

1. Review of Definition of Language Syntax

CS 336

West Virginia University

Lecture 1

August 22, 2000

Frances L. Van Scoy fvanscoy@wvu.edu

Alphabets and Languages

An alphabet is a finite non-empty set.

Let S and T be alphabets.

$$S \bullet T = \{ s t \mid s \in S, t \in T \}$$

(We'll often write ST for $S \bullet T$.)

λ = empty string, string of length zero

$$S^0 = \{ \lambda \}$$

$$S^1 = S$$

$$S^n = S^{(n-1)} \bullet S, n > 1$$

$$S^+ = S^1 \cup S^2 \cup S^3 \cup \dots$$

$$S^* = S^0 \cup S^+$$

A language L over an alphabet S is a subset of S^* .

How Many Languages Are There?

How many languages over a particular alphabet are there?

Uncountably infinitely many!

Then, any finite method of describing languages can not include all of them.

Formal language theory gives us techniques for defining some languages over an alphabet.

Methods for Defining Languages

- Grammar
 - Rules for defining which strings over an alphabet are in a particular language
- Automaton (plural is automata)
 - A mathematical model of a computer which can determine whether a particular string is in the language

Definition of a Grammar

A grammar G is a 4 tuple $G = (N, \Sigma, P, S)$, where

N is an alphabet of nonterminal symbols

Σ is an alphabet of terminal symbols

N and Σ are disjoint

S is an element of N ; S is the start symbol or initial symbol of the grammar

P is a set of productions of the form $\alpha \rightarrow \beta$ where

α is in $(N \cup \Sigma)^* N (N \cup \Sigma)^*$

β is in $(N \cup \Sigma)^*$

Definition of a Language Generated by a Grammar

We define \Rightarrow by

$\gamma \alpha \delta \Rightarrow \gamma \beta \delta$ if

$\alpha \rightarrow \beta$ is in P , and

γ and δ are in $(N \cup \Sigma)^*$

\Rightarrow^+ is the transitive closure of \Rightarrow

\Rightarrow^* is the reflexive transitive closure of \Rightarrow

The language L generated by grammar $G = (N, \Sigma, P, S)$, is defined by

$$L = L(G) = \{ x \mid S \Rightarrow^+ x \text{ and } x \text{ is in } \Sigma^* \}$$

Classes of Grammars (The Chomsky Hierarchy)

Type 0, Phrase Structure (same as basic grammar definition)

Type 1, Context Sensitive

- (1) $\alpha \rightarrow \beta$ where α is in $(N \cup \Sigma)^* N (N \cup \Sigma)^*$,
 β is in $(N \cup \Sigma)^+$, and $\text{length}(\alpha) \leq \text{length}(\beta)$
- (2) $\gamma A \delta \rightarrow \gamma \beta \delta$ where A is in N , β is in $(N \cup \Sigma)^+$, and
 γ and δ are in $(N \cup \Sigma)^*$

Type 2, Context Free

$A \rightarrow \beta$ where A is in N , β is in $(N \cup \Sigma)^*$

Linear

$A \rightarrow x$ or $A \rightarrow x B y$, where A and B are in N and x and y are in Σ^*

Type 3, Regular Expressions

- (1) left linear $A \rightarrow B a$ or $A \rightarrow a$, where A and B are in N and a is in Σ
- (2) right linear $A \rightarrow a B$ or $A \rightarrow a$, where A and B are in N and a is in Σ

Comments on the Chomsky Hierarchy (1)

Definitions (1) and (2) for context sensitive are equivalent.

Definitions (1) and (2) for regular expressions are equivalent.

If a grammar has productions of all three of the forms described in definitions (1) and (2) for regular expressions, then it is a linear grammar.

Each definition of context sensitive is a restriction on the definition of phrase structure.

Every context free grammar can be converted to a context sensitive grammar with satisfies definition (2) which generates the same language except the language generated by the context sensitive grammar cannot contain the empty string λ .

The definition of linear grammar is a restriction on the definition of context free.

The definitions of left linear and right linear are restrictions on the definition of linear.

