AUTOMATA THEORY AND FORMAL LANGUAGES

2022-23

UNIT 6: PUSH-DOWN AUTOMATA

José Luis Mira Peidro jmira@inf.uc3m.es
Universidad Carlos III de Madrid

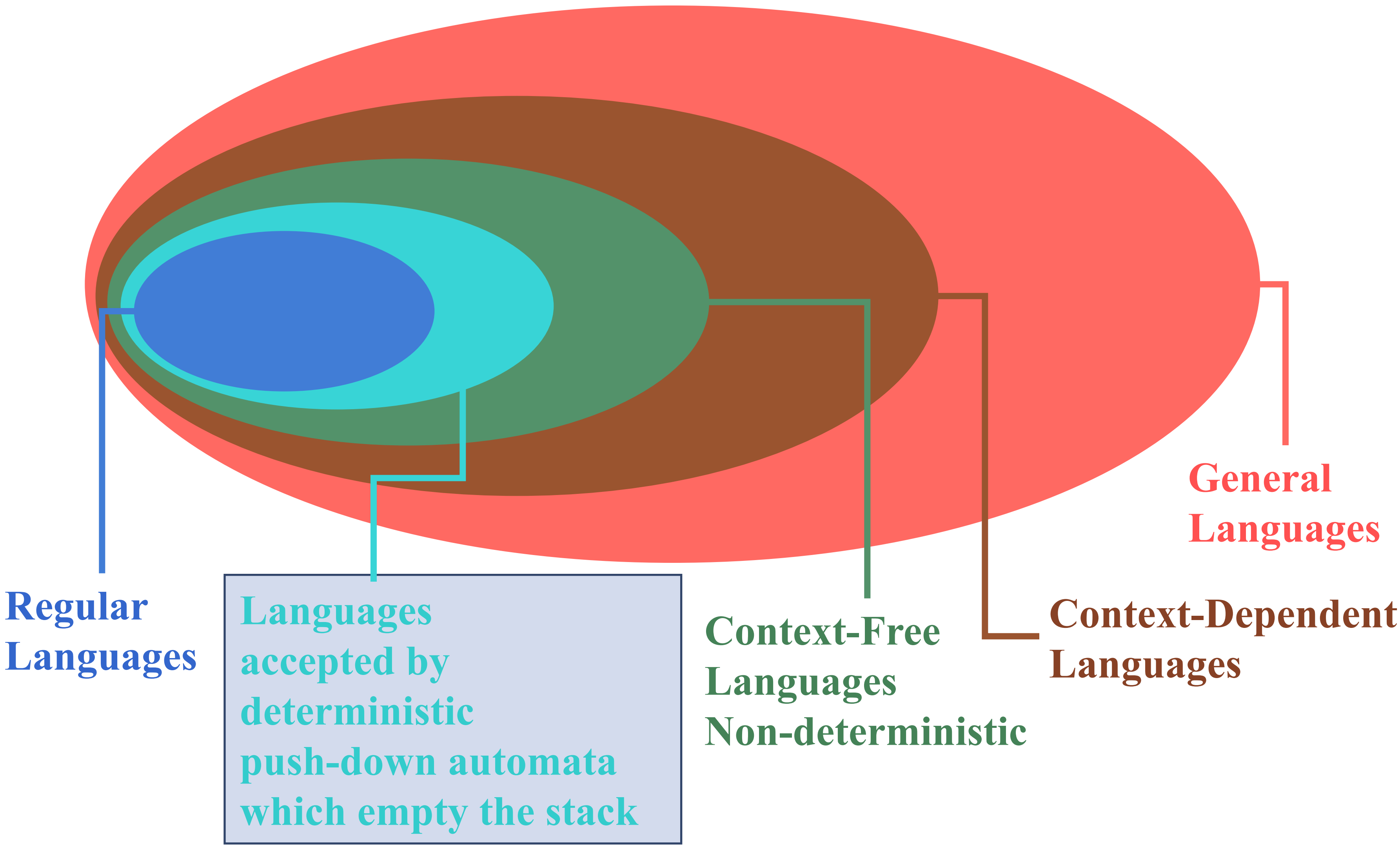
Bibliography

- Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. Teoría de autómatas y lenguajes formales. McGraw-Hill (2007). Chapter 4 and Section 8.1.
- John E. Hopcroft, Rajeev Motwani, Jeffrey D.Ullman. Introduction to Automata Theory, Languages, and Computation (3rd edition). Ed, Pearson Addison Wesley. Chapter 6.
- Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. Teoría de lenguajes, gramáticas y autómatas. Publicaciones R.A.E.C. 1997. Chapter 10.

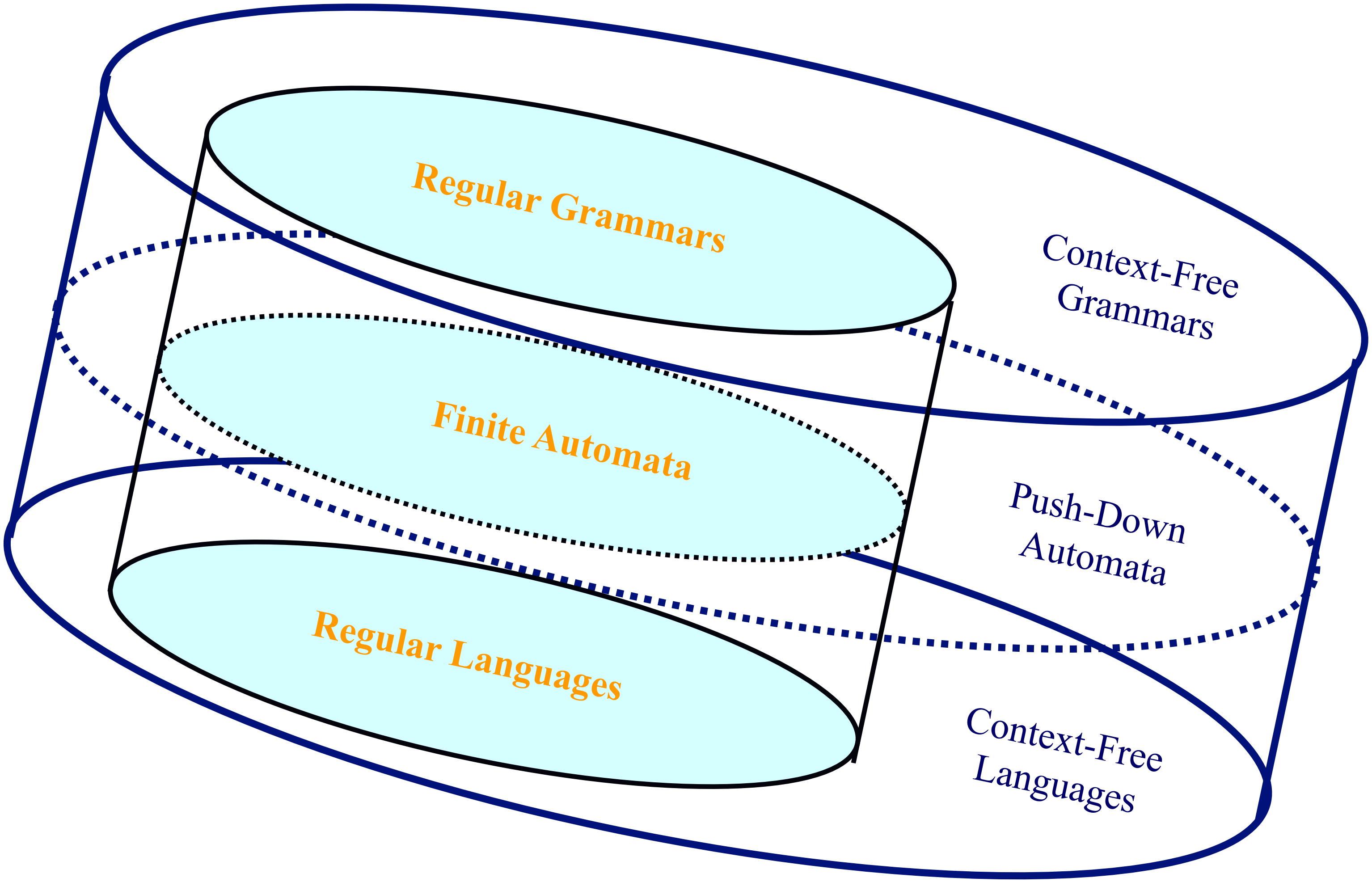
Outline

- **Introduction**
- Definition of Push-Down Automata
  - Acceptance in final states or when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Introduction. Language Hierarchy



Introduction



Introduction

- **Limitations of FA's:**
  - Only repetition sentences can be recognized.
    - E. g. $a^n b^n$, $a^n b^n c^n$
  - It is not possible to determine if a program is correct.
  - It is not possible to determine syntax errors present in natural language.

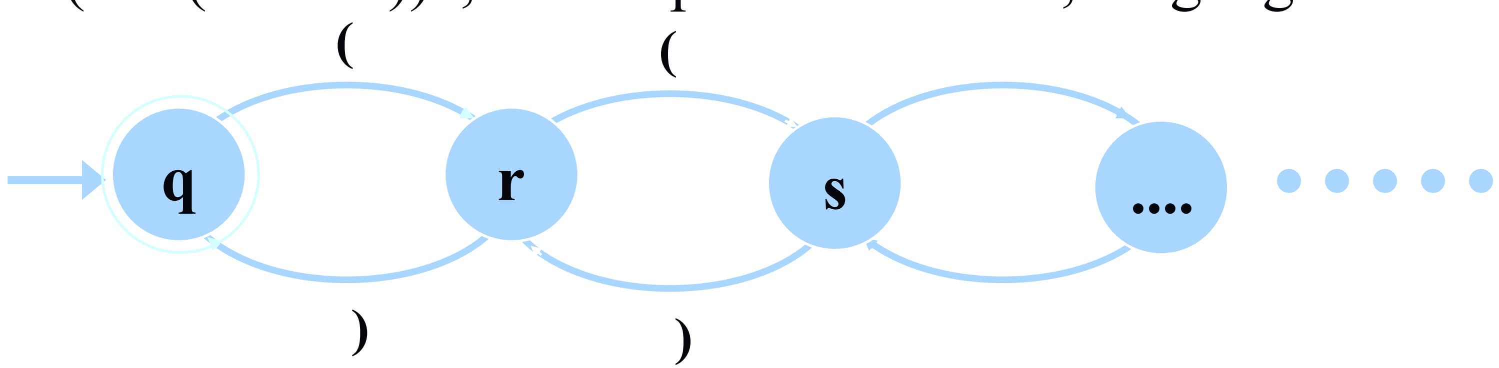
Introduction

Limitations of FA's: Explanation.

Lack of Memory

Mathematical expressions cannot be recognized,

e.g. "(2x+(2+n/25))", nested paired brackets, language X^nY^n



Push-Down Automata and Languages

- Function: Analyze words to know if they belong to Type-2 languages: **Accept or not accept.**

- Same structure that a finite automata adding a stack (auxiliary memory).

