$\boxed{\textbf{Unrestricted Grammar Example}}$

Definition

By now you have seen grammars that are called Context Free Grammars which by definition only have a single variable on the left side in each of the production (replacement rules).

It is interesting to explore the possibilities once this restriction is removed. In fact, such grammars where any combination of variables and terminals can appear on the left side are called unrestricted grammars.

1 Example

While you have probably seen how to construct a context free grammar for

$\{ww^R | w \in \{a, b\}^*\}$

if you try to make a context free grammar for $D = \{ww | w \in \{a, b\}^*\}$ however, it will prove to be impossible.

In fact, using the pumping lemma, it is possible to prove that this language D is not context free.

The interesting question, is whether an unrestricted grammar can help us achieve this.

Since the solution is quite tricky we will provide the rules and see how they are derived. This example solution is taken from reference.

We begin by just adapting the solution for ww^R to have some indicators for the beginning and the end of the reversed portion.

Enter the following in JFLAP

$$\begin{aligned} S &\to Q\# \\ Q &\to aQa \\ Q &\to bQb \\ Q &\to T \end{aligned}$$

This will produce strings where the reverse of the first part can be found between T and $\#$.

The key now is to add rules that will help in reversing this portion of the string.

The idea is to add a rule $T \to TR$. Now the R can be used to slowly reverse the next two characters. This is where the unrestricted nature of the grammar truly plays a part. For instance if we have the characters a and b following the R, we can move the R forward and swap those two characeters. $Rab \to bRa$.

This adds the following rules to the grammar

$$\begin{aligned}
Raa &\to aRa \\
Rab &\to bRa \\
Rba &\to aRb \\
Ra\# &\to \#Ra \\
Rb\# &\to \#Rb
\end{aligned}$$

By using these rules to reverse the piece between T and $\#$, we will eventually end up with $wT\#w$. Clearly, the rule to add to get rid of the $T\#$ is $T\# \to \lambda$.

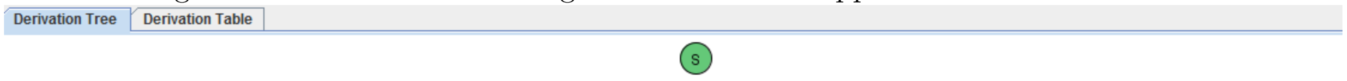
2 Parsing using JFLAP

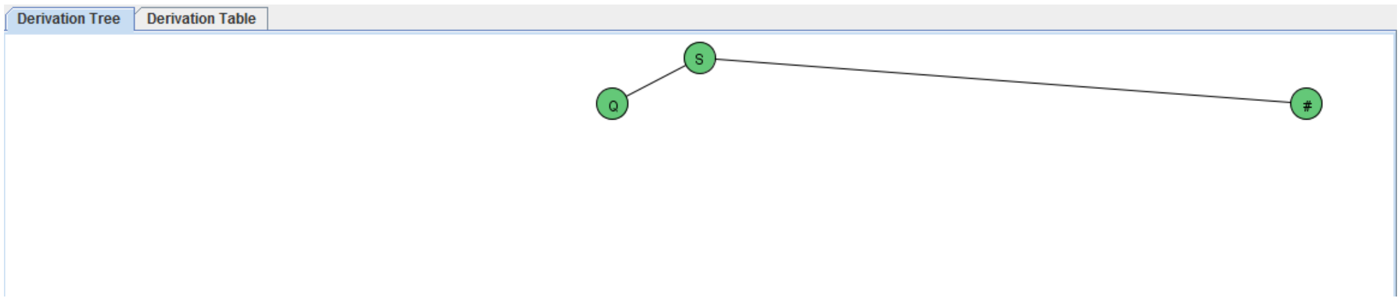
JFLAP can be used to parse strings using the unrestricted grammar. Similar to parsing of context free grammars, click on brute-force parse and enter a string into the text box. Let us try to parse *abab* which should be part of the language.

Similar to context free grammar parse, the brute force parsing is done by starting with the start symbol S and just trying every single possible next step.

