Push-down Automata
= FA + Stack

PDA

Definition

A push-down automaton M is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is the input alphabet (of terminal symbols, terminals)
Γ is the stack alphabet
$\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

PDA

intrinsically nondeterministic

Definition

A push-down automaton M is tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is the input alphabet (of terminal symbols, terminals)
Γ is the stack alphabet
$\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

PDA

intrinsically nondeterministic

Definition

A push-down automaton M is tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet (of terminal symbols, terminals)
- Γ is the stack alphabet
- $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
- q_0 is the initial state, $q_0 \in Q$
- F is a set of final states, $F \subseteq Q$

$(r,c) \in \delta(q,a,b)$ means that in a state q, reading input symbol a and popping b from the stack, the PDA may change to state r and push c on the stack

PDA

Compute via configurations

PDA

Compute via configurations

Given M = $(Q, \Sigma, \Gamma, \delta, q_0, F)$ a configuration of M is an element in

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

an initial configuration is (q_0, w, ε).

PDA

Compute via configurations

Given M = $(Q, \Sigma, \Gamma, \delta, q_0, F)$ a configuration of M is an element in

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

an initial configuration is (q_0, w, ε).

$(q,w,u) \vDash (r,w_1, u_1)$ if and only if
$w = aw_1$, $u = bu_2$, and $u_1 = cu_2$

PDA

Compute via configurations

Given $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ a configuration of M is an element in

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

an initial configuration is (q_0, w, ε).

one-step computation

$(q,w,u) \vDash (r,w_1, u_1)$ if and only if
$w = aw_1$, $u = bu_2$, and $u_1 = cu_2$

PDA

Compute via configurations

Given $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ a configuration of M is an element in

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

one-step computation

an initial configuration is (q_0, w, ε).

$(q,w,u) \vDash (r,w_1, u_1)$ if and only if
$w = aw_1$, $u = bu_2$, and $u_1 = cu_2$

Definition

The language recognised / accepted by a push-down automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, \varepsilon) \vDash^* (f, \varepsilon, \varepsilon) \text{ for some } f \in F\}$$

PDA

Compute via configurations

Given $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ a configuration of M is an element in

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

an initial configuration is (q_0, w, ε).

one-step computation

$(q,w,u) \vDash (r,w_1, u_1)$ if and only if
$w = aw_1$, $u = bu_2$, and $u_1 = cu_2$

Definition

zero-or-more-steps computation

The language recognised / accepted by a pu
$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, \varepsilon) \vDash^* (f, \varepsilon, \varepsilon) \text{ for some } f \in F\}$$