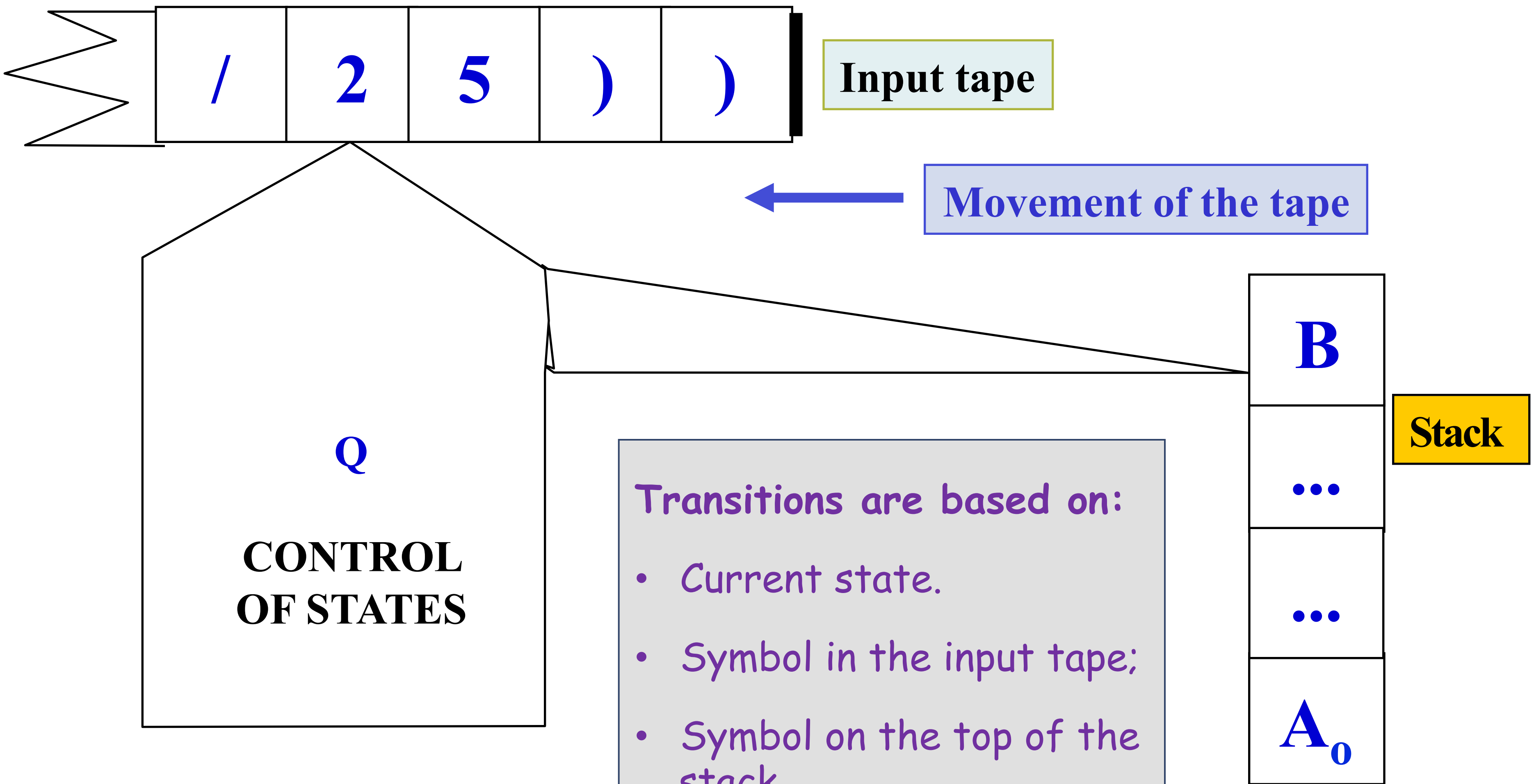
Push-Down Automata and Languages

- Theorems:
  - For each context-free grammar G, there is a push-down automaton M that fulfills L(G)=L(M)
  - For each push-down automata M, there is a context-free grammar G that fulfills L(M)=L(G)
  - There are context-free languages that cannot be recognized by any deterministic push-down automaton.

Outline

- Introduction
- **Definition of Push-Down Automata**
  - Acceptance in final states
  - Acceptance when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

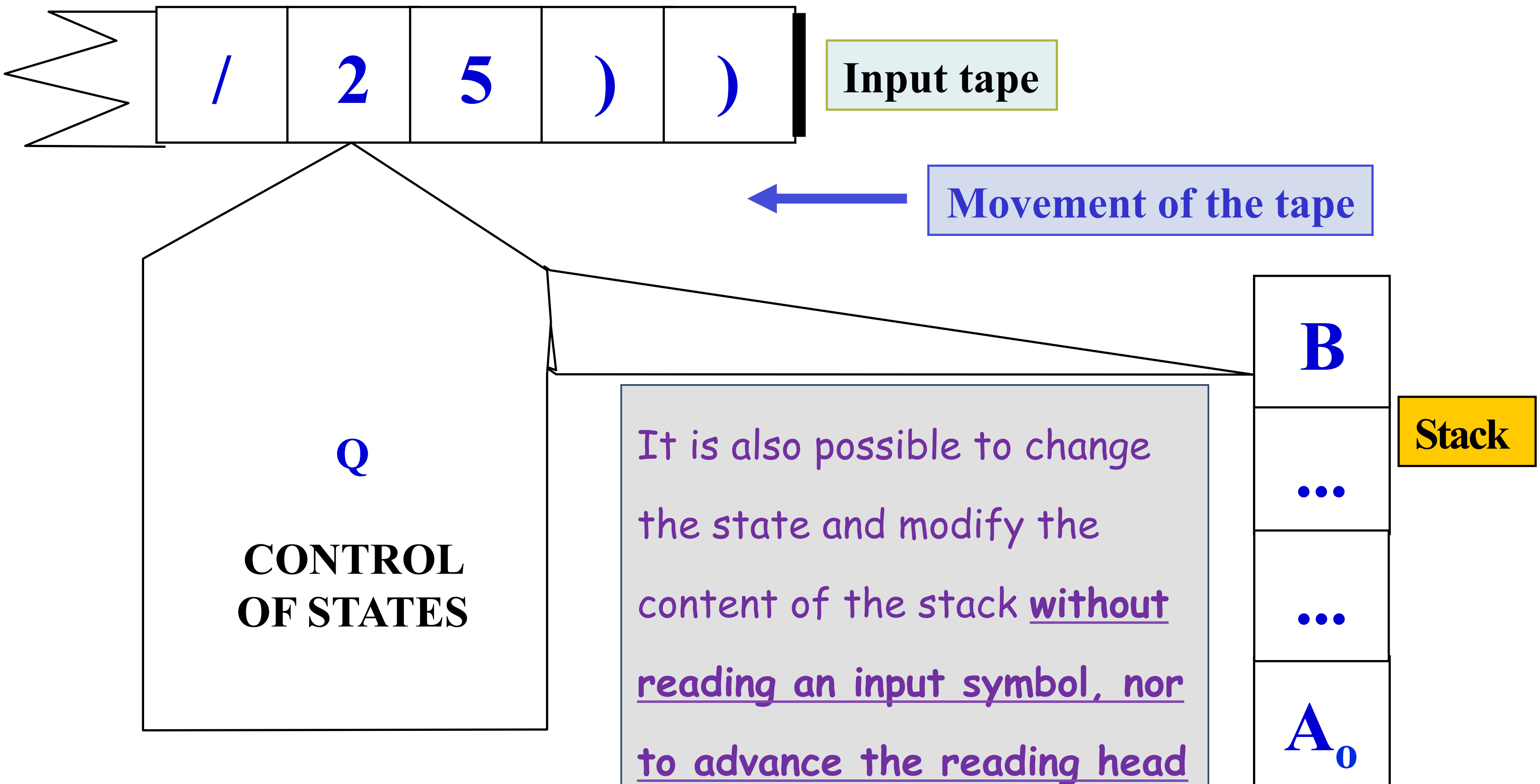
Definition of Push-Down Automaton



Transitions are based on:

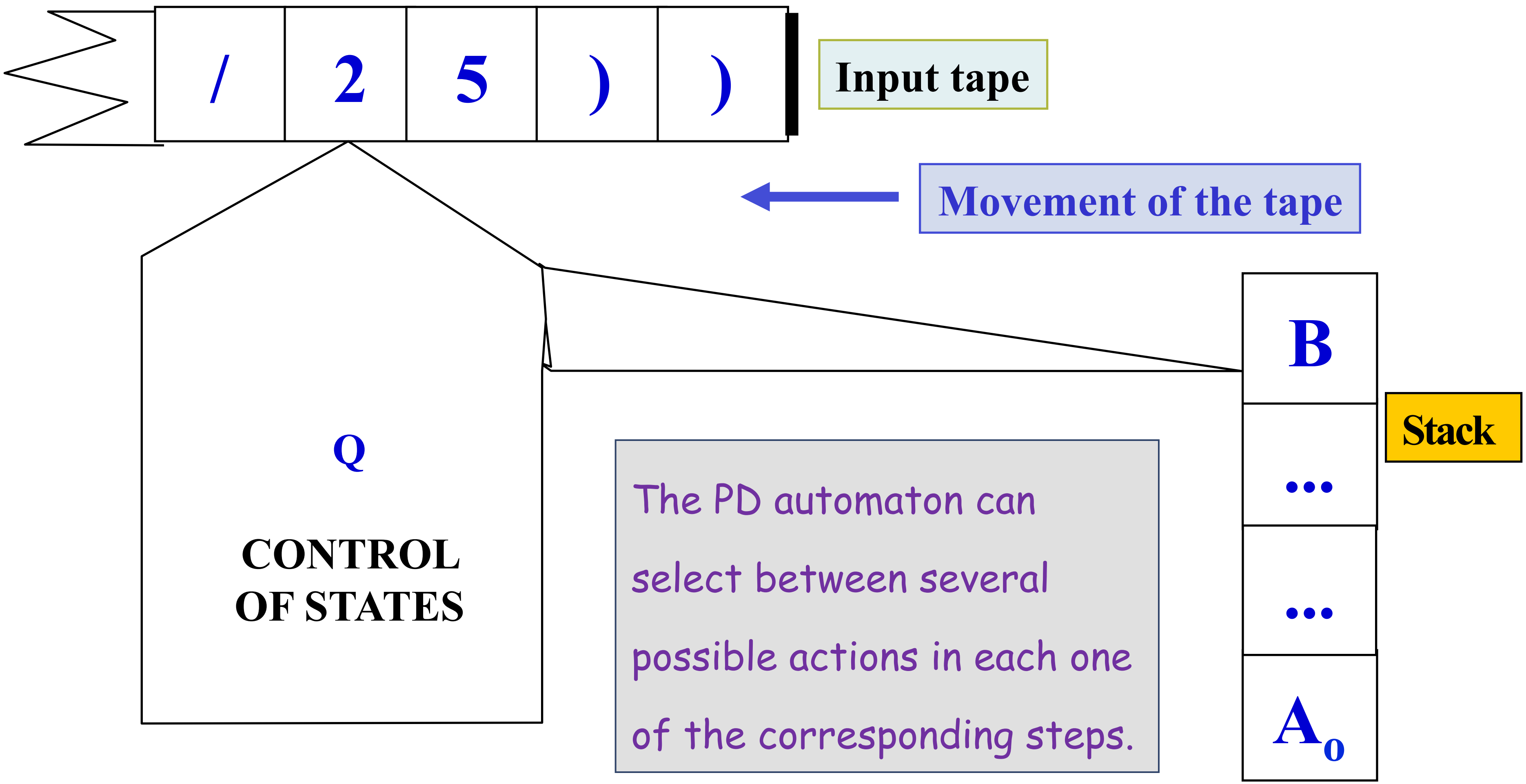
- Current state.
- Symbol in the input tape;
- Symbol on the top of the stack.

