HW#6

Goals

- Semantics
 - Gain better understanding of semantic representations
 - Develop experience with lambda calculus and FOL
 - Create semantic attachments
 - Understand semantic composition

Compositional Semantics

• Part 1:

- Manually create target semantic representations
- Use Neo-Davidsonian event representation
 - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

Compositional Semantics

• Part 1:

- *Manually* create target semantic representations
- Use Neo-Davidsonian event representation
 - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

• Part 2:

- Create semantic attachments to reproduce (NLTK)
- Add to grammatical rules to derive sentence representations

Compositional Semantics

• Part 1:

- *Manually* create target semantic representations
- Use Neo-Davidsonian event representation
 - e.g. verb representation with event variable, argument conjuncts
- Can use as test cases for part 2

• Part 2:

- Create semantic attachments to reproduce (NLTK)
- Add to grammatical rules to derive sentence representations
- Note: Lots of ambiguities (scope, etc)
 - Only need to produce one

Semantics in NLTK

- Grammar files:
 - .fcfg extension
 - Example format in <u>NLTK Book Chapter 10</u>
 - /corpora/nltk/nltk-data/grammars/book_grammars/simple-sem.fcfg
 - Note: Not "event-style"
- Parsing:
 - Use nltk.parse.FeatureChartParser (or similar)

Semantics in NLTK

Printing semantic representations:

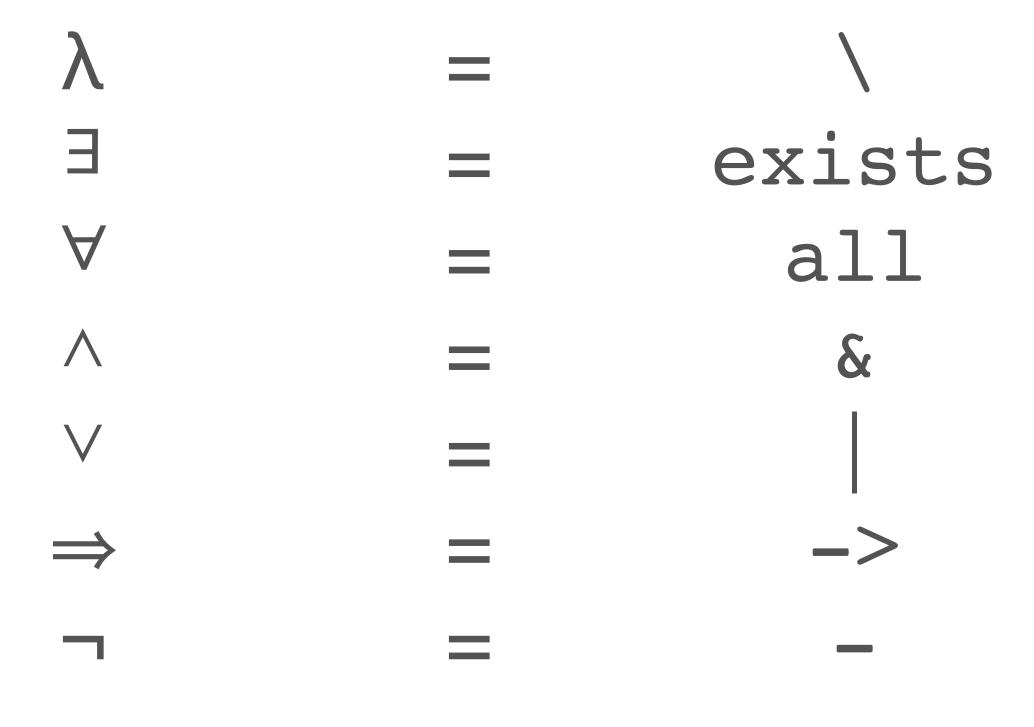
```
item.label()['SEM'].simplify()
all x.(dog(x) -> exists e.(barking(e) & barker(e,x)))
```

• Also nltk.sem.util.root_semrep(item)

Semantic attachments in NLTK:

- a,b,e,x
 - lowercase variables can be arguments:
 - $\bullet \ \ x.dog(x)$

- P,Q,X
 - uppercase lambda variables are functors
 - \P.P(john)



More NLTK Logic Format

- Added to typical CFG rules
 - Composing semantics:
 - $S[SEM=<?np(?vp)>] \rightarrow NP[SEM=?np] VP[SEM=?vp]$

More NLTK Logic Format

- Added to typical CFG rules
 - Composing semantics:
 - S[SEM=<?np(?vp)>] -> NP[SEM=?np] VP[SEM=?vp]
- Creating lambdas:
 - IV[SEM=<\x.exists e.(barking(e) & barker(e,x))>] -> 'barks'

More NLTK Logic Format

- Added to typical CFG rules
 - Composing semantics:
 - S[SEM=<?np(?vp)>] -> NP[SEM=?np] VP[SEM=?vp]
- Creating lambdas:
 - IV[SEM=<\x.exists e.(barking(e) & barker(e,x))>] -> 'barks'
- Nested lambdas:
 - \x.\y. Etc → \x y.
 Can remove '.' between sequences of lambda elements
 Keep '.' between sections: lambdas, quantifiers, body