PDA vs. CFG

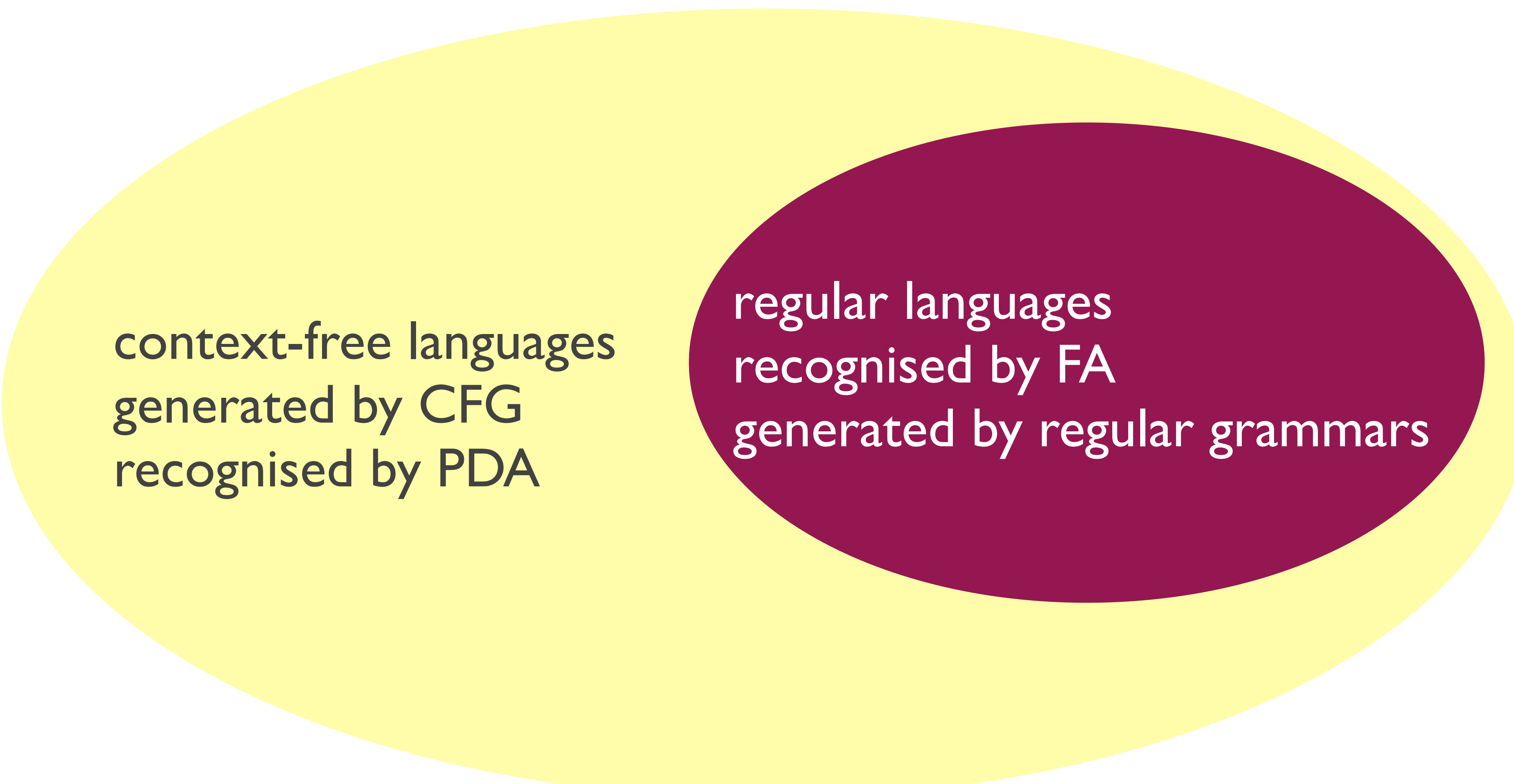
Theorem PDA-CFG

A language is context-free if and only if it is recognised by a push-down automaton.

PDA vs. CFG

Theorem PDA-CFG

A language is context-free if and only if it is recognised by a push-down automaton.



DPDA

Definitions

DPDA

Definitions

Two words u, v $\in \Sigma^*$ are consistent if none is a prefix of the other.

Two PDA transitions ((q,a,b),(r,c)) and ((p,d,e),(s,g)) are compatible if a and d, as well as b and e are inconsistent.

A PDA M is deterministic if no two different transitions are compatible.

DPDA

$(r,c) \in \delta(q,a,b)$

Definitions

Two words u, v $\in \Sigma^*$ are consistent if none is a prefix of the other.

Two PDA transitions ((q,a,b),(r,c)) and ((p,d,e),(s,g)) are compatible if a and d, as well as b and e are inconsistent.

A PDA M is deterministic if no two different transitions are compatible.

DPDA

Definitions

$(r,c) \in \delta(q,a,b)$

Two words $u, v \in \Sigma^*$ are consistent if none is a prefix of the other.

Two PDA transitions ((q,a,b),(r,c)) and ((p,d,e),(s,g)) are compatible if a and d, as well as b and e are inconsistent.

A PDA M is deterministic if no two different transitions

can fire at the same time

DPDA

Definitions

$(r,c) \in \delta(q,a,b)$

Two words $u, v \in \Sigma^*$ are consistent if none is a prefix of the other.

Two PDA transitions $((q,a,b),(r,c))$ and $((p,d,e),(s,g))$ are compatible if a and d, as well as b and e are inconsistent.

A PDA M is deterministic if no two different transitions can fire at the same time

Definition

A language L is a deterministic context-free language if there exists a DPDA M that recognises

$$L\$ = \{w\$ \mid w \in L\}$$

We say then that M recognises L and write $L = L(M)$.

DPDA

Definitions

Two words u, v $\in \Sigma^*$ are consistent if none is a prefix of the other.

(r,c) $\in \delta$(q,a,b)

Two PDA transitions ((q,a,b),(r,c)) and ((p,d,e),(s,g)) are compatible if a and d, as well as b and e are inconsistent.

A PDA M is deterministic if no two different transitions can fire at the same time

Definition

A language L is a deterministic context-free language if there exists a DPDA M that recognises

$$L\$ = \{w\$ \mid w \in L\}$$

We say then that M recognises L and write L = L(M).

\$ is a fresh symbol, not in Σ