Comments on the Chomsky Hierarchy (2)

- Every language generated by a left linear grammar can be generated by a right linear grammar, and every language generated by a right linear grammar can be generated by a left linear grammar.
- Every language generated by a left linear or right linear grammar can be generated by a linear grammar.
- Every language generated by a linear grammar can be generated by a context free grammar.
- Let L be a language generated by a context free grammar. If L does not contain λ , then L can be generated by a context sensitive grammar. If L contains λ , then $L - \{\lambda\}$ can be generated by a context sensitive grammar.
- Every language generated by a context sensitive grammar can be generated by a phrase structure grammar.

Type 3: Regular Expressions
Right Linear Grammars
Left Linear Grammars
Finite State Automata

A Left Linear Grammar for Identifiers

$S \rightarrow S a$

$S \Rightarrow a$

$S \rightarrow S b$

$S \rightarrow S 1$

$S \Rightarrow S 1 \Rightarrow a 1$

$S \rightarrow S 2$

$S \rightarrow a$

$S \Rightarrow S 2 \Rightarrow S b 2$

$S \rightarrow b$

$\Rightarrow S 1 b 2 \Rightarrow a 1 b 2$

A Right Linear Grammar for Identifiers

$S \rightarrow a T$

$S \rightarrow b T$

$S \rightarrow a$

$S \rightarrow b$

$T \rightarrow a T$

$T \rightarrow b T$

$T \rightarrow 1 T$

$T \rightarrow 2 T$

$T \rightarrow a$

$T \rightarrow b$

$T \rightarrow 1$

$T \rightarrow 2$

$S \Rightarrow a$

$S \Rightarrow a T \Rightarrow a 1$

$S \Rightarrow a T \Rightarrow a 1 T$

$\Rightarrow a 1 b T \Rightarrow a 1 b 2$

Definition of a Deterministic Finite State Automaton

A deterministic finite state automaton M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

Q is a finite, nonempty set of states

Σ is the finite nonempty tape alphabet

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

q_0 is an element of Q , the initial state

F is a subset of Q , the set of final states

Definition of a Language Accepted by a Deterministic Finite State Automaton

Define $\delta': Q \times \Sigma^* \rightarrow Q$ by

$$\delta'(q, a) = \delta(q, a)$$

$$\delta'(q, ax) = \delta'(\delta(q, a), x)$$

$$\delta'(q, \lambda) = q$$

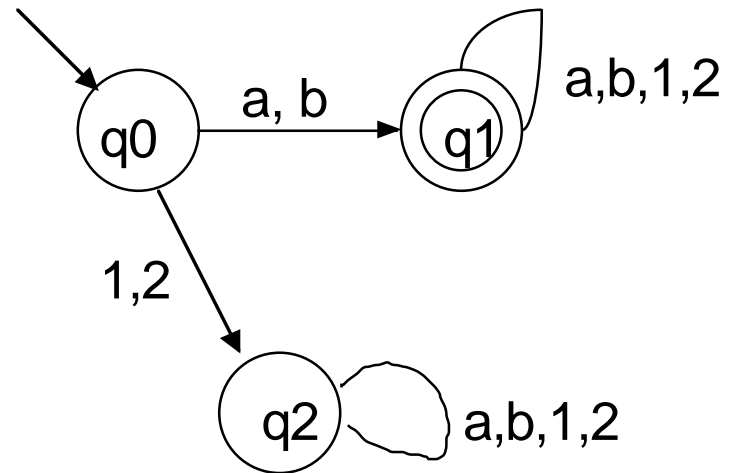
for q in Q , a in Σ , x in Σ^*

The language L accepted by the deterministic finite state automaton $M =$

$(Q, \Sigma, \delta, q_0, F)$ is defined by $L = T(M) = \{ x \mid x \text{ in } \Sigma^* \text{ and } \delta'(q_0, x) \text{ is in } F.$

