Pushdown Automata

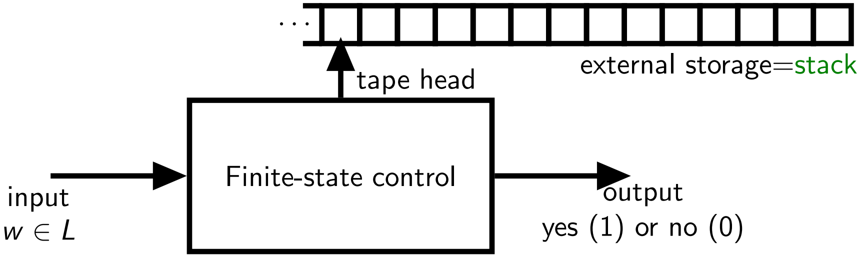
5DV037 — Fundamentals of Computer Science

Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

The Idea of a Pushdown Automaton

- The model of accepter for CFLs is called a *pushdown automaton* or *PDA*.
- It is basically an NFA with an auxiliary stack.
- The stack is a true stack; only push and pop operations are allowed.
- Only one stack is allowed.



Formal Definition of a Pushdown Automaton

- In this context, nondeterministic machines will be considered first.
- Deterministic machines will be considered as a special case later.
- A *nondeterministic pushdown automaton* or *NPDA* is a seven tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$$

  in which
  - Q is finite set of *states*;
  - Σ is an alphabet, called the *input alphabet*;
  - Γ is an alphabet, called the *stack alphabet*;
  - $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{Q \times \Gamma^*}_{\text{finite}}$ is a total function, the *state-transition function*;
  - $q_0 \in Q$ is the *initial state*;
  - $z_0 \in \Gamma$ is the *initial stack symbol*;
  - $F \subseteq Q$ is the set of *final* or *accepting states*.
- Here 2^X_{finite} denotes the set of *finite* subsets of X.

The Operation of an NPDA

$$\underbrace{Q}_{\text{Current state}} \quad \times \quad \underbrace{\Sigma \cup \{\lambda\}}_{\text{Input to be consumed}} \quad \times \quad \underbrace{\Gamma}_{\text{Current top of stack}} \quad \to \quad 2_{\text{finite}}^{\underbrace{Q}_{\text{New state}} \quad \times \quad \underbrace{\Gamma^*}_{\text{New top of stack}}}$$

- The number of possibilities at each step must be finite.
- Γ^* is an infinite set.
- Hence the restriction to finite subsets.

Formal Representation:

- Instead of an extended transition function, it is convenient to represent the operation of an NPDA with the move relation.
- First, a review of this notion for finite automata is given.

Review: IDs and the Move Relation for NFAs

- An *instantaneous description* (or *machine configuration* or *ID*) for the NFA $M = (Q, \Sigma, \delta, q_0, F)$ is a pair $(q, \alpha) \in Q \times \Sigma^*$ in which:
  - q represents the current state;
  - α represents the part of the input string which has not yet been read.
- $\mathrm{ID}\langle M\rangle = Q \times \Sigma^*$; the set of all possible IDs of M.
- The *move relation* $\vdash_M \subseteq \mathrm{ID}\langle M\rangle \times \mathrm{ID}\langle M\rangle$ represents one step of M and is defined by $(q_1, \alpha_1) \vdash_M (q_2, \alpha_2)$ iff
  - $\alpha_2 = \mathrm{Rest}\langle\alpha_1\rangle$ and $q_2 \in \delta(q_1, \mathrm{First}\langle\alpha_1\rangle)$; or
  - $\alpha_2 = \alpha_1$ and $q_2 \in \delta(q_1, \lambda)$.
- $\vdash_M^*$ is the reflexive and transitive closure of $\vdash_M$:
  - $(q, \alpha) \vdash_M^* (q, \alpha)$;
  - $(q_1, \alpha_1) \vdash_M^* (q_2, \alpha_2),\ (q_2, \alpha_2) \vdash_M^* (q_3, \alpha_3) \ \Rightarrow\ (q_1, \alpha_1) \vdash_M^* (q_3, \alpha_3)$.
- Thus $(q, \alpha_1\alpha_2) \vdash_M^* (\delta^*(q, \alpha_1), \alpha_2)$.

