Chapter 6
Pushdown automata

Outline

- 6.0 Introduction
- 6.1 Definition of PDA
- 6.2 The Language of a PDA
- 6.3 Equivalence of PDA's and CFG's
- 6.4 Deterministic PDA's

6.0 Introduction

- **Basic concepts:**
  - CFL's may be accepted by pushdown automata (PDA's)
  - A PDA is an ε-NFA with a stack.
  - The stack can be read, pushed, and popped only at the top.
  - Two different versions of PDA's ---
    - Accepting strings by "entering an accepting state"
    - Accepting strings by "emptying the stack"

6.0 Introduction

- **Basic concepts (cont'd)**
  - The original PDA is *nondeterministic*.
  - There is also a subclass of PDA's which are *deterministic* in nature.
  - Deterministic PDA's (DPDA's) resembles parsers for CFL's in compilers.

6.0 Introduction

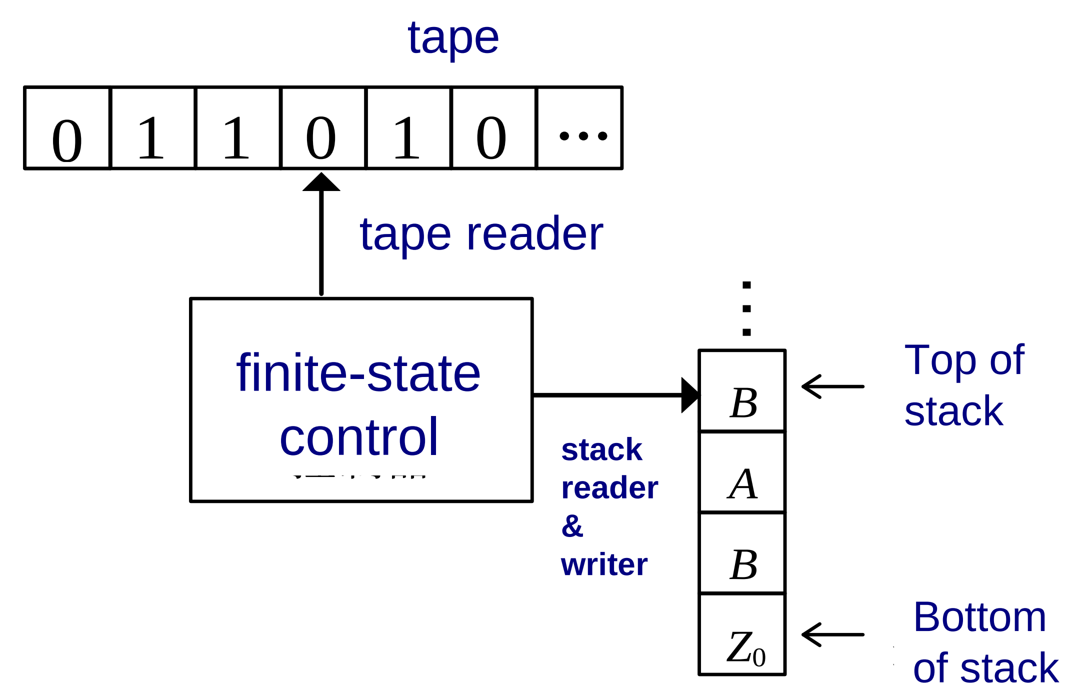
- **Basic concepts (cont'd)**
  - It is interesting to know what "language constructs" which a DPDA can accept.
  - The stack is *infinite* in size, so can be used as a "memory" to eliminate the weakness of "finite states" of NFA's, which cannot accept languages like $L = \{a^n b^n \mid n \geq 1\}$.

6.1 Definition of PDA

- **6.1.1 Informal Introduction**
  - Advantage of the stack --- the stack can "remember" an *infinite* amount of information.
  - Weakness of the stack --- the stack can only be read in a *first-in-last-out* manner.
  - Therefore, it can accept languages like $L_{wwr} = \{ww^R \mid w \text{ is in } (\mathbf{0} + \mathbf{1})^*\}$, but not languages like $L = \{a^n b^n c^n \mid n \geq 1\}$.

6.1 Definition of PDA

- **6.1.1 Informal Introduction**
  - A graphic model of a PDA



A graph model of a PDA

6.1 Definition of PDA

- **6.1.1 Informal Introduction**
  - The input string on the "tape" can only be read.
  - But operations applied to the stack is complicated; we may replace the top symbol by any *string* ---
    - By a single symbol
    - By a string of symbols
    - By the empty string ε which means the top stack symbol is "popped up (eliminated)."

