VISUALIZING WITH JFLAP

Background

My interest the issues associated to learning began back in the nineties, before I became aware of JFLAP. I attended a PKAL session in Williamsburg, VA and I was teaching a Freshman Seminar for students majoring in computing. I told the students in the seminar that I had two goals:

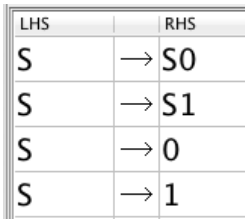
1. To give them a good opportunity to graduate with their class.
2. Determine whether majoring in computing was for them and, if not, make sure they are properly guided to a major that would be approriate for them.

I had an interest in the issues of how students learn. At the PKAL meeting, during a small group discussion, a member of the group mentioned Felder-Silverman Learning Styles, see http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Learning_Styles.html. Simple stated, Felder-Silverman Learning Styles view that different people learn in different ways and if they understand how they learn they may be able to adjust heir personal learning process. Felder-Silverman Learning Styles view learning as a combination of four independent learning axes, see http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/styles.htm:

- **Active/Reflective Axis**: Active learners tend to retain and understand information best by doing something active with it--discussing or applying it or explaining it to others. Reflective learners prefer to think about it quietly first. "Let's try it out and see how it works" is an **active** learner's phrase; "Let's think it through first" is the **reflective** learner's response.
- **Sensing/Intuitive Axis**: Sensing learners tend to like learning facts, intuitive learners often prefer discovering possibilities and relationships. Sensors often like solving problems by well-established methods and dislike complications and surprises; intuitors like innovation and dislike repetition. Sensors are more likely than intuitors to resent being tested on material that has not been explicitly covered in class.
- **Visual/Verbal Axis**: Visual learners remember best what they see--pictures, diagrams, flow charts, time lines, films, and demonstrations. Verbal learners get more out of words--written and spoken explanations. Everyone learns more when information is presented both visually and verbally.
- **Sequential/Global Axis**: Sequential learners tend to gain understanding in linear steps, with each step following logically from the previous one. Global learners tend to learn in large jumps, absorbing material almost randomly without seeing connections, and then suddenly "getting it."

I have focused on the Visual/Verbal axis, and found that visualization tools, like JFLAP, play a fundamental role in assisting students in mentally moving along this axis between the visualization of what they want to accomplish and understanding the corresponding code, or text, that associates to that visualization.

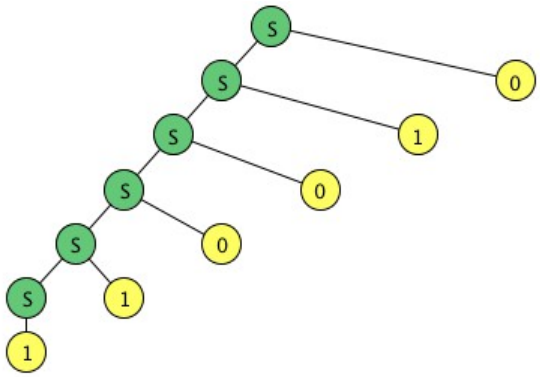
PRODUCTIONS TO STRINGS



LHS		RHS
S	→	S0
S	→	S1
S	→	0
S	→	1

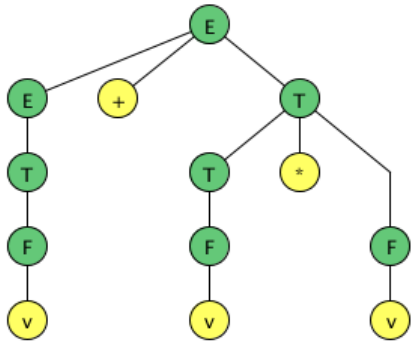
An important use I make of JFLAP is using it to provide students with some insight into starting with a grammar and its productions and seeing how one might generate a string in the language. I prefer applying the *noninverted tree* to the *Brute Parse Table*. Unfortunately, the *noninverted tree* is not currently up and running in JFLAP8.

To illustrate, let's start with a simple grammar, one that generates binary strings, illustrated to the right. Consider the parse structure that generates the string 110010. Generating a brute force parse in JFLAP 7 yields the structure illustrated. A simple question that should be posed when students generate brute force parses of a regular grammars is: Determine a simple way of expressing the structure generated by a parse of a regular grammar. I float this question over and over again, but do not request a definitive answer until we have covered Contest Free grammars.



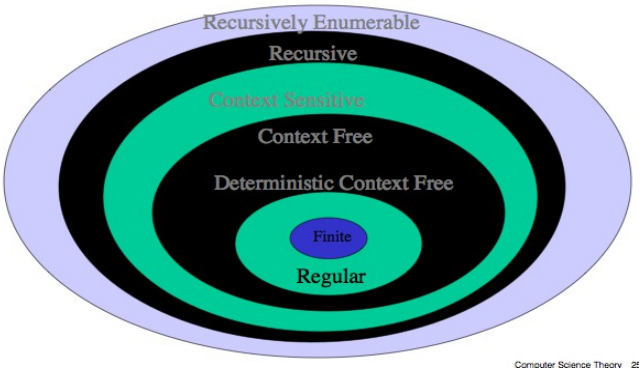
LHS		RHS
E	→	T
E	→	E+T
E	→	E−T
T	→	F
T	→	T*F
T	→	T/F
F	→	(E)
F	→	−F
F	→	v
S	→	E

When we move to context free grammars the structure of the parse is apparent to all of us, namely, a tree. For illustrations, consider the parse for a grammar of arithmetic expressions. The structure of the parse of a context free grammar is so ubiquitous that, without question we refer to it as a parse tree. As an example, consider the parse of the expression, v+v*v, where "v" represents any primitive variable or value. When you compare the parse of a string in a regular grammar, see above, to the parse of a string in a context free grammar, to the right, you could say they are both trees, and the regular string produces a very wimpy tree! Look again. The parse of strings in regular grammars form linked lists and all parses of strings in context free grammars are trees. When you look at the parse of a string in a regular grammar may be viewed as a linked list with the terminal symbols in the language contained in the head of each head-tail pair in the list structure. Those are defining characteristics of regular languages and context free languages.



