Improving the Capabilities of JFLAP: Creating Effective User Interfaces in Learning for Theoretical Computer Science

Ian McMahon | Advisor: Susan Rodger – Computer Science Department, Duke University, Durham, NC

This work was supported by NSF grant DUE-1044191

Automata and Formal Language Theory

· One of the foundations of theoretical computer science.
· Study of abstract machines, languages, and grammars, and their interactions.
· Due to its complex, abstract nature, is difficult to receive immediate feedback without some aid.

Software and Algorithm Visualization

· Methodology of creating graphical constructs to represent algorithms and data structures.
· Facilitates human understanding of computer software.
· In education, works as an aid to enhance learning through user interaction.
· Levels of engagement, effectiveness depending on knowledge level, and specific effective characteristics of SV all research topics.

JFLAP – Java Formal Languages and Automata Package

· Create and experiment with:
  · Automata – DFA, PDA, Turing machines
  · Grammars – regular context-free, unrestricted
  · Parsing – Brute farce, LL, SLR, CYK
  · Pumping Lemma – regular and CFG
  · L-Systems
· Experiment with proofs:
  · NFA to DFA to minimal state DFA
  · DFA to regular expression to regular grammar
  · NPDA to CFG, CFG to NPDA
· Started under Prof. Rodger in early 1990s as a tool for visualizing NPDAs.

Limitations of JFLAP v7.0

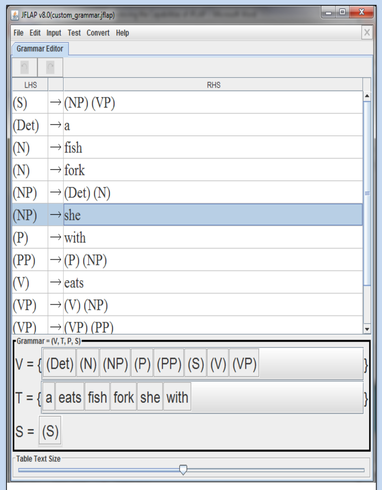
· Symbols represented as single alpha-numeric characters. This limits complexity of formal definitions due to finite number of symbols, lack of complex multi-character or symbolic languages, and direct String manipulation.
· Lack of explicit formal definitions hides underlying state from the user, separating it from textbook descriptions.
· Lack of GUIs for CYK and Brute Force parsing algorithms for grammars.
· Disjoint codebase resulting from 20 years of development.

Research goals

· Restructuring of JFLAP hierarchy
· Implementation and improvement of GUI
· Adding new functionality

Restructuring of JFLAP Hierarchy

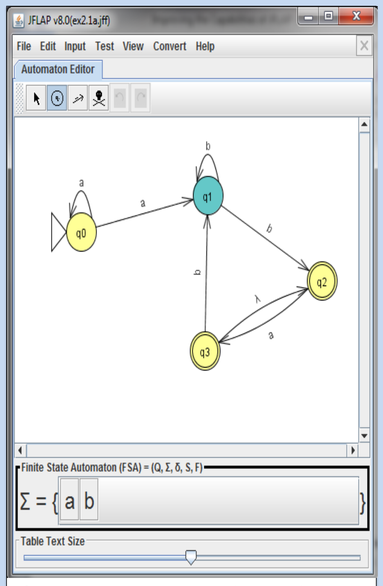
· JFLAP hierarchy started in 2011 under Julian Genkins with the introduction of the *Symbol*, *SymbolString*, *Alphabet* hierarchy. This provided an underlying base on which all formal definitions were able to consist of complex *Symbols* that did not conflict with previous single-character definitions.
· A common algorithm interface was introduced, allowing for all algorithms to be treated as similar objects.
· A stricter MVC architecture has been enforced, allowing for users to directly access model code without dealing with graphical tools.



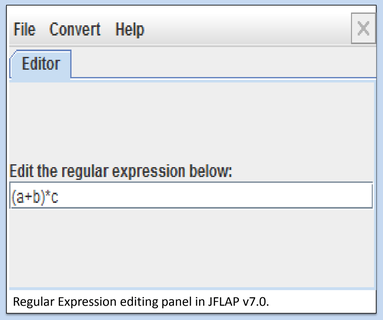
A custom-mode grammar using complex symbols to create a sentence. Note the explicit formal definition panel consisting of Variable and Terminal Alphabets and the Start Symbol.

GUI Implementation and Improvement

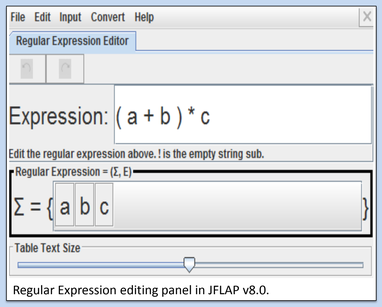
· Automata – More flexible editing capabilities and fixes including a new State tool that can move states as well as create them, transition editing without the use of control points, multiple selection using control/shift clicking, deletion using the delete key, disallowing objects to be dragged offscreen, and a more capable undo/redo functionality.
· Regular Expressions – New updated GUI, with undo/redo capabilities, expression checking, and alphabet bar.
· L-Systems – improved axiom entry, an obvious button to access the "hidden" parameter menu, rendering of context-sensitive L-Systems, and image saving to include entire rendering.
· Addition of Pumping Lemmas, most algorithms, an improved preference menu, and better dialog navigation.