6.1 Definition of PDA

- **6.1.1 Informal Introduction**
  - **Example 6.1** - Design a PDA to accept the language $L_{wwr} = \{ww^R \mid w \text{ is in } (\mathbf{0} + \mathbf{1})^*\}$.
    - In start state q_0, copy input symbols onto the stack.
    - At any time, *nondeterministically* guess **whether** the middle of ww^R is reached and enter q_1, *or* continue copying input symbols.
    - In q_1, compare remaining input symbols with those on the stack one by one.
    - If the stack can be so emptied, then the matching of w with w^R succeeds.

6.1 Definition of PDA

- **6.1.2 Formal Definition**

  A PDA is a 7-tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where

  – Q: a finite set of states
  – Σ: a finite set of input symbols
  – Γ: a finite stack alphabet
  – δ: a transition function such that $\delta(q, a, X)$ **is a set of pairs** (p, γ) where
    - $q \in Q$ (the current state)
    - $a \in \Sigma$ or $a = \varepsilon$ (an input symbol or an empty string)
    - $X \in \Gamma$
    - $p \in Q$ (the next state)

(cont'd in the next page)

6.1 Definition of PDA

- **6.1.2 Formal Definition**

  (continued from last page)

  - $\gamma \in \Gamma^*$ which replaces X on the top of the stack:

    when $\gamma = \varepsilon$, the top stack symbol is popped up

    when $\gamma = X$, the stack is unchanged

    when $\gamma = YZ$, X is replaced by Z, and Y is pushed to the top

    when $\gamma = \alpha Z$, X is replaced by Z and string α is pushed to the top

  - q_0: the start state
  - Z_0: the start symbol of the stack
  - F: the set of accepting or final states

6.1 Definition of PDA

- **6.1.2 Formal Definition**
  - **Example 6.2 (**cont'd from Example 6.1) - Designing a PDA to accept the language L_{ww^R}.
    - Need a start symbol Z of the stack and a 3rd state q_2 as the accepting state.
    - $P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$ such that
      - $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$, $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$

        (initial pushing steps with Z_0 to mark stack bottom)
      - $\delta(q_0, 0, 0) = \{(q_0, 00)\}$, $\delta(q_0, 0, 1) = \{(q_0, 01)\}$, $\delta(q_0, 1, 0) = \{(q_0, 10)\}$, $\delta(q_0, 1, 1) = \{(q_0, 11)\}$

        (continuing pushing)

6.1 Definition of PDA

- **6.1.2 Formal Definition**
  - **Example 6.2** (cont'd)
    - $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$

      (check if input is ε which is in L_{ww^R})
    - $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$, $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$

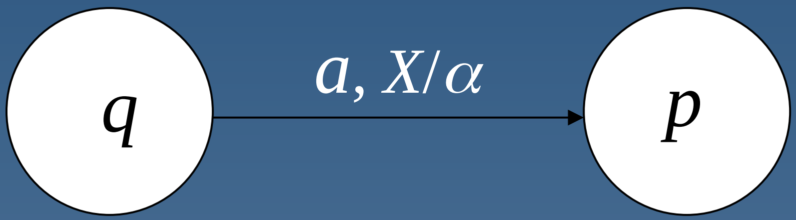
      (check the string's middle)
    - $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$, $\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$

      (matching pairs)
    - $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$

      (entering final state)

6.1 Definition of PDA

- **6.1.3 A Graphic Notation for PDA's**
  - The transition diagram of a PDA is easier to follow.
  - We use "$a, X/\alpha$" on an arc from state p to q to represent that "transition $\delta(q, a, X)$ contains (p, α)"



  - **Example 6.3** The transition diagram of the PDA of Example 6.2 is as shown in Fig. 6.2 (see next page) (in p. 230 of the textbook).

