Semantics

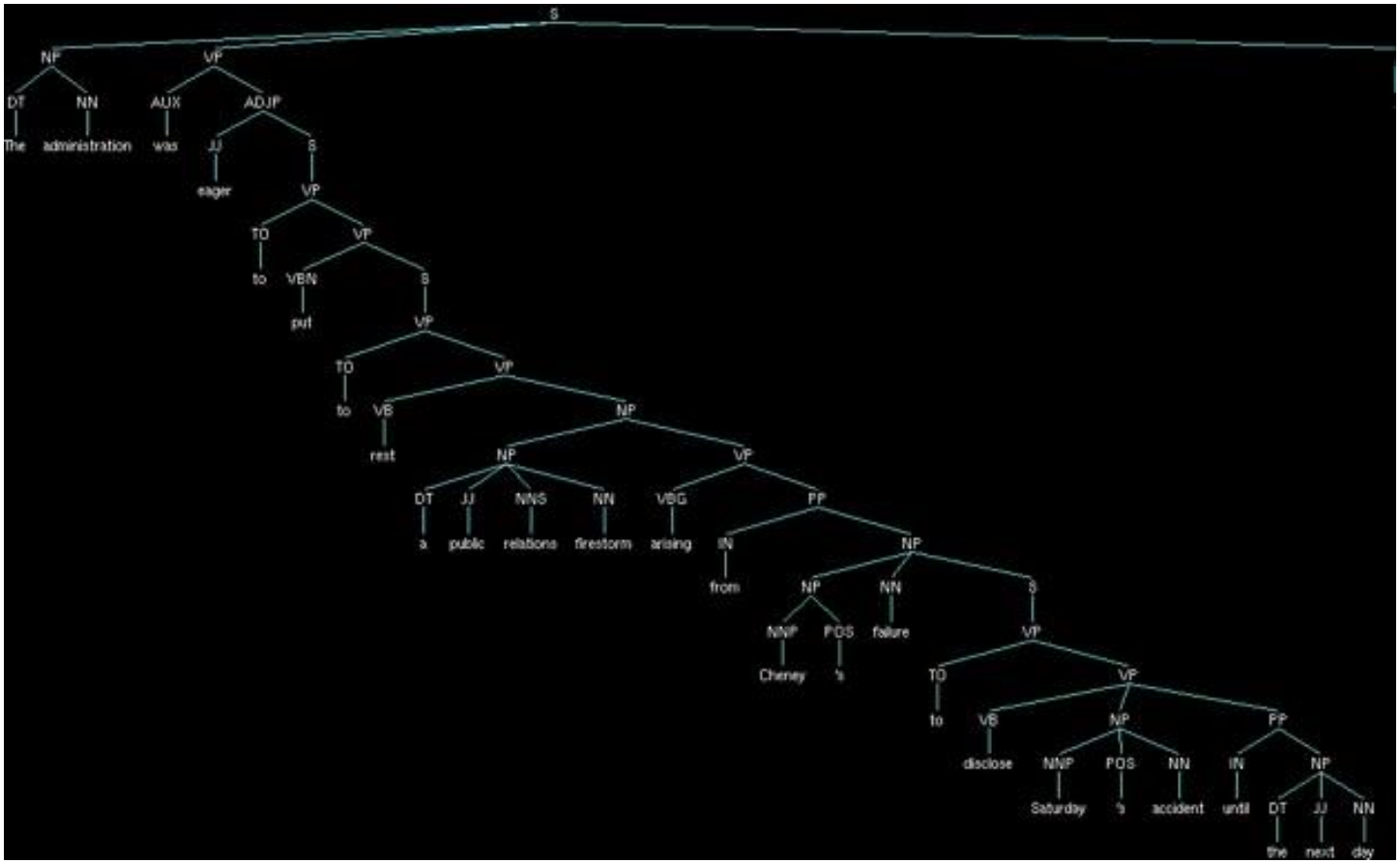
Slides by James Martin, adapted by Diana Inkpen
for CSI 5386 @ uOttawa

Transition

- First we did words (morphology)
- Then simple sequences of words
- Then we looked at syntax
- Now we're moving on to meaning
  - Where some would say we should have started to begin with.

Example

Even if this is the right tree, what does that tell us about the meaning?