IDs and the Move Relation of an NPDA

- An *instantaneous description* (or *machine configuration* or *ID*) for the NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ is a triple $(q, \alpha, \gamma) \in Q \times \Sigma^* \times \Gamma^*$ in which:
  - q represents the current state;
  - α represents the part of the input string which has not yet been read.
  - γ represents the contents of the stack, top to bottom.
- $\mathrm{ID}\langle M\rangle = Q \times \Sigma^* \times \Gamma^*$; the set of all possible IDs of M.
- The *move relation* $\vdash_M \subseteq \mathrm{ID}\langle M\rangle \times \mathrm{ID}\langle M\rangle$ represents one step of M and is defined by $(q_1, \alpha_1, \gamma_1) \vdash_M (q_2, \alpha_2, \gamma_2)$ iff $\gamma_1 \neq \lambda$ and
  - $\alpha_2 = \mathrm{Rest}\langle\alpha_1\rangle$ and $(q_2, \gamma_2') \in \delta(q_1, \mathrm{First}\langle\alpha_1\rangle, \mathrm{First}\langle\gamma_1\rangle)$
    for some $\gamma_2' \in \Gamma^*$ with $\gamma_2 = \gamma_2' \cdot \mathrm{Rest}\langle\gamma_1\rangle$; or
  - $\alpha_2 = \alpha_1$ and $(q_2, \gamma_2') \in \delta(q_1, \lambda, \mathrm{First}\langle\gamma_1\rangle)$
    for some $\gamma_2' \in \Gamma^*$ with $\gamma_2 = \gamma_2' \cdot \mathrm{Rest}\langle\gamma_1\rangle$;
- $\vdash_M^*$ is the reflexive and transitive closure of $\vdash_M$:
  - $(q, \alpha, \gamma) \vdash_M^* (q, \alpha, \gamma)$;
  - $(q_1, \alpha_1, \gamma_1) \vdash_M^* (q_2, \alpha_2, \gamma_2),\ (q_2, \alpha_2, \gamma_2) \vdash_M^* (q_3, \alpha_3, \gamma_3)$
    $\Rightarrow\ (q_1, \alpha_1, \gamma_1) \vdash_M^* (q_3, \alpha_3, \gamma_3)$.

Acceptance by an NPDA

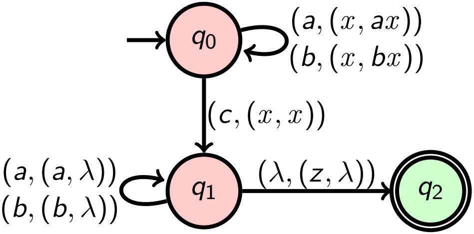
Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

- There are three common notions of acceptance by M of a string $\alpha \in \Sigma^*$.
  - Acceptance by *final state*:
    $\mathcal{L}_A(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \gamma)$ for some $q \in F$ and $\gamma \in \Gamma^*\}$
  - Acceptance by *empty stack*:
    $\mathcal{L}_E(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda)$ for some $q \in Q\}$
  - Acceptance by *final state and empty stack*.
    $\mathcal{L}_{AE}(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda)$ for some $q \in F\}$
- All three are equivalent in expressive power; this will be established later.
- The textbook uses only acceptance by final state, so this will be taken to be the default: $\mathcal{L}(M) = \mathcal{L}_A(M)$.

Example of an NPDA

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ which accepts L.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1, q_2\}$; $F = \{q_2\}$.
- The transition function δ may be described either by table or by diagram.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	c	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ

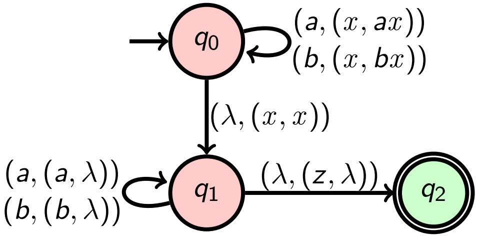


- The symbol x is used as a wildcard to reduce the number of entries.
- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

A Second Example of an Accepter

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ which accepts L.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1\}$; $F = \{q_2\}$.
- The solution is almost the same as for the previous example.
- Guess that the middle of string has been reached.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	λ	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ



- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

Basic Nondeterministic Top-Down Parsing

Algorithm (Basic top-down parsing): Given a CFG $G = (V, \Sigma, S, P)$, build an NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}(M) = \mathcal{L}(G)$.

