Formales

Alexander Vargas y Juan Esteban Orrego

2025-05-14

Introduction

This report documents the challenges and solutions encountered during the development of a combined LL(1)/SLR(1) grammar analyzer written in Python. The analyzer classifies input grammars as LL(1), SLR(1), both, or neither, and allows parsing of strings using the appropriate parser.

Objective

Our goal was to build a functional, interactive command-line grammar analyzer that can:

- Accept context-free grammars in a strict format (e.g., `E->E + T | T`).
- Determine if the grammar is LL(1), SLR(1), both, or invalid.
- Allow the user to analyze strings interactively using the appropriate parser.

Main Problems Encountered

- **Uninitialized Parser Mode Variable**
  In cases where only one parser type was valid, the variable controlling parser selection was not initialized, leading to runtime errors.
- **No SLR(1) Conflict Detection (Initially)**
  The original version of the program did not detect shift/reduce or reduce/reduce conflicts when building the SLR(1) action table. Conflicting actions were silently overwritten.
- **Grammar Type Overclassification**
  LL(1) grammars were sometimes wrongly classified as "both" simply because an SLR(1) table existed—even if it was not valid.
- **Strict Grammar Input Format**
  Grammar input was highly sensitive to spacing (e.g., no spaces around arrows and consistent spacing between tokens). Invalid formatting would silently fail or misbehave.
- **Ambiguous Parser Selection Prompt**
  The program asked for LL or SLR parser choice even when only one was valid, which could confuse users.
- **Frequent Indentation and Logic Bugs**
  Several iterations had broken parser logic or failed table construction due to indentation errors or incorrect control flows.

Solutions Applied

- **SLR(1) Conflict Detection Implemented**
  The program now explicitly checks for existing actions in the SLR(1) table before inserting new ones. If a conflict is found, a warning is printed and the table is invalidated.
- **Safe Parser Selection Logic**
  The parser selection variable is now always defined. When only one parser is valid, it is automatically selected without prompting the user.
- **Improved Classification Logic**
  The analyzer now correctly distinguishes among LL(1)-only, SLR(1)-only, both, or neither based on valid parsing table construction.
- **Enhanced Input Formatting Guidance**
  Clear instructions are now provided to users for correct grammar input, including how to use the '->', '|', and spaces.
- **Conflict Warnings for Debugging**
  When conflicts are detected during SLR(1) table construction, the program prints the state, symbol, and conflicting actions.
- **Consistent Indentation and Error Handling**
  All code blocks were reviewed for indentation correctness and defensive programming was added to avoid runtime crashes.

Remaining Limitations

- **No Grammar Minimization or Cleanup**
  There is no detection or elimination of unreachable non-terminals, useless symbols, or left recursion (except as indirectly handled by LL(1) validation).
- **LL(1) vs BOTH Classification is Still Conservative**
  In some borderline cases, the tool may still conservatively classify a grammar as "LL(1) and SLR(1)" if both tables build successfully—even if SLR(1) has looser restrictions.

Conclusion

The final version of the LL(1)/SLR(1) grammar analyzer is now significantly more robust and reliable. It supports safe parser selection, real SLR(1) conflict detection, and accurate grammar classification. While further enhancements are possible (e.g., grammar rewriting or visualization), the current implementation fulfills its educational and functional goals successfully.