HW #3

LING 571

Deep Processing Techniques for NLP

CKY Parsing: Goals

- Complete implementation of CKY parser
- Implement dynamic programming approach
- Incorporate/follow backpointers to recover parse

Implementation

- Build full parser
- You may use existing data structures for rules, trees

  e.g. NLTK has nice **`tree`** data structure

  CKY algorithm must be your own
- Dynamic programming table filling crucial!
- Will use smaller grammar (similar to HW #1)
- Back to ATIS for HW #4

Implementation

- For CKY Implementation:
  - NLTK's **`CFG.productions()`** method:
    - optional `rhs=` argument *only looks at first token of RHS*
    - Be-ware: NOT the entire RHS

Notes

- Teams:

  You may work in teams of two on this assignment

- Test grammar

  Pre-converted to CNF

  Start symbol: **TOP**

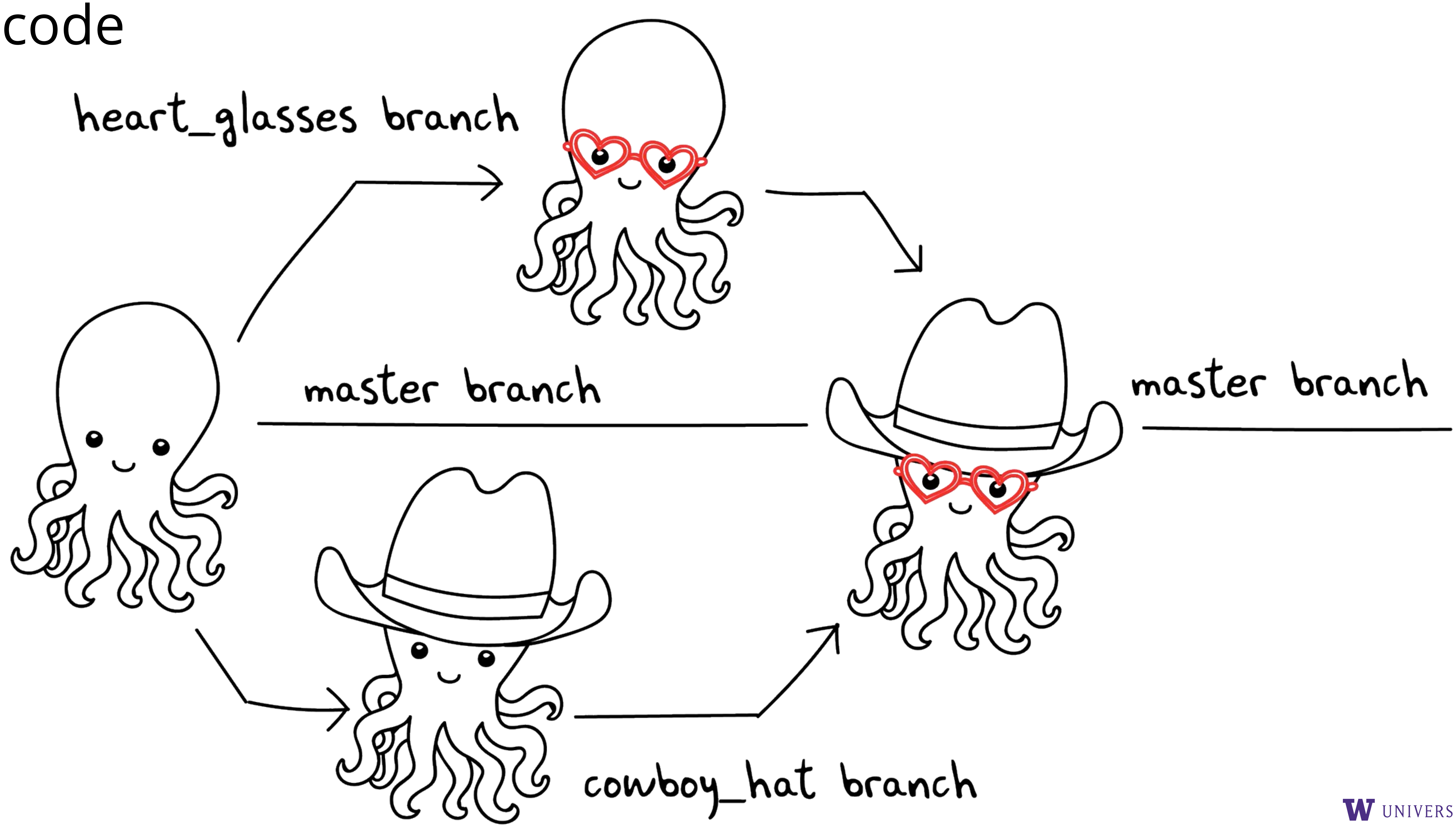
  Parse should span input and be rooted at: **TOP**

- In general:
  - Read grammars with nltk.data.load()
  - Access start symbol with .start()

Some Collaboration Basics

Git Branches

- Good for semi-isolating your development code from the shared, reviewed code



Recommended Git Flow

- Initialize a git repository, with a `main` branch
  - (Create initial commit, if necessary)
- Create a new branch, maybe “`adding_rule_objects`”
- Make regular commits on your branch (like saving)
- Switch to `main` branch, and “pull”
- Merge your branch to main
- …rinse & repeat
- If using GitHub (or GitLab, etc): **MUST BE PRIVATE REPO!**

Communication: Check-ins

- For check-ins, three main points:
  - What have you been working on?
  - What do you plan to work on next?
  - Is there anything “blocking” you?
- In industry, these brief check-ins among small teams are often done daily