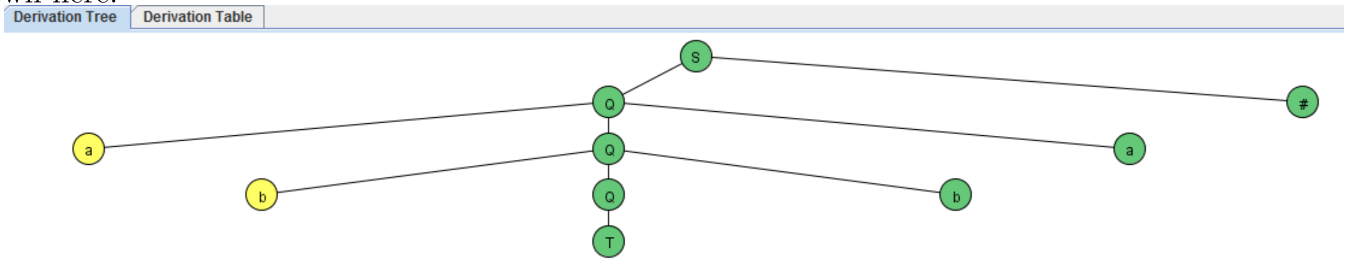
By clicking on step, we get the possible next steps in the derivation. In the parse table, it is a little hard to really appreciate how the unrestricted grammar is working. So, for this example, just click complete to finish up the brute force parse table. Then click into the derivation tree view.

The following screenshots show how the grammar rules are applied

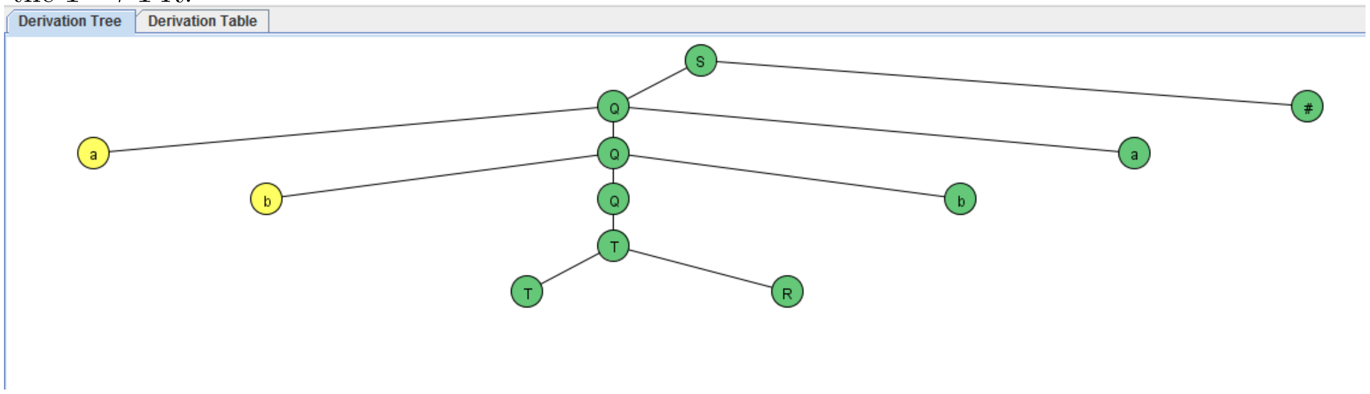




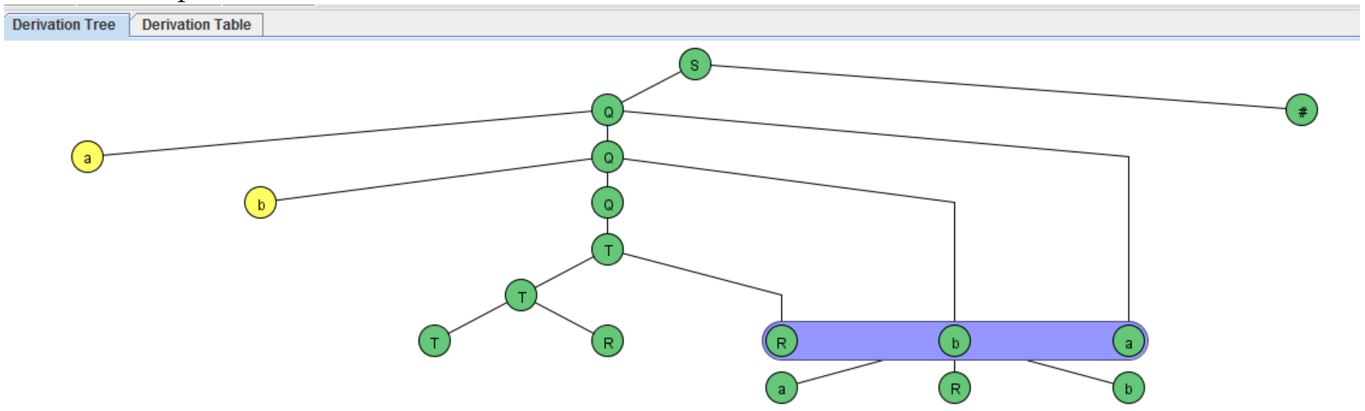
As mentioned before, the first step is to get something like ww^R in place and that is shown here.