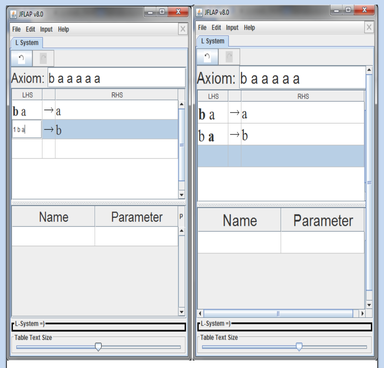
A finite automaton in JFLAP v8.0 using the state tool to manipulate state location. Note the explicit formal definition panel



Regular Expression editing panel in JFLAP v7.0.



Regular Expression editing panel in JFLAP v8.0.



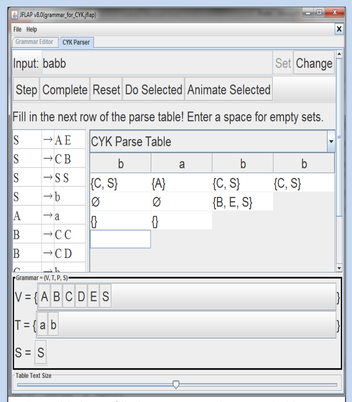
A simple context-sensitive L-System being edited by the user (left) and with cells rendered (right).

New functionality

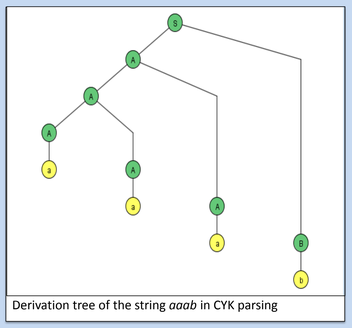
· CYK parsing algorithm implementation and GUI.
· Brute Force parsing algorithm fit to algorithm hierarchy, new visualization
· Language Generation algorithm and visualization
· Block and Universal Turing Machines, S-Option Removal algorithm and visualization.
· New file format, conversion of files in previous format.

CYK Parsing Algorithm

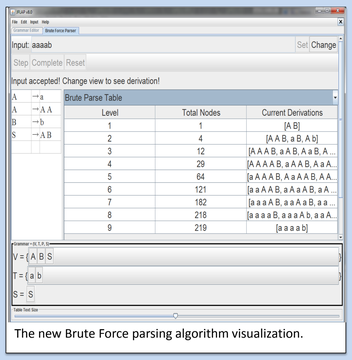
· Cocke-Younger-Kasami parsing algorithm.
$O(n^3 |G|)$, dynamic programming
· Considers all possible subsequences of a string, starting with single symbols, strings of 2, and so on.
· Due to CNF form, these partitions are easy to determine



A CYK table being filled out to parse the string *babb.*



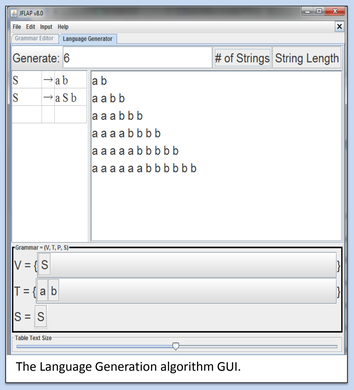
Derivation tree of the string *aaab* in CYK parsing



The new Brute Force parsing algorithm visualization.

Language Generation

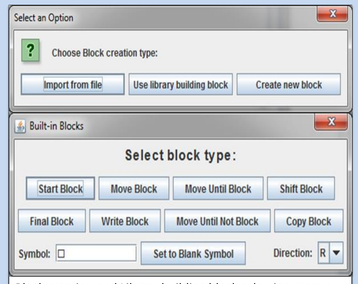
· Generates all strings of a specified length, or a number of strings in a language
· Works on grammars, automata, and regular expressions
· Uses CYK (for infinite context-free languages) and brute force (for finite, context-sensitive, and unrestricted languages) parsing algorithms
· Efficient for small example, allows for a new way to interact with languages



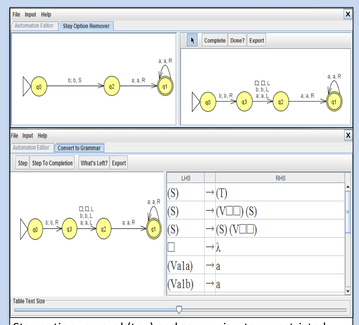
The Language Generation algorithm GUI.

Turing Machine Functionality

· Reconstruction of Building Blocks
· Built-in library of flexible blocks
· Universal TM conversion algorithm and blocks
· S-Option Removal for Unrestricted Grammar conversion



Block creation and Library building block selection menus in JFLAP v8.0.



Stay option removal (top) and conversion to unrestricted grammar (bottom) for a single-tape TM.

New File Format

· Conversion of files in previous format.
· Does not work on Block Turing Machines due to distinct changes in how the program deals with them

Assessment

· Small survey, 7 of 8 responses from students in Prof. Rodger's CompSci 334 class. 6 completed the whole survey, 1 completed part of it.
· Students were asked to do a set of exercises using new version of JFLAP and answer questions about their experiences.
· Results were all very positive, indicating that the changes add to the program's flexibility and use as an educational tool.

Statement	Mean
The Language Generator improves the capabilities of JFLAP as a teaching tool.	1.57
The new version of JFLAP improves the flexibility of parsing strings in a language.	1.5
The new version of JFLAP makes the formal definition of automata very clear.	1.5
The new version of JFLAP increases flexibility with undo/redo capabilities.	2.0

Four of the questions from the assessment survey, where strongly agree is 2, strongly disagree is -2