6.1 Definition of PDA

- **6.1.3 A Graphic Notation for PDA's**

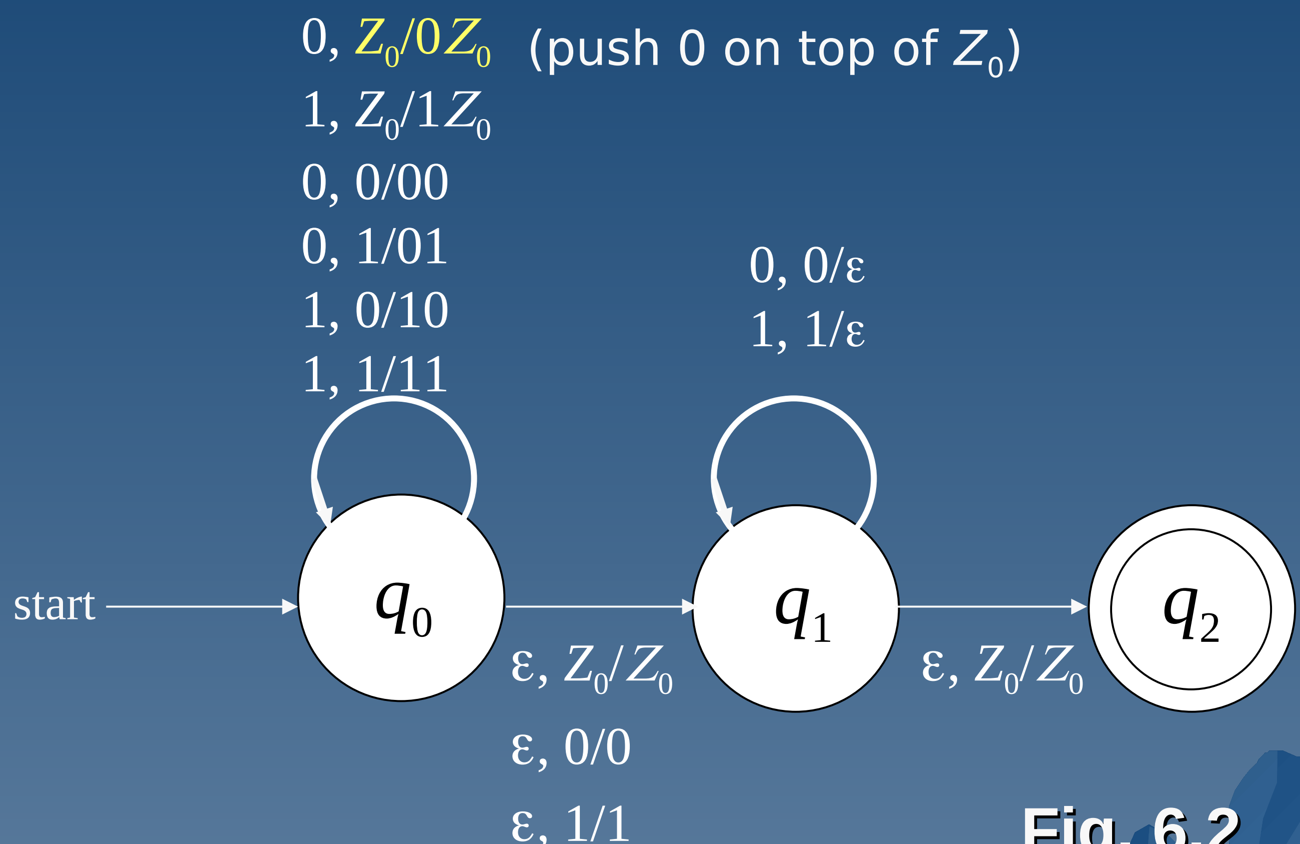


Fig. 6.2

- Where is the nondeterminism?

6.1 Definition of PDA

- **6.1.4 Instantaneous Descriptions of a PDA**
  - The *configuration* of a PDA is represented by a 3-tuple (q, w, γ) where
    - q is the state;
    - w is the remaining input; and
    - γ is the stack content.
  - Such a 3-tuple is called an *instantaneous description* (*ID*) of the PDA.

6.1 Definition of PDA

- **6.1.4 Instantaneous Descriptions of a PDA**
  - The change of an ID into another is called a *move*, denoted by the symbol $\underset{P}{\vdash}$, or $\vdash$ when P is understood.
  - So, if $\delta(q, a, X)$ contains (p, α), then the following is a corresponding move:

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

  - We use $\overset{*}{\underset{P}{\vdash}}$ or $\overset{*}{\vdash}$ to indicate zero or more moves.

6.1 Definition of PDA

- **6.1.4 Instantaneous Descriptions of a PDA**
  - **Example 6.4** (cont'd from Example 6.2) -
    - See Fig. 6.3
    - Moves for the PDA to accept input w = 1111:

$$(q_0, 1111, Z_0) \vdash (q_0, 111, 1Z_0)$$
$$\vdash (q_0, 11, 11Z_0) \quad \vdash (q_1, 11, 11Z_0)$$
$$\vdash (q_1, 1, 1Z_0)$$
$$\vdash (q_1, \varepsilon, Z_0) \quad \vdash (q_2, \varepsilon, Z_0)$$

- There are other paths entering dead ends (not shown).

