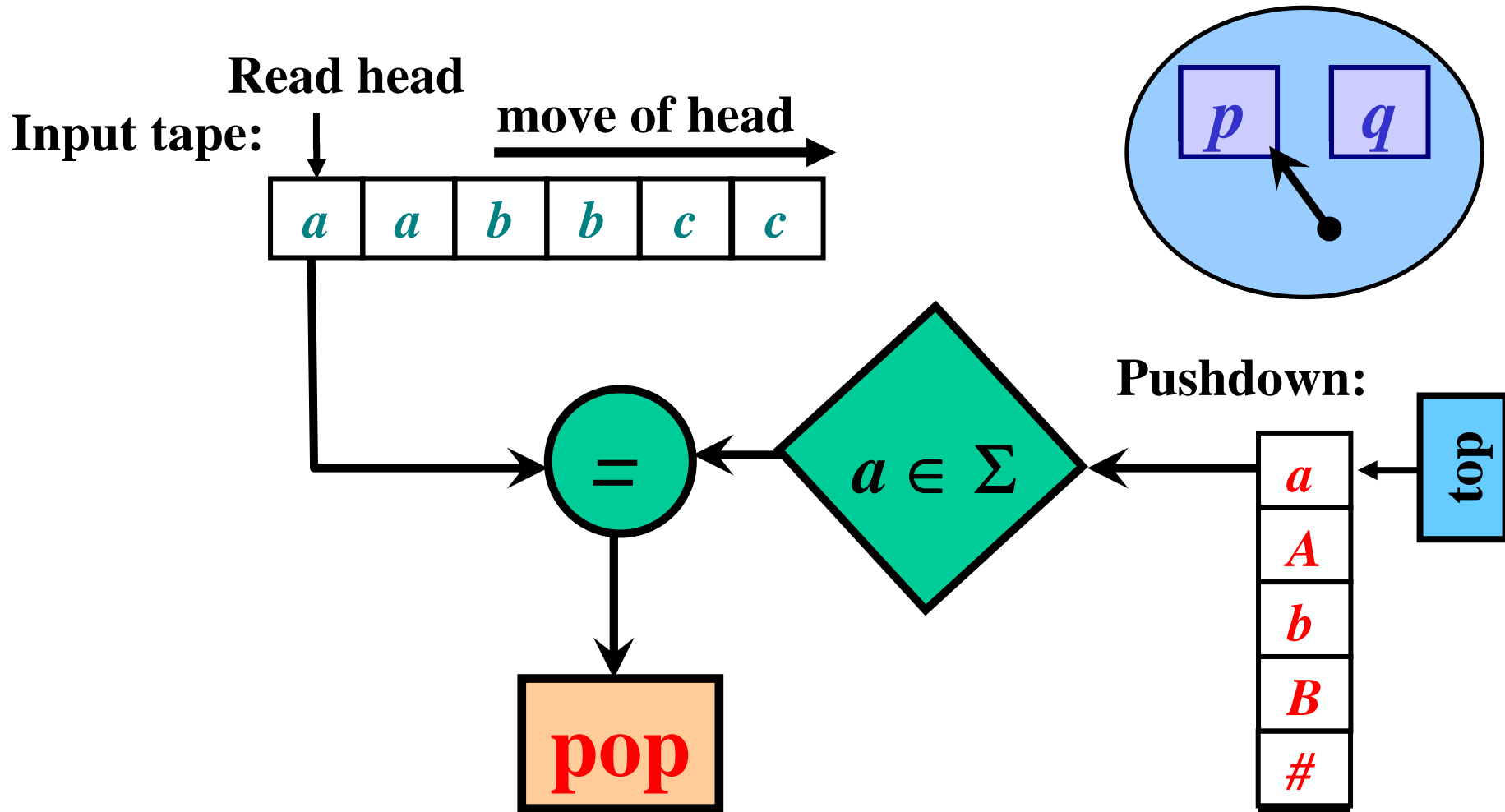
Pop: Illustration

Move: $(p, aabbcc, aAbB\#) \xRightarrow{p} (p, abbcc, AbB\#)$



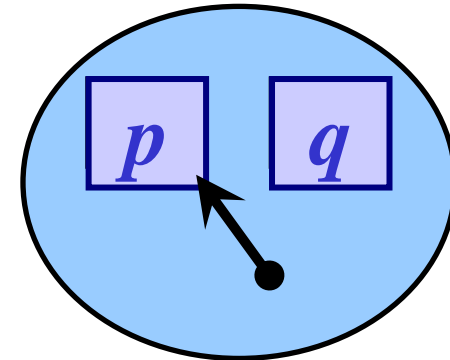
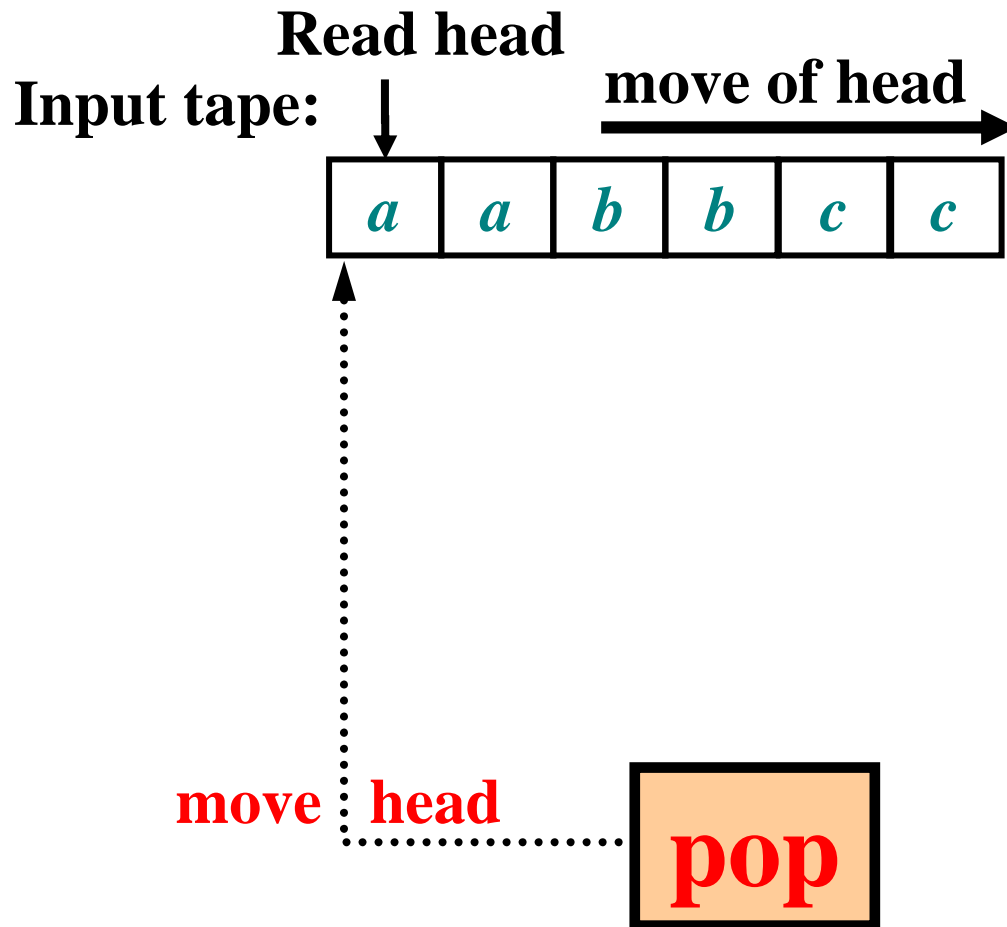
Pop: Illustration

Move: $(p, aabbcc, aAbB\#)_p \Rightarrow (p, abbcc, AbB\#)$