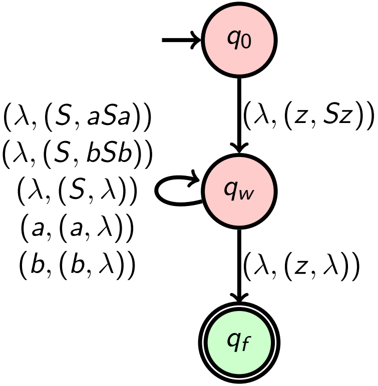
- Define: $Q = \{q_0, q_w, q_f\}$; $F = \{q_f\}$; $\Gamma = \Sigma \cup V$.
- The transition function δ is defined by two main operations and two auxiliary operations:
  Initialize: $(q_w, Sz) \in \delta(q_0, \lambda, z)$.
  Conjecture: For each $A \to \alpha \in V$, $(q_w, \alpha) \in \delta(q_w, \lambda, A)$.
  Match: For each $a \in \Sigma$, $(q_w, \lambda) \in \delta(q_w, a, a)$.
  Accept: $(q_f, \lambda) \in \delta(q_w, \lambda, z)$.

Theorem: Given any CFL L, there is an NPDA M with $\mathcal{L}_A(M) = L$. $\Box$

- This form of parsing is best illustrated by example.

An Illustration of Basic Top-Down Parsing

- Let $\Sigma = \{a, b\}$ and $G = \{\{S\}, \Sigma, S, \{S \rightarrow aSa, \mid bSb \mid \lambda\}\}$.
- $\mathcal{L}(G) = \{\alpha \cdot \alpha^R \mid \alpha \in \Sigma^*\}$.
- The algorithm on the previous slide yields the following machine:
- with the acceptance of *baaaab* shown to the right.



	$(q_0, baaaab, z)$	
$\vdash$	$(q_w, baaaab, Sz)$	Initialize
$\vdash$	$(q_w, baaaab, bSbz)$	Conjecture $S \rightarrow bSb$
$\vdash$	$(q_w, aaaab, Sbz)$	Match b
$\vdash$	$(q_w, aaaab, aSabz)$	Conjecture $S \rightarrow aSa$
$\vdash$	$(q_w, aaab, Sabz)$	Match a
$\vdash$	$(q_w, aaab, aSaabz)$	Conjecture $S \rightarrow aSa$
$\vdash$	$(q_w, aab, Saabz)$	Match a
$\vdash$	$(q_w, aab, aabz)$	Conjecture $S \rightarrow \lambda$
$\vdash$	(q_w, ab, abz)	Match a
$\vdash$	(q_w, b, bz)	Match a
$\vdash$	(q_w, λ, z)	Match b
$\vdash$	(q_f, λ, z)	Accept

Basic Top-Down Parsing is not a Practical Solution

- From a practical point of view, there are two major problems with basic top-down parsing:

Nondeterminism: The process is inherently nondeterministic

- The correct production must be chosen for each shift step.

Unbounded descent: If the grammar is left recursive, the algorithm may never terminate.

- This problem may be resolved by using grammars in Greibach normal form.

- Still, this form of parsing is useful because it proves that every CFG is accepted by some NPDA.

- More practical parsing will be examined briefly later.

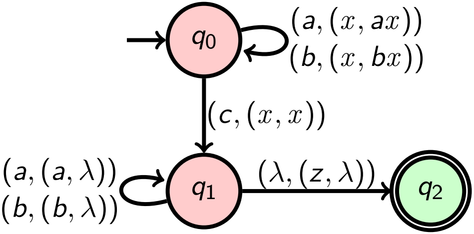
Constructing a CFG from an NPDA — Conditions

- The construction of a CFG from an NPDA is substantially more complex than the construction of a parser for a CFG.
- There is no easy proof.
- However, it is easier if acceptance by empty store is allowed.
- In the textbook, acceptance by empty store is covered only in an exercise (17 of Sec. 17.1).
- The proof of equivalence is very easy and will be covered here.
- Notions of acceptance by the NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$;
    - Acceptance by *final state*:
      $\mathcal{L}_A(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$
    - Acceptance by *empty stack*:
      $$\mathcal{L}_E(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda) \text{ for some } q \in Q\}$$
    - Acceptance by *final state and empty stack*.
      $$\mathcal{L}_{AE}(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda) \text{ for some } q \in F\}$$

