بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ــ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۰۱)

نظریهٔ زبان‌ها و ماشین‌ها

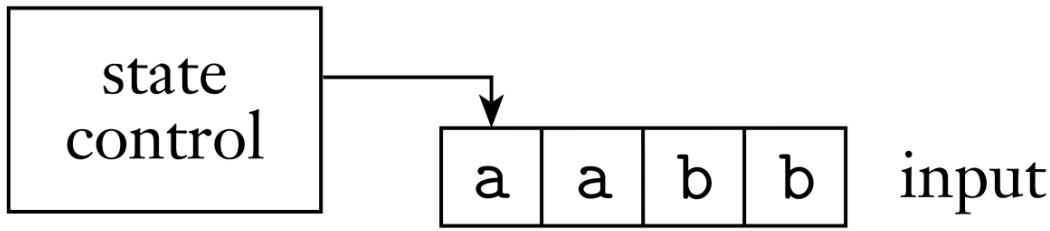
حسین فلسفین

Pushdown Automaton (PDA)

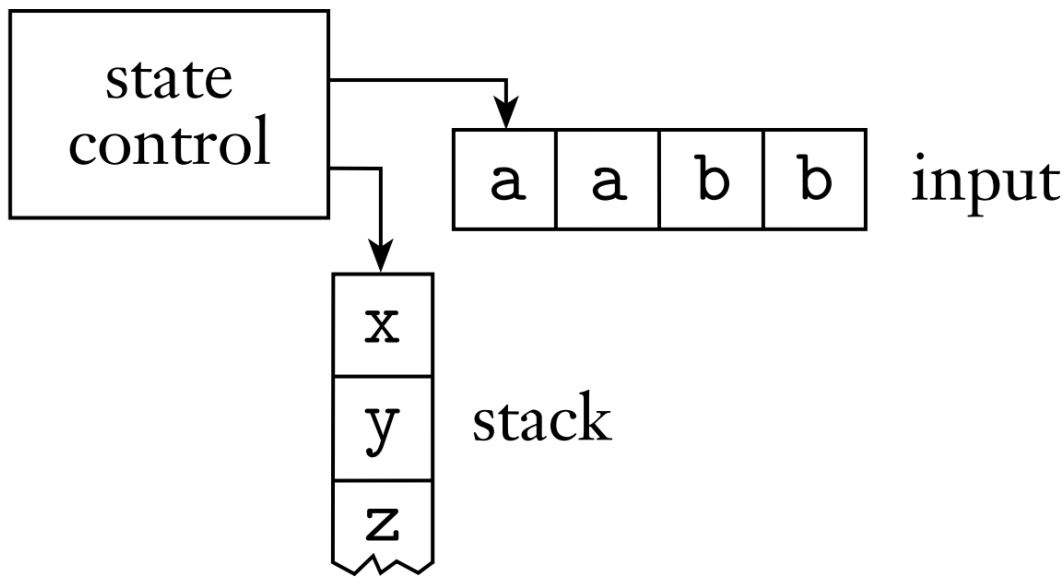
In this session we introduce a new type of computational model called pushdown automata. These automata are like NFAs but have an extra component called a stack. The stack provides additional memory beyond the finite amount available in the control.

Pushdown automata are equivalent in power to context-free grammars. This equivalence is useful because it gives us two options for proving that a language is context free. We can give either a context-free grammar generating it or a pushdown automaton recognizing it. Certain languages are more easily described in terms of generators, whereas others are more easily described by recognizers.

Schematic of a finite automaton:



Schematic of a pushdown automaton:



Last-in-first-out (LIFO) queue

A pushdown automaton (PDA) can write symbols on the stack and read them back later. Writing a symbol "pushes down" all the other symbols on the stack. At any time the symbol on the top of the stack can be read and removed. The remaining symbols then move back up. Writing a symbol on the stack is often referred to as pushing the symbol, and removing a symbol is referred to as popping it. Note that all access to the stack, for both reading and writing, may be done only at the top. In other words a stack is a "last in, first out" storage device. If certain information is written on the stack and additional information is written afterward, the earlier information becomes inaccessible until the later information is removed.

استک نامحدود است:

A stack is valuable because it can hold an unlimited amount of information. Recall that a finite automaton is unable to recognize the language $\{0^n 1^n | n \geq 0\}$ because it cannot store very large numbers in its finite memory. A PDA is able to recognize this language because it can use its stack to store the number of 0s it has seen. Thus the unlimited nature of a stack allows the PDA to store numbers of unbounded size.

PDAهای قطعی و غیرقطعی در قدرت توصیف با یکدیگر تمایز دارند

As mentioned earlier, pushdown automata may be nondeterministic. Deterministic and nondeterministic pushdown automata are not equivalent in power. Nondeterministic pushdown automata recognize certain languages that no deterministic pushdown automata can recognize.

Recall that deterministic and nondeterministic finite automata do recognize the same class of languages, so the pushdown automata situation is different. We focus on nondeterministic pushdown automata because these automata are equivalent in power to context-free grammars.

