Linguistics 401, section 2
Hale December 3, 2003
Is English a finite state language?

Formal languages

In **formal language theory**, a language is a set of strings. A string is just a sequence of symbols chosen from an agreed-upon set of symbols, called the vocabulary or **lexicon**.

formal language		(non-)examples
any sequence of English words from the OED	$\in$	"at by green maybe", "vanity, all vanity"
	$\notin$	"I think que nous allons tromper the lesson"
strings over $\{a, b, c\}$ starting with a	$\in$	*abbb*, *a*, *a* followed by a million *b*'s
	$\notin$	*bcacc*, the zero-length string ϵ
strings over $\{a, b, c, d\}$ in alphabetical order	$\in$	*abd*, *ad*, *bcd*, *b*, *abcd*
	$\notin$	*dbcd*, *ba*

The idea of **generative grammar** is to use grammars to define a set that closely resembles a natural language – for instance, all and only the acceptable English sentences. However, not all sets are definable by all types of grammars. What kind of rules do we require to accurately describe natural languages?

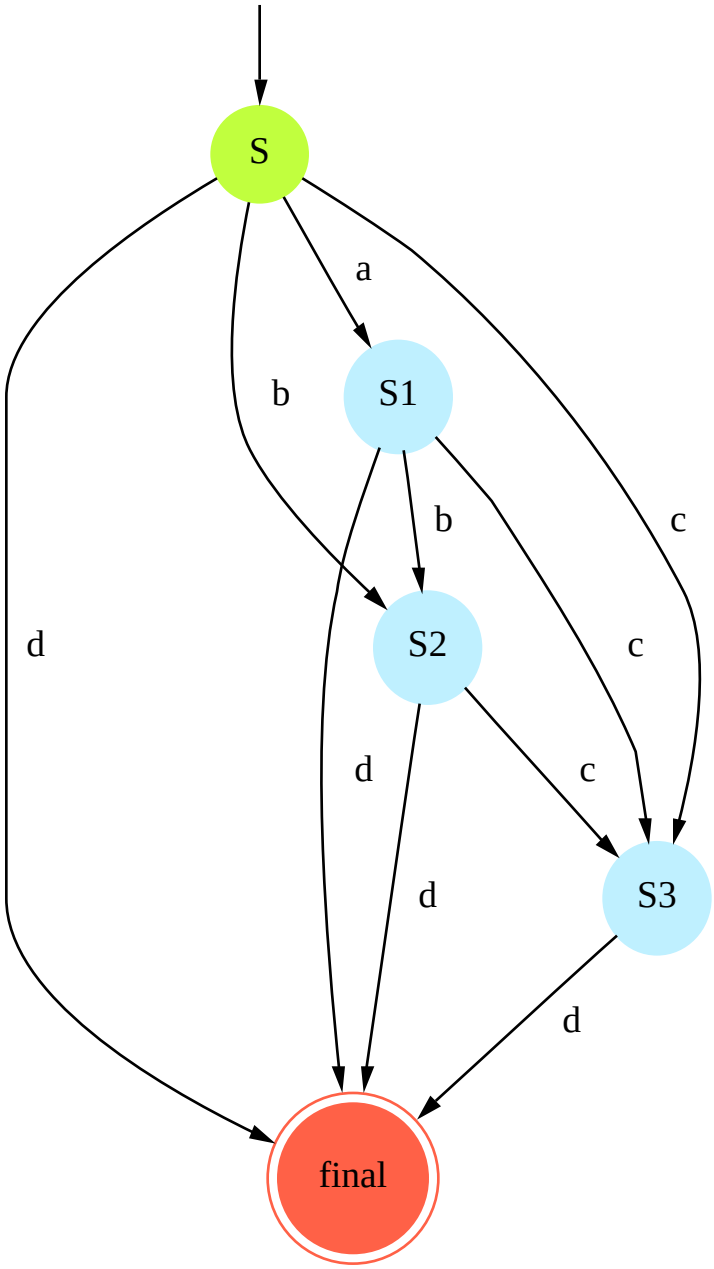
Finite-state grammars

To define ("generate") strings over $\{a, b, c, d\}$ in alphabetical order, let S be the start symbol of the grammar in (1).

(1)

S	→	a S1
S	→	b S2
S	→	c S3
S	→	d
S1	→	b S2
S1	→	c S3
S1	→	d
S2	→	c S3
S2	→	d
S3	→	d

Grammar (1) has a corresponding **finite state machine** that recognizes all and only the sentences it generates.


S
a
b
S1
c
b
d
S2
c
d
c
d
S3
d
final

The general form of finite state grammars

Any grammar having only rules of the form $A \rightarrow bC$ where A, B are nonterminals and b is a terminal has a corresponding finite state machine. Given a string, if a path can be found through the machine, the string is generated by the grammar and vice versa.

There are some languages that cannot be recognized by finite state machines.

formal language		(non-)examples
sequence of a's followed by an equal number of b's	$\in$	*ab*, *aabb*, *aaabbb*, ...
	$\notin$	*aabbb*, *aaaaaaaaaaaabbb*

Call the number of a's in the sentence being analyzed n. A finite state machine would need to 'remember' this number n while waiting for the end of the b's. But, by definition, a finite state machine will only have enough states to remember some fixed number of a's. Hence there exists neither a finite-state grammar nor a finite state machine for the language $a^n b^n$.

English is just like $a^n b^n$

Consider

(2)
a. The cat died.
b. The cat the dog chased died.
c. The cat the dog the rat bit chased died.
d. The cat the dog the rat the elephant admired bit chased died
⋮

If $A = \{\text{the cat}, \text{the dog}, \text{the rat}, \text{the elephant}, \text{the kangaroo}, ...\}$ and $B = \{\text{chased}, \text{bit}, \text{admired}, \text{ate}, \text{befriended}, ...\}$ it's clear that (2) have the structure

(3) $a^n b^{n-1}$died

Chomsky and Miller (1963) argued that the obligatory paired dependencies presented by *either...or*, *if...then* or the agreement between verbs and subjects can nest inside one another to an arbitrary depth.

(4) Anyone$_1$ who feels that if$_2$ so-many$_3$ more$_4$ students$_5$ whom we$_6$ haven't$_6$ actually admitted are$_5$ sitting in on the course than$_4$ ones we have that$_3$ the room had to be changed, then$_2$ probably auditors will have to be excluded, is$_1$ likely to agree that the curriculum needs revision.

This is analogous to the reversal language $\left\{xx^R | x \in \{a, b\}^*\right\}$, which also cannot be generated by a finite automaton.

Grammar (5) does generate the language $a^n b^n$. It is a **context-free** grammar, and as such is capable of deriving any number of center-embeddings.

(5)
S → a b
S → a S b

There is a corresponding machine that accepts the context-free languages: the **push down automaton**.

The Chomsky Hierarchy

Let A, B be nonterminals, b a terminal, α a nonempty sequence of either kind of symbol, and γ, δ possibly empty sequences of either kind of symbol. The **Chomsky Hierarchy** is a classification of languages in a subset relationship. Each language has corresponding class of machine that recognizes it.

language	rule type	machine
finite-state	$A \to bC$	finite state machine
context-free	$A \to \alpha$	push down automaton
context-sensitive	$\gamma A \delta \to \gamma \alpha \delta$	linear bounded automaton
unrestricted	no restriction	Turing machine

The cardinality of the set of languages definable by a grammar formalism is called its **generative capacity**. Natural languages are believed to reside somewhere between context-free and context-sensitive.