Recall this Example of an NPDA

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ which accepts L.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1, q_2\}$; $F = \{q_2\}$.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	c	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ

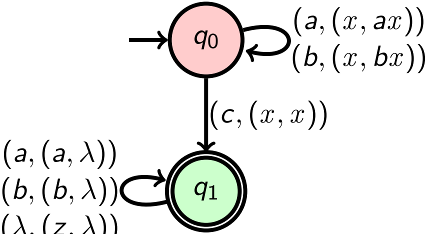


- The symbol x is used as a wildcard to reduce the number of entries.
- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

Example of Acceptance by Empty Stack

- With acceptance by empty stack, q_2 is not necessary.
- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}_E(M) = L$.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1\}$; $F = \emptyset$. $F = \cancel{\emptyset}\ \{q_1\}$

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	c	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_1	λ



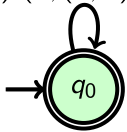
- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_1 an accepting state.

A Single-State Acceptor by Empty Stack

- In fact, q_1 is not necessary either.
- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a one-state NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}_E(M) = L$.
- $\Gamma = \{a, b, z, A\}$; $Q = \{q_0\}$; $F = \emptyset$. $F = \cancel{\emptyset}\ \{q_0\}$

Current			Next	
State	Input	Stack	State	Stack
q_0	a	z	q_0	Aaz
q_0	b	z	q_0	Abz
q_0	a	A	q_0	Aa
q_0	b	A	q_0	Ab
q_0	c	z	q_0	λ
q_0	c	A	q_0	λ
q_0	a	a	q_0	λ
q_0	b	b	q_0	λ
q_0	λ	z	q_0	λ

$(a, (z, Aaz))$ $(b, (z, Abz))$ $(a, (A, Aa))$
$(b, (A, Ab))$ $(c, (z, \lambda))$ $(c, (A, \lambda))$
$(a, (a, \lambda))$ $(b, (b, \lambda))$ $(\lambda, (z, \lambda))$

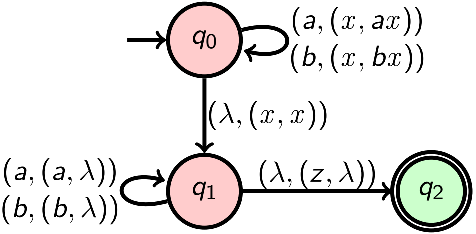


- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_0 an accepting state.

Recall a Second Example of an Accepter

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ which accepts L.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1\}$; $F = \{q_1\}$.
- The solution is almost the same as for the $\alpha \cdot c \cdot \alpha^R$ example.
- Guess that the middle of string has been reached.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	λ	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ

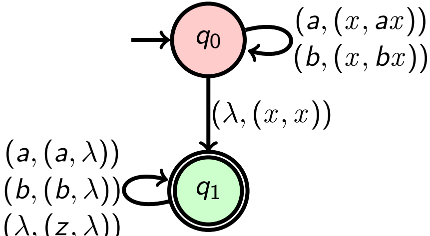


- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

A Second Example of Acceptance by Empty Stack

- With acceptance by empty stack, q_2 is not necessary.
- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}_E(M) = L$.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1\}$; $F = \emptyset$. $F = \cancel{\emptyset}\ \{q_1\}$

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	λ	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_1	λ



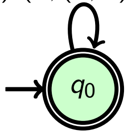
- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_1 an accepting state.

A Second Example of a Single-State Acceptor

- In fact, q_1 is not necessary either.
- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- Design a one-state NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}_E(M) = L$.
- $\Gamma = \{a, b, z, A\}$; $Q = \{q_0\}$; $F = \emptyset$. $F = \cancel{\emptyset}\ \{q_0\}$

Current			Next	
State	Input	Stack	State	Stack
q_0	a	z	q_0	Aaz
q_0	b	z	q_0	Abz
q_0	a	A	q_0	Aa
q_0	b	A	q_0	Ab
q_0	λ	A	q_0	λ
q_0	a	a	q_0	λ
q_0	b	b	q_0	λ
q_0	λ	z	q_0	λ

$(a, (z, Aaz))$ $(b, (z, Abz))$
$(a, (A, Aa))$ $(b, (A, Ab))$ $(\lambda, (A, \lambda))$
$(a, (a, \lambda))$ $(b, (b, \lambda))$ $(\lambda, (z, \lambda))$



- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_0 an accepting state.