Definition of Push-Down Automaton



It is also possible to change the state and modify the content of the stack **<u>without reading an input symbol, nor to advance the reading head of the tape.</u>**

Definition of Push-Down Automaton

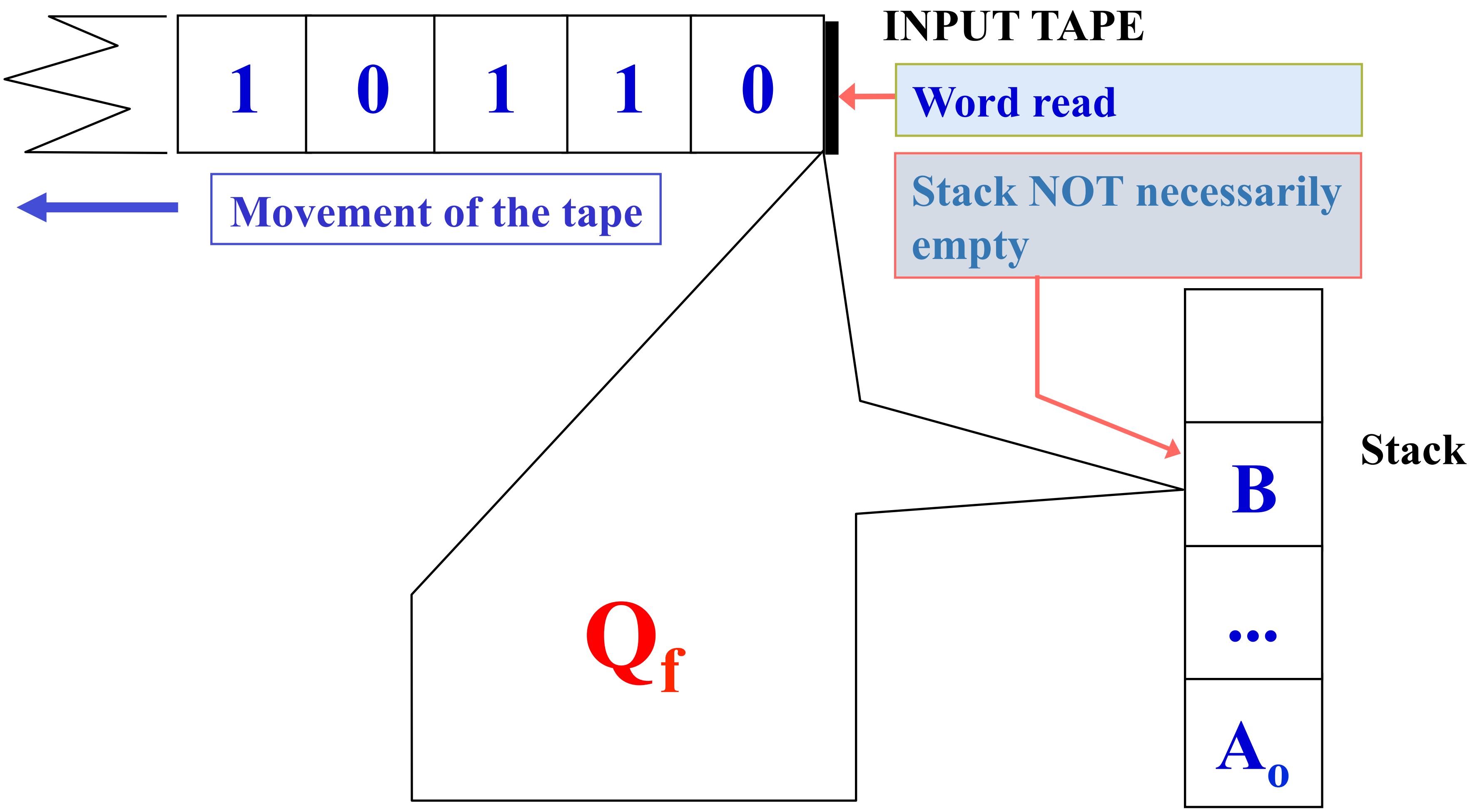


The PD automaton can select between several possible actions in each one of the corresponding steps.

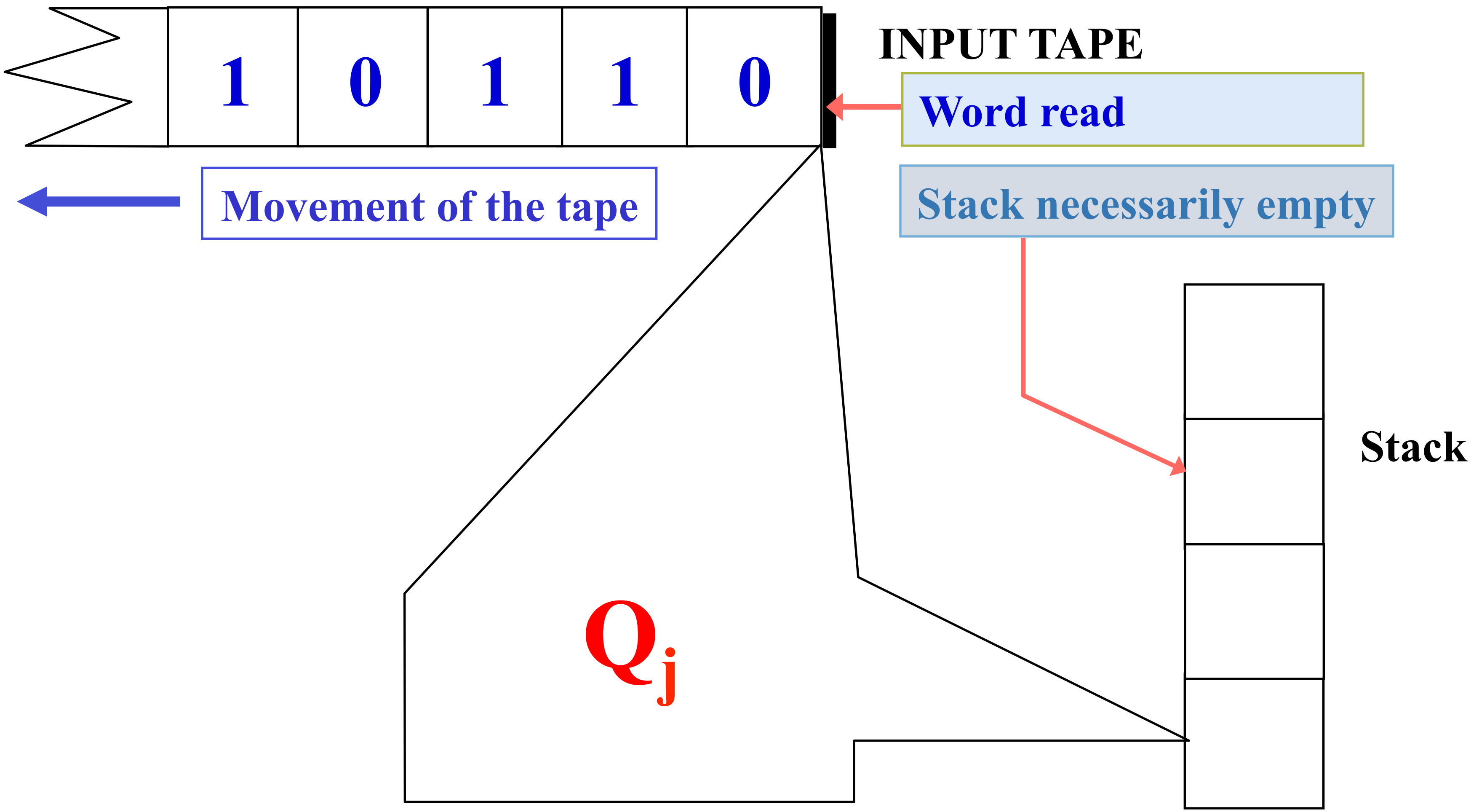
Outline

- Introduction
- **Definition of Push-Down Automata**
  - **Acceptance in final states or when the stack is empty**
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Acceptation in final states



