

Context-Sensitive Grammars

Context-sensitive languages are specified with a context-sensitive grammar (CSG), in which every production has the form:

$$\alpha \rightarrow \beta, \text{ where } |\beta| \geq |\alpha|$$

An example CSG:

$$\begin{array}{ll} S & \rightarrow aBCT|aBC \\ T & \rightarrow ABCT|ABC \\ BA & \rightarrow AB \\ CA & \rightarrow AC \\ CB & \rightarrow BC \\ aA & \rightarrow aa \\ aB & \rightarrow ab \\ bB & \rightarrow bb \\ bC & \rightarrow bc \\ cC & \rightarrow cc \end{array}$$

Context-Sensitive Languages

Some example derivations for the previous grammar:

$$\begin{aligned} S &\Rightarrow aBC \\ &\Rightarrow abC \\ &\Rightarrow abc \end{aligned}$$

$$\begin{aligned} S &\Rightarrow aBCT \\ &\Rightarrow aBCABC \\ &\Rightarrow aBACBC \\ &\Rightarrow aABCBC \\ &\Rightarrow aABBCC \\ &\Rightarrow aaBBCC \\ &\Rightarrow aabBCC \\ &\Rightarrow aabbCC \\ &\Rightarrow aabbcC \\ &\Rightarrow aabbcc \end{aligned}$$

$$L(G) = \{a^n b^n c^n : n \geq 1\}$$

What does this say about the relationship between CSG and CFG?

Linear Bounded Automata

A *linear bounded automaton* (LBA) is an automaton which has a finite length store (usually called a *tape*). Characters can be read or written at any position on this tape. It therefore has a read-write head which can move both left and right.

The same tape is normally used for both the input and the store. The special symbols $<$ and $>$ are used to mark the finite bounds of the tape beyond which the read-write head cannot move. These special symbols cannot be overwritten.

At each step, the LBA will perform one of the actions $\mathcal{A} \in \{Y, N, L, R\}$, where:

- Y denotes “Yes”, accept the input string
- N denotes “No”, do not accept the input string
- L denotes “Left”, move the read-write head one space to the left
- R denotes “Right”, move the read-write head one space to the right

Theorem: A language is context-sensitive iff it can be recognized by a linear bounded automaton.

LBA Definition

A linear bounded automaton is a 5-tuple

$M = (Q, \Sigma, \Gamma, q_0, \delta)$, where:

- Q is a finite set of *states*.
- Σ is an alphabet (*input symbols*).
- Γ is an alphabet (*store symbols*).
- $q_0 \in Q$ is the *initial state*.
- δ , the *transition function*, is from $Q \times (\Gamma \cup \{<, >\})$ to $Q \times (\Gamma \cup \{<, >\}) \times \mathcal{A}$.

If $((q, a), (q', b, action)) \in \delta$, then when in state q with a at the current read position on the tape, M may replace a with b on the tape, perform the specified *action*, and enter state q' .

M accepts $w \in \Sigma^*$ iff it starts with configuration $(q_0, \underline{\leq} w >)$ and the action Y is taken.

LBA Example

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$\text{Let } M = (Q, \Sigma, \Gamma, q_0, \delta)$$

- $Q = \{s, t, u, v, w\}$
- $\Sigma = \{a, b, c\}$
- $\Gamma = \{a, b, c, x\}$
- $q_0 = s$
 - $((s, <), (t, <, R))$
 - $((t, >), (t, >, Y))$
 - $((t, x), (t, x, R))$
 - $((t, a), (u, x, R))$
 - $((u, a), (u, a, R))$
 - $((u, x), (u, x, R))$
 - $((u, b), (v, x, R))$
- $\delta =$
 - $((v, b), (v, b, R))$
 - $((v, x), (v, x, R))$
 - $((v, c), (w, x, L))$
 - $((w, c), (w, c, L))$
 - $((w, b), (w, b, L))$
 - $((w, a), (w, a, L))$
 - $((w, x), (w, x, L))$
 - $((w, <), (t, <, R))$

LBA Example

The intuition behind the previous example is that on each pass through the input string, we match one a , one b and one c and replace each of them with an x until there are no a 's, b 's or c 's left.

Each of the states can be explained as follows:

- State t looks for the leftmost a , changes this to an x , and moves into state u . If no symbol from the input alphabet can be found, then the input string is accepted.
- State u moves right past any a 's or x 's until it finds a b . It changes this b to an x , and moves into state v .
- State v moves right past any b 's or x 's until it finds a c . It changes this c to an x , and moves into state w .
- State w moves left past any a 's, b 's, c 's or x 's until it reaches the start boundary, and moves into state t .