Single-State Basic Top-Down Parsing

Algorithm (Basic top-down parsing): Given a CFG $G = (V, \Sigma, S, P)$, build a one-state NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with $\mathcal{L}_E(M) = \mathcal{L}(G)$.

- Define: $Q = \{q_0\}$; $\Gamma = \Sigma \cup V \cup \{z_f\}$ with $z_f \notin \Gamma \cup \Sigma$; $F = \emptyset$. $F = \cancel{\emptyset}\ \{q_0\}$

- The transition function δ is similar to that for the multi-state version.
  Initialize: $(q_0, Sz_f) \in \delta(q_0, \lambda, z)$.
  Conjecture: For each $A \to \alpha \in V$, $(q_0, \alpha) \in \delta(q_0, \lambda, A)$.
  Match: For each $a \in \Sigma$, $(q_0, \lambda) \in \delta(q_0, a, a)$.
  Accept: $(q_0, \lambda) \in \delta(q_0, \lambda, z_f)$.

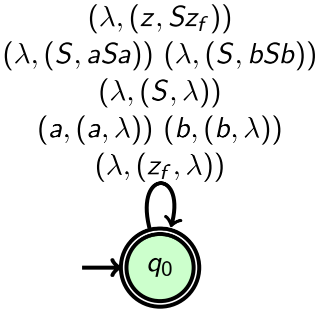
- Two "bottom-of-stack" symbols are used, z and z_f, to ensure that the machine does not accept λ without using the grammar.

Theorem: For any CFL L, there is a one-state NPDA M with $\mathcal{L}_E(M) = L$. $\square$

- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_0 an accepting state.

An Illustration of Basic One-State Top-Down Parsing

- Let $\Sigma = \{a, b\}$ and $G = \{\{S\}, \Sigma, S, \{S \to aSa, \mid bSb \mid \lambda\}\}$.
- $\mathcal{L}(G) = \{\alpha \cdot \alpha^R \mid \alpha \in \Sigma^*\}$.
- The operation is almost identical to that of the multi-state version.
- The acceptance of *baaaab* is shown to the right.



- To get $\mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$, make q_0 an accepting state.

	$(q_0, baaaab, z)$	
$\vdash$	$(q_0, baaaab, Sz_f)$	Initialize
$\vdash$	$(q_0, baaaab, bSbz_f)$	Conjecture $S \to bSb$
$\vdash$	$(q_0, aaaab, Sbz_f)$	Match b
$\vdash$	$(q_0, aaaab, aSabz_f)$	Conjecture $S \to aSa$
$\vdash$	$(q_0, aaab, Sabz_f)$	Match a
$\vdash$	$(q_0, aaab, aSaabz_f)$	Conjecture $S \to aSa$
$\vdash$	$(q_0, aab, Saabz_f)$	Match a
$\vdash$	$(q_0, aab, aabz_f)$	Conjecture $S \to \lambda$
$\vdash$	(q_0, ab, abz_f)	Match a
$\vdash$	(q_0, b, bz_f)	Match a
$\vdash$	(q_0, λ, z_f)	Match b
$\vdash$	(q_0, λ, λ)	Accept

Recall Acceptance by an NPDA

Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

- There are three common notions of acceptance by M of a string $\alpha \in \Sigma^*$.
  - Acceptance by *final state*:
    $\mathcal{L}_A(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \gamma) \text{ for some } q \in F \text{ and } \gamma \in \Gamma^*\}$
  - Acceptance by *empty stack*:
    $\mathcal{L}_E(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda) \text{ for some } q \in Q\}$
  - Acceptance by *final state and empty stack*.
    $\mathcal{L}_{AE}(M) = \{\alpha \in \Sigma^* \mid (q_0, \alpha, z) \vdash_M^* (q, \lambda, \lambda) \text{ for some } q \in F\}$

Theorem: For any $L \subseteq \Sigma^*$, the following are equivalent:

(i) $L = \mathcal{L}_A(M')$ for some NDPA M'.

(ii) $L = \mathcal{L}_E(M')$ for some NDPA M'.

(iii) $L = \mathcal{L}_{AE}(M')$ for some NDPA M'.

Furthermore, there are algorithms to convert between the forms.

Proof: Algorithms follow on the next slides.