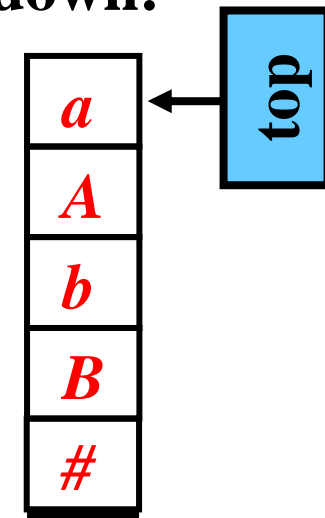


Pop: Illustration

Move: $(p, aabbcc, aAbB\#)_p \Rightarrow (p, abbcc, AbB\#)$

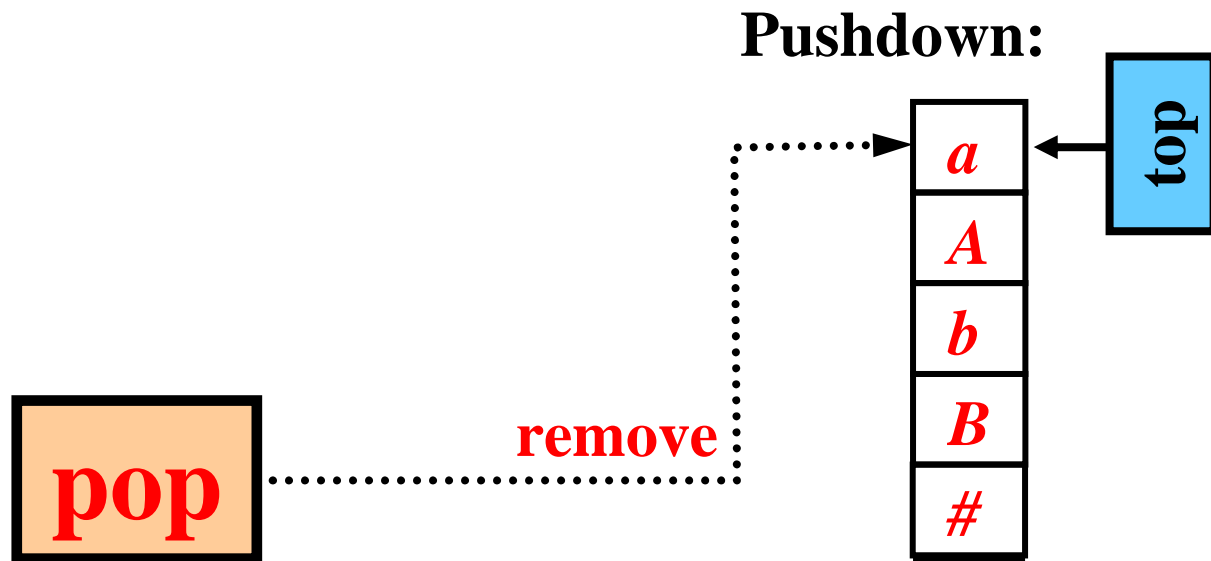
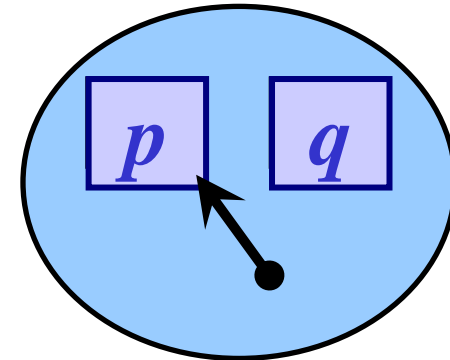
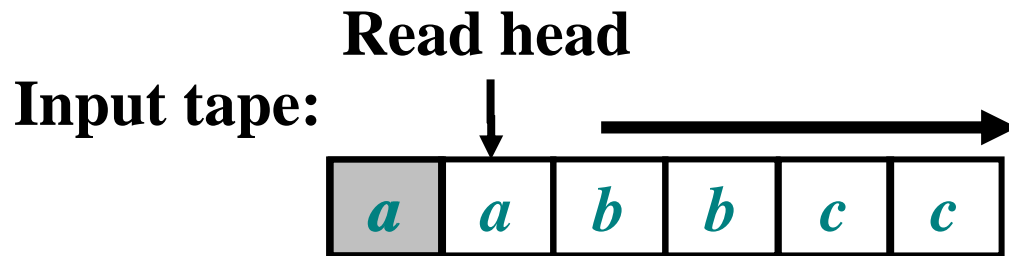