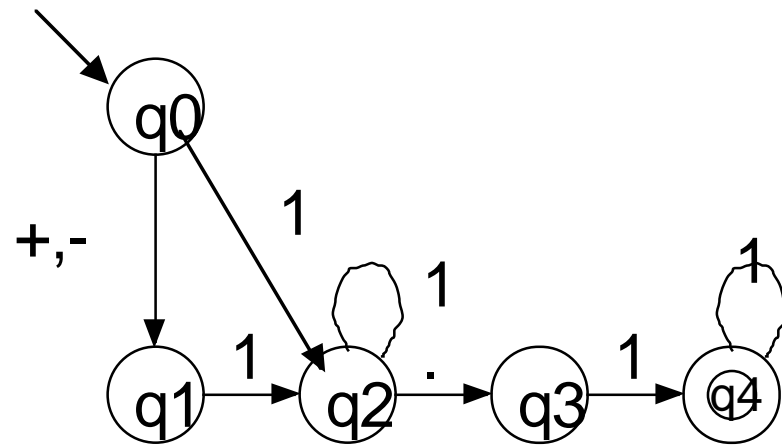
(We generally use δ and δ' interchangeably.)

A Deterministic Finite State Automaton for Identifiers



A Deterministic Finite State Automaton for Real Numeric Literals

	+	-	1	.
q0	q1	q1	q2	error
q1	error	error	q2	error
q2	error	error	q2	q3
q3	error	error	q4	error
q4 *	error	error	q4	error



Definition of a Nondeterministic Finite State Automaton

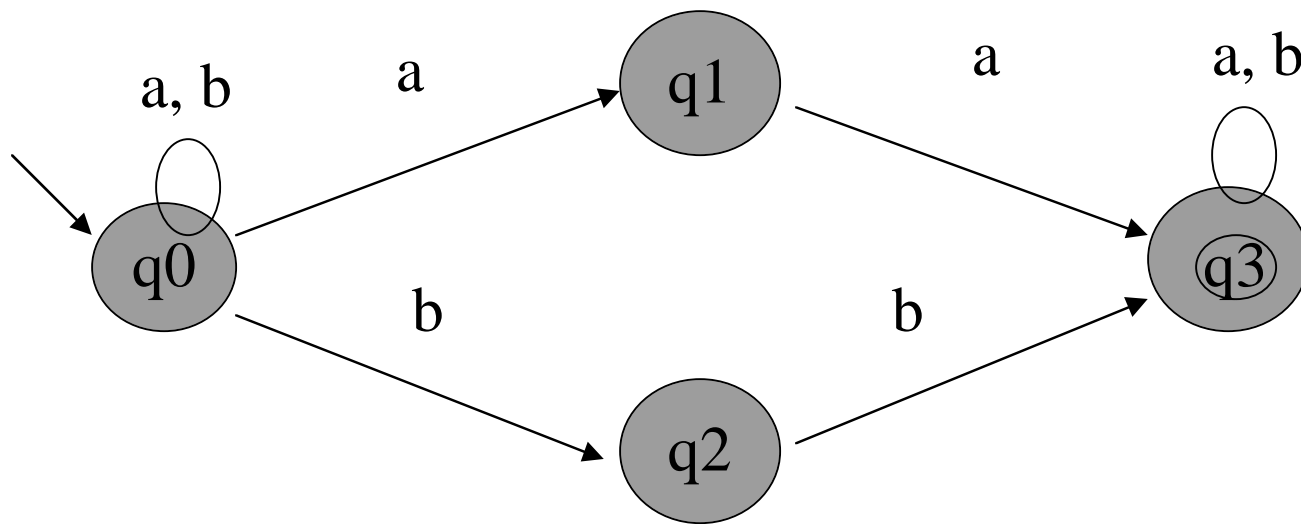
Let $P(S)$ denote the powerset of S , the set of all subsets of S .