پس تمرکز ما روی NPDAها خواهد بود، نه DPDAها. نیز، منظور ما از یک PDA در حقیقت NPDA است.

الفبای استک، ممکن است متمایز از الفبای ورودی باشد:

The stack is a device containing symbols drawn from some alphabet. The machine may use different alphabets for its input and its stack, so now we specify both an input alphabet Σ and a stack alphabet Γ.

نمادهای کتاب سیپسر: $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$ و $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

☞ *The domain of the transition function is* $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon$. *Thus the current state, next input symbol read, and top symbol of the stack determine the next move of a pushdown automaton. Either symbol may be* ε, *causing the machine to move without reading a symbol from the input or without reading a symbol from the stack.*

☞ *For the range of the transition function we need to consider what to allow the automaton to do when it is in a particular situation. It may enter some new state and possibly write a symbol on the top of the stack. The function* δ *can indicate this action by returning a member of* Q *together with a member of* Γ_ε, *that is, a member of* $Q \times \Gamma_\varepsilon$. *Because we allow nondeterminism in this model, a situation may have several legal next moves. The transition function incorporates nondeterminism in the usual way, by returning a set of members of* $Q \times \Gamma_\varepsilon$, *that is, a member of* $\mathcal{P}(Q \times \Gamma_\varepsilon)$. *Putting it all together, our transition function* δ *takes the form* $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \mapsto \mathcal{P}(Q \times \Gamma_\varepsilon)$.

تعریف رسمی یک PDA

Definition: *A pushdown automaton is a* 6*-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, F)$*, where* Q*,* Σ*,* Γ*, and* F *are all finite sets, and*

1. Q *is the set of states,*
2. Σ *is the input alphabet,*
3. Γ *is the stack alphabet,*
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \mapsto \mathcal{P}(Q \times \Gamma_\varepsilon)$ *is the transition function,*
5. $q_0 \in Q$ *is the start state, and*
6. $F \subseteq Q$ *is the set of accept states.*

تعریف رسمی مفهوم پذیرش توسط یک PDA

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows. It accepts input w if w can be written as $w = w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions. The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

1. **$r_0 = q_0$ and $s_0 = \varepsilon$. This condition signifies that M starts out properly, in the start state and with an empty stack.**
2. **For $i = 0, \ldots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$. This condition states that M moves properly according to the state, stack, and next input symbol.**
3. **$r_m \in F$. This condition states that an accept state occurs at the input end.**

توصیف وضعیت یک PDA حین پردازش یک رشته

We introduce a succinct notation for describing the successive configurations of an NPDA during the processing of a string. The relevant factors at any time are the current state of the control unit, the unread part of the input string (the portion of the input string that has not yet been read), and the current contents of the stack. The triplet (q, w, u), where q is the state of the control unit, w is the unread part of the input string, and u is the stack contents (with the leftmost symbol indicating the top of the stack), is called an instantaneous description of an NPDA. (The stack contents will be represented by a string of stack symbols, and the leftmost symbol is assumed to be the one on top.) In fact, a configuration of the PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a triple (q, x, α) where $q \in Q$, $x \in \Sigma^*$, and $\alpha \in \Gamma^*$.

حرکت از یک کانفیگ به کانفیگ دیگر

We write $(p, x, \alpha) \vdash_M (q, y, \beta)$ ***to mean that one of the possible moves in the first configuration takes*** M ***to the second. This can happen in two ways, depending on whether the move reads an input symbol or is a*** ε***-transition. In the first case,*** $x = \sigma y$ ***for some*** $\sigma \in \Sigma$***, and in the second case*** $x = y$***. We can summarize both cases by saying that*** $x = \sigma y$ ***for some*** $\sigma \in \Sigma_\varepsilon$***. If*** $\alpha = X\gamma$ ***for some*** $X \in \Gamma_\varepsilon$ ***and some*** $\gamma \in \Gamma^*$***, then*** $\beta = \xi\gamma$ ***for some string*** $\xi \in \Gamma_\varepsilon$ ***for which*** $(q, \xi) \in \delta(p, \sigma, X)$***.***

بیان دیگر:

A move from one instantaneous description to another will be denoted by the symbol $\vdash$***; thus*** $(q_1, aw, bx) \vdash (q_2, w, yx)$ ***is possible if and only if*** $(q_2, y) \in \delta(q_1, a, b)$***.***

معرفی چند نماد

More generally, we write $(p, x, \alpha) \vdash_M^n (q, y, \beta)$ *if there is a sequence of* n *moves taking from* M *from the first configuration to the second, and* $(p, x, \alpha) \vdash_M^* (q, y, \beta)$ *if there is a sequence of zero or more moves taking* M *from the first configuration to the second. In the three notations* $\vdash_M$, $\vdash_M^n$, *and* $\vdash_M^*$, *if there is no confusion we usually omit the subscript* M. *(On occasions where several automata are under consideration we will use* $\vdash_M$ *to emphasize that the move is made by the particular automaton* M. *)*

Moves involving an arbitrary number of steps will be denoted by $\vdash^*$. *The expression* $(q_1, w_1, x_1) \vdash^* (q_2, w_2, x_2)$ *indicates a possible configuration change over a number of steps.*

نحوهٔ دیگر بیان مفهوم پذیرش توسط یک PDA

If $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ and $x \in \Sigma^*$, the string x is accepted by M if

$$(q_0, x, \varepsilon) \vdash_M^* (q, \varepsilon, \alpha)$$

for some $\alpha \in \Gamma^*$ and some $q \in F$. A language $L \subseteq \Sigma^*$ is said to be accepted by M if L is precisely the set of strings accepted by M; in this case, we write $L = L(M)$. (In order to be accepted, a string must have been read in its entirety by the PDA.)

