CSE 401/M501 – Compilers

Section 2: Project Infrastructure
Nate Yazdani & Aaron Johnston
Spring 2018
Welcome

• The guy talking is Nate
  – O.H. 2-3pm Mon./Fri. in CSE 220

• The other fellow is Aaron
  – O.H. 1:30-2:30pm Tue. in CSE 021 and 12-1pm Thurs. in CSE 220

• We’ll be leading most sections this quarter
Agenda

• Quick refresher on git revision control  
  – See handouts/references on website for more
• Walk through the starter code
• Practice with ambiguity of formal grammars
Git Review – SSH Keys

• An SSH key lets a git server remember a specific client computer
• If git asks for a password to push or pull, you need to setup an SSH key
• Typically just need to do the following:
  – ssh-keygen -t rsa -C "you@cs.washington.edu" -b 4096
  – Copy ~/.ssh/id_rsa.pub into your GitLab account
• Full setup and troubleshooting instructions: https://gitlab.cs.washington.edu/help/ssh/README
Git Review – Revision Control

- The “official” repo (a.k.a., the remote) lives on the CSE GitLab server
- **Cloning** a repo gives you a private, local copy
- **Committing** saves local changes into the local repo’s revision history
- **Push** to send local commits to remote repo
- **Pull** to bring remote commits to local repo
- Beware **merge conflicts** – pull frequently
Git Review – The Team Repository

• Each project pair is given a repo to collaborate
  – Starts out empty, unlike CSE 331, 333, etc.
  – Tagging is how you submit project phases, like CSE 331

• One person from each pair should download the starter code and push it to the shared repo
  – Then the other person pulls to get the starter code
MiniJava Project – Getting Started

• On course website, go to “Compiler project” → “Starter code” (at top) to grab starter code
  – Or just pull your team repo, if already pushed 😊
• One person from each pair should download the starter code and push it to their team’s repo
  – Then the other person pulls to get the starter code
• Everybody have a local copy of the starter code?
MiniJava Project – Walk Through

Together, we’re going to do the following:
1. Unarchive starter code and push to repo*
2. Try out the demo scanner
3. Get to know the CUP/JFlex infrastructure
4. Run a main program as in the scanner phase
5. Try making some changes to lexical spec.

* if applicable
Ambiguity of a Formal Grammar

• Recall from lecture:
  – A formal grammar is *ambiguous* when a sentence in the language has multiple leftmost (or rightmost) derivations (*i.e.*, multiple parse trees).

• Now some exercises selected from a past exam...
Ambiguity – 4.a (15wi midterm)

Question 4. Context-free grammars (14 points) Consider the following syntax for expressions involving addition and field selection:

\[
\begin{align*}
expr &::= expr + field \\
expr &::= field \\
field &::= expr \ . \ id \\
field &::= id
\end{align*}
\]

(a) (8 points) Show that this grammar is ambiguous.
Question 4. Context-free grammars (14 points) Consider the following syntax for expressions involving addition and field selection:

\[
\begin{align*}
expr &::= expr + field \\
expr &::= field \\
field &::= expr \cdot id \\
field &::= id \\
\end{align*}
\]

(b) (6 points) Give an unambiguous context-free grammar that fixes the problem(s) with the grammar in part (a) and generates expressions with \texttt{id}, field selection and addition. As in Java, field selection should have higher precedence than addition and both field selection and addition should be left-associative (i.e., \texttt{a+b+c} means \texttt{(a+b)+c}).
Ambiguity – 4.a solution (example)

(a) (8 points) Show that this grammar is ambiguous.

Here are two derivations of id+id.id:
(b) (6 points) Give an unambiguous context-free grammar that fixes the problem(s) with the grammar in part (a) and generates expressions with \texttt{id}, field selection and addition. As in Java, field selection should have higher precedence than addition and both field selection and addition should be left-associative (i.e., \texttt{a+b+c} means \texttt{(a+b)+c}).

The problem is in the first rule for \texttt{field}, which creates an ambiguous precedence. Here is a reasonably simple fix.

```
expr ::= expr + field
expr ::= field
field ::= field . id
field ::= id
```
Ambiguity in Practice

• Comes down to the existence of multiple, legal derivation alternatives for some sentences
  – e.g., do we pick \( expr ::= field \) or \( expr ::= expr + field \)?
• Frequent cause of shift/reduce and reduce/reduce conflicts
• Typically just need to incorporate precedence and/or associativity