6.1 Definition of PDA

- **6.1.4 Instantaneous Descriptions of a PDA**
  - **Theorem 6.5**

If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, and

$$(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta),$$

then for any string w in Σ^* and γ in Γ^*, it is also true that

$$(q, xw, \alpha\gamma) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta\gamma).$$

(The reverse is not true; but if γ is taken away, the reverse is true, as shown by the next theorem)

6.1 Definition of PDA

- **6.1.4 Instantaneous Descriptions of a PDA**
  - **Theorem 6.6**

If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, and

$$(q, xw, \alpha) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta),$$

then it is also true that

$$(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta).$$

6.2 The Language of a PDA

- **Some important facts:**
  - Two ways to define languages of PDA's: by final state and by empty stack, as mentioned before.
  - It can be proved that a language L has a PDA that accepts it by final state *if and only if* L has a PDA that accepts it by empty stack.
  - For a given PDA P, the language that P accepts by final state and by empty stack are usually *different*.
  - In this section, we show *conversions* between the two ways of language acceptances.

6.2 The Language of a PDA

- **6.2.1 Acceptance by Final State**
  - **Definition:**

    If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA. Then $L(P)$, the language accepted by *P by final state*, is

    $$\{w \mid (q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \alpha), q \in F\}$$

    for any α.
  - **Example 6.7** - Proving the PDA shown in Example 6.2 indeed accepts the language L_{ww^R} (see the detail in the textbook by yourself).

6.2 The Language of a PDA

- **6.2.2 Acceptance by Empty Stack**
  - **Definition:**

    If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA. Then $N(P)$, the language accepted by P *by empty stack*, is

    $$\{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)\}$$

    for any q.
  - The set of final states F may be dropped to form a 6-tuple, instead of a 7-tuple, for a PDA.

6.2 The Language of a PDA

- **6.2.2 Acceptance by Empty Stack**
  - **Example 6.8**

    The PDA of Example 6.2 may be modified to accept L_{ww^R} by empty stack:

    simply change the original transition

    $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$

    to be

    $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$.

    (just eliminate Z_0)

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Theorem 6.9 (1/3)**

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F = such that $L = L(P_F)$.

Proof. The idea is to use Fig. 6.4 below.

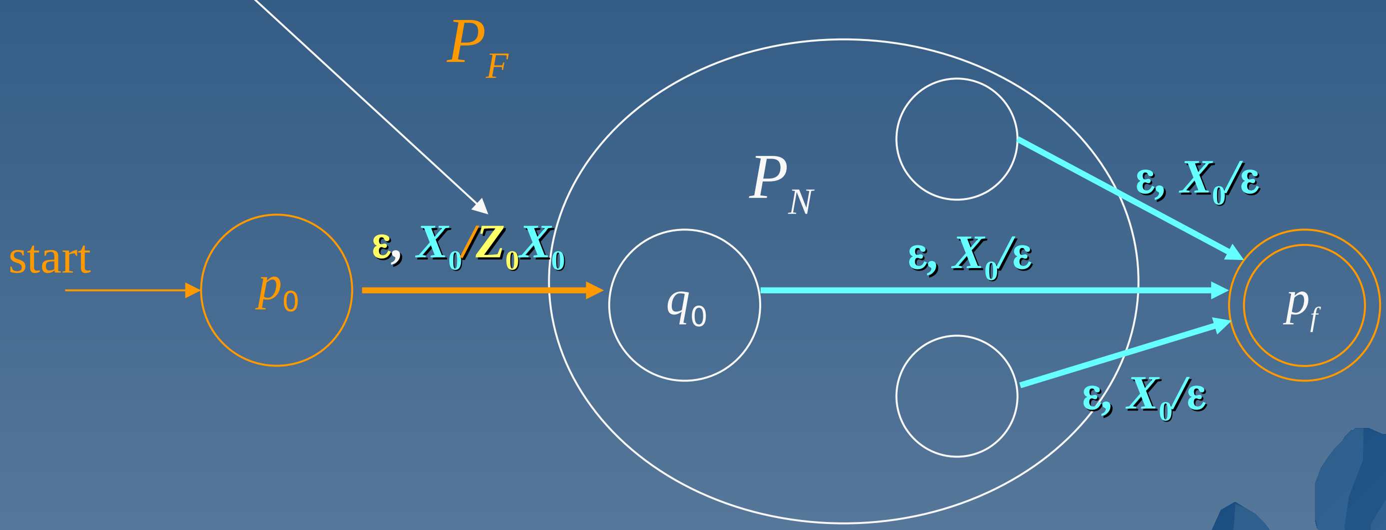


Fig. 6.4 P_F simulating P_N and accepts if P_N empties its stack

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Theorem 6.9 (2/3)**

    If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F = such that $L = L(P_F)$.

    Proof. (cont'd)

    Define $P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$ where δ_F is such that

    - $\delta_F(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$. (用 X_0 墊底).
    - For all $q \in Q$, $a \in \Sigma$ or $a = \varepsilon$, and $Y \in \Gamma$, $\delta_F(q, a, Y)$ contains all the pairs in $\delta_N(q, a, Y)$.
    - $\delta_F(q, \varepsilon, X_0)$ contains (p_f, ε) for every state q in Q.