Conversion from Acceptance by Final State

Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

Algorithm: Construct an NPDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', z', F')$ with

$$\mathcal{L}_A(M') = \mathcal{L}_E(M') = \mathcal{L}_{AE}(M') = \mathcal{L}_A(M).$$

- $Q' = Q \cup \{q_0', q_f'\}$, with $q_0, q_f' \notin Q$.
- $\Gamma' = \Gamma \cup \{z'\}$, with $z' \notin \Gamma$.
- $F' = \{q_f'\}$
- The transition function $\delta' : Q' \times \Sigma \cup \{\lambda\} \times \Gamma' \to Q' \times \Gamma'^*$ is defined by:
  - Prepare to simulate: $\delta'(q_0', \lambda, z') = \{(q_0, zz')\}$.
  - Simulate M:
    $$\delta(q, x, y) \subseteq \delta'(q, x, y) \text{ for all } (q, x, y) \in Q \times \Sigma^* \cup \{\lambda\} \times \Gamma.$$
  - Guess that input has ended:
    $$(q_f', \lambda) \in \delta'(q, \lambda, y) \text{ for all } q \in F \text{ and } y \in \Gamma.$$
  - Empty the stack: $\delta'(q_f', \lambda, y) = \{(q_f', \lambda)\}$ for all $y \in \Gamma'$. $\square$
- z' prevents the simulation from emptying the stack.

Conversion from Acceptance by Final State + Empty Stack

Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

Algorithm: Construct an NPDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', z', F')$ with

$$\mathcal{L}_A(M') = \mathcal{L}_E(M') = \mathcal{L}_{AE}(M') = \mathcal{L}_{AE}(M).$$

- $Q' = Q \cup \{q_0', q_f'\}$, with $q_0, q_f' \notin Q$.
- $\Gamma' = \Gamma \cup \{z'\}$, with $z' \notin \Gamma$.
- $F' = \{q_f'\}$
- The transition function $\delta' : Q' \times \Sigma \cup \{\lambda\} \times \Gamma' \to Q' \times \Gamma'^*$ is defined by:
  - Prepare to simulate: $\delta'(q_0', \lambda, z') = \{(q_0, zz')\}$.
  - Simulate M:
    $$\delta'(q, x, y) = \delta(q, x, y) \text{ for all } (q, x, y) \in Q \times \Sigma \cup \{\lambda\} \times \Gamma.$$
  - When the simulated stack is empty and the simulated state of M is accepting, delete z' and move to the accepting state of M':
    $$\delta'(q, \lambda, z') = \{(q_f', \lambda)\} \text{ for all } q \in F. \ \Box$$
- z' prevents the simulation from emptying the stack.

Conversion from Acceptance by Empty Stack

Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

Algorithm: Construct an NPDA $M' = (Q', \Sigma, \Gamma', \delta', q_0', z', F')$ with

$$\mathcal{L}_A(M') = \mathcal{L}_E(M') = \mathcal{L}_{AE}(M') = \mathcal{L}_E(M).$$

- Just apply the previous algorithm with $Q = F$.
- In that case, $\mathcal{L}_{AE}(M) = \mathcal{L}_E(M)$.

Theorem: For any $L \subseteq \Sigma^*$, the following are equivalent:

(i) $L = \mathcal{L}_A(M')$ for some NDPA M'.

(ii) $L = \mathcal{L}_E(M')$ for some NDPA M'.

(iii) $L = \mathcal{L}_{AE}(M')$ for some NDPA M'.

Furthermore, there are algorithms to convert between the forms. $\square$

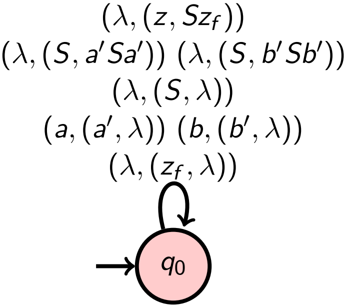
Obtaining a CFG from a One-State NPDA

Context: A one-state NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$.

Algorithm: Construct a CFG $G = (V, \Sigma, S, P)$ with $\mathcal{L}_E(M) = \mathcal{L}(L)$.

- Without loss of generality, assume that $\Sigma \cap \Gamma = \emptyset$.
- Define: $V = \Gamma$; $S = z$;
- Define
  $P = \{y \to x\beta \mid y \in \Gamma \text{ and } x \in \Sigma^* \cup \{\lambda\} \text{ and } (q_0, \beta) \in \delta(q_0, x, y)\}$. $\square$
- This algorithm is best illustrated by example.