Pushdown:



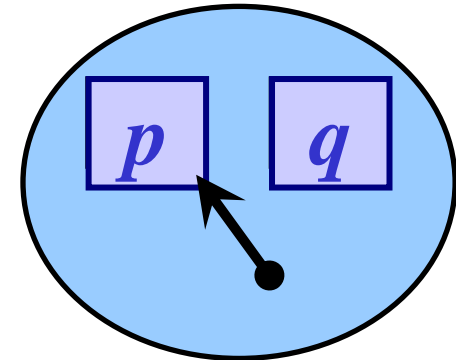
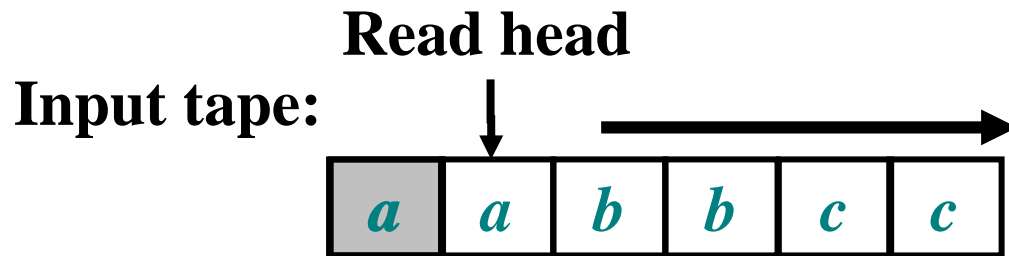
Pop: Illustration

Move: $(p, aabbcc, aAbB\#)_p \Rightarrow (p, abbcc, AbB\#)$

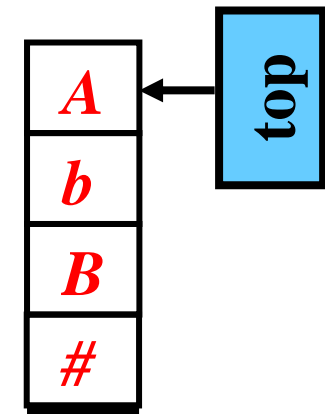


Pop: Illustration

Move: $(p, aabbcc, aAbB\#)_p \Rightarrow (p, abbcc, AbB\#)$



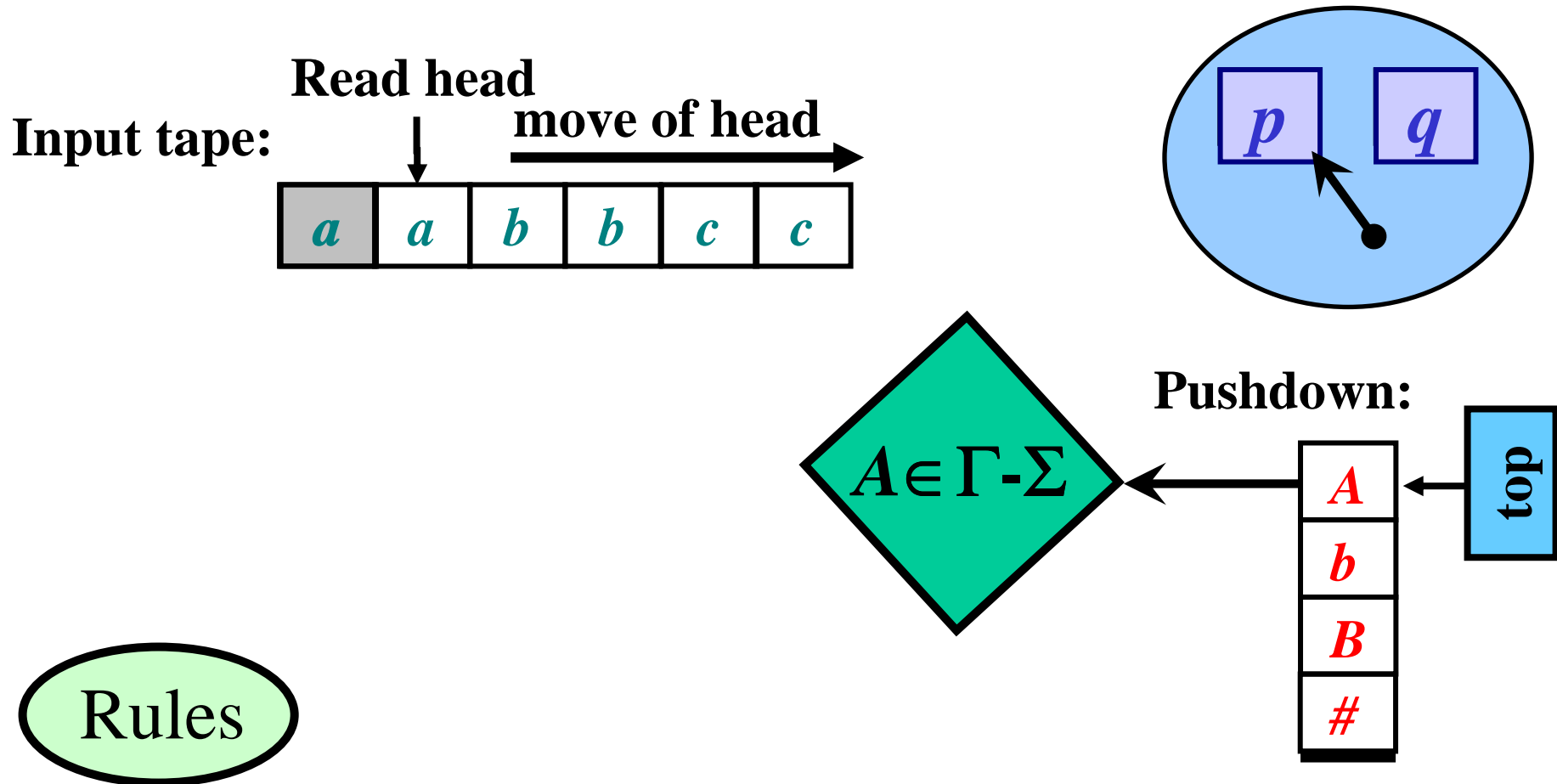
Pushdown:



pop ✓

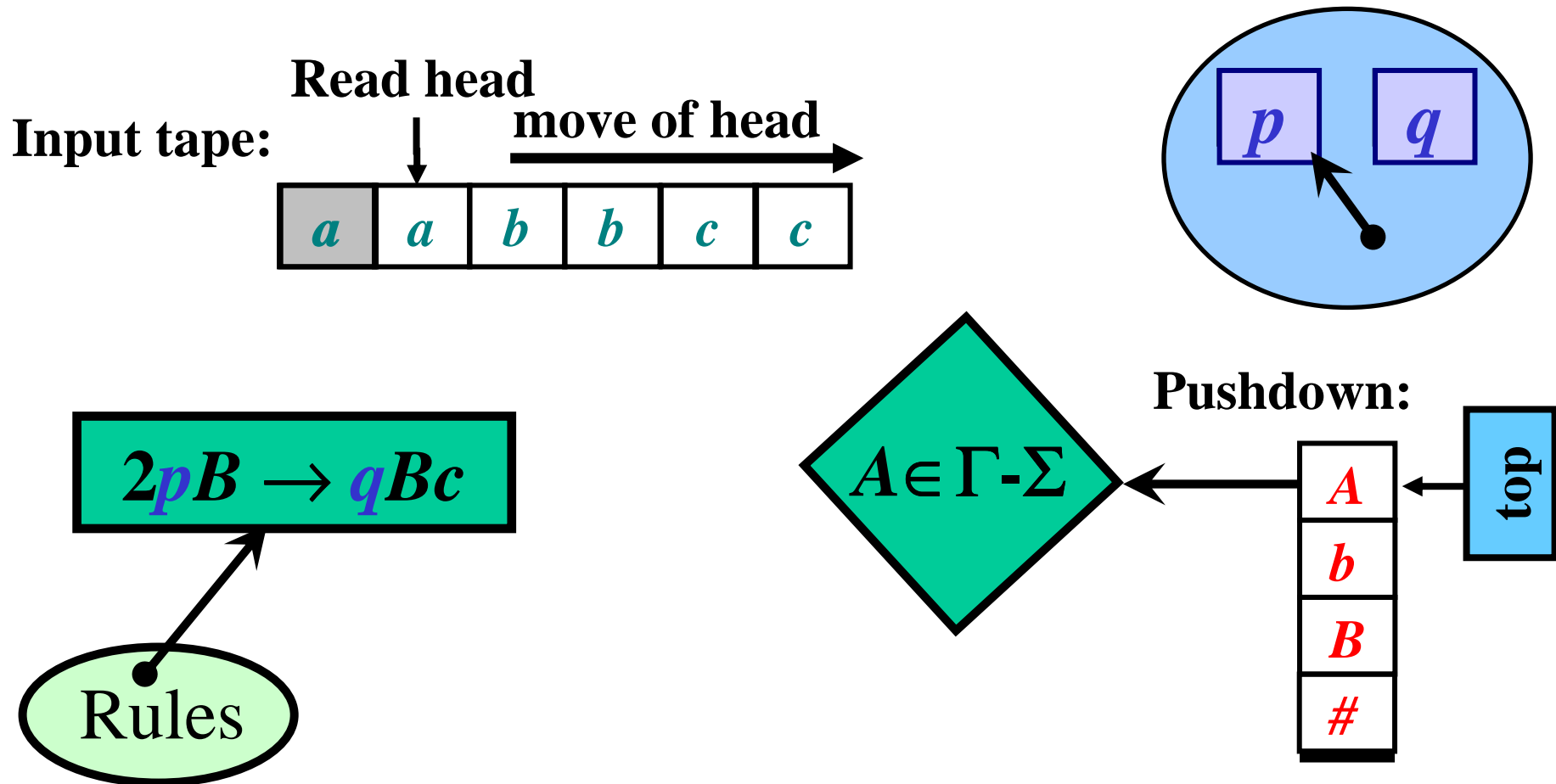
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