A nondeterministic finite state automaton is defined the same as a deterministic finite state automaton, except

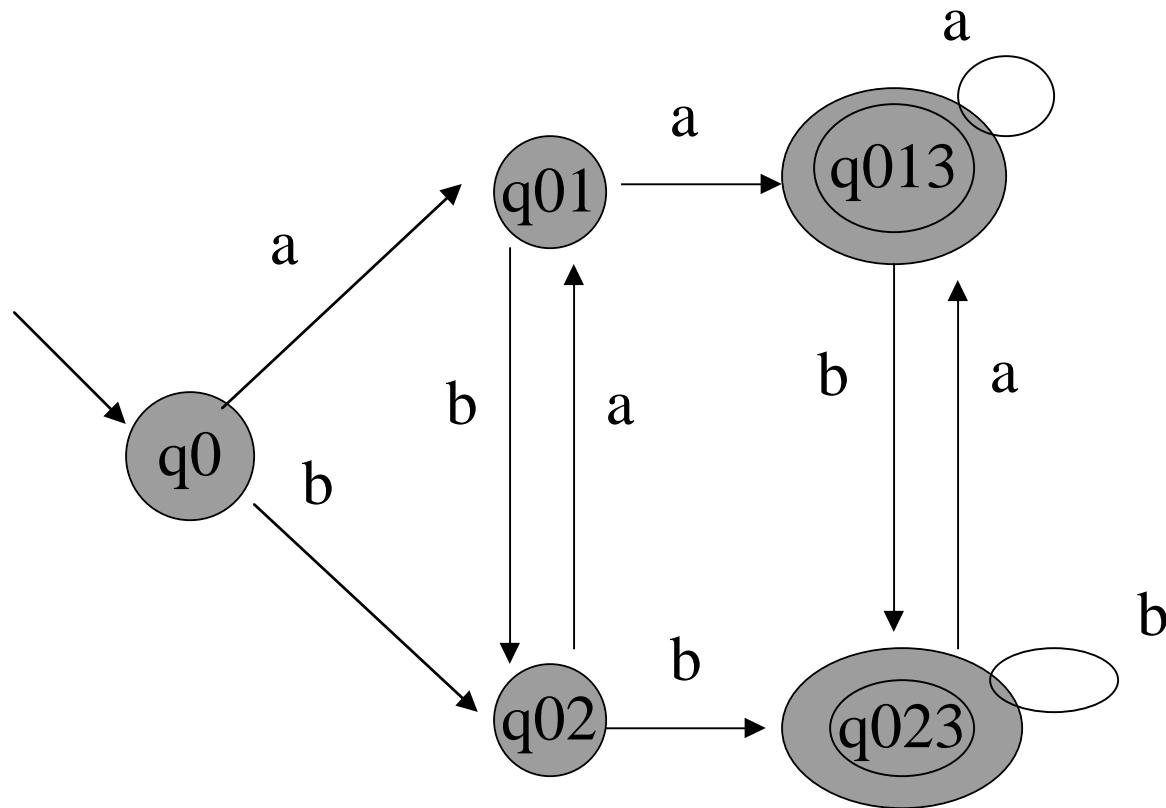
$$\delta: P(Q) \times \Sigma \rightarrow P(Q)$$

$$T(M) = \{ x \mid x \text{ in } \Sigma^* \text{ and } \delta(q_0, x) \text{ intersection } F \neq \Phi \}$$

Example of a Nondeterministic Finite State Automaton



Example of Conversion of a NDFSA to a DFSA



Class 2:

Context Free Grammars

Pushdown Automata

A Context Free Grammar for Expressions

$S \rightarrow E$	$F \rightarrow (E)$	$S \Rightarrow E \Rightarrow E + T$
$E \rightarrow E + T$	$F \rightarrow a$	$\Rightarrow E - T + T$
$E \rightarrow E - T$	$F \rightarrow b$	$\Rightarrow T - T + T$
$E \rightarrow T$	$F \rightarrow c$	$\Rightarrow F - T + T$
$T \rightarrow T * F$	$F \rightarrow d$	$\Rightarrow a - T + T$
$T \rightarrow T / F$	$F \rightarrow e$	$\Rightarrow a - T * F + T$
$T \rightarrow F$		$\Rightarrow a - F * F + T$
		$\Rightarrow a - b * F + T$
		$\Rightarrow a - b * c + T$
		$\Rightarrow a - b * c + F$
		$\Rightarrow a - b * c - d$

ex. grammar G1 (left recursive)