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Theorem 6.9 (3/3)**

If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F = such that $L = L(P_F)$.

Proof. (cont'd)

- It can be proved that W is in $L(P_F)$ if and only if w is in $N(P_N)$ (see the textbook).

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Example 6.10 -** Design a PDA which accepts the if/else errors by empty stack.
    - Let *i* represents if; *e* represents else.
    - The PDA is designed in such a way that

      if the number of *else* (#*else*) > the number of *if* (#*if*),

      then the stack will be emptied.

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Example 6.10 (cont'd)**
    - A PDA *by empty stack* for this is as follows:

$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z)$$

when an "*if*" is seen, push a "*Z*";

when an "*else*" is seen, pop a "*Z*";

when (#*else*) > (#*if* + 1), the stack is emptied and the input sting is accepted.

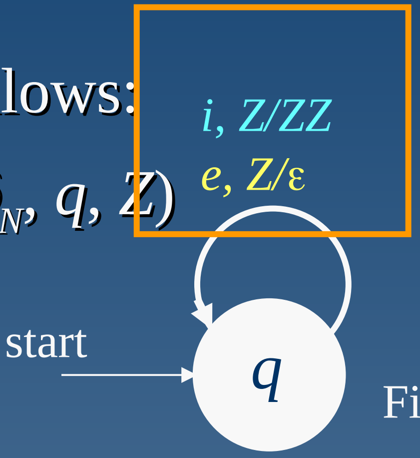


Fig. 6.5

- For example, for input string $w = iee$, the moves are:

$$(q, iee, Z) \vdash (q, ee, ZZ) \vdash (q, e, Z) \vdash (q, \varepsilon, \varepsilon) \text{ accept !}$$

(how about $w = eei$?)

6.2 The Language of a PDA

- **6.2.3 From Empty Stack to Final State**
  - **Example 6.10 (cont'd)**
    - A PDA *by final state* as follows:

$$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\})$$

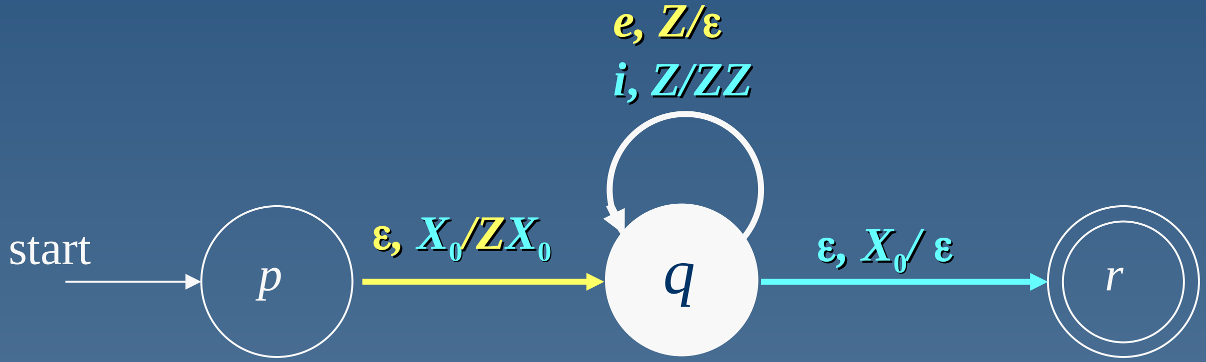


Fig. 6.6

- For input $w = iee$, the moves are:

$(p, iee, X_0) \vdash (q, iee, ZX_0) \vdash (q, ee, ZZX_0) \vdash (q, e, ZX_0) \vdash (q, \varepsilon, X_0) \vdash (r, \varepsilon, \varepsilon)$ accept !

6.2 The Language of a PDA

- **6.2.4 From Final State to Empty Stack**
  - **Theorem 6.11**

Let L be $L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Then there is a PDA P_N such that $L = N(P_N)$.

Proof. The idea is to use Fig. 6.7 below (in final states of P_F, pop up the remaining symbols in the stack).

