Grammars

Grammars define languages by generating strings, rather than consuming them.

Def: A (Chomsky) *grammar* is a 4-tuple (V, Σ, R, S), where

- V is a finite set of *variables* (or "nonterminal symbols"),
- Σ is a finite set, disjoint from V, of *terminals* (or "terminal symbols"),
- R is a finite set of *rules*. Each rule has the form $l \to r$, where l and r are elements of $(V \cup \Sigma)^*$ and l must contain at least one variable.
- $S \in V$ is the *start variable* (or "start symbol").

Elements of $(V \cup \Sigma)^*$ are sometimes called *sentential forms*.

Derivations

Consider a grammar $G = (V, \Sigma, R, S)$.

Def: If u, v, and w are sentential forms then we say u *yields* v ($u \Rightarrow v$) if for some sentential forms w_1 and w_2 and some rule $l \to r$ of G, we have $u = w_1 l w_2$ and $v = w_1 r w_2$ (*i.e.* v is obtained by replacing l by r in u).

We say that u *derives* v ($u \overset{*}{\Rightarrow} v$) if either $u = v$ or there exists u_1, u_2, . . . , u_k for $k \geq 0$ such that

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$$

(this is called a *derivation* of v from u).

The *language of G* is $\{w \in \Sigma^*.\ S \overset{*}{\Rightarrow} w\}$.

Classes of Grammars

- An arbitrary grammar is called *unrestricted*.
- A grammar G is *context-sensitive* if $|l| \leq |r|$ for every rule $l \to r$ in G.
- A grammar G is *context-free* if $l \in V$ for every rule $l \to r$ in G.
- A grammar G is *regular* (or "right-linear") if it is context-free and each rule $l \to r$ in G has either $r = w$ or $r = wB$ for some $w \in \Sigma^*$ and some $B \in V$.

For now, we are only concerned with context-free and regular grammars.

Example

Let $G = (V, \Sigma, R, S)$, where:

- $V = \{S\}$
- $\Sigma = \{0, 1\}$.
- $R = \{S \to \epsilon, S \to 0S1\}$.

Note: G is context-free.

The following is a derivation of G:

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000111$$

It is easy to see that $L(G) = \{0^n1^n.\ n \geq 0\}$.

Example Grammar (Sipser, 2.4)

Let $G = (V, \Sigma, R, E)$, where:

- $V = \{E, T, F\}$ "Expression", "Term", "Factor"
- $\Sigma = \{a, +, \times, (,)\}$.
- The set of rules R contains the following:

$$\begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T \times F \mid F \\ F & \rightarrow & (\ E \ ) \mid a \end{array}$$

The vertical bar | ("or") has been used to abbreviate multiple rules with the same left-hand-side.

Example Derivation

Rules:

$$\begin{array}{rcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T \times F \mid F \\ F & \rightarrow & (\ E\ ) \mid a \end{array}$$

Derivation:

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow E + F \Rightarrow E + (E) \Rightarrow T + (E) \Rightarrow F + (E) \\ &\Rightarrow F + (T) \Rightarrow F + (T \times F) \Rightarrow F + (F \times F) \Rightarrow F + (a \times F) \\ &\Rightarrow F + (a \times a) \Rightarrow a + (a \times a) \end{aligned}$$

Shows $a + (a \times a) \in L(G)$.

Leftmost Derivations

In general, a given string w will have multiple derivations that differ in the choice of which variable is expanded in each step.

These uninteresting differences can be avoided by considering only derivations that follow a definite rule.

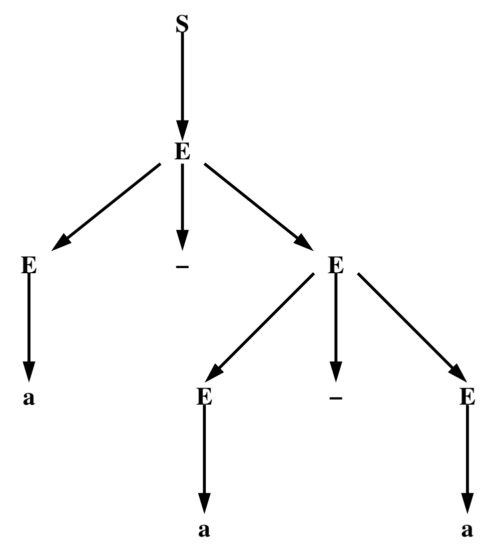
Def. A derivation is called *leftmost* if at each step it is the leftmost variable that is replaced.

