COMP 442/6421 Compiler Design

Instructor: Dr. Joey Paquet paquet@cse.concordia.ca
TA: Zachary Lapointe zachary.lapointe@mail.Concordia.ca

LAB 6 – LR GRAMMARS AND PARSING – AN OVERVIEW

Grammars

- Context free grammars are a general class of formal grammars, whose rules can be applied regardless of context
  - In the context of compiler design, we are interested in grammars which are deterministic
  - For the project, you have used an LL(1) grammar
  - There exist grammars which are deterministic, which can not be parsed by LL(k) parsers
    - These can however be parsed by LR(k) Parsers

Grammars - LR(1)

- **L**eft to Right
  - Traversing input from left to right
- **R**ightmost parse derivation
  - In the derivation, the leftmost terminals are reduced to first
- **1** lookahead token
  - A unique shift/reduce action can be selected and applied, by looking at the next (only **1**) terminal symbol

- LR(*k*) grammars can exist for $k \geq 0$, and all are deterministic
- They are a proper subset of context free grammars, and a proper superset of LR(k) grammars
- All deterministic context free grammars can be LR(k)

How LR(k) grammars work

- LR(k) parsing works by waiting until a RHS is complete before committing to a production, without requiring grammar transformations to do so.
  - LL(k) parsers must choose a production when they see its leftmost symbol.
  - Many of the ambiguities in the project's language can be handle by LR(k) parsers

- They do so with two main operations
  - **Shift**: Proceed to the next input symbol
  - **Reduce**: Reduce a set of symbols into a single symbol, using a grammar rule

  - An example . . .

LR(k) grammars - types

- All of the following operate using the same grammar, and are more expressive than LL(k) grammars
  - They differ in the algorithms they use
- Simple LR: **SLR**
  - Simple LR parser -> small memory requirement
  - Fails on Shift/Reduce and Reduce/Reduce conflicts
- Look Ahead LR: **LALR**
  - A simplified CLR parser, often obtained by merging similar states
    - Smaller
    - May introduce reduce/reduce conflicts
- Canonical LR: **CLR**
  - Can parse any deterministic CFG
    - No conflicts
  - Large memory requirement (often exponential)
    - Recent advancements somewhat negate this
      so-called Minimal LR(1) parsers have memory requirements comparable to the above parsers