Now we need to step through the reversed part of the string. The first step of this is to use the $T \to TR$.

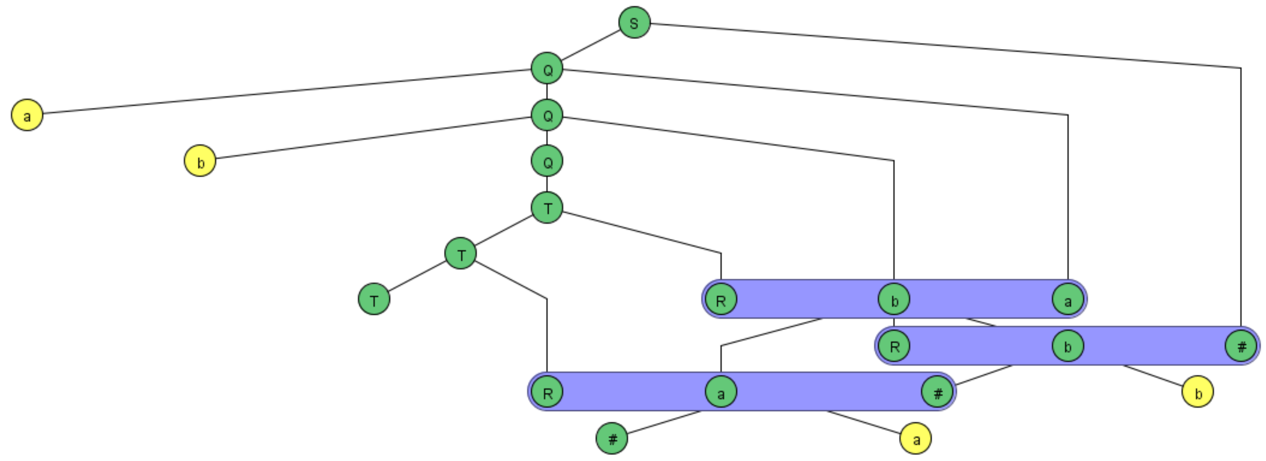


The next steps are where the unrestriced nature is used.

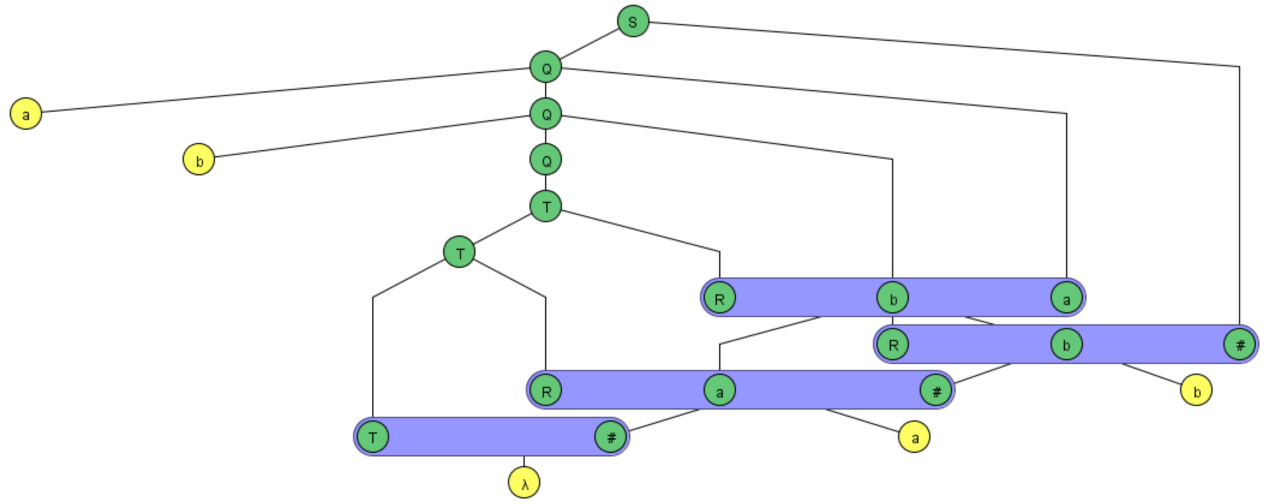


Notice how JFLAP shows a grouping of variables and terminals being replaced by a sequence of variables and terminals.

The Rs is being moved to the end of the string and each move will create a part of the reversal process.



Finally, we want to get rid of the $T\#$, which is easily done with $T\# \to \lambda$ which completes the derivation as shown below.



One of the key takeaways is that by allowing a combination of variables and terminals to be replaced, a larger class of languages can be obtained. In particular the language D which is impossible to generate using a context free grammar, is obtainable with an unrestricted grammar.