Meaning Representations

- We're going to take the same basic approach to meaning that we took to syntax and morphology
- We're going to create representations of linguistic inputs that capture the meanings of those inputs.
- But unlike parse trees, these representations aren't primarily descriptions of the structure of the inputs…

Meaning Representations

- In most cases, they are simultaneously representations of the meanings of utterances and representations of some potential state of affairs in some world.

Meaning Representations

- What could this mean…
  - representations of linguistic inputs that capture the meanings of those inputs
- For us it means
  - Representations that permit or facilitate semantic processing

Semantic Processing

- Ok, so what does that mean?
- Representations that
  - Permit us to reason about their truth (i.e., their relationship to some world)
  - Permit us to answer questions based on their content
  - Permit us to perform inference (answer questions and determine the truth of things we don't already know to be true)

Semantic Processing

- Touchstone application is often question answering
  - Can a machine answer questions involving the meaning of some text or discourse?
  - What kind of representations do we need to mechanize that process?

Semantic Processing

- We're going to discuss 2 ways to attack this problem (just as we did with parsing)
  - There's the principled, theoretically motivated approach…
    - Computational/Compositional Semantics
      - Chapters 17 and 18
  - And there are limited, practical approaches that have some hope of actually being useful
    - Information extraction
      - Chapter 22

Semantic Analysis

- Compositional Analysis
  - Create a FOL representation that accounts for all the entities, roles and relations present in a sentence.
    - Similar to our approach to full parsing
- Information Extraction
  - Do a superficial analysis that pulls out only the entities, relations and roles that are of interest to the consuming application.
    - Similar to chunking

Information Extraction (preview)

Investigators worked leads Monday in Riverside County where the car was reported stolen and reviewed security tape from Highway 241 where it was abandoned, said city of Anaheim spokesman John Nicoletti.

Information Extraction
Named Entities

- Investigators worked leads Monday in Riverside County where the car was reported stolen and reviewed security tape from Highway 241 where it was abandoned, said city of Anaheim spokesman John Nicoletti.

Investigators worked leads [Monday] in [Riverside County] where the car was reported stolen and reviewed security tape from [Highway 241] where it was abandoned, said city of [Anaheim] spokesman [John Nicoletti].

Information Extraction Events

- Investigators worked leads Monday in Riverside County where the car was reported stolen and reviewed security tape from Highway 241 where it was abandoned, said city of Anaheim spokesman John Nicoletti.

Investigators worked leads Monday in Riverside County where the car was reported stolen and reviewed security tape from Highway 241 where it was abandoned, said city of Anaheim spokesman John Nicoletti.

Representational Schemes

- We're going to make use of First Order Logic (FOL) as our representational framework
  - Not because we think it's perfect
  - Many of the alternatives turn out to be either too limiting or
  - They turn out to be notational variants

FOL

- Allows for…
  - The analysis of truth conditions
    - Allows us to answer yes/no questions
  - Supports the use of variables
    - Allows us to answer questions through the use of variable binding
  - Supports inference
    - Allows us to answer questions that go beyond what we know explicitly

FOL

- This choice isn't completely arbitrary or driven by the needs of practical applications
- FOL reflects the semantics of natural languages because it was designed that way by human beings
- In particular…

Meaning Structure of Language

- Natural languages convey meaning through the use of
  - Predicate-argument structures
  - Variables
  - Quantifiers
  - A partially compositional semantics
  - And a host of other techniques

Predicate-Argument Structure

- Events, actions and relationships can be captured with representations that consist of predicates and arguments to those predicates.
- Languages display a division of labor where some words and constituents (typically) function as predicates and some as arguments.

Predicate-Argument Structure

- Predicates
  - Primarily Verbs, VPs, Sentences
  - Sometimes Nouns and NPs
- Arguments
  - Primarily Nouns, Nominals, NPs, PPs
  - But also everything else; as we'll see it depends on the context

Example

- *Mary gave a list to John.*
- Giving(Mary, John, List)
- More precisely
  - *Gave* conveys a three-argument predicate
  - The first argument is the subject
  - The second is the recipient, which is conveyed by the NP inside the PP
  - The third argument is the thing given, conveyed by the direct object

Note

- *Giving(Mary, John, List)* is pretty the same as
  - Subj(Giving, Mary), Obj(Giving, John), IndObj(Giving, List)
  - Which should look an awful lot like.... what?

