PPL 2/19

Reagan Shirk

February 19, 2020

- I really need to fucking pay attention today so I locked myself out of my phone. Here goes nothing.

LL Parsing

- It's been two minutes and I already opened GroupMe on my laptop send help

Removing Left Recursion (important to remove it for some reason)

- This has left recursion:

$$\begin{aligned} \text{id-list} &\rightarrow \text{id} \\ &\mid \text{id-list}, \text{id} \end{aligned}$$

- This fixes the left recursion

$$\begin{aligned} \text{id-list} &\rightarrow \text{id id-list-tail} \\ \text{id-list-tail} &\rightarrow \text{, id id-list-tail} \\ &\mid \text{epsilon} \end{aligned}$$

- You can solve the common prefixes issue by "left-factoring" which is demonstrated as shown:

$$\text{Example: stmt} \rightarrow \text{id} := \text{id (arg-list)}$$

$$\begin{aligned} \text{Fixed: stmt} &\rightarrow \text{id id-stmt-trail} \\ \text{id-stmt-trail} &\rightarrow := \text{expr} \\ &\mid \text{(arg-list)} \end{aligned}$$

- There's also an issue with the "dangling else" that prevents grammars from being LL(1)
  - Dangling else basically means when there's an else that could go to two different if statements and you don't know which one it belongs to
- The example in the slides changes to ambiguous, dangling else grammar into a less natural grammer that can be parsed from bottom up but not top down, that's a lot of LaTex to write so I'll try to remember to do it later
- You can employ explicit end-markers which is a better solution and fixes the issues with both of the previous grammars
- Usually you can use the ambiguous grammar with a *disambiguating rule* that says the else should go with the closest then, or the first two possible productions is the one to predict/reduce
- I still ended up getting distracted sorry friends