Examples of Pushdown Automata

Example 1: ***The following is the formal description of an NPDA that recognizes the language*** $\{0^n 1^n | n \geq 0\}$. ***Let*** M_1 ***be*** $(Q, \Sigma, \Gamma, \delta, q_1, F)$, ***where*** $Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, \$\}$, $F = \{q_1, q_4\}$, ***and*** δ ***is given by the following table, wherein blank entries signify*** $\varnothing$.

Input:	0			1			ε		
Stack:	0	$	ε	0	$	ε	0	$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

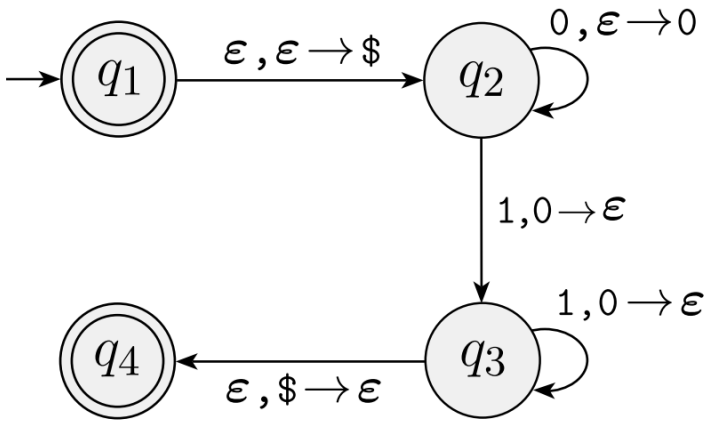
State Diagram

We can also use a state diagram to describe a PDA. Such diagrams are similar to the state diagrams used to describe finite automata, modified to show how the PDA uses its stack when going from state to state. We write "$a, b \to c$" to signify that when the machine is reading an a from the input, it may replace the symbol b on the top of the stack with a c. Any of a, b, and c may be ε.

☞ If a is ε, the machine may make this transition without reading any symbol from the input.

☞ If b is ε, the machine may make this transition without reading and popping any symbol from the stack.

☞ If c is ε, the machine does not write any symbol on the stack when going along this transition.