PARSING CONTEST SENSITIVE AND UNRESTRICTED GRAMMARS

I have one minor gripe when it comes to JFLAP. It is that JFAP refers to any grammar beyond CFG as an unrestricted grammar. Technically, that is not true. In fact (tongue in cheek) if one would present a truly unrestricted grammar that is not contexts sensitive they would win $800,000! Recall, that all we have is an existence theorem about such a grammar.



The figure to the right comes from a computer science theory course presentation at RIT by Rob Duncan. The next step out is context sensitive grammar. Consider the grammar for the language,

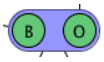
$$L = \{xx \mid x \varepsilon \{0,1\}^*\}$$

LHS		RHS
S	→	0TZ
S	→	1TO
T	→	0TA
T	→	1TB
AZ	→	PZ
BZ	→	PO
AO	→	QZ
BO	→	QO
P	→	0
Q	→	1
AP	→	PA
AQ	→	QA
BP	→	PB
BQ	→	QB
T	→	λ
Z	→	0
O	→	1

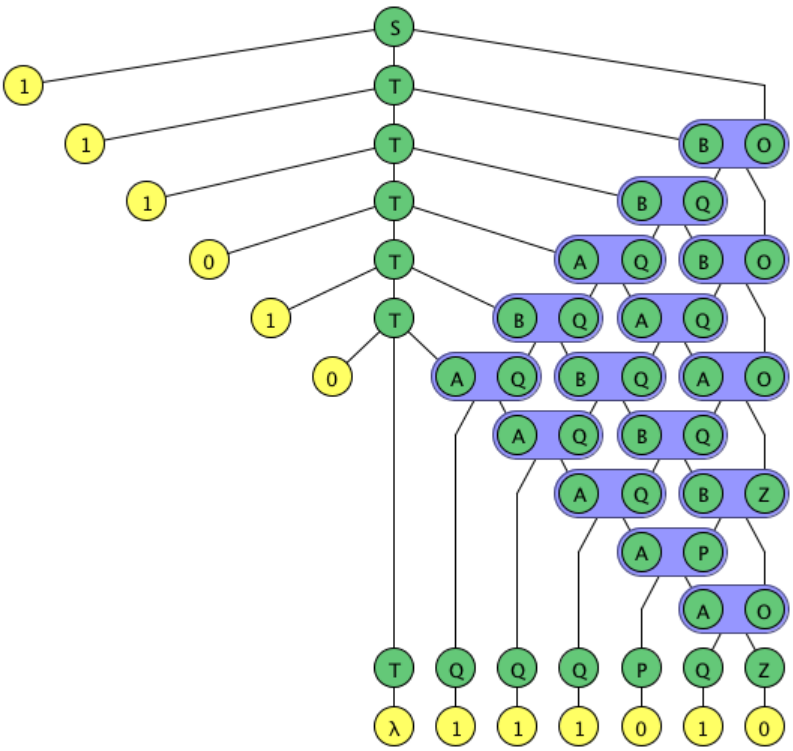
Let's continue our look at parsing and take a look at the grammar for the language, L. The productions for one possible grammar for L appear on the left. It is a contexts sensitive grammar because many of the productions have more than one entry on the left side of the production. The productions generate a string as follows:

1. The nonterminal, S, starts the productions, placing a 0 or 1 on the left and its corresponding right marker, Z (for zero) and O (for one). The nonterminal T builds the rest of the x with 0s and 1s, placing corresponding As and Bs in the right half of the string.
2. During the parse of the string there will aways be a Z or O marking the right side of the string. When a Z or O swaps positions with an A or B, note the four productions that do this. The Z is replaced by the nonterminal symbol P and the O is replaced by the nonterminal symbol Q, while the A or B are replaced by the nonterminals Z or O, respectively. The result is that during parsing the parse graph will always have a Z or O on the right and nowhere else.
3. Note that if the nonterminals P. Q, Z, and O are replaced by a 0 or 1 before items are in their proper places the productions will not terminate in a string of terminals.

Application of the productions to produce a string in the language is demonstrated below with the string, 111010111010. Note the righthand side of the parse graph. Each blue highlight focuses on the application of one of the non context free grammar productions. For example, consider the blue highlight,



It is highlighting the production whose left hand side. BO, with the right hand side, QO, which are the Q and O appearing immediately below the B and O.



SUMMARY

Simply stated, the parse graph of a grammar is:

- Regular Grammar – A linked list.
- Context Free Grammar – A tree.
- Context Sensitive Grammar and Unrestricted Grammar – A graph.

Now all we have to do (!) is classify the difference in structure of the parse graphs of contexts sensitive grammars and unrestricted grammars.