$S \rightarrow E$

$E \rightarrow E - T$

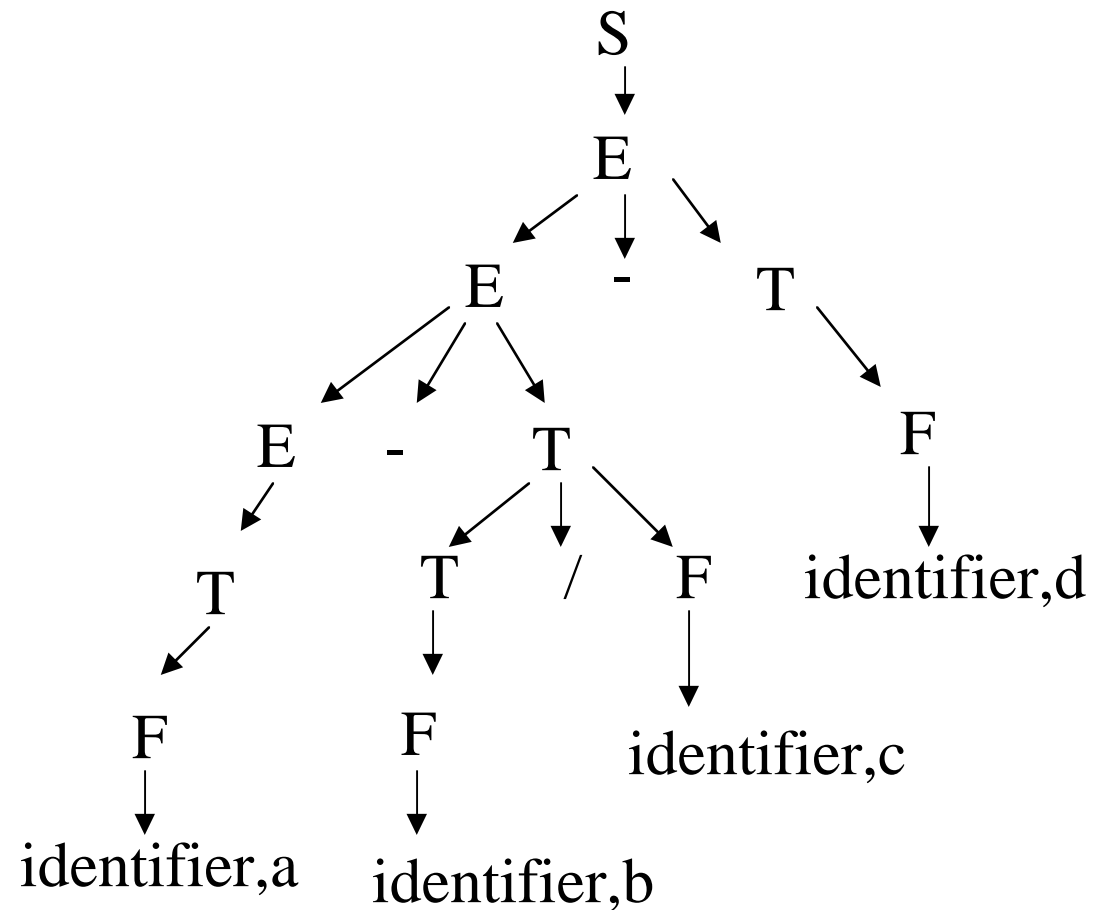
$E \rightarrow T$

$T \rightarrow T / F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow \text{identifier}$



ex. grammar G2 (right recursive)