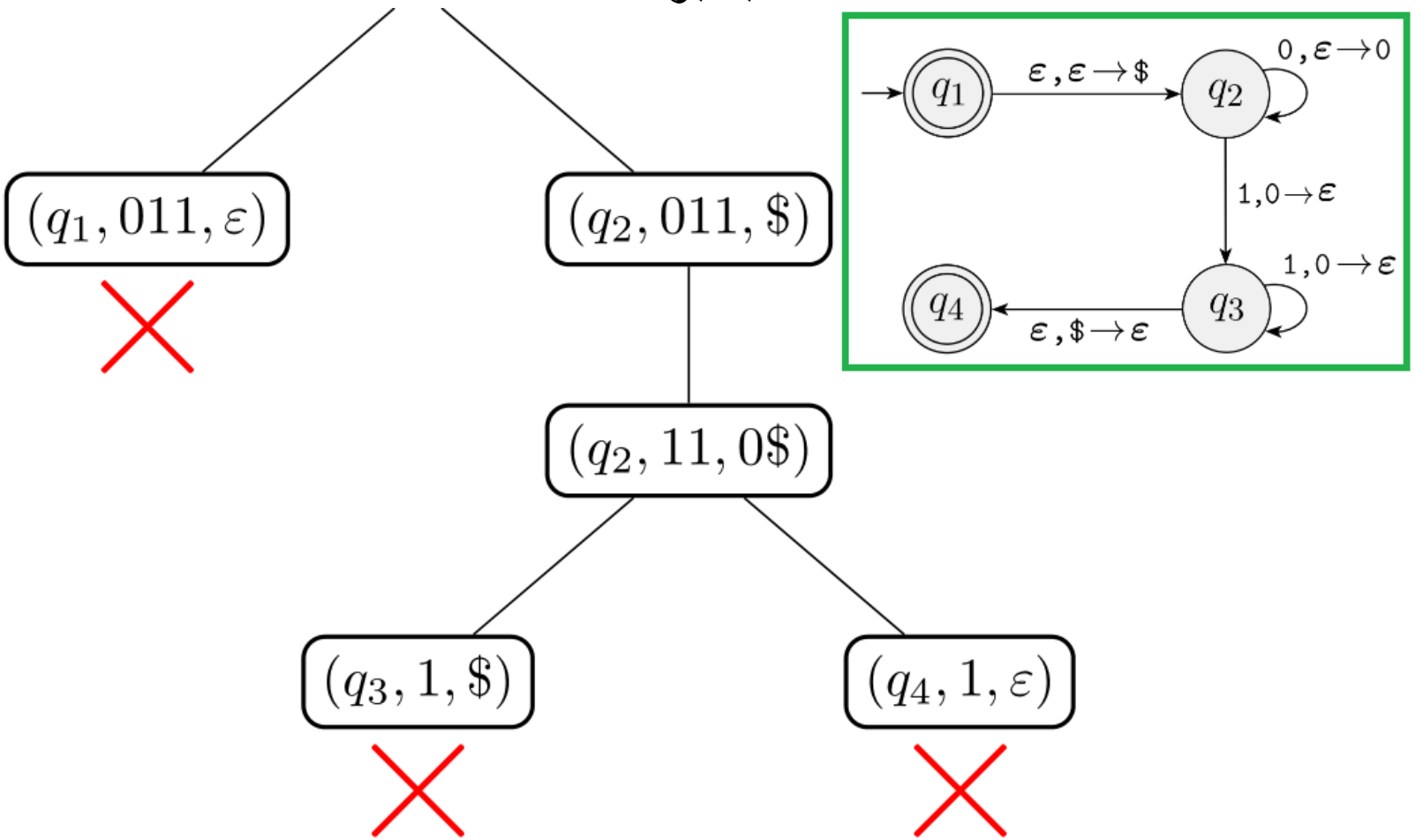


The formal definition of a PDA contains no explicit mechanism to allow the PDA to test for an empty stack. This PDA is able to get the same effect by initially placing a special symbol $ on the stack. Then if it ever sees the $ again, it knows that the stack effectively is empty.

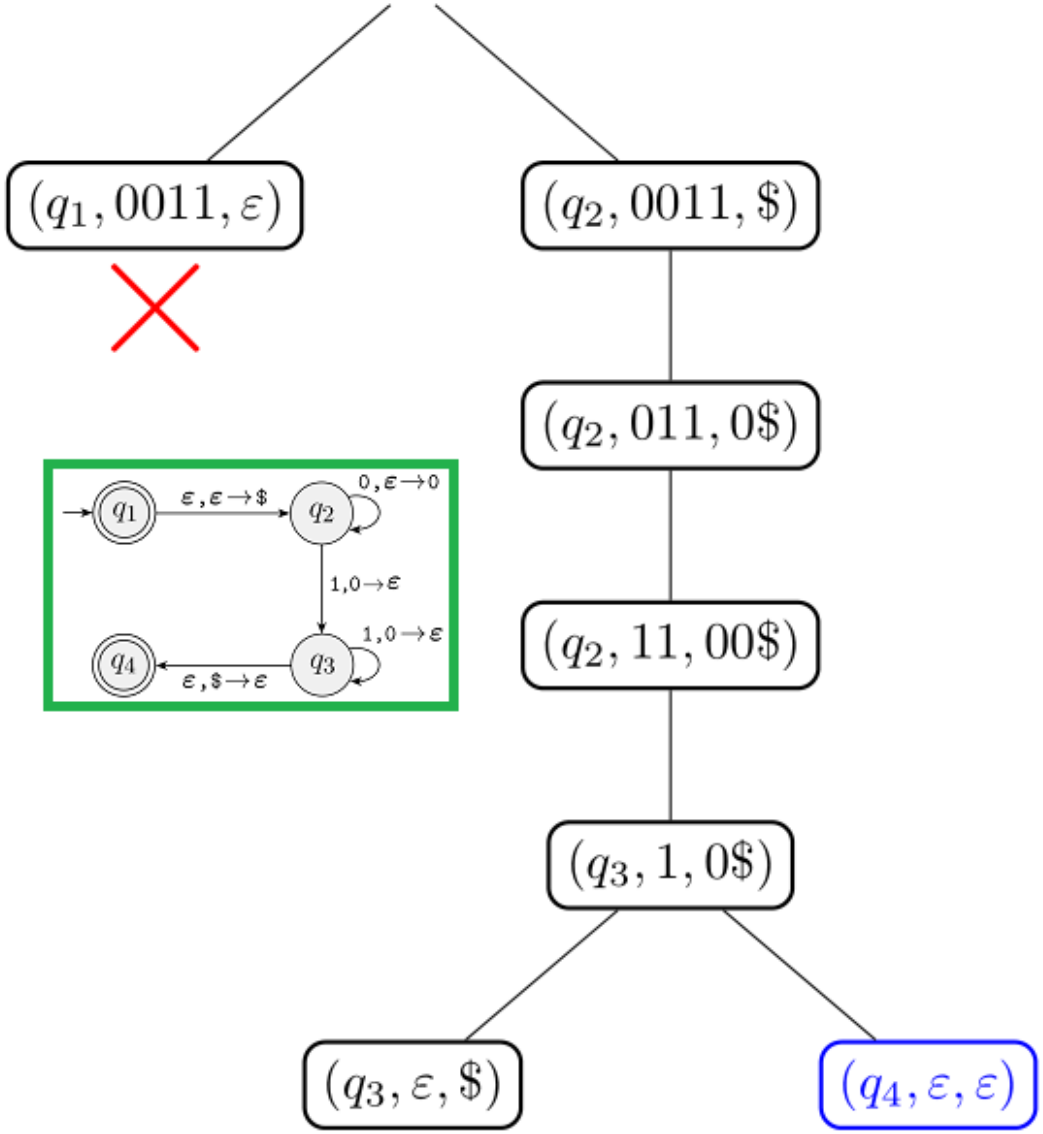
$$(q_1, 0011, \varepsilon) \vdash (q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$) \vdash$$
$$(q_3, 1, 0\$) \vdash (q_3, \varepsilon, \$) \vdash (q_4, \varepsilon, \varepsilon)$$