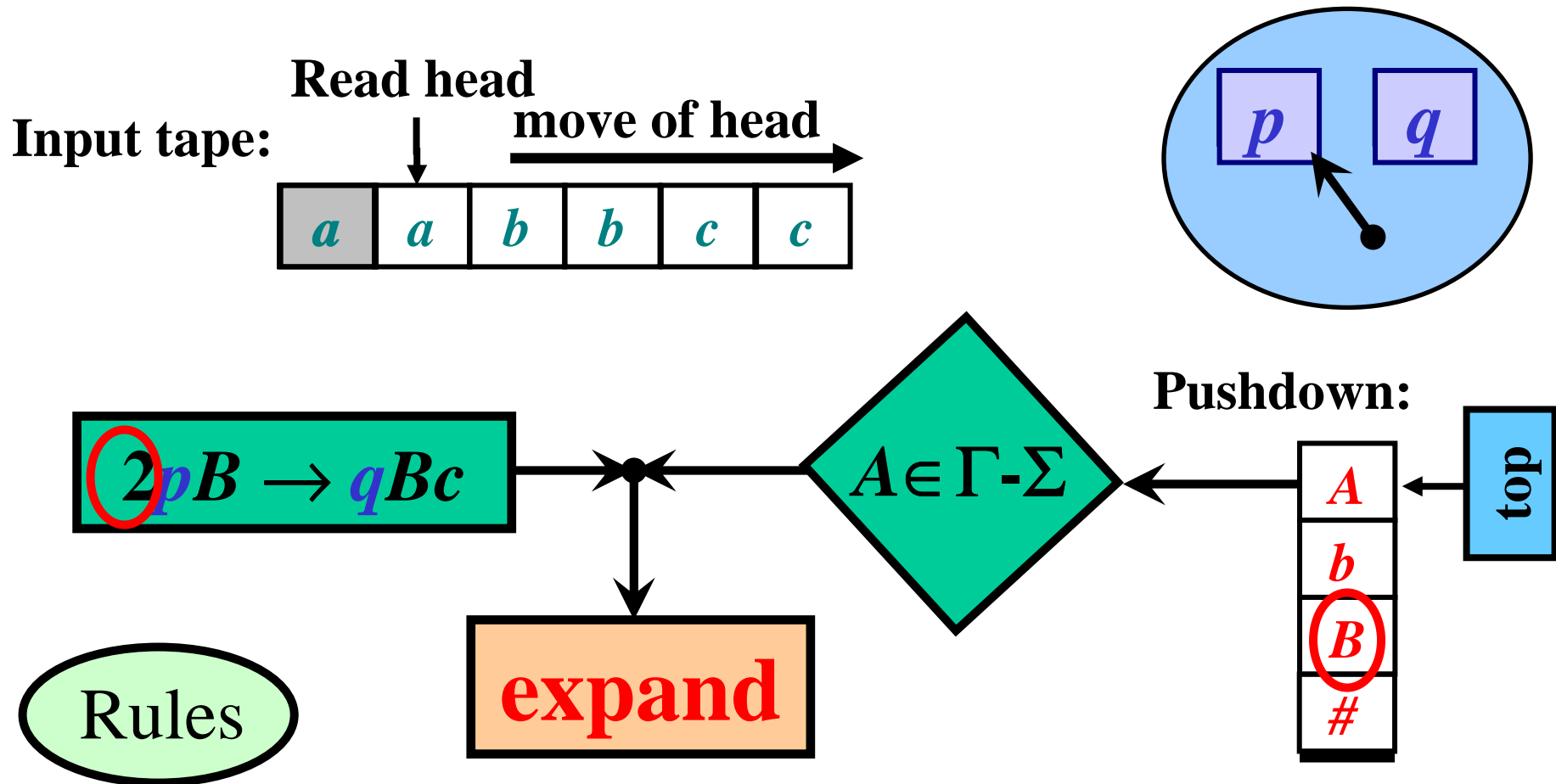
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



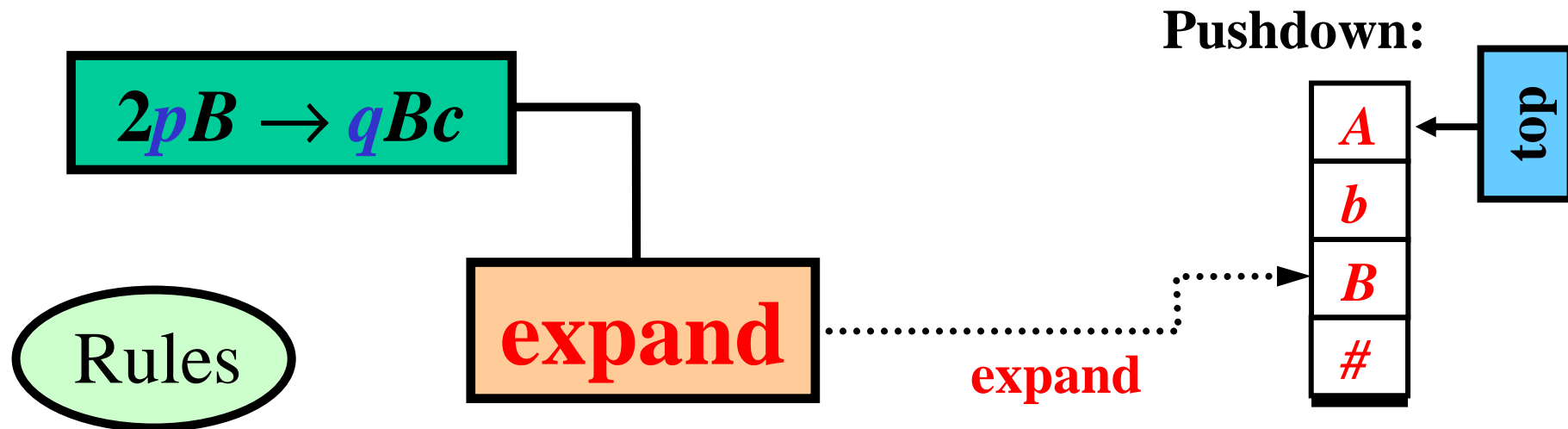
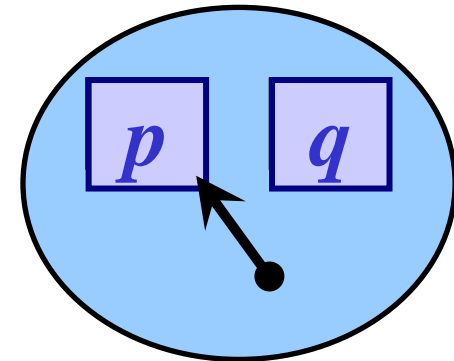
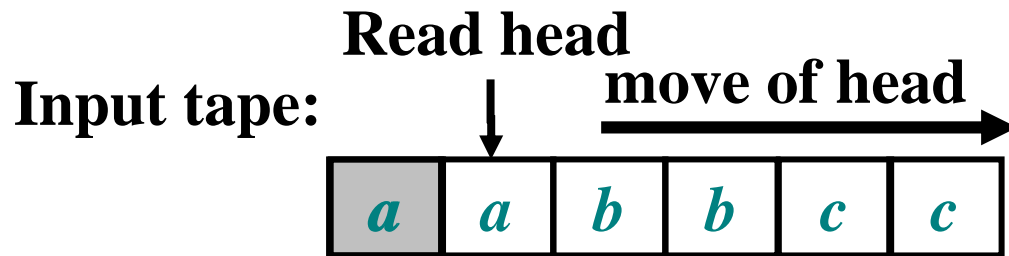
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