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + F \\ &\Rightarrow a + (E) \Rightarrow a + (T) \Rightarrow a + (T \times F) \Rightarrow a + (F \times F) \\ &\Rightarrow a + (a \times F) \Rightarrow a + (a \times a) \end{aligned}$$

Parse Trees

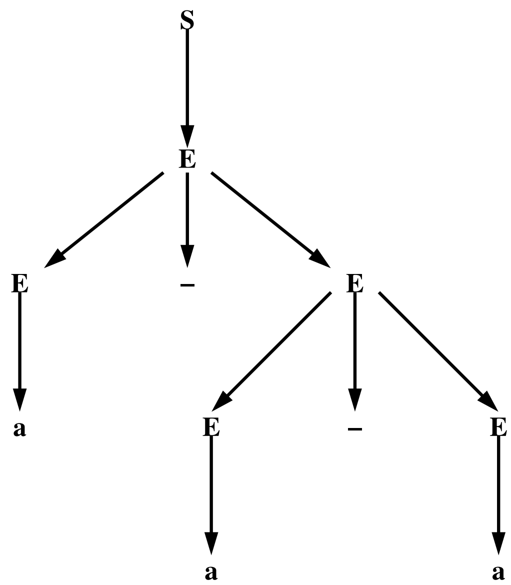
Def: Let G be a CFG. A *parse tree* for G is an ordered tree whose interior nodes are labeled by variables and whose leaves are labeled by terminals, such that for each node n labeled by variable A there is a rule $A \to r$ of G such that n has $|r|$ children, labeled by $r_1, r_2, \ldots, r_{|r|}$.

$$
\begin{aligned}
S &\to E \\
E &\to E - E \\
E &\to a
\end{aligned}
$$

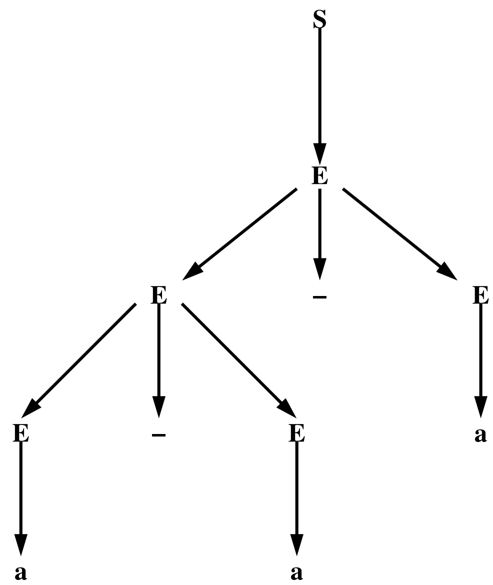


Leftmost Derivations and Parse Trees

Leftmost derivations and parse trees are in one-to-one correspondence:



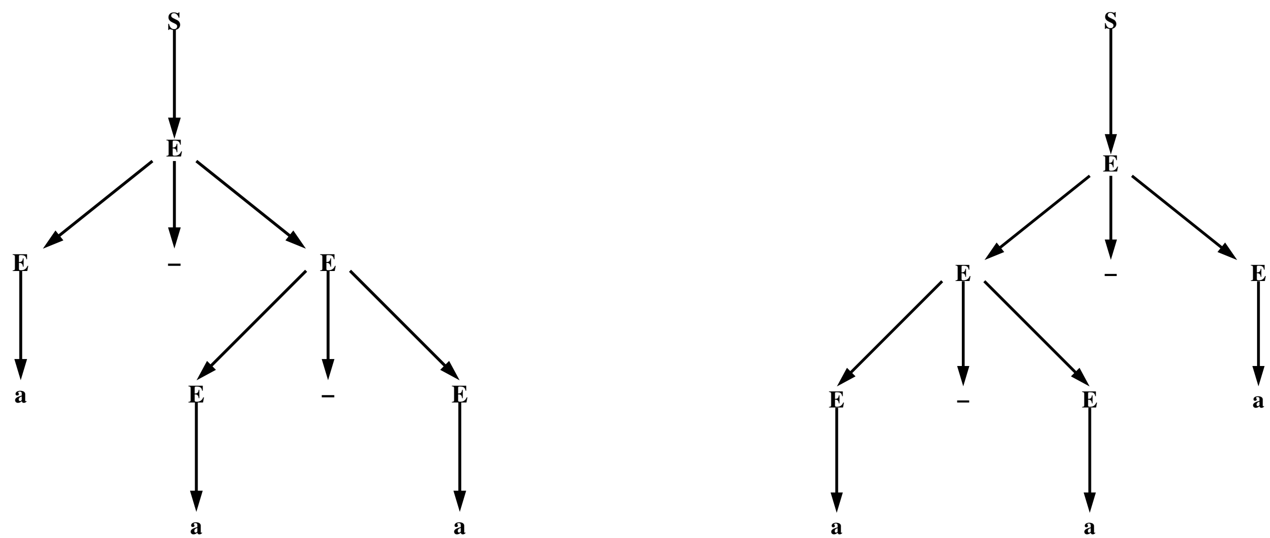
$$
\begin{aligned}
S &\Rightarrow E \Rightarrow E - E \\
&\Rightarrow a - E \Rightarrow a - E - E \\
&\Rightarrow a - a - E \Rightarrow a - a - a
\end{aligned}
$$



$$
\begin{aligned}
S &\Rightarrow E \Rightarrow E - E \\
&\Rightarrow E - E - E \Rightarrow a - E - E \\
&\Rightarrow a - a - E \Rightarrow a - a - a
\end{aligned}
$$

Ambiguity

Def. A context-free grammar is called *ambiguous* if there exists a string $w \in \Sigma^*$ such that w has more than one parse tree (equivalently, more than one leftmost derivation).



Ambiguity is an issue because the meaning of an expression (*e.g.* in a PL), can depend on the chosen parse tree.

Regular Grammars and Regular Languages

Thm: A language L is regular if and only $L = L(G)$ for some regular grammar G.

Proof: $(\rightarrow)$ Suppose L is regular. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes L. Define $G = (Q, \Sigma, R, q_0)$, where

$$R = \{q \rightarrow ar.\ \delta(q, a) = r\} \cup \{q \rightarrow \epsilon.\ q \in F\}.$$

Note: G is a regular grammar.

We can show (formal proof using induction omitted):

- For all $w \in \Sigma^*$, input w takes M from q to r if and only if $q \stackrel{*}{\Rightarrow} wr$.
- For all $w \in \Sigma^*$, M accepts w if and only if $q_0 \stackrel{*}{\Rightarrow} w$.
  Thus, $L(M) = L(G)$.

($\leftarrow$) Suppose $L = L(G)$ where $G = (V, \Sigma, R, S)$ is a regular grammar. We assume, without loss of generality, that every rule R has one of the the following two forms:

1. $A \rightarrow \epsilon$

2. $A \rightarrow aB$, where $a \in \Sigma \cup \{\epsilon\}$.

For any regular grammar we can construct an equivalent one satisfying this assumption by "splitting up" the right-hand sides of rules; e.g.

$$A \rightarrow w_1 \ldots w_n B \qquad \Longrightarrow \qquad \begin{array}{l} A \rightarrow w_1 B_1 \\ B_1 \rightarrow w_2 B_2 \\ \ldots \\ B_{n-1} \rightarrow w_n B \end{array}$$

Define an NFA

$$N = (V \cup \{f\}, \Sigma, \delta, S, \{f\}),$$

where

$$\begin{aligned} \delta(A, w) &= \{B \in V.\ R \text{ contains rule } A \to wB\} \cup \\ &\quad \{f.\ R \text{ contains rule } A \to w\}. \end{aligned}$$

Then N accepts $w \in \Sigma^*$ if and only if $S \overset{*}{\Rightarrow} w$.