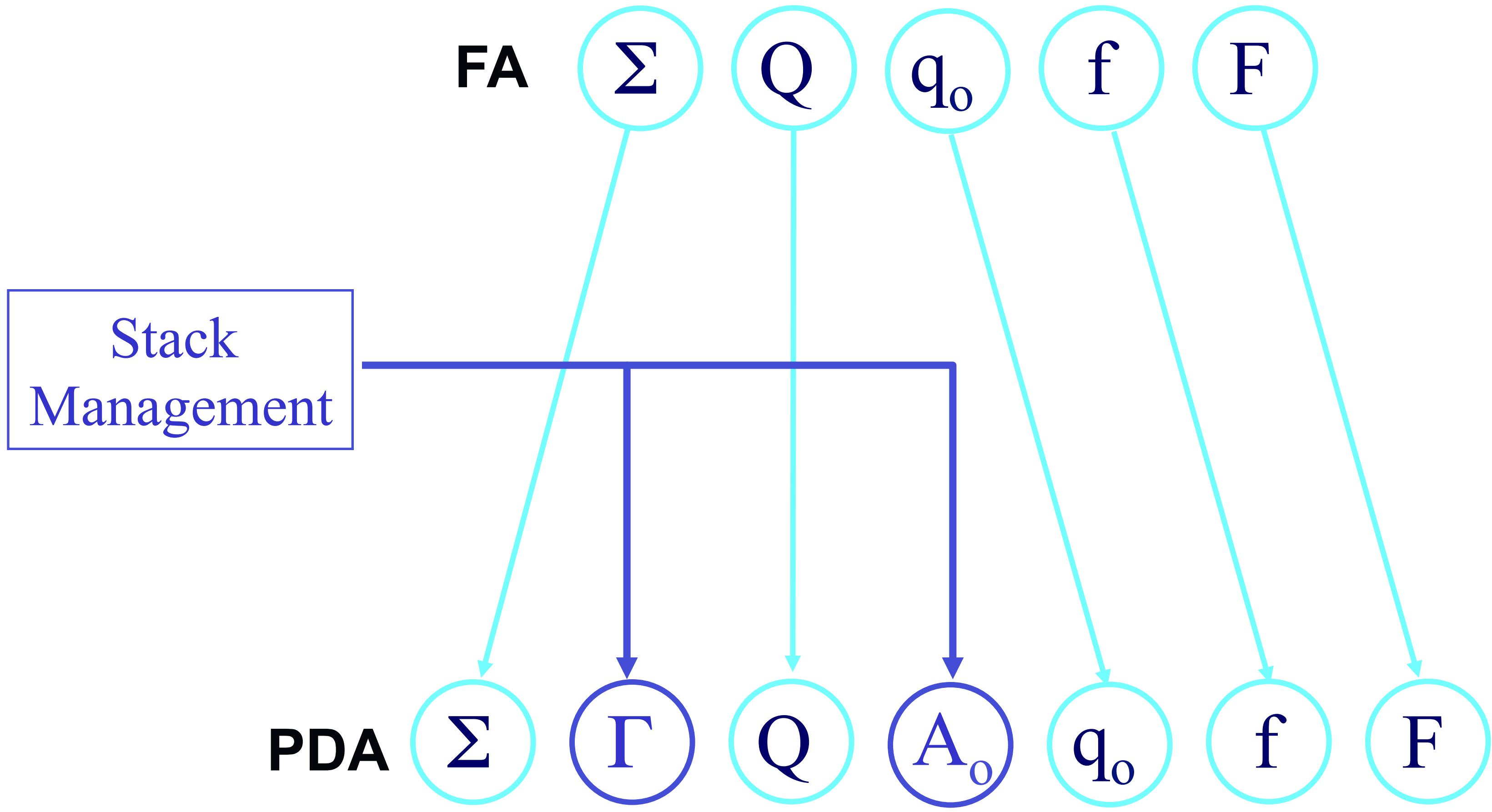
Acceptation when the stack is empty



Outline

- Introduction
- **Definition of Push-Down Automata**
  - Acceptance in final states or when the stack is empty
  - **Formal definition**
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Formal definition of Push-Down Automaton



Formal definition of Push-Down Automaton

PDA: $(\Sigma, \Gamma, Q, A_o, q_o, f, F)$

- **Σ : input alphabet (tape)** **Input Words:** x, y, z, ax, ay… $\in \Sigma^*$.
- **Γ : stack alphabet** **Words in the stack:** X, Y, Z, AX, AY… $\in \Gamma^*$
- **Q : finite set of states** **Q =** {p,q,r,…}
- **$A_o \in \Gamma$: initial symbol in the stack**
- **$q_o \in Q$: initial state of the automaton**
- **f : transition function**
- **$F \subset Q$: set of final states**

Outline

- Introduction
- **Definition of Push-Down Automata**
  - Acceptance in final states or when the stack is empty
  - Formal definition
  - **Transitions**
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Push-Down Automaton. Transitions

- **<u>Transition function</u>:**

$$f : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P\,(Q \times \Gamma^{*})$$

For each state, input symbol in the tape or empty word, and symbol on the top of the stack, the automaton determines the transition to another state and decides the symbols to be written in the stack.

$P\,(Q \times \Gamma^{*})$ are the parts of $Q \times \Gamma^{*}$

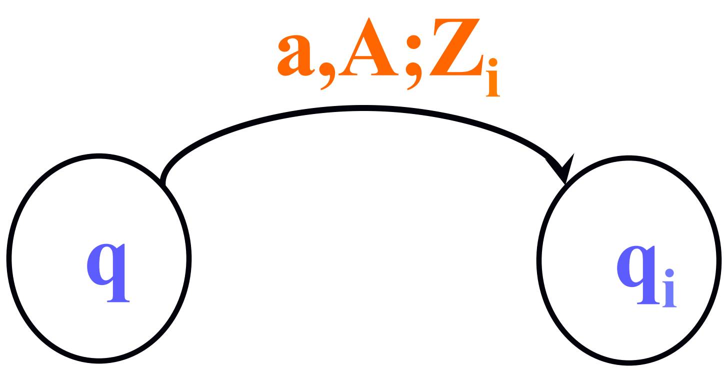
Example, $Q=\{p,q\}$ $\Gamma=\{A,B\}$,

$Q \times \Gamma^{*}=\{(p, \lambda),(q, \lambda), (p,A), (p,B), (q,A), (q,B), (p,AA), (p,AB)\ldots \}$

$\{(p, \lambda), (p,A), (p,B)\} \in P\,(Q \times \Gamma^{*})$

Push-Down Automaton. Transitions

- Transitions in a push-down automaton follow the following sequence:
  - Read an input symbol.
  - Extract a symbol from the stack.
  - Insert a word in the stack.
  - Transit to a new state.



- Definition:
  - $f(q,a,A)=\{(q_1,Z_1),(q_2, Z_2),\ldots,(q_n, Z_n)\}$
  - Another notation: $(q,a,A;\ q_i,Z_i)$

    where $q,\ q_i \in Q,\ \ a \in \Sigma,\ A \in \Gamma,\ \ Z_i \in \Gamma^*$

Push-Down Automaton. Transitions

$$f : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$$

Transitions that depend on the input

$Q \times \Sigma \times \Gamma$

Transitions that do not depend on the input

$Q \times \lambda \times \Gamma$

Deterministic Push-Down Automata

$Q \times \Gamma^*$

Non-Deterministic Push-Down Automata

$P(Q \times \Gamma^*)$

Push-Down Automaton. Transitions

Transitions that do not depend on the input

- Given the transition:

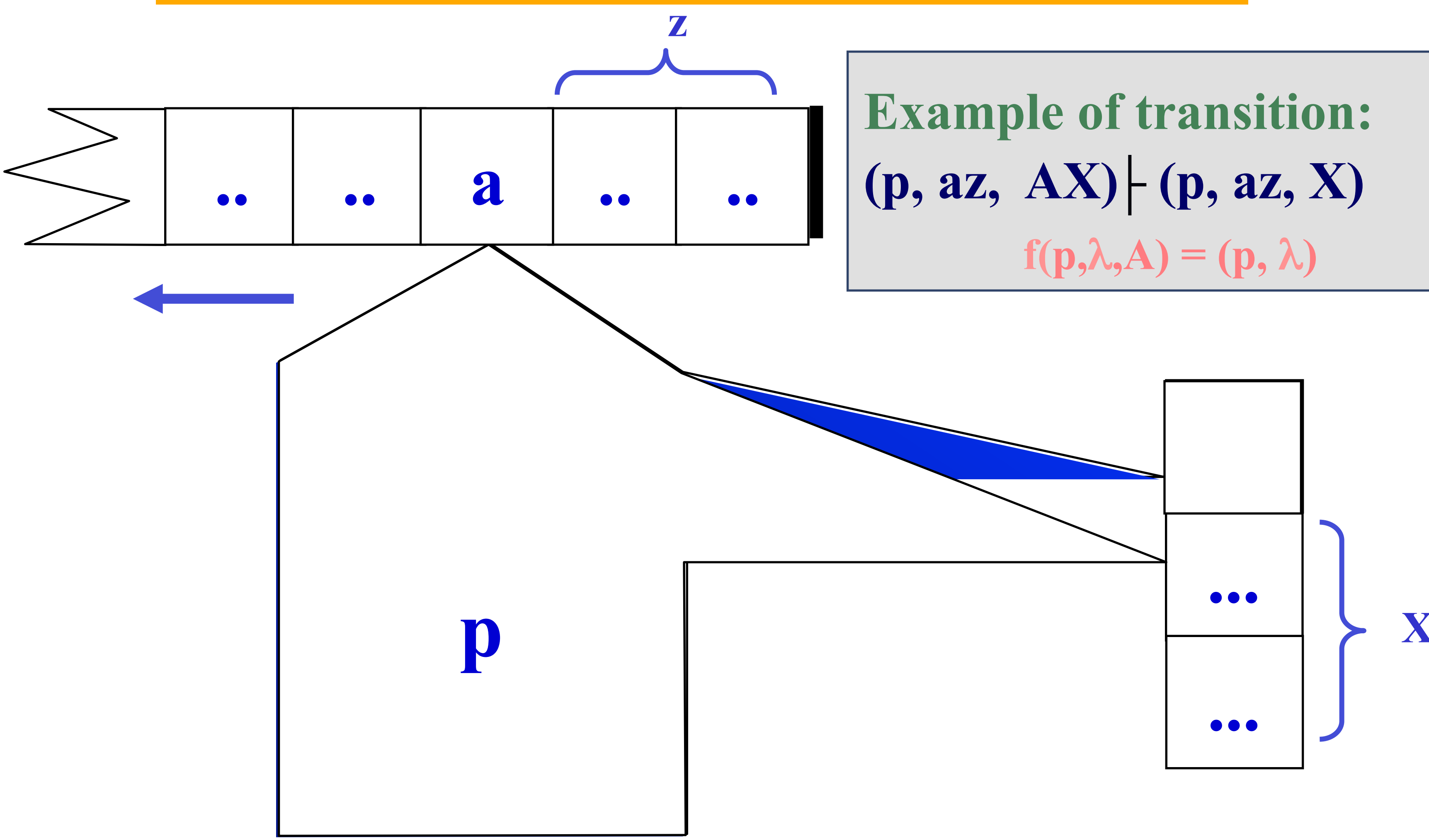
  $$f(q, \lambda, A) = \{(q_1, Z_1), (q_2, Z_2), \ldots, (q_n, Z_n)\}$$

  where:

  - $q, q_i \in Q$
  - $A \in \Gamma$
  - $Z_i \in \Gamma^*$

Push-Down Automaton. Transitions

Transitions that do not depend on the input



Example of transition:

$(p, az, AX) \vdash (p, az, X)$

$f(p,\lambda,A) = (p, \lambda)$

Push-Down Automaton. Transitions

Transitions that depend on the input

- **Given the transition:**

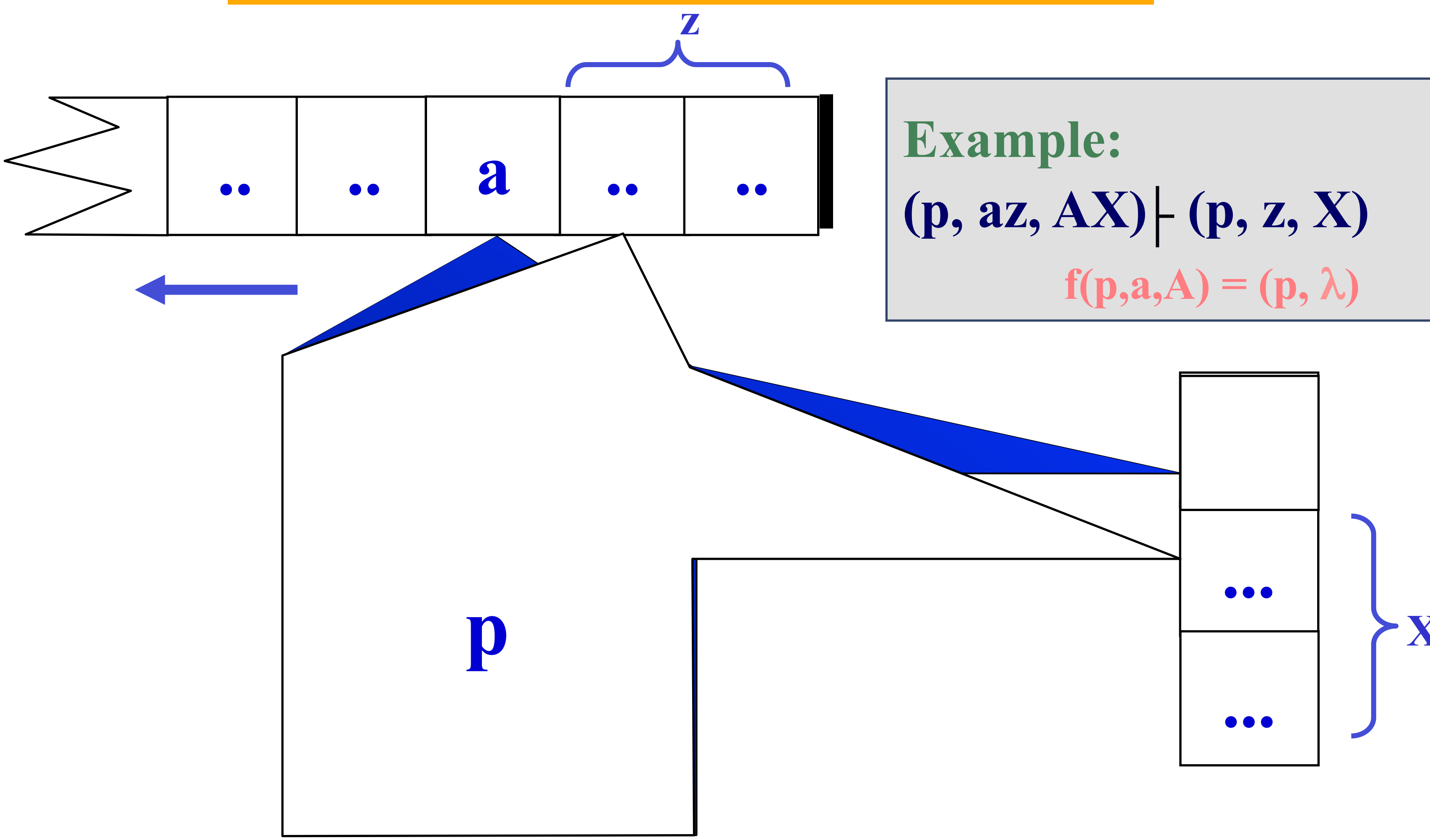
  $$f(q,a,A) = \{(q_1,Z_1), (q_2,Z_2),\ldots,(q_n,Z_n)\}$$

  where:

  - $q, q_i \in Q$
  - $a \in \Sigma$
  - $A \in \Gamma$
  - $Z_i \in \Gamma^*$

Push-Down Automaton. Transitions

Transitions that depend on the input



Example:

(p, az, AX) ├ (p, z, X)

f(p,a,A) = (p, λ)

Outline

- Introduction.
- **Definition of Push-Down Automata**
  - Acceptance in Final States or when the stack is empty
  - Formal definition
  - Transitions
  - **Instantaneous Description, Movement**
  - Deterministic Push-Down Automata
  - Language Accepted by a PD Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Instantaneous description

- It is used to easily describe the configuration of a Push-Down automaton in each moment.
  - Group of three (q,x,z)

    where $q \in Q$, $x \in \Sigma^*$, $z \in \Gamma^*$

  - It contains:
    - the current state (q);
    - the part of the input word that is still to be read (x);
    - the symbols on the stack (z).

Instantaneous description

- **Instantaneous description (q,x,z)** where $q \in Q$, $x \in \Sigma^*$, $z \in \Gamma^*$

- **Movement $(q,ay,AX) \vdash (p,y,YX)$** describes the transition from an instantaneous description to another.

  (q,ay,AX) precedes (p,y,YX) if f(q,a,A)=(p,Y)

- **Succession of movements: $(q,ay,AX) \;{}_{*}\!\vdash (p,y,YX)$** represents that the second instantaneous description can be reached from the first one.

Outline

- Introduction
- **Definition of Push-Down automata**
  - Acceptance in Final States or when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - **Deterministic Push-Down Automata**
  - Language Accepted by a PD Automaton
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Deterministic Push-Down Automaton

- $(\Sigma,\Gamma,Q,A_0,q_0,f,F)$ is deterministic if verifies:
  - $\forall q \in Q,\ A \in \Gamma,\ |f\ (q,\lambda,A)| > 0 \Rightarrow f\ (q,a,A) = \Phi\ \forall a \in \Sigma$
    - If there is a λ-transition, given a state *q* and a stack symbol *A*, then there is not any transition with any other input symbol.
  - $\forall q \in Q,\ A \in \Gamma,\ \forall a \in \Sigma \cup \{\lambda\},\ |\ f\ (q,a,A)\ | < 2$
    - There is **only one transition** given a state and a symbol on the top of the stack: f (q,a,A) = (p,X)
    - If (p, x, y; q, z) and (p, x, y; r, w) are transitions of a deterministic push-down automaton, then:

$$q \equiv r, \quad z = w$$

Outline

- Introduction
- **Definition of Push-Down Automata**
  - Acceptance in final states or when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - **Language Accepted by a Push-Down Automaton**
  - Examples
- Equivalence between PD Automata and Context-Free Languages

Language accepted by a Push-Down Automaton

- **Acceptance by empty stack:**
  - $L(PDA_E) = \{x \mid (q_0, x, A_0) \ {}_*\vdash (p, \lambda, \lambda), p \in Q, x \in \Sigma^*\}$
  - When the acceptance is when the stack is empty, the set of final states is irrelevant, and usually it is empty ($F = \emptyset$).

- **Acceptance by final state:**
  - $L(PDA_F) = \{x \mid (q_0, x, A_0) \ {}_*\vdash (p, \lambda, X), p \in F, x \in \Sigma^*, X \in \Gamma^*\}$

Outline

- Introduction
- **Definition of Push-Down Automata**
  - Acceptance in final states or when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - **Examples**
- Equivalence between PD Automata and Context-Free Languages

Definition of Push-Down Automaton. Example

LANGUAGE: set of sentences

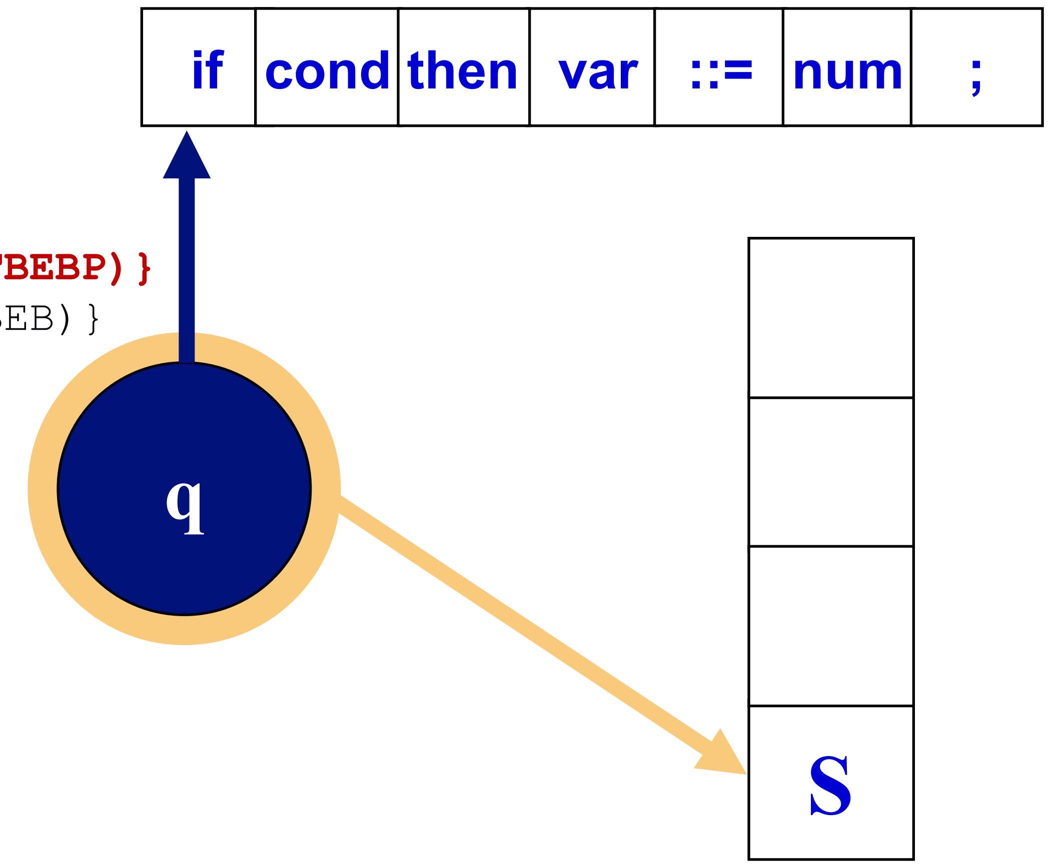
```
var ::= num;

if cond
then
     Assignation or IF

if cond
then
     Assignation or IF
else
     Assignation or IF
```

Definition of Push-Down Automaton. Example

AP= ({if, then, else, ::=, var, num, cond, ;},
{S, B, C, F, N, P, T, E}, {q}, q, S, f, ϕ)

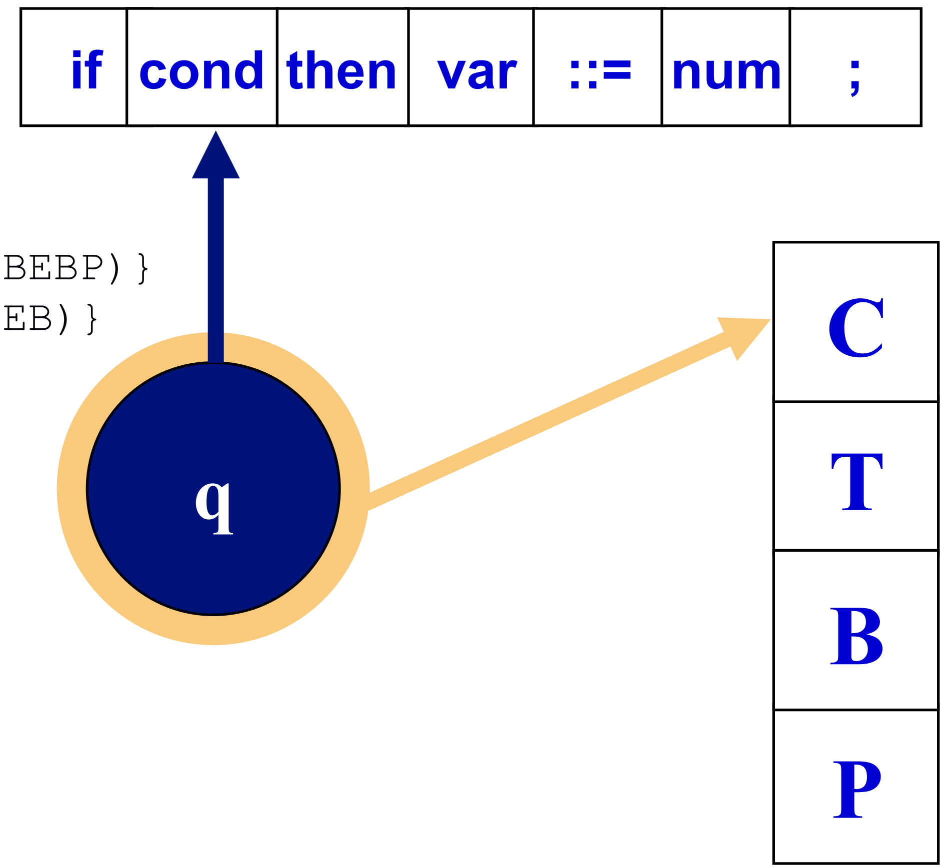
f(q, var, S) = {(q, FNP)}
f(q, if, S) = {(q, CTBP), (q, CTBEBP)}
f(q, if, B) = {(q, CTB), (q, CTBEB)}
f(q, var, B) = {(q, FN)}
f(q, cond, C) = {(q, λ)}
f(q, ::=, F) = {(q, λ)}
f(q, num, N) = {(q, λ)}
f(q, ;, P) = {(q, λ)}
f(q, then, T) = {(q, λ)}
f(q, else, E) = {(q, λ)}