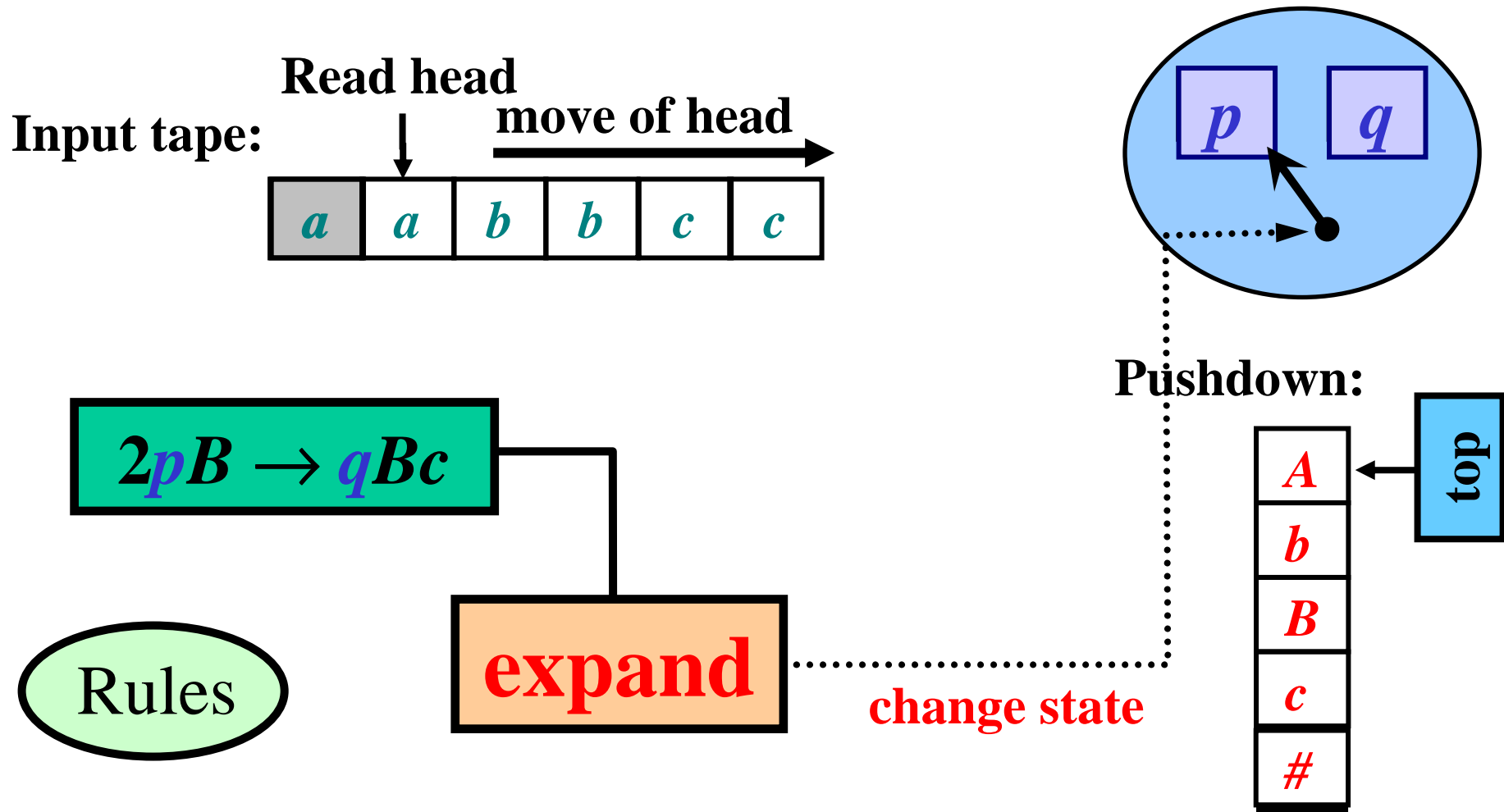
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