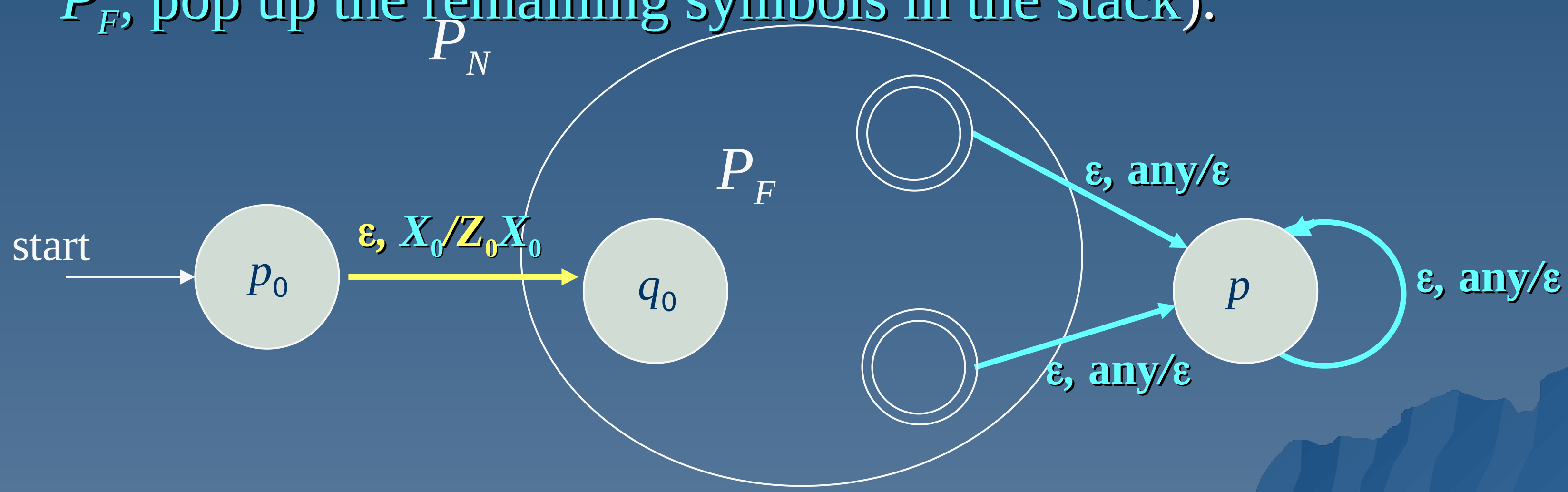


Fig. 6.7 P_N simulating P_F and empties its stack when and only when P_N enters an accepting state.

6.3 Equivalence of PDA's and CFG's

- **Equivalences to be proved:**
  1) CFL's defined by CFG's
  2) Languages accepted by final state by some PDA
  3) Languages accepted by empty stack by some PDA

  – Equivalence of 2) and 3) above have been proved.

6.3 Equivalence of PDA's and CFG's

- **6.3.1 From Grammars to PDA's**
  - Given a CFG $G = (V, T, Q, S)$, construct a PDA P that accepts $L(G)$ by empty stack in the following way:
  - $P = (\{q\}, T, V \cup T, \delta, q, S)$ where the transition function δ is defined by:
    - for each nonterminal A,

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is a production of } G\};$$

    - for each terminal a, $\delta(q, a, a) = \{(q, \varepsilon)\}$.

6.3 Equivalence of PDA's and CFG's

- **6.3.1 From Grammars to PDA's**
  - **Theorem 6.13**

    If PDA P is constructed from CFG G by the construction above, then $N(P) = L(G)$.

    Proof. See the textbook.

6.3 Equivalence of PDA's and CFG's

- **6.3.1 From Grammars to PDA's**
  - **Example 6.12** - Construct a PDA from the expression grammar of Fig. 5.2:

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1;$$
$$E \rightarrow I \mid E*E \mid E+E \mid (E).$$

The transition function for the PDA is as follows:

a) $\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$

b) $\delta(q, \varepsilon, E) = \{(q, \mathrm{I}), (q, E+E), (q, E*E), (q, (E))\}$

c) $\delta(q, d, d) = \{(q, \varepsilon)\}$ where d may any of the terminals a, b, 0, 1, (,), +, *.

6.3 Equivalence of PDA's and CFG's

- **6.3.2 From PDA's to Grammars**
  - **Theorem 6.14**

    Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ be a PDA. Then there is a context-free grammar G such that $L(G) = N(P)$.

  Proof. Construct $G = (V, T, P, S)$ where the set of nonterminals consists of:

    - the special symbol S as the start symbol;
    - all symbols of the form $[pXq]$ where p and q are states in Q and X is a stack symbol in Γ.

6.3 Equivalence of PDA's and CFG's

- **6.3.2 From PDA's to Grammars**
  - **Theorem 6.14**

Proof. (cont'd)

The productions of G are as follows.

(a) For all states p, G has the production $S \rightarrow [q_0 Z_0 p]$.

(b) Let $\delta(q, a, X)$ contain the pair $(r, Y_1 Y_2 \ldots Y_k)$, where

- a is either a symbol in Σ or $a = \varepsilon$;
- k can be any number, including 0, in which case the pair is (r, ε).

