What's cfg-vis, Anyway?

Daniel Guilak

Very semi-formally, a Context-Free Grammar[1] (CFG) G is a 4-tuple of the form

$$G = (V, \Sigma, R, S) \tag{1}$$

where

1. V is a finite set of variables (so in the prototype Python file `prototype/cfg-vis.py`, S is the only variable)
2. Σ is a finite set of terminals (the terminals I was using were a and b).
3. R is a relationship from V (a variable) to any number/permutation of different elements from both sets V and Σ (the nonterminals). For example a relationship of the form
$$S \to a|S \tag{2}$$
means that the non-terminal value a or the variable S can be derived from the variable S (which is a redundant rule, but you get the idea).
4. Finally, S is the start variable (I was using a variable of the same name in my file).

The simplified example I'm working on right now is a CFG that describes (can "generate") the language of palindromes over an alphabet that consists only of a and b (so a, aba, $baab$ are all valid palindromes). So it would have a formal form of $G = (V, \Sigma, R, S)$ where

1. $V = S$ – The only variable is S.
2. $\Sigma = a, b$ – the only two non-terminals are the characters a and b, so strings generated by the CFG can only have a's and b's.
3. R consists of only one rule (this is labeled "production" in `prototype/cfg-vis.py`), $S \to a|b|\epsilon|aSa|bSb$ which means five different expressions can be generated by S:
   - The non-terminals a or b,
   - The empty string ϵ,

[1] Sipser, M. (2006). Introduction to the Theory of Computation (Vol. 27). Boston, MA: Thomson Course Technology.

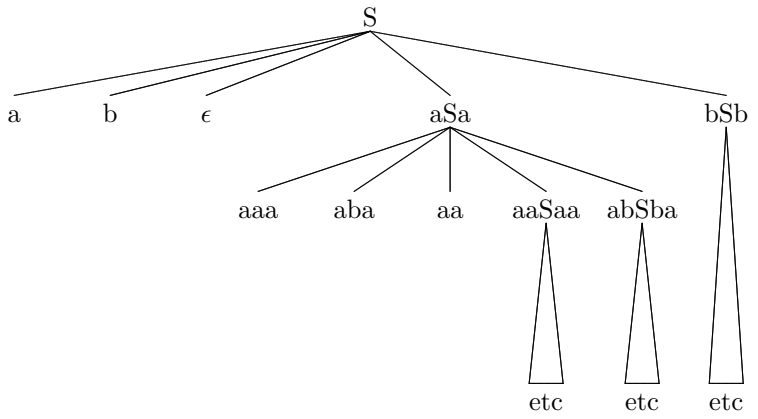
- The expressions aSa or aSb.

4. The start variable is S.

So as an example, if we wanted to use the CFG to generate the palindrome $baab$ starting with the variable S, we would follow these steps:

1. $S \rightarrow bSb$
2. $\rightarrow baSab$ (the middle S is replaced with aSa)
3. $\rightarrow baab$ (the middle S is replaced with ϵ, the empty string).

The visualization I've been working on for this grammar is that of an interactive tree to allow users to explore what a CFG can generate and how. If you imagine that the start variable S is the root, each of the possible expressions that can be derived from it could be represented as children of the root (so there would be a, b, ϵ, aSa, etc child nodes). Each of those child nodes would have their own children if they have at least one variable present, otherwise they would be leaves. So I hope to make something that looks like:



where start with just the root node displayed, and upon being clicked it will generate and display all of its children nodes which are also clickable and expandable, thus hopefully making CFGs a little easier to visualize! And it will be trivial to expand to more variables and production rules, but will obviously make the parse tree a little more complicated.