Application to the Previous Top-Down Example



- The stack symbols a and b are renamed to a' and b'.

Transition(δ)	Production
$(q_0, \lambda, z) \mapsto (q_0, Sz_f)$	$z \to Sz_f$
$(q_0, \lambda, S) \mapsto (q_0, a'Sa')$	$S \to a'Sa'$
$(q_0, \lambda, S) \mapsto (q_0, b'Sb')$	$S \to b'Sb'$
$(q_0, \lambda, S) \mapsto (q_0, \lambda)$	$S \to \lambda$
$(q_0, a, a') \mapsto (q_0, \lambda)$	$a' \to a$
$(q_0, b, b') \mapsto (q_0, \lambda)$	$b' \to b$
$(q_0, \lambda, z_f) \mapsto (q_0, \lambda)$	$z_f \to \lambda$

- The start symbol of the grammar is z, not S.
- The task is to extend this construction to general NPDAs.
- The approach is to show that for every NPDA M, there is an one-state NPDA M' with $\mathcal{L}_E(M) = \mathcal{L}_E(M')$.

Simulation of an NPDA with a One-State Unit

Context: $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ an NPDA.

- The idea is to simulate the states of M with stack symbols of a one-state NPDA M'.
- A *transition triple* for M is a triple $\langle q, y, q' \rangle$ in which:
  - $q, q' \in Q$;
  - $y \in \Gamma$;
  - $(q, \alpha, y \cdot \gamma) \vdash_M^* (q', \alpha', \gamma)$ for some $\alpha, \alpha' \in \Sigma^*$ and $\gamma \in \Gamma^*$.
- The stack alphabet of the simulating one-state NPDA consists of transition triples (plus a start symbol).
- If $\gamma = y_1 y_2 \ldots y_k$ is the stack contents of M, and the state is q, then the stack contents of M' in the simulation is of the form $\langle q, y_1, q_1 \rangle \langle q_1, y_2, q_2 \rangle \ldots \langle q_{k-1}, y_k, q_k \rangle$ for some $q_1, q_2, \ldots, q_k \in Q$.
- In effect, the state of M is simulated in M' as an entry in the stack symbol.

Formal Construction:NPDA $\rightarrow$ One-State NPDA

Context: An NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$.

Algorithm: Construct a one-state NPDA with
$\mathcal{L}_{AE}(M') = \mathcal{L}_E(M') = \mathcal{L}_E(M)$.

- $Q' = q_0 = F'$.
- $\Gamma' = (Q \times \Gamma \times Q) \cup \{z'\}$
- The transition function $\delta' : Q' \times \Sigma \cup \{\lambda\} \times \Gamma' \rightarrow Q' \times \Gamma'^*$ is defined by:
  - Initialize: $\delta'(q_0', \lambda, z') = \{(q_0', \langle q_0, z, q\rangle) \mid q \in Q\}$
  - Simulate:

$$
\begin{aligned}
&\delta'(q_0', x, \langle p, y, q\rangle) = \\
&\qquad \{(q_0', \beta) \mid \beta = \langle q_1, b_1, q_2\rangle\langle q_2, b_2, q_3\rangle, \ldots, \langle q_k, b_k, q_{k+1}\rangle \\
&\qquad\quad \text{and } p = q_1 \text{ and } q = q_{k+1} \text{ and } (q, b_1 b_2 \ldots b_k) \in \delta(p, x, y)\} \\
&\qquad \cup \{(q_0', \lambda) \mid (q, \lambda) \in \delta(q, x, y)\}
\end{aligned}
$$

for $x \in \Sigma \cup \{\lambda\}$ and $\langle p, y, q\rangle \in Q \times \Gamma \times Q$.