$S \rightarrow E$

$E \rightarrow T - E$

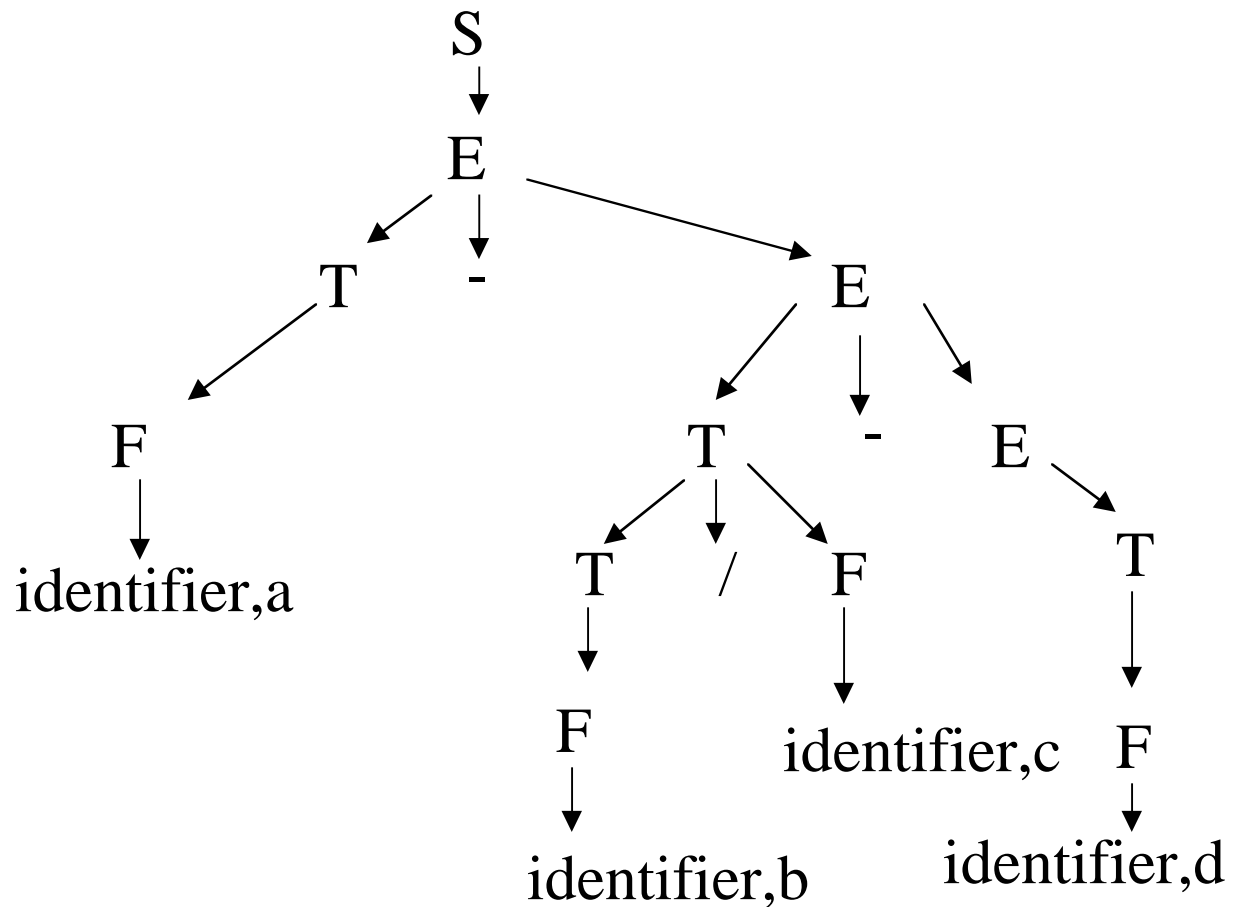
$E \rightarrow T$

$T \rightarrow F / T$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow \text{identifier}$



ex. grammar G3 (ambiguous)

$S \rightarrow E$

$E \rightarrow E - E$

$E \rightarrow E / E$

$E \rightarrow (E)$

$E \rightarrow \text{identifier}$

Ambiguity

- Given a language L and a grammar G such that L is generated by G ,
 - if x is in L and there are two distinct parse trees of x with respect to G , then grammar G is an ambiguous grammar
 - if every grammar which generates L is ambiguous, then L is an ambiguous language

An Ambiguous Context Free Language

- $\{a^j b^j c^k \mid j, k > 1\} \cup \{a^j b^k c^k \mid j, k > 1\}$ is an ambiguous language.
- We can prove that regardless of grammar, the string $\{a^j b^j c^j \mid j > 1\}$ will always have two distinct parse trees.