Context-Free Languages

Def: A language L is *context-free* if $L = L(G)$ for some context-free grammar G.

Prop: Every regular language is context-free.
Proof: If L is regular, then we just showed $L = L(G)$ for some regular grammar G, and every regular grammar is a context-free grammar.

Prop: There exists a non-regular context-free language.
Proof: We showed $\{0^n1^n.\ n \geq 0\}$ is context-free. It is not regular (proved using the Pumping Lemma).

Closure Properties of the Class of Context-Free Languages

Thm: The class of context-free languages is closed under union, concatenation, and star.

Union: Suppose L_1 and L_2 are context-free. Obtain CFG's G_1 and G_2 such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Assume (renaming, if necessary) that V_1 and V_2 are disjoint. Form

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}, S).$$

Then $L(G) = L_1 \cup L_2$.

Concatenation: Suppose L_1 and L_2 are context-free. Obtain CFG's G_1 and G_2 such that $L_1 = L(G_1)$ and $L_2 = L(G_2)$. Assume (renaming, if necessary) that V_1 and V_2 are disjoint. Form

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_1 \cup R_2 \cup \{S \to S_1 S_2\}, S).$$

Then $L(G) = L_1 \cup L_2$.

Star: Suppose L_1 is context-free. Obtain CFG G_1 such that $L_1 = L(G_1)$. Form

$$G = (V_1 \cup \{S\}, \Sigma, R_1 \cup \{S \to \epsilon, S \to S_1 S\}, S).$$

Then $L(G) = L_1^*$.

Chomsky Normal Form

It is often useful in proofs to be able to suppose that a CFG is given in an especially simple form.

Def. A CFG is in *Chomsky normal form* if every rule has one of the following three forms:

$$\begin{array}{lll} A & \rightarrow & BC \\ A & \rightarrow & a \\ S & \rightarrow & \epsilon \end{array}$$

(B, C are not the start symbol S)

Theorem: *(Sipser, 2.9)* Every CFG is equivalent to one in Chomsky normal form.

Proof: Let CFG $G = (V, \Sigma, R, S)$ be given. Use the following procedure to transform G into G':

1. Add a new start symbol S_0 and a new rule $S_0 \to S$ (to ensure that the new start symbol does not occur on the RHS of any rule).

2. Remove an "ϵ-rule" $A \to \epsilon$, where A is not S, and add instead rules obtained by deleting occurrences of A on the RHS of other rules in all possible ways. That is:

$R \to uAv$	results in	$R \to uv$
$R \to uAvAw$	results in	$R \to uvw$
		$R \to uAvw$
		$R \to uvAw$
$R \to A$	results in	$R \to \epsilon$
	(if not already removed)	

Repeat until all ϵ-rules have been removed, except for $S \to \epsilon$ (if present).

3. Remove “unit rule” $A \rightarrow B$ and add new rule $A \rightarrow u$ for all rules $B \rightarrow u$, unless $A \rightarrow u$ is a unit rule that was previously removed.

   Repeat until all unit rules have been removed.

4. Convert remaining rules into the proper form:

$A \to u_1 u_2 \ldots u_k$	replaced by	$A \to u_1 A_1$ $A_1 \to u_2 A_2$ $\ldots$ $A_{k-1} \to u_{k-1} u_k$

(for $k \geq 3$, u_i terminals or nonterminals)

$A \to u_1 u_2$	replaced by	$A \to U_1 U_2$ $U_1 \to u_1$ $U_2 \to u_2$
$A \to u_1 C$	replaced by	$A \to U_1 C$ $U_1 \to u_1$
$A \to B u_2$	replaced by	$A \to B U_2$ $U_2 \to u_2$

(for $k = 2$, u_i terminals)

Notes

- None of the new rules added allow anything new to be derived.
- It is somewhat trickier to argue that steps (2) and (3) don't "lose" some of the strings that can originally be derived. (Sipser does not address this.)
- A complete proof would have to show how every derivation of the original grammar can be systematically converted into a derivation from the new grammar.