We can draw a computation tree for an NPDA, showing the configuration and choice of moves at each step.

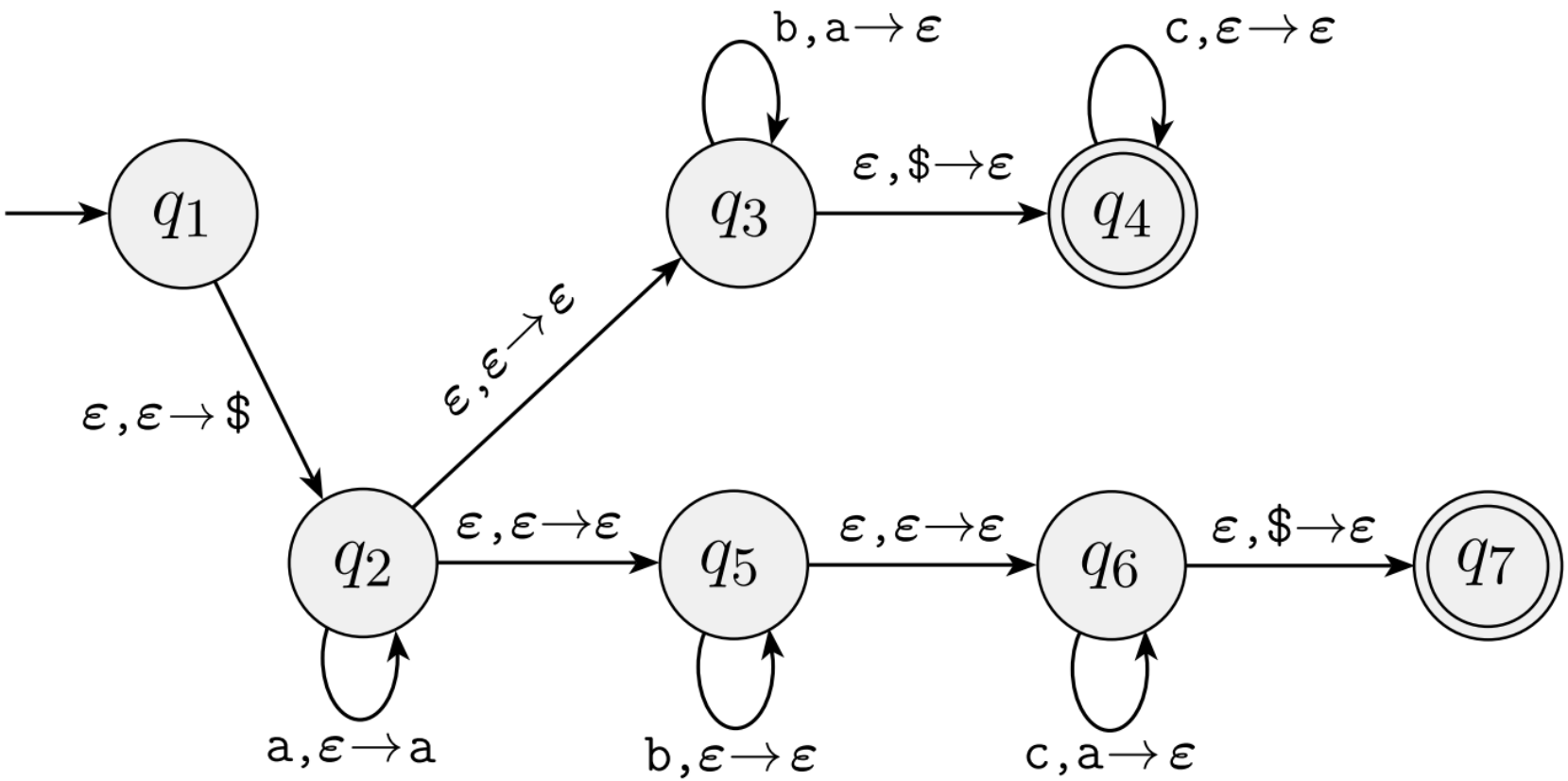
درخت محاسبه برای رشتهٔ 011:



درخت محاسبه برای رشتهٔ 0011:



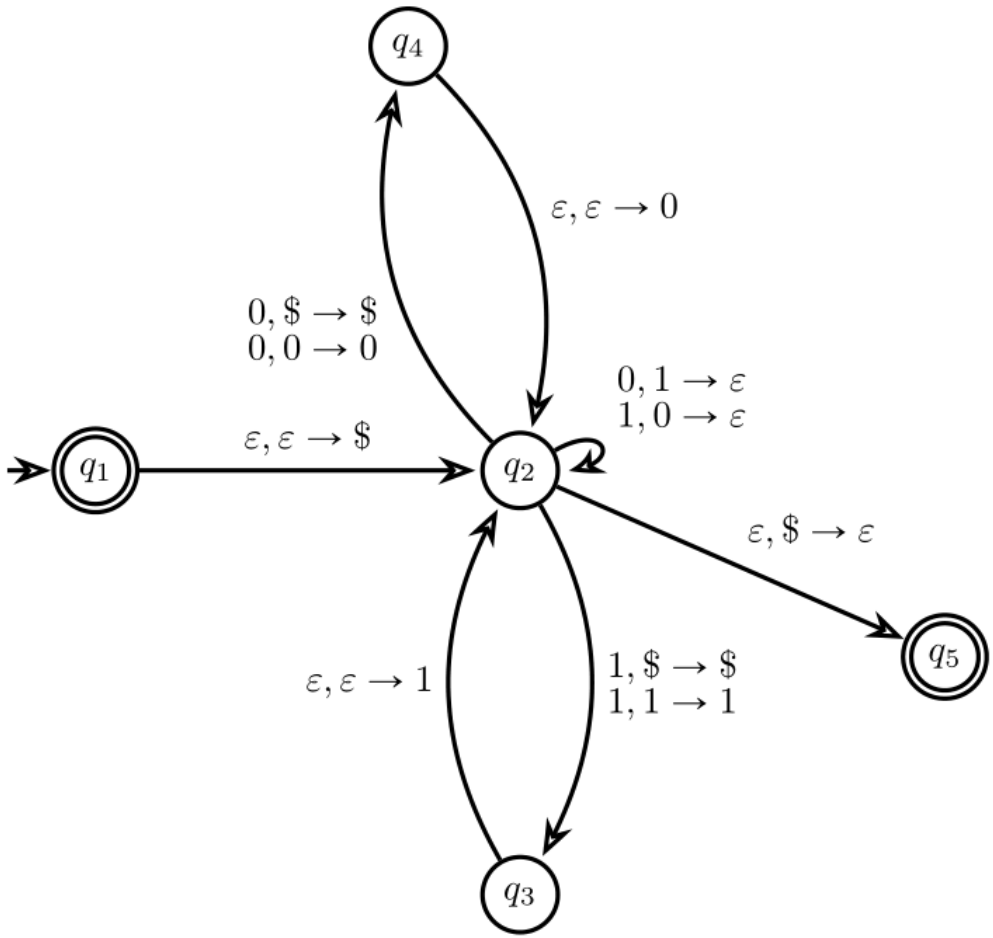
Example 2: ***This example illustrates a pushdown automaton that recognizes the language*** $\{a^i b^j c^k | i, j, k \geq 0$ ***and*** $i = j$ ***or*** $i = k\}$**.**



Problem 2.57 asks you to show that nondeterminism is essential for recognizing this language with a PDA.

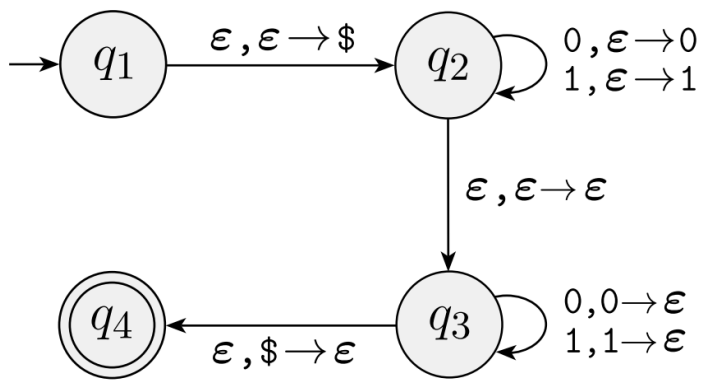
Example 3: ***Consider the language consisting of strings that have equal numbers of 0's and 1's, which is expressed as***

