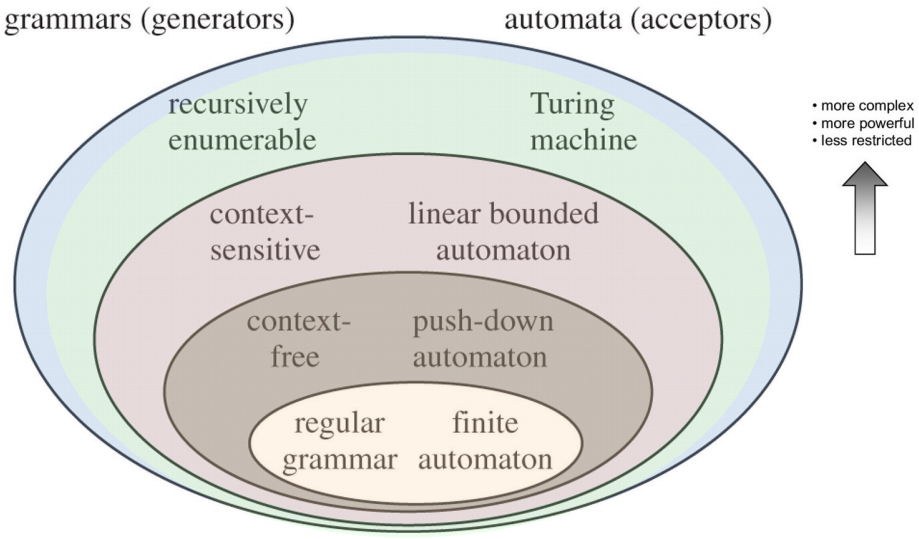
ECE 374 CSG vs CFG

Ranjani Ramesh

September 2025

1 What is the difference between Context Free and Context Sensitive Grammars?

First detail to remember: **All Context Free Grammars (CFGs) are also Context Senstive Grammars (CSGs).**
As seen in Chompsky's Hierarchy (below), CFGs are a special subset of CSGs



Now, coming to actual differences between them. The main difference lies in the way the production rules are defined for each of these Grammars.

1.1 CFG Production Rule Definition

$$A \to \alpha$$

- Here A is a non-terminal. Non-terminals are our variables such as the usual start symbol S and other variables such as A, B, and so on.

- α can belong to any string of non-terminals/terminals of any length. Most importantly, α is **independent of what surrounds A, thus making it context free.**

1.2 CSG Production Rule Definition

$$uAv \rightarrow u\gamma v$$

- Here, A is a non-terminal.
- The main difference here, is the difference of these u and v characters. These are the **context** that makes these grammars context sensitive. They are strings of non-terminals or terminals of any length.
- γ denotes what A becomes in the context of u and v. γ can be either a terminal or non-terminal string of any length.

2 Examples

Our main question during Office Hours was why does something like $L = \{a^i b^j c^k | i = j = k\}$ constitute only a CSG (and NOT a CFG), while something like $L = \{a^i b^j c^k | i = k \text{ or } j = k\}$ (Lab 6, Q6) constitutes a CFG (and a CSG).

This is because of the fact that in the first case we have to track i, j, and k and their equality to each other, while in the second case, we have to track two separate relationships i = k and j = k.

Considering the automatons that describe these grammars (refer to the Chomsky's hierarchy above), for CFGs they use PDAs while CSGs use something called Linear Bounded Automatons (LBAs). PDAs utilize a single stack and thus can track a single dependency since they can pop a symbol and simultaneously push another symbol. However, they have limitations since they cannot track more than one dependency (while LBAs can do this).

Here are the grammars for these questions to illustrate this better:

2.1 CSG Example

First, for $L = \{a^i b^j c^k | i = j = k, i, j, k > 0\}$

$S \rightarrow aSBC \mid aBC$	To create strings looking aBCBCBC and the base case aBC since i, j , k > 0
$CB \rightarrow BC$	Previously, BCs occur together, so we need to get the Bs and Cs to come together
$aB \rightarrow ab$	Context is important here, since B becomes b only if there is a a before the B.
$bB \rightarrow bb$	As previous, but if only there is a B before. For runs of b.
$bC \rightarrow bc$	For the first c after the run of bs ensuring c is in the right place (after run of bs)
$cC \rightarrow cc$	For the c in the run of cs

I would recommend generating some of your strings with these production rules. Here is a sample one:

$S \to aSBC \to aaSBCBC \to aaaBCBCBC$ - Rule 1

$aaaBCBCBC \to aaaBBCCBC \to aaaBBCBCC \to aaaBBBCCC$ - Rule 2

$aaaBBBCCC \to aaabBBCCC$ - Rule 3

$aaabBBCCC \to aaabbBCCC \to aaabbbCCC$ - Rule 4

$aaabbbCCC \to aaabbbcCC$ - Rule 5

$aaabbbcCC \to aaabbbccC \to aaabbbccc$ - Rule 6

2.2 CFG Example

Similar to Lab 6 Q 6, $L = \{a^i b^j c^k | i = k \text{ or } j = k\}$:

$$\begin{array}{rcl} S & \to & AB \mid XY \\ A & \to & a\,A\,b \mid \varepsilon \\ B & \to & c\,B \mid \varepsilon \\ X & \to & a\,X \mid \varepsilon \\ Y & \to & b\,Y\,c \mid \varepsilon \end{array}$$

The main thing to notice here, is that in each of the production rules, there is no context to maintain similar to the previous production rules!

Hope this clears up your question! Email me at rr37@illinois.edu if you have further questions or come to Office Hours (preferably on Thursday since it is less crowded)