Definition of Push-Down Automaton. Example



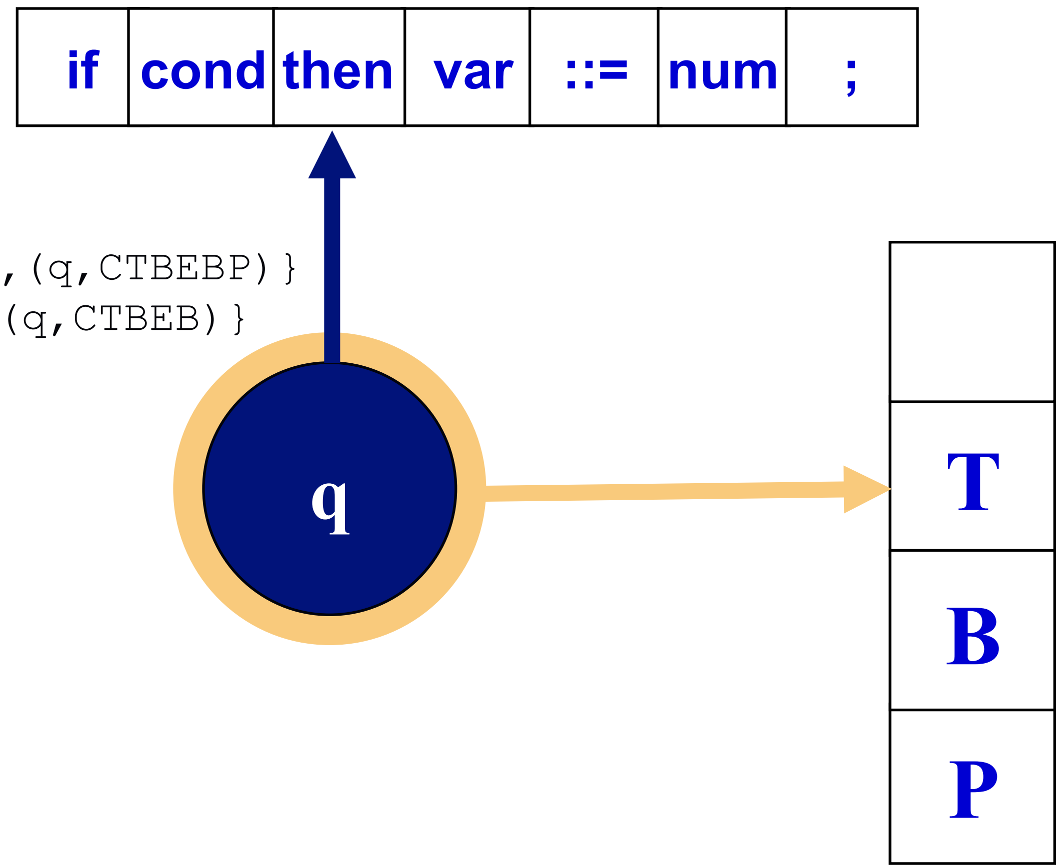

```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```

Definition of Push-Down Automaton. Example




```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```

Definition of Push-Down Automaton. Example

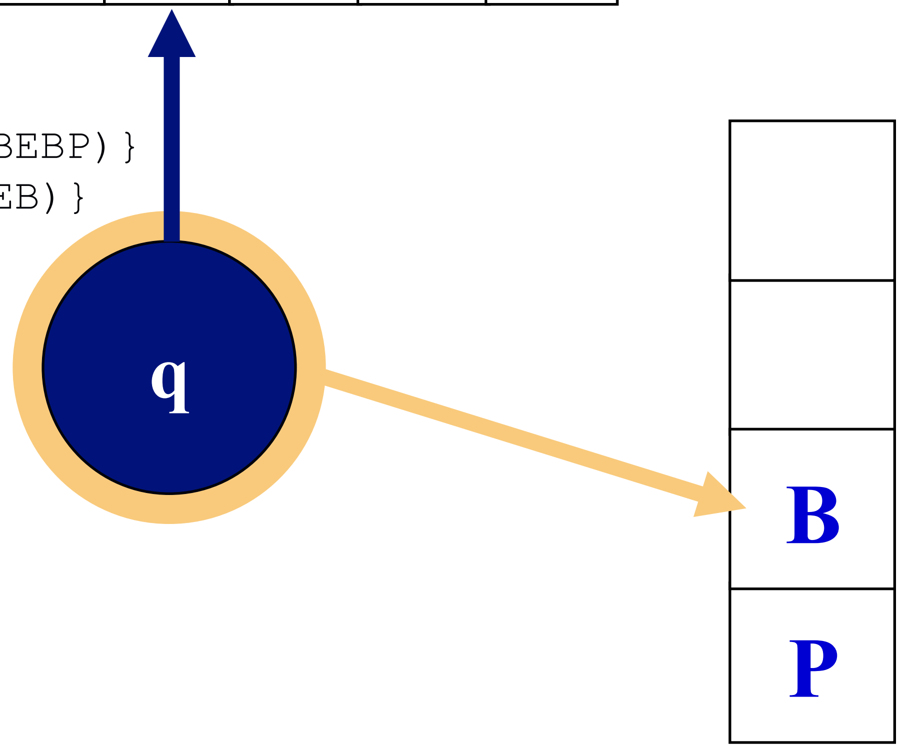



```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```

Definition of Push-Down Automaton. Example

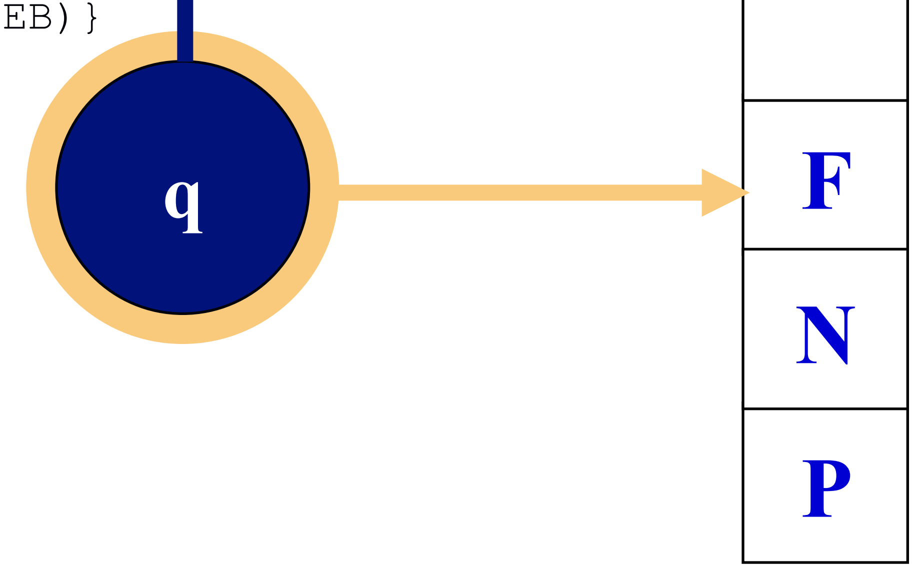
if	cond	then	var	::=	num	;

```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```



Definition of Push-Down Automaton. Example

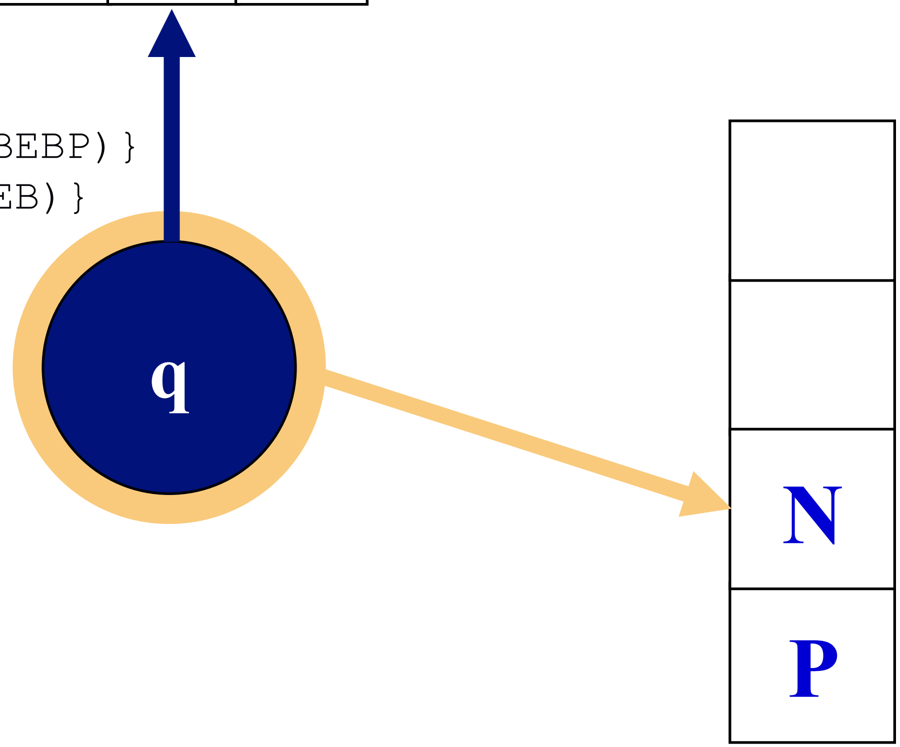

```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```



Definition of Push-Down Automaton. Example

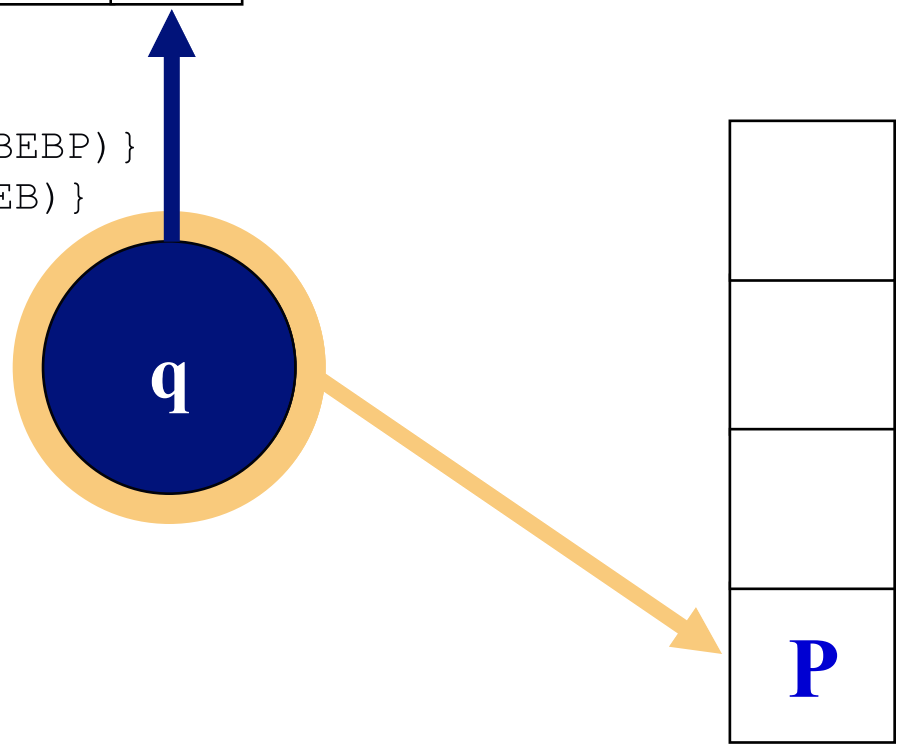
if	cond	then	var	::=	num	;