Example (Sipser Ex. 2.10)

$$
\begin{array}{lcl}
S & \rightarrow & ASA \\
S & \rightarrow & aB \\
A & \rightarrow & B \\
A & \rightarrow & S \\
B & \rightarrow & b \\
B & \rightarrow & \epsilon
\end{array}
$$

Step 1:

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \\
S &\rightarrow aB \\
A &\rightarrow B \\
A &\rightarrow S \\
B &\rightarrow b \\
B &\rightarrow \epsilon
\end{aligned}
$$

Step 2(a):

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \\
S &\rightarrow aB \\
S &\rightarrow a \\
A &\rightarrow B \\
A &\rightarrow \epsilon \\
A &\rightarrow S \\
B &\rightarrow b \\
B &\rightarrow \epsilon
\end{aligned}
$$

Step 2(b):

$$
\begin{array}{lcl}
S_0 & \rightarrow & S \\
S & \rightarrow & ASA \\
S & \rightarrow & SA \\
S & \rightarrow & AS \\
S & \rightarrow & S \\
S & \rightarrow & aB \\
S & \rightarrow & a \\
A & \rightarrow & B \\
A & \rightarrow & \epsilon \\
A & \rightarrow & S \\
B & \rightarrow & b
\end{array}
$$

Step 3:

$$
\begin{array}{lcl}
S_0 & \to & S \\
S & \to & ASA \\
A & \to & ASA \\
S & \to & SA \\
A & \to & SA \\
S & \to & AS \\
A & \to & AS \\
S & \to & S \\
S & \to & aB \\
A & \to & aB \\
S & \to & a \\
A & \to & a \\
A & \to & B \\
A & \to & S \\
B & \to & b \\
A & \to & b
\end{array}
$$

Step 4(a):

$$
\begin{array}{lcl}
S_0 & \rightarrow & S \\
S & \rightarrow & ASA \\
S & \rightarrow & AS_1 \\
S_1 & \rightarrow & SA \\
A & \rightarrow & ASA \\
A & \rightarrow & AA_1 \\
A_1 & \rightarrow & SA \\
S & \rightarrow & SA \\
A & \rightarrow & SA \\
S & \rightarrow & AS \\
A & \rightarrow & AS \\
S & \rightarrow & aB \\
A & \rightarrow & aB \\
S & \rightarrow & a \\
A & \rightarrow & a \\
B & \rightarrow & b \\
A & \rightarrow & b
\end{array}
$$

Step 4(b):

$$
\begin{array}{lcl}
S_0 & \rightarrow & S \\
S & \rightarrow & AS_1 \\
S_1 & \rightarrow & SA \\
A & \rightarrow & AA_1 \\
A_1 & \rightarrow & SA \\
S & \rightarrow & SA \\
A & \rightarrow & SA \\
S & \rightarrow & AS \\
A & \rightarrow & AS \\
S & \rightarrow & aB \\
S & \rightarrow & A_2B \\
A_2 & \rightarrow & a \\
A & \rightarrow & aB \\
S & \rightarrow & A_2B \\
S & \rightarrow & a \\
A & \rightarrow & a \\
B & \rightarrow & b \\
A & \rightarrow & b
\end{array}
$$

Final:

$$
\begin{array}{lcl}
S_0 & \rightarrow & S \\
S & \rightarrow & AS_1 \\
S_1 & \rightarrow & SA \\
A & \rightarrow & AA_1 \\
A_1 & \rightarrow & SA \\
S & \rightarrow & SA \\
A & \rightarrow & SA \\
S & \rightarrow & AS \\
A & \rightarrow & AS \\
S & \rightarrow & A_2B \\
A_2 & \rightarrow & a \\
S & \rightarrow & A_2B \\
S & \rightarrow & a \\
A & \rightarrow & a \\
B & \rightarrow & b \\
A & \rightarrow & b
\end{array}
$$