Then for all lists of states $r_1, r_2, \ldots, r_k$, G has the production

$$[qXr_k] \rightarrow a[rY_1r_1][r_1Y_2r_2]\ldots[r_{k-1}Y_kr_k].$$

6.3 Equivalence of PDA's and CFG's

- **6.3.2 From PDA's to Grammars**
  - **Example 6.15 ---** Convert the PDA of Example 6.10 (below) to a grammar.

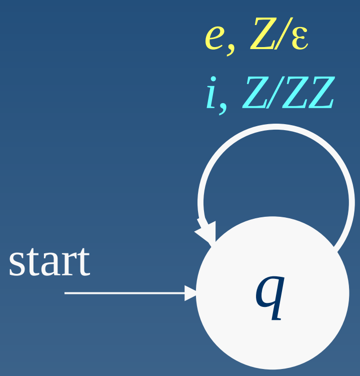


Fig. 6.5

Nonterminals include only two symbols, S and $[qZq]$.

Productions:

1. $S \rightarrow [qZq]$ (for the start symbol S);

2. $[qZq] \rightarrow i[qZq][qZq]$ (from $(q, ZZ) \in \delta_N(q, i, Z)$)

3. $[qZq] \rightarrow e$ (from $(q, \varepsilon) \in \delta_N(q, e, Z)$)

6.3 Equivalence of PDA's and CFG's

- **6.3.2 From PDA's to Grammars**
  - **Example 6.15 ---** (cont'd)

    If we replace $[qZq]$ by a simple symbol A, then the productions become

    $$1.\ S \rightarrow A$$
    $$2.\ A \rightarrow iAA$$
    $$3.\ A \rightarrow e$$

    Obviously, these productions can be simplified to be

    $$1.\ S \rightarrow iSS$$
    $$2.\ S \rightarrow e$$

    And the grammar may be written simply as

    $$G = (\{S\}, \{i, e\}, \{S \rightarrow iSS \mid e\}, S)$$

6.4 Deterministic PDA's

- **6.4.1 Definition of a Deterministic PDA**
  - Intuitively, a PDA is deterministic if there is never a choice of moves (including *ε-moves*) in any situation.
  - Formally, a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is said to be deterministic (a DPDA) if and only if the following two conditions are met:
    - $\delta(q, a, X)$ has at most one element for any $q \in Q$, $a \in \Sigma$ or $a = \varepsilon$, and $X \in \Gamma$. ("一定要有")
    - If $\delta(q, a, X)$ is nonempty for some $a \in \Sigma$, then $\delta(q, \varepsilon, X)$ must be empty. ("不能多於一個")

6.4 Deterministic PDA's

- **6.4.1 Definition of a DPDA**
  - **Example 6.16 –**
    - There is no DPDA for L_{ww^R} of Example 6.2.
    - But there is a DPDA for a modified version of L_{ww^R} as follows, which is not an RL (proved later):

$$L_{wcw^R} = \{wcw^R \mid w \in L((\mathbf{0} + \mathbf{1})^*)\}.$$

    - To recognize wcw^R, just store 0's & 1's in stack until center marker c is seen. Then, match the remaining input w^R with the stack content (w).
    - The PDA can so be designed to be deterministic by searching the center marker *without trying matching all the time nondeterministically* .

6.4 Deterministic PDA's

- **6.4.1 Definition of a DPDA**
  - **Example 6.16** (cont'd) A desired DPDA is as follows. (The difference is just the blue c.)