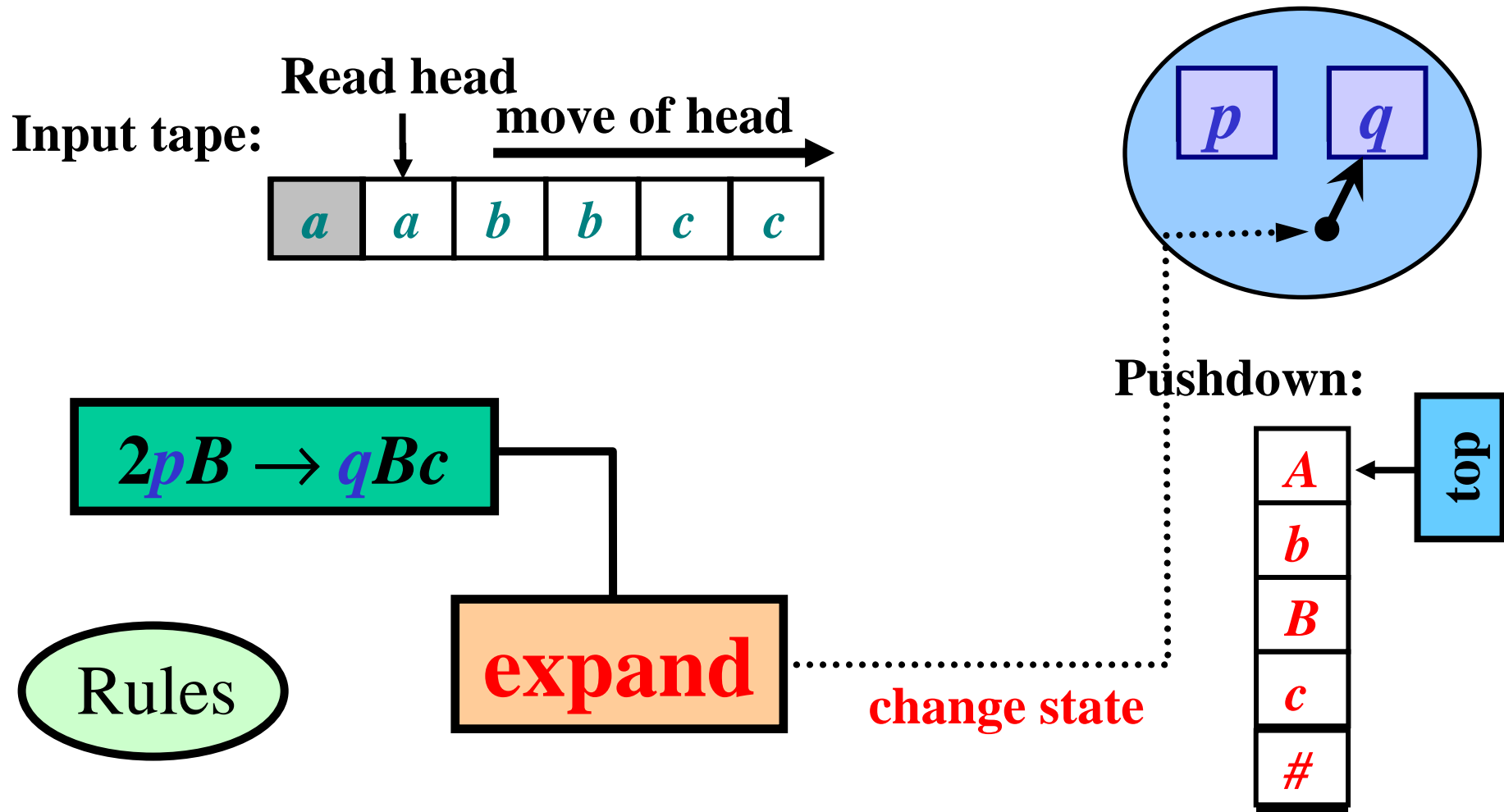
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



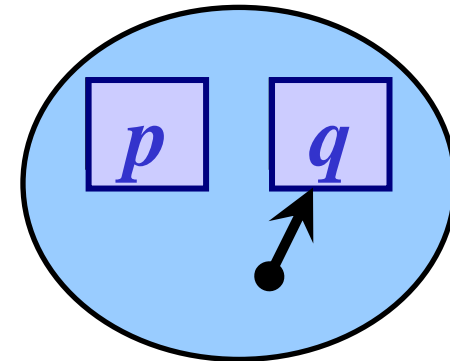
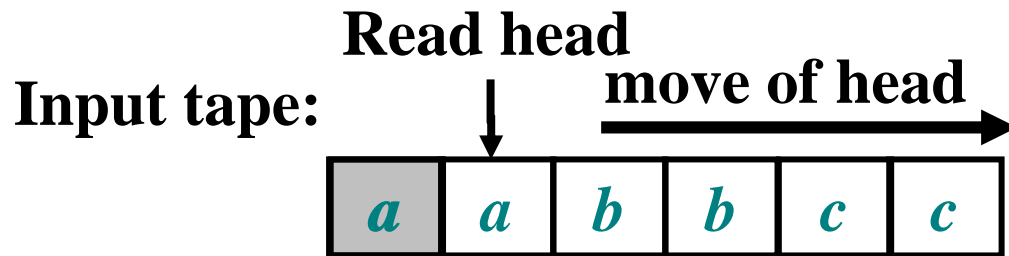
Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]



Deep Expansion: Illustration

Move: $(p, abbcc, AbB\#) \Rightarrow (q, abbcc, AbBc\#)$ [$2pB \rightarrow qBc$]

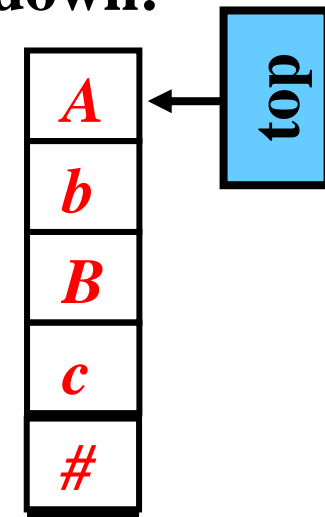


$2pB \rightarrow qBc$

expand ✓

Rules

Pushdown:



Example: Deep PDA

Deep PDA M :

- [1]. $1sS \rightarrow qAB$
- [2]. $1qA \rightarrow paAb$
- [3]. $1qA \rightarrow fab$
- [4]. $2pB \rightarrow qBc$
- [5]. $1fB \rightarrow fc$