An Ambiguous English Sentence

Time flies like a green arrow.

subject = time, verb = flies

subject = you (understood), verb = flies

subject = flies, verb = like

ex. grammar G4, right linear

- $S \rightarrow \text{identifier } T$
- $S \rightarrow \text{identifier}$
- $T \rightarrow - S$
- $T \rightarrow / S$

Definition of Pushdown Automaton

- A pushdown automaton M is a 7 tuple $M = (Q, S, T, q_0, F, \delta, \epsilon)$ where
- Q is a finite nonempty set of states
- S is a finite nonempty set of terminal symbols (or tape symbols)
- T is a finite nonempty set of stack symbols
- q_0 in Q is the initial state
- F subset of Q is the set of final states
- δ in T
- $\epsilon: Q \times S \times T \rightarrow Q \times T^*$

Language Recognized by PDA

- $T(M) = \{ x \text{ in } S^* \mid d(q_0, x, t_0) = (q, t_0) \text{ where } q \text{ is in } F \}$

ex. Recognizing Odd Palindromes with Center Marker

$S \rightarrow a S a$

$S \rightarrow b S b$

$S \rightarrow \$$

- $d(q_0, a, x) = (q_0, xa)$
- $d(q_0, b, x) = (q_0, xb)$
- $d(q_0, \$, x) = (q_1, x)$
- $d(q_1, a, a) = (q_1, \text{lambda})$
- $d(q_1, b, b) = (q_1, \text{lambda})$

ex. Recognizing Even Palindromes (without Center Marker)

$S \rightarrow a S a$

$S \rightarrow b S b$

$S \rightarrow a a$

$S \rightarrow b b$

Determinism vs. Nondeterminism

- Nondeterministic and deterministic finite state automata have the same power.
 - That is, given a nondeterministic finite state automaton, I can produce a deterministic finite state automaton which recognizes exactly the same language.
- Nondeterministic pushdown automata have greater power than deterministic pushdown automata.
 - That is, there are some nondeterministic pushdown automata for which no equivalent deterministic pushdown automaton exists.

ex. Some English Palindromes

- Madam, I'm Adam.
- Able was I ere I saw Elba.
- Pa's a sap.
- Won't lovers revolt now?
- A man, a plan, a canal, Panama.

Class 1:
Context Sensitive Grammar
Linear Bounded Automata
2-Stack Pushdown Automata

ex. Grammar for $\{a^n b^n c^n \mid n > 0\}$

$S \rightarrow a S B C$

$S \rightarrow a B C$

$a B \rightarrow a b$

$b B \rightarrow b B$

$b C \rightarrow b c$

$c C \rightarrow c c$

$C B \rightarrow B C$

$S \Rightarrow a B C \Rightarrow a b C \Rightarrow$
 $a b c$

$S \rightarrow a S B C \Rightarrow a a B C$
 $B C \Rightarrow a a b C B C \Rightarrow$
 $a a b B C C \Rightarrow a a b b$
 $C C \Rightarrow a a b b c C \Rightarrow$
 $a a b b c c$

Some Other Context Sensitive Languages

$\{ w w \}$

(Recall $\{ w w^R \}$ is linear.)

$\{ a^m b^n c^{mn} \mid m, n > 0 \}$

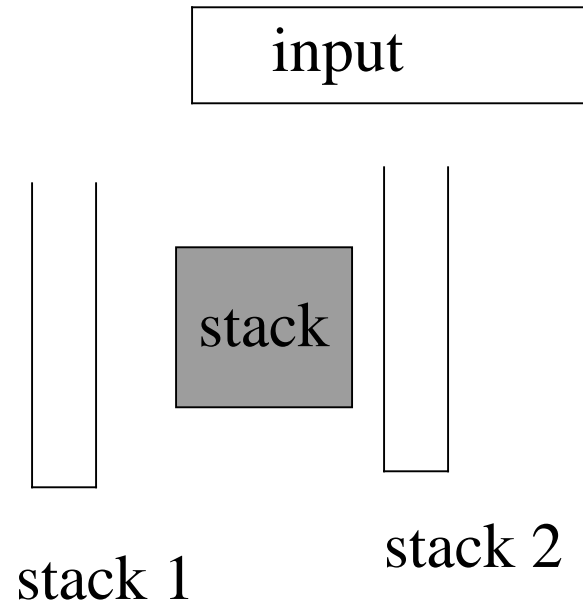
(Recall $\{ a^m b^n c^{m+n} \mid m, n > 0 \}$ is linear.)