```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```

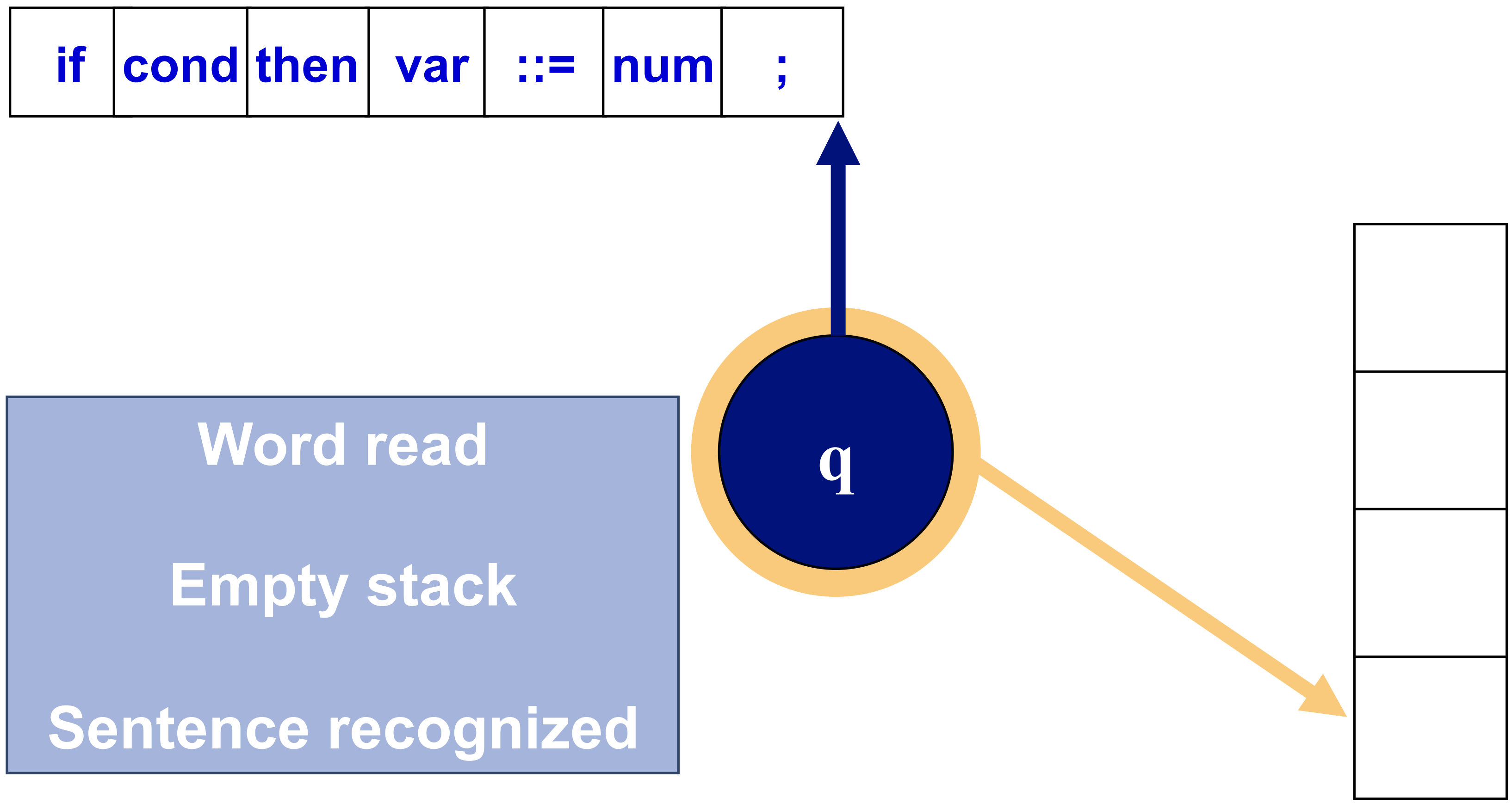


Definition of Push-Down Automaton. Example


```
f(q,var,S)   = {(q,FNP)}
f(q,if,S)    = {(q,CTBP),(q,CTBEBP)}
f(q,if,B)    = {(q,CTB),(q,CTBEB)}
f(q,var,B)   = {(q,FN)}
f(q,cond,C)  = {(q,λ)}
f(q,::=,F)   = {(q,λ)}
f(q,num,N)   = {(q,λ)}
f(q,;,P)     = {(q,λ)}
f(q,then,T)  = {(q,λ)}
f(q,else,E)  = {(q,λ)}
```



Definition of Push-Down Automaton. Example



Outline

- Introduction
- Definition of Push-Down Automata
  - Acceptance in final states or when the stack is empty
  - Formal definition
  - Transitions
  - Instantaneous Description, Movement
  - Deterministic Push-Down Automata
  - Language Accepted by a Push-Down Automaton
  - Examples
- **Equivalence between PD Automata and Context-Free Languages**

Push-Down Automaton. Theorem

- For each push-down automaton accepting strings without emptying the stack (**PDA$_F$**), there is an equivalent automaton that empties the stack accepting these strings (**PDA$_E$**).

$$\mathbf{L(PDA_F) = L(PDA_E)}$$

Equivalence PDA_E and PDA_F

From PDA_F to PDA_E

$$PDA_F = (\Sigma, \Gamma, Q, A_0, q_0, f, F)$$

$$PDA_E = (\Sigma, \boxed{\Gamma \cup \{B\}}, \boxed{Q \cup \{p, r\}}, \boxed{B}, \boxed{p}, f', \boxed{\phi})$$

New symbol B for the stack

Two new states

Initial value on the stack

New initial state

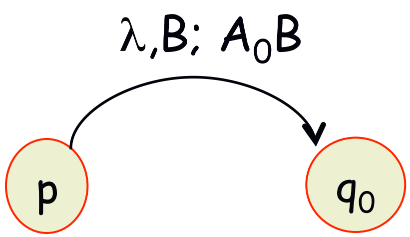
WITHOUT FINAL STATES

Equivalence PDA_E and PDA_F

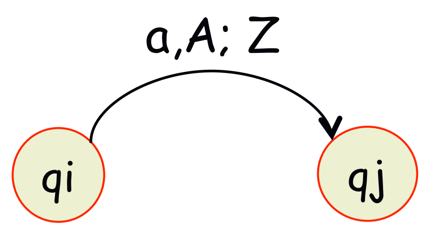
$PDA_F = (\Sigma, \Gamma, Q, A_0, q_0, f, F)$

$PDA_E = (\Sigma, \Gamma \cup \{B\}, Q \cup \{p, r\}, B, p, f', \phi)$

f' is defined as following:



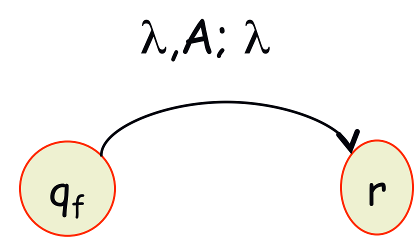
Transition independent of the input of the PDA_E with the first symbol of the stack transiting to the state q_0 of the PDA_F and putting A_0 on the stack.



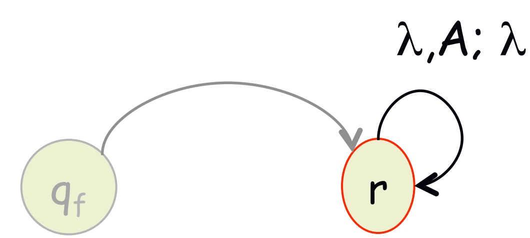
$q_i, q_j \in Q, \quad a \in \Sigma \cup \{\lambda\}$,
$A \in \Gamma, Z \in \Gamma^*$,

The transitions in the PDA_F are kept.

The characteristics of acceptance of this state are removed.