M accepts $aabbcc$:

$(s, aabbcc, S\#)$

- $e \Rightarrow (q, aabbcc, AB\#)$ [1]
- $e \Rightarrow (p, aabbcc, aAbB\#)$ [2]
- $p \Rightarrow (p, abbcc, AbB\#)$
- $e \Rightarrow (q, abbcc, AbBc\#)$ [4]
- $e \Rightarrow (f, abbcc, abbBc\#)$ [3]
- $p \Rightarrow (f, bbcc, bbBc\#)$
- $p \Rightarrow^2 (f, cc, Bc\#)$
- $e \Rightarrow (f, cc, cc\#)$ [5]
- $p \Rightarrow (f, c, c\#)$
- $p \Rightarrow (f, \varepsilon, \#)$

$$L(M) = \{a^n b^n c^n : n \geq 1\} \in PD_2$$

Definition 1/3

A deep pushdown automaton is a 7-tuple

$$M = (Q, \Sigma, \Gamma, R, s, S, F), \text{ where}$$

- Q – states,
- $\Sigma \subseteq \Gamma$ – input alphabet,
- Γ – pushdown alphabet, bottom symbol $\# \in \Gamma - \Sigma$
- R – finite set of rules of the form

$$mqA \rightarrow pw \text{ or } mq\# \rightarrow pv\#$$

- $s \in Q$ – start state
- $S \in \Gamma$ – start pushdown symbol
- $F \subseteq Q$ – final states

Definition 2/3

- if an input symbol is on pd top, **M pops** the pd as

$$(q, au, az)_p \Rightarrow (q, u, z), \quad a \in \Sigma$$

- no explicit rule needed in R

- if a non-input symbol is on pd top, **M expands** the pd as

$$(q, w, uAz)_e \Rightarrow (p, w, uvz) [mqA \rightarrow pv],$$

where u contains $m - 1$ non-input symbols

Definition 3/3

- M is *of depth n* , denoted by ${}_nM$, if n is the minimal positive integer such that each of M 's rules is of depth n or less.

- Language accepted by ${}_nM$, $L({}_nM)$, is defined as

$$L({}_nM) = \{w \in \Sigma^* : (s, w, S\#) \Rightarrow^* (f, \varepsilon, \#) \text{ in } {}_nM \text{ with } f \in F\}.$$

Main Result and its Proof

- PD_n – the language family defined by DeepPDAs of depth n .

Theorem: $PD_n \subset PD_{n+1}$, for all $n \geq 1$.

Proof (Sketch):

- State grammars (Kasai, 1970) are needed in the proof
- State grammar is a modification of CFG based on rules of the form

$$(q, A) \rightarrow (p, v)$$

Proof 1/6: State Grammar

- *State grammar* $G = (V, W, T, P, S)$

- V – total alphabet, W – states, $T \subseteq V$ – terminals,
- P – set of rules of the form $(q, A) \rightarrow (p, v)$
- $S \in (V - T)$ – start symbol,

- *Configuration* – (q, x)

- *Derivation step:*

$$(q, uAz) \Rightarrow (p, uvz) [(q, A) \rightarrow (p, v)]$$

and for every nonterminal B in u , P contains no rule with (q, B) on the left-hand side

Proof 2/6: n -limited Step

- *n -limited derivation step:*

each derivation step within the first n non-terminals

$$(q, uAz) \xRightarrow{n} (p, uvz) \text{ and}$$

uA has n or fewer non-terminals

- *n -limited state language:*

$$L(G, n) = \{w \in T^* : (q, S) \xRightarrow{n}^* (p, w)\}$$

- ST_n – the family of n -limited state languages

Proof 3/6: Example

State Grammar G :

- [1]. (1, S) \rightarrow (2, AC)
- [2]. (2, A) \rightarrow (3, aAb)
- [3]. (2, A) \rightarrow (4, ab)
- [4]. (3, C) \rightarrow (2, Cc)
- [5]. (4, C) \rightarrow (4, c)

$$W = \{1, 2, 3, 4\}$$

G generates $aabbcc$:

$$\begin{aligned}
 (S, 1) &\Rightarrow (AC, 2) && [1] \\
 &\Rightarrow (aAbC, 3) && [2] \\
 &\Rightarrow (aAbCc, 2) && [4] \\
 &\Rightarrow (aabbCc, 4) && [3] \\
 &\Rightarrow (aabbcc, 4) && [5]
 \end{aligned}$$

$$L(G) = \{a^n b^n c^n : n \geq 1\} \in ST_2$$

Proof 4/6: $PD_n \subseteq ST_n, n \geq 1$

- G simulates the application of $ipA \rightarrow qy \in R$:
 - make a left-to-right scan of the pd until the i th occurrence of a non-terminal
 - if $X_i = A$, then replace A with y and return to the beginning of the sentential form
 - rightmost symbol is always a special a' , and G completes the simulation by changing a' to a

Proof 5/6: $ST_n \subseteq PD_n, n \geq 1$

- ${}_nM$ simulates the n -limited derivations of G in pd:
 - always records the first n non-terminals from the current sentential form of G in its state
 - fewer than n non-terminals are extended by #s
 - reads the string, empties pd, enters $\$ \in F$

Proof 6/6: $PD_n \subset PD_{n+1}$, $n \geq 1$

1) As $PD_n \subseteq ST_n$ and $ST_n \subseteq PD_n$
for all $n \geq 1$, $ST_n = PD_n$.

2) Kasai (1970): $ST_n \subset ST_{n+1}$, for all $n \geq 1$.

For all $n \geq 1$, $PD_n = ST_n \subset ST_{n+1} = PD_{n+1}$

Q. E. D.

Open Problem Areas

- Determinism
- Rules of the form $m q A \rightarrow p \varepsilon$

Discussion and End

- Any questions, please?