Discussion of the Formal Construction

- Each stack symbol of the simulator (except the initial stack symbol) encodes three pieces of information

$$\langle \quad p \quad , \quad b \quad , \quad q \quad \rangle \quad \in \quad Q \times \Gamma \times Q$$

State of M
in simulation

Stack symbol
in M

The state that M must
go to when b is replaced by λ

- In the first step
  of the simulation, a triple of the following form is placed on the stack of M'.

$$\langle \quad q_0 \quad , \quad z \quad , \quad q \quad \rangle \quad \in \quad Q \times \Gamma \times Q$$

Start simulation
in Q of M

Start simulation
with initial stack symbol of M

End simulation
in some state of M

Discussion of the Formal Construction — 2

- To pop this triple off of the stack of the simulator directly, M must have the transition $(q, z) \in \delta(q_0, x, z)$ with x either the current input symbol or else λ.
- To pop this triple off of the stack of the simulator indirectly, conjecture that M goes through intermediate transitions.
- The first step must be to replace the initial stack symbol z with some string $\beta \in \Gamma^*$ and go to some state $q \in Q$: $(q, \beta) \in \delta(q_0, x, z)$ for input $x \in \Sigma \cup \{\lambda\}$.
- $\langle q_0, z, q\rangle \rightsquigarrow \langle q_0, b_1, p_1\rangle\langle p_1, b_2, p_2\rangle \ldots \langle p_k, b_k, p\rangle$ with $\beta = b_1 b_2 \ldots b_k$.
- The process continues, possibly replacing $\langle q_0, b_1, p_1\rangle$ with another string of transition triples.
- In an acceptance, the stack of the simulator M' will eventually be emptied.

Deterministic PDAs and CFLs

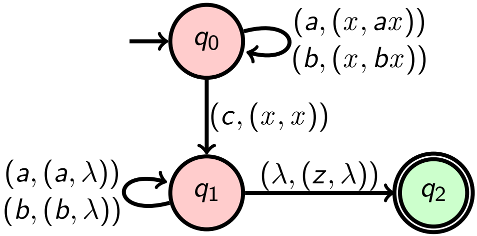
- An NPDA is *deterministic* if there is at most one possible move from any ID.
- Specifically, this means the following:
  - $\text{Card}(\delta(q, a, y)) \leq 1$ for all $(q, a, y) \in Q \times \Sigma^* \cup \{\lambda\} \times \Gamma$.
  - For any $q \in Q$ and $y \in \Gamma$,

    if $\delta(q, \lambda, y) \neq \emptyset$, then $\delta(q, a, y) = \emptyset$ for all $a \in \Sigma$.
- The abbreviation *DPDA* is used for deterministic NPDA.
- A CFL L is *deterministic* if there is a DPDA M with $\mathcal{L}(M) = L$.

Example of a DPDA

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot c \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- The accepter given earlier is also a DPDA.
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1, q_2\}$; $F = \{q_2\}$.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	c	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ

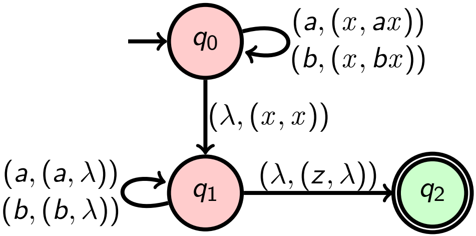


- The symbol x is used as a wildcard to reduce the number of entries.
- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

An Example Which Does Not Admit a DPDA Accepter

- Let $\Sigma = \{a, b, c\}$ and let $L = \{\alpha \cdot \alpha^R \mid \alpha \in \{a, b\}^*\}$.
- For this language, it is not possible to design a DPDA which accepts it..
- $\Gamma = \{a, b, z\}$; $Q = \{q_0, q_1\}$; $F = \{q_2\}$.
- Guessing is essential.

Current			Next	
State	Input	Stack	State	Stack
q_0	a	x	q_0	ax
q_0	b	x	q_0	bx
q_0	λ	x	q_1	x
q_1	a	a	q_1	λ
q_1	b	b	q_1	λ
q_1	λ	z	q_2	λ



- $\mathcal{L}_A(M) = \mathcal{L}_E(M) = \mathcal{L}_{AE}(M) = L$.

Characterization of Deterministic CFLs

- Determinism for a CFL is important in practice, because it means that it may be parsed with a deterministic PDA.

Theorem: Every deterministic CFL is unambiguous, but the converse fails to hold. □

- For a proof, consult an advanced textbook.
- In general, the languages accepted by NPDAs are represented by CFLs.

Question: Is there a similar characterization of the languages accepted by DPDAs?

Answer: Yes, the class of *LR(k) grammars*.

- These grammars are extremely important in practice, and are used in the construction of practical parsers.
- They will be discussed briefly later in a following set of slides.