$\{ a^m b^n c^m d^n \mid m, n > 0 \}$

(Recall $\{ a^m b^n c^n d^m \mid m, n > 0 \}$ is linear and $\{ a^m b^m c^n d^n \mid m, n > 0 \}$ is context free.)

2-Stack Pushdown Automaton

- input tape
- memory (state)
- two stacks



ex. Parsing

$$\{ \omega \$ \omega \mid \omega \text{ in } (\Sigma - \{\$\})^+ \}$$

- state q_0 (processing characters to left of \$)
 - if input is any symbol except \$ or eof, push it onto stack 1 and remain in state q_0
 - if input is \$, go to state q_1
 - if input is eof, error
- state q_1 (processing characters to right of \$)
 - if input is any symbol except \$ or eof, push it onto stack 2 and remain in state q_1
 - if input is \$, error
 - if input is eof, go to state q_2
- state q_2 (at end of input, comparing contents of two stacks)
 - if top of stack 1 = top of stack 2, pop both stacks and remain in q_2
 - if both stacks are empty, success

ex. Parsing
 $\{ \omega \omega \mid \omega \text{ in } \Sigma^+ \}$

- same as previous example, except everything is done nondeterministically!

Ex. Parsing

$$\{ a^m b^n c^m d^n \mid m, n > 0 \}$$

- state q0
 - if input is a, push onto stack 1 and stay in q0
 - if input is b, push onto stack 2 and go to q1
- state q1
 - if input is b, push onto stack 2 and stay in q1
 - if input is c, don't erase c from input string, go to q2
- state q2
 - if stack 1 is not empty pop top of stack 1 and push value onto state 2 and stay in q2
 - if stack 1 is empty, go to q3
- state q3
 - if input is c and top of stack is a, pop top of stack 2 and stay in q3
 - if input is d and top of stack is b, pop top of stack 2 and go to q4
- state q4
 - if input is d and top of stack is b, pop top of stack 2 and stay in q4
 - if input is eof and stack 2 is empty, success
- Otherwise, error

Linear Bounded Automaton

- if input has n characters, then input tape has space for $k * n$ characters and lba can write to tape as well as read from tape
- memory (state)
- 1 pushdown stack

The Determinism Question

- Deterministic and nondeterministic finite state automata are equivalent in power.
- There are some languages which can be recognized by a nondeterministic pushdown automaton but not by a deterministic pushdown automaton.
- It is an open question whether deterministic and nondeterministic linear bounded automata are equivalent in power.
- (Deterministic and nondeterministic Turing machines are equivalent in power. Turing machines and phrase structure grammars define the same class of languages.)

Class 0:

Phrase Structure Grammar

Turing Machine

Turing Machine

- finite memory
- no stack
- (one-way) infinite tape
- read/write head on
tape unit

The Church-Turing Thesis

- Alonso Church and Alan Turing
- The algorithms implementable by a Turing machine are precisely those which are computable.

Summary

- We can define a language by a grammar or automaton.
- Different classes of languages can be defined by restrictions on the form of the productions of the grammar.
- Sometimes the syntactic definition of a language has semantic implications.

Quiz

Construct grammars for each of the following languages.

$$\{ a^i b^j c^k d^l \mid i, j, k, l \geq 1 \}$$

$$\{ a^j b^j \mid j \geq 1 \}$$

$$\{ a^j b^k c^k d^j \mid j, k \geq 1 \}$$

$$\{ a^j b^j c^k d^k \mid j, k \geq 1 \}$$

$$\{ a^k b^k c^k \mid k \geq 1 \}$$