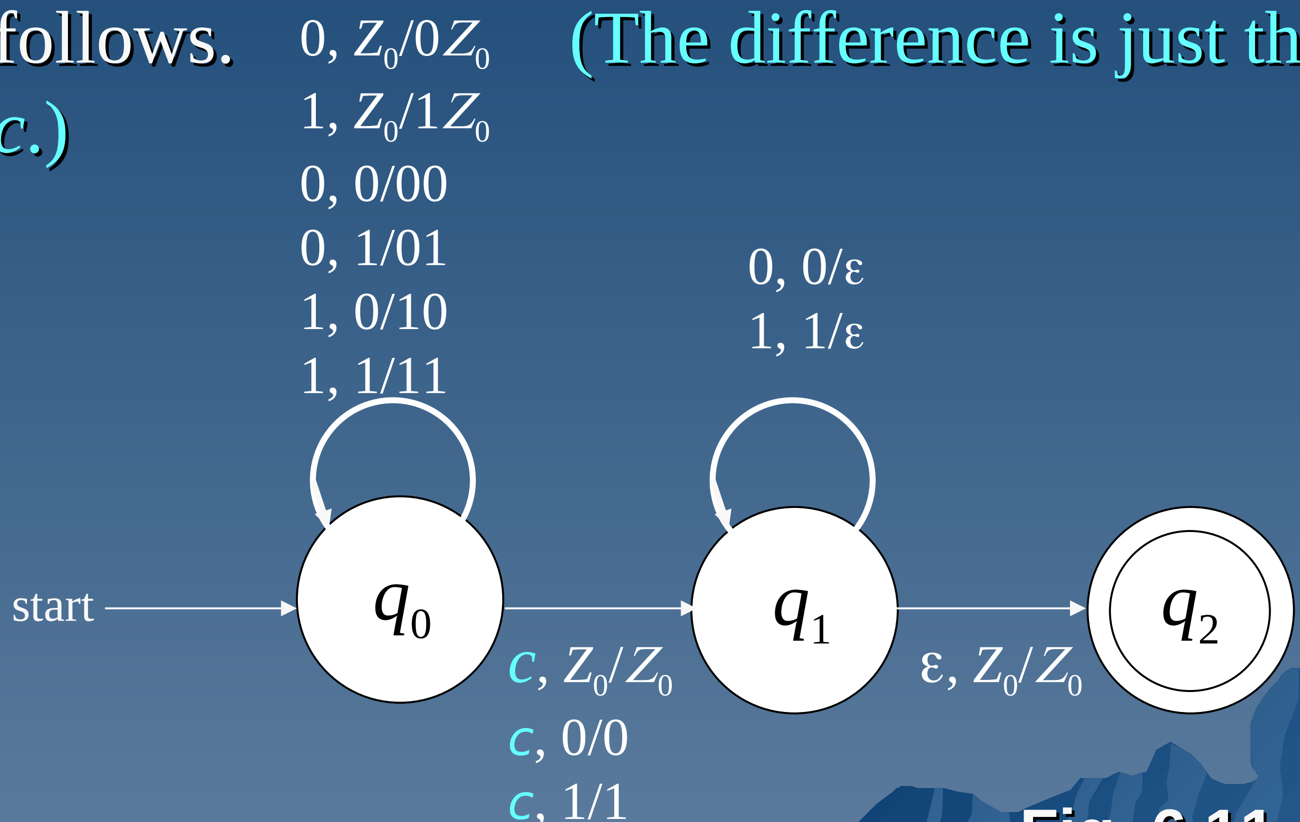


Fig. 6.11

6.4 Deterministic PDA's

- **6.4.2 Regular Languages and DPDA's**
  - The DPDA's accepts a class of languages that is *between* the RL's and the CFL's, as proved in the following.
  - **Theorem 6.17**

    If L is an RL, then $L = L(P)$ for some DPDA P (accepting by final state).

    Proof. Easy. Just use a DPDA to simulate a DFA as follows.

    If DFA $A = (Q, \Sigma, \delta_A, q_0, F)$ accepts L, then construct DPDA $P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$ where δ_P is such that $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$ for all states p and q in Q such that $\delta_A(q, a) = p$.

6.4 Deterministic PDA's

- **6.4.2 Regular Languages and DPDA's**
  - The language-recognizing capability of the *DPDA by empty stack* is *rather limited*.
  - **Theorem 6.19**

    A language L is $N(P)$ for some DPDA P if and only if L has the prefix property and L is $L(P')$ for some DPDA P' (for proof, do exercise 6.4.3).
  - A language L is said to have the prefix property if there are no two different strings x and y in L such that x is a prefix of y.

    (For examples of such languages, see Example 6.18)

6.4 Deterministic PDA's

- **6.4.3 DPDA's and CFL's**
  - DPDA's can be used to accept non-RL's, for example, L_{wcw^R} mentioned before.
    - It can be proved by the pumping lemma that L_{wcw^R} *is not an RL* (see the textbook, pp. 254~255).
  - On the other hand, DPDA's *by final state* cannot accept certain CFL's, for example, L_{ww^R}.
    - It can be proved that L_{ww^R} cannot be accepted by a DPDA by final state (see an informal proof in the textbook, p. 255).

6.4 Deterministic PDA's

- **6.4.3 DPDA's and CFL's**
  - *Conclusion*:

    The languages accepted by DPDA's by final state properly include RL's, but are properly included in CFL's.

6.4 Deterministic PDA's

- **6.4.4 DPDA's and Ambiguous Grammars**
  - **Theorem 6.20**

    If $L = N(P)$ (*accepting by empty stack*) for some DPDA P, then L has an unambiguous CFG.

    Proof. See the textbook.
  - **Theorem 6.21**

    If $L = L(P)$ for some DPDA P (*accepting by final state*), then L has an unambiguous CFG.

    Proof. See the textbook.