$\boxed{\forall\, q_f \in F,}\ A \in \Gamma \cup \{B\}$



$\forall\, A \in \Gamma \cup \{B\}$

Equivalence PDA_E and PDA_F

PDA_F = $(\Sigma, \Gamma, Q, A_0, q_0, f, F)$

PDA_E =$(\Sigma, \Gamma \cup \{B\}, Q \cup \{p,r\}, B, p, f', \phi)$ **f' is defined as following:**

- $f'(p, \lambda, B) = (q_0, A_0 \cdot B)$
  - A new initial state is incorporated and a transition from this new state to the original initial state of the PDA_F, the transition inserts A to which already existed: A_0B
- $f'(q, a, A) = f(q, a, A)$ $\forall q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $A \in \Gamma$
  - Keep old transitions, but eliminate the characteristic of acceptance of the final states.
- $f'(q, \lambda, A) = (r, \lambda)$ $\forall q \in F$, $A \in \Gamma \cup \{B\}$
  - A new state r is added receiving the transitions from final states, with all stack symbols, without reading, extracting or inserting symbols.
- $f'(r, \lambda, A) = (r, \lambda)$ $\forall A \in \Gamma \cup \{B\}$
  - For each $A \in \Gamma$, add the transition $(r,\lambda,A; r,\lambda)$

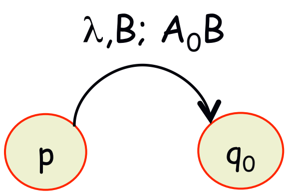
Equivalence PDA_E and PDA_F

From PDA_E to PDA_F

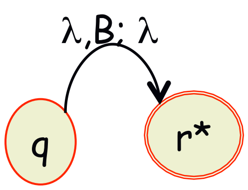
$PDA_E=(\Sigma, \Gamma, Q, A_0, q_0, f, \phi) \rightarrow$

$PDA_F=(\Sigma, \Gamma\cup\{B\}, Q \cup\{p, r\}, B, p, f', \{r\})$

f' is defined as following:



- $f'(p, \lambda, B) = (q_0, A_0 \cdot B)$
- $f(q, a, A) = f'(q, a, A) \quad \forall\, q \in Q,\; a \in \Sigma \cup \{\lambda\},\; A \in \Gamma$



- $f'(q, \lambda, B)=(r, \lambda) \quad \forall\, q \in Q,$

Equivalence PDA_E and PDA_F

$PDA_E = (\Sigma, \Gamma, Q, A_0, q_0, f, \phi) \rightarrow$

$PDA_F = (\Sigma, \Gamma \cup \{B\}, Q \cup \{p, r\}, B, p, f', \{r\})$

f' is defined as following:

- $f'(p, \lambda, B) = (q_0, A_0 \cdot B)$
  - The first transition of the PDA_F is to go to q0 of the PDA_V and write A_0B on the stack, verifying that B is on the top the stack.

- $f(q, a, A) = f'(q, a, A) \quad \forall\, q \in Q,\ a \in \Sigma \cup \{\lambda\},\ A \in \Gamma$
  - The transitions of the PDA_E are kept (the original PDA)

- $f'(q, \lambda, B) = (r, \lambda) \quad \forall\, q \in Q,$
  - When there is no input, it goes to the final state of the PDA_F: in the stack only remains B (that was introduced at the beginning) .

Push-Down Automaton and Type-2 Languages

- **Given a G2 in GNF, construct a PDA_E:**

$G = (\Sigma_T, \Sigma_N, S, P)$

input stack Initial symbol in the stack

- $PDA_E = (\Sigma_T, \Sigma_N, \{q\}, S, q, f, \phi)$ We obtain an PDA_E with only one state.

i.e., $\boxed{\begin{array}{l} f(q, a, A) = (q, Z) \\ f(q, a, A) = (q, \lambda) \end{array}}$ if there is a production with the form A ::= aZ. if there is a production with the form A ::= a

$f(q, a, A) = \{(q, Z), (q, \lambda)\}$

Given a production A::= aZ | aD | b $\Rightarrow$ $f(q, a, A)= \{(q, Z), (q, D)\}$
$f(q, b, A) = (q, \lambda)$

- **If $S::= \lambda \Rightarrow f(q, \lambda, S)= (q, \lambda)$**

Push-Down Automata and Type-2 Languages

- **Given a G2, construct a PDA_F:**
  - $G = (\Sigma_T, \Sigma_N, S, P)$
  - $PDA_E = (\Sigma_T, \Gamma, Q, A_0, q_0, f, \{q_2\})$
- Where:
  - $\Gamma = \Sigma_T \cup \Sigma_N \cup \{A_0\}$, where $A_0 \notin \Sigma_T \cup \Sigma_N$
  - $Q=\{q_0, q_1, q_2\}$, q_0 is the initial state, q_1 is the state from which transitions are carried out and q_2 is the final state.

- *f* is defined as follows:
  - $f(q_0, \lambda, A_0)=\{q_1, SA_0\}$
  - $\forall A \in \Sigma_N$, if $A ::= \alpha \in P$, $(\alpha \in \Sigma^*) \Rightarrow f(q_1, \lambda, A)=(q_1, \alpha)$
  - $\forall\ a \in \Sigma_T$, $f(q_1, a, a)=(q_1, \lambda)$
  - $f(q_1, \lambda, A_0) = (q_2, A_o)$

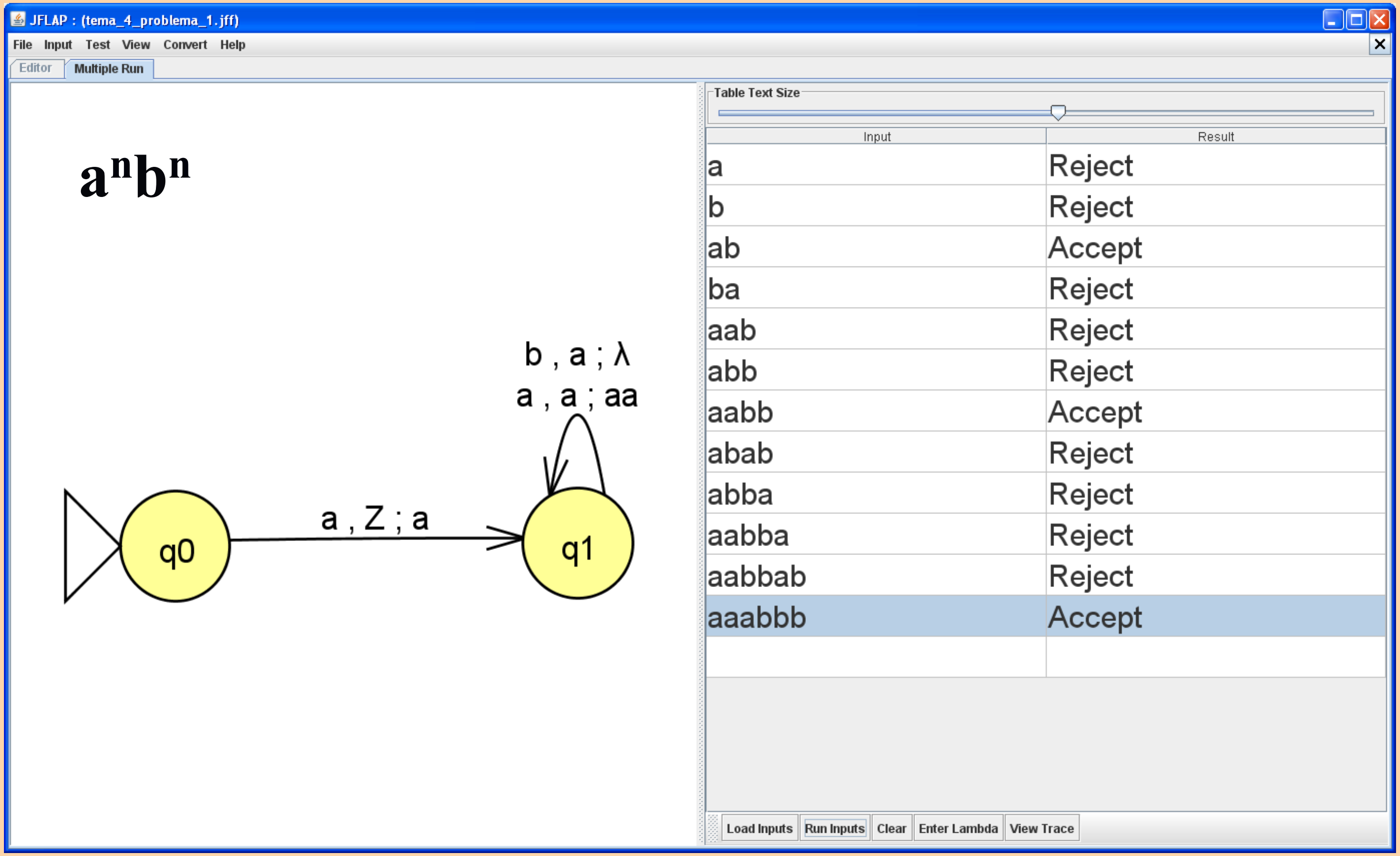
Push-Down Automata and Type-2 Languages

- **Given a PDA_E, construct a G2 that fulfills $L(G2) = L(PDA_E)$**
  - $PDA_E = (\Sigma, \Gamma, Q, A_0, q_o, f, \phi)$
  - $G = (\Sigma_T, \Sigma_N, S, P)$

- $\Sigma_N = \{S\} \cup \{ (p,A,q) \mid p,q \in Q,\ A \in \Gamma\}$

- **To construct P:**
  1. $S ::= (q_0, A_0, q)$ $\quad \forall q \in Q$ (select those that begins with q_0A_0)
  2. From each transition $f(p,a,A) = (q, BB'B''....B''')$ where $A,B,B',B'',\ldots,B''' \in \Gamma$; $a \in \Sigma \cup \{\lambda\}$
     - $(p\ A\ z) ::= a\ (q\ B\ r)\ (r\ B'\ s)\ s \ldots y\ (y\ B'''\ z)$
  3. From each transition $f(p, a, A) = (q, \lambda)$, we obtain: $(p,A,q) ::= a$

Push-Down Automaton and Type-2 Languages



- Determine if it is valid and propose other valid PDAs with acceptance when empty stack or final states.

Push-Down Automaton and Type-2 Languages

G2 in GNF →PDA$_E$

G2 = ({a,b,c,d,e,0,1}, {S,A,B,C,D,E,F,G,H}, S, P)

P = { S::= bDG/ cDG / bDH/ cDH/ bG/ cG/ bH/ cH

G::= aB

H::= aC

A::= bD/ cD/ b/ c

D::= bD/ cD/ b/ c

B::= 0E/ 0

E::= dCE/ dC

C::= 1F/ 1

F::= eBF/ eB}

Push-Down Automaton and Type-2 Languages

G2 in GNF →PDA_E

WELL-FORMED

G2 = ({a,b,c,d,e,0,1}, {S,A,B,C,D,E,F,G,H}, S, P)

P = { S::= bDG/ cDG / bDH/ cDH/ bG/ cG/ bH/ cH

G::= aB

H::= aC

A::= bD/ cD/ b/ c

D::= bD/ cD/ b/ c

B::= 0E/ 0

E::= dCE/ dC

C::= 1F/ 1

F::= eBF/ eB}

Push-Down Automaton and Type-2 Languages

Given a PDA_E, construct a G2 grammar that describes the same recognized language

- Example (Alfonseca _1997 Page 230)

Given the Apv = ({a,b}, {A,B}, {p,q}, A, p, f, Ø), where f

$f(p,a,A) = \{ (p,BA)\}$

$f(p,a,B) = \{ (p,BB)\}$

$f(p,b,B) = \{ (q, \lambda)\}$

$f(q,b,B) = \{ (q, \lambda)\}$

$f(q, \lambda,B) = \{ (q,\lambda)\}$

$f(q, \lambda,A) = \{ (q,\lambda)\}$

Calculate the G2 grammar that describes the language recogniced by the PDA_E

Limitations of Push-Down Automata

- Is there a Context-Free Grammar which is able to recognize the language $a^n b^n c^n$?
- Is there a Push-Down automaton which is able to recognize the language $a^n b^n c^n$?