$$\{w \in \{0,1\}^* | n_0(w) = n_1(w)\}.$$

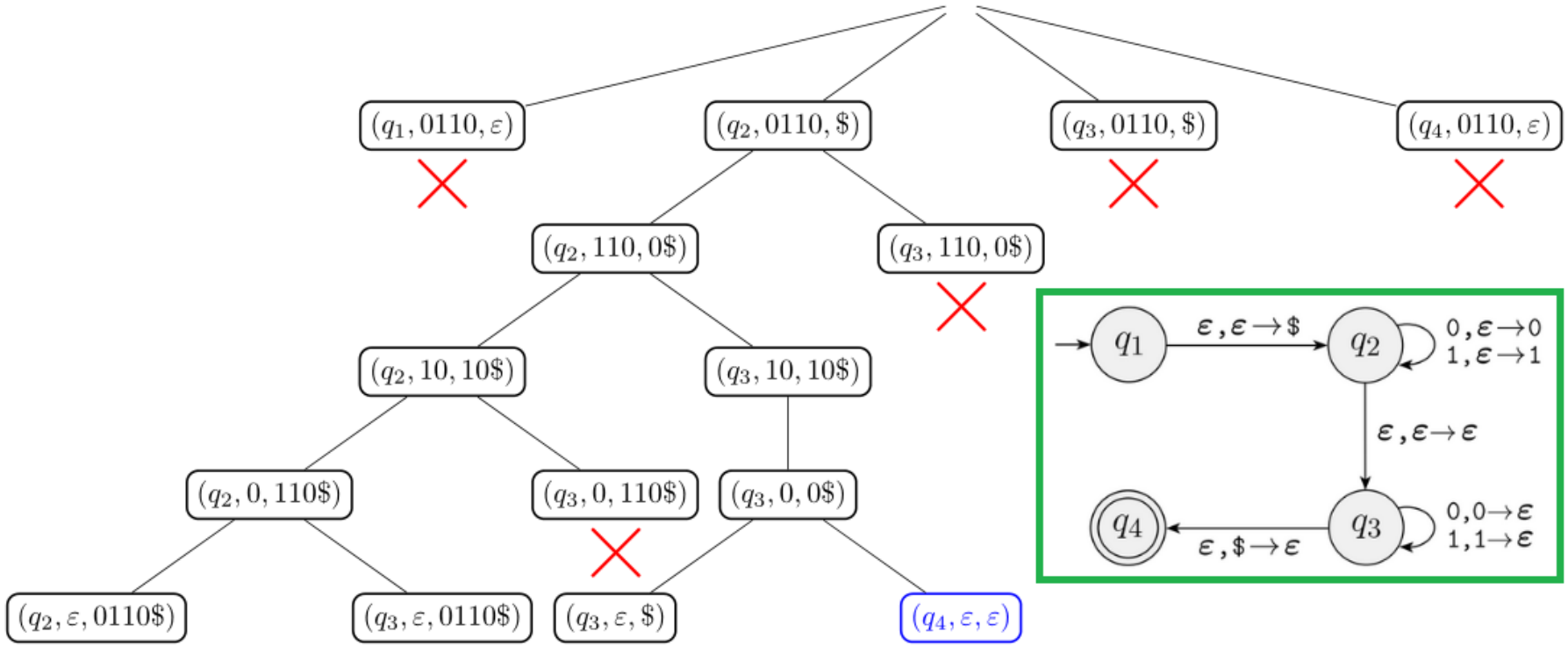


Example 4: *In this example we give a PDA M_3 recognizing the language $\{ww^R | w \in \{0,1\}^*\}$.*

The informal description and state diagram of the PDA follow. Begin by pushing the symbols that are read onto the stack. At each point, nondeterministically guess that the middle of the string has been reached and then change into popping off the stack for each symbol read, checking to see that they are the same. If they were always the same symbol and the stack empties at the same time as the input is finished, accept; otherwise reject. Problem 2.58 shows that this language requires a nondeterministic PDA.



درخت محاسبه برای رشتهٔ 0110:



Some Brief Remarks

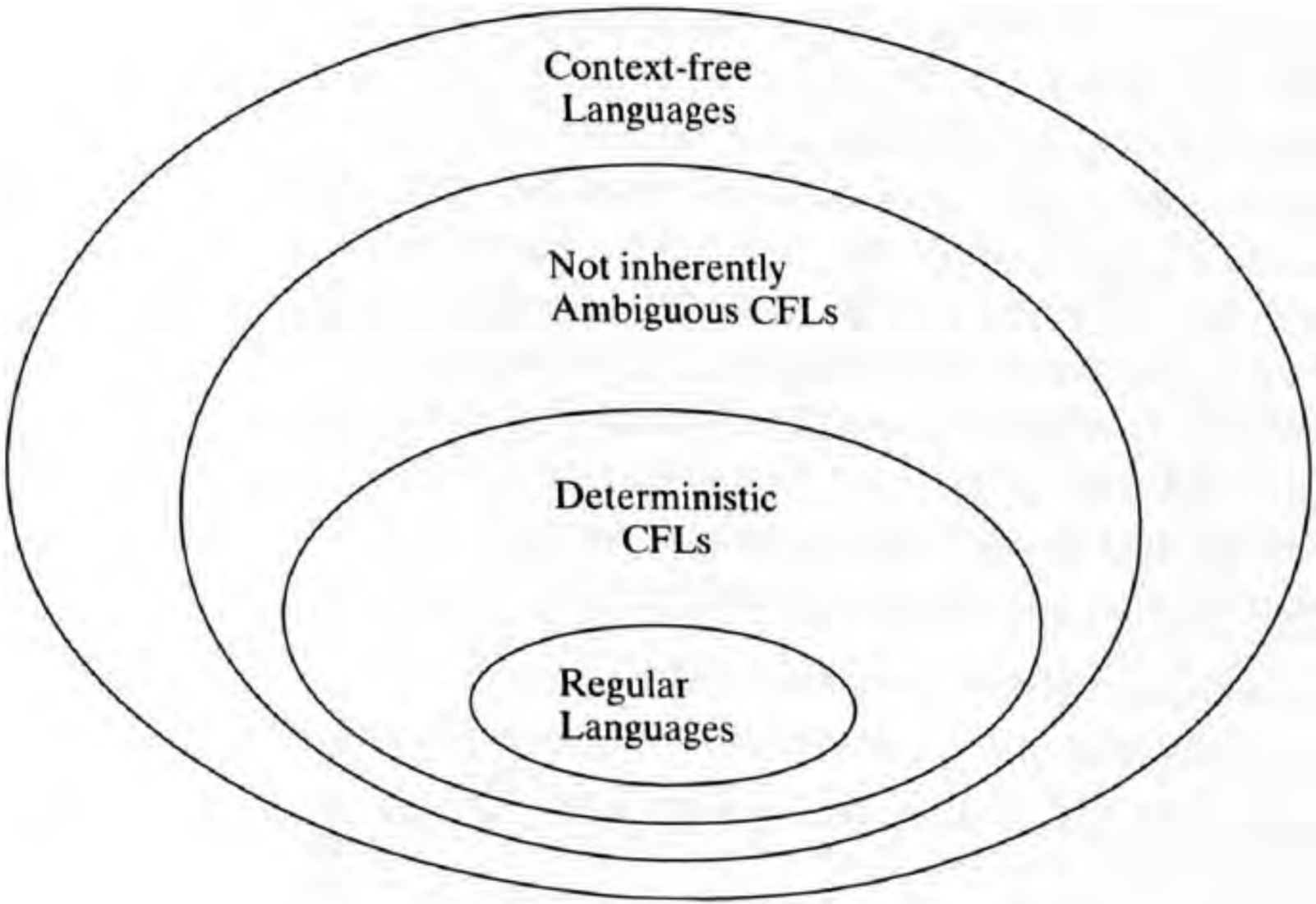
☞ *We can show that context-free grammars and pushdown automata are equivalent in power. Both are capable of describing the class of context-free languages.*

☞ *In contrast to finite automata, deterministic and nondeterministic pushdown automata are not equivalent. There are context-free languages that are not deterministic.*

☞ *A language L is said to be a deterministic context-free language (DCFL) if and only if there exists a DPDA M such that $L = L(M)$.*

☞ *Because every regular language is recognized by a finite automaton and every finite automaton is automatically a pushdown automaton that simply ignores its stack, we now know that every regular language is also a context-free language. Every regular language is context free.*

A hierarchy within the class of context-free languages.



Theorem: *Let L_1 be a context-free language and L_2 be a regular language. Then $L_1 \cap L_2$ is context free.*

Proof: *Let $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, F_1)$ be an NDPA that accepts L_1 and $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ be a DFA that accepts L_2. We construct a push-down automaton*

$$\widehat{M} = (\widehat{Q}, \Sigma, \Gamma, \widehat{\delta}, \widehat{q_0}, \widehat{F})$$

that simulates the parallel action of M_1 and M_2: Whenever a symbol is read from the input string, $\widehat{M}$ simultaneously executes the moves of M_1 and M_2. To this end we let

$$\widehat{Q} = Q \times P,$$

$$\widehat{q_0} = (q_0, p_0),$$

$$\widehat{F} = F_1 \times F_2,$$

and define $\widehat{\delta}$ *such that* $((q_k, p_l), x) \in \widehat{\delta}((q_i, p_j), a, b)$ *if and only if* $(q_k, x) \in \delta_1(q_i, a, b)$ *and*

$$p_l = \begin{cases} p_j, & \text{if } a = \varepsilon, \\ \delta_2(p_j, a), & \text{otherwise.} \end{cases}$$

In other words, the states of $\widehat{M}$ *are labeled with pairs* (q_i, p_j)*, representing the respective states in which* M_1 *and* M_2 *can be after reading a certain input string. It is a straightforward induction argument to show that* $((q_0, p_0), w, \varepsilon) \vdash^*_{\widehat{M}} ((q_r, p_s), \varepsilon, x)$*, with* $q_r \in F_1$ *and* $p_s \in F_2$ *if and only if* $(q_0, w, \varepsilon) \vdash^*_{M_1} (q_r, \varepsilon, x)$*, and* $\delta^*(p_0, w) = p_s$*. Therefore, a string is accepted by* $\widehat{M}$ *if and only if it is accepted by* M_1 *and* M_2*, that is, if it is in* $L(M_1) \cap L(M_2) = L_1 \cap L_2$*.*

تمرین:

$$L = \{a^n b^m c^{n+m} : n \geq 0, m \geq 0\}.$$