Better

- Turns out this representation isn't quite as useful as it could be.
- Better would be

$$\$e,y\ Giving(e) \wedge Giver(e,Mary) \wedge Given(e,y) \wedge Givee(e,John) \wedge List(y)$$

Predicates

- The notion of a predicate just got more complicated…
- In this example, think of the verb/VP providing a template like the following

$$\$e,x,y,z Giving(e) \wedge Giver(e,x) \wedge Given(e,y) \wedge Givee(e,z)$$

- The semantics of the NPs and the PPs in the sentence plug into the slots provided in the template

Two Issues

- How can we create this kind of representation in a principled way
- What makes that representation a "meaning" representation, as opposed say to a parse tree?

Semantic Analysis

- Semantic analysis is the process of taking in some linguistic input and assigning a meaning representation to it.
  - There a lot of different ways to do this that make more or less (or no) use of syntax
  - We're going to start with the idea that syntax does matter
    - The compositional rule-to-rule approach

Compositional Analysis

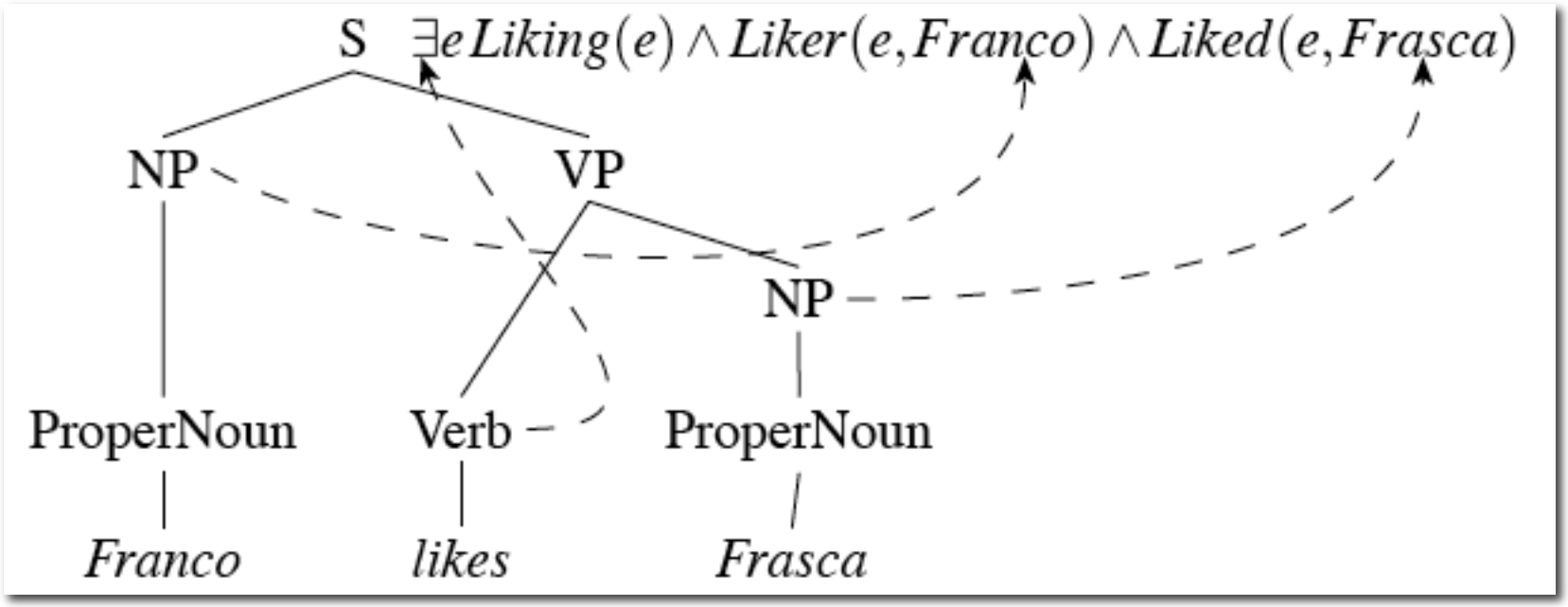
- Principle of Compositionality
  - The meaning of a whole is derived from the meanings of the parts
- What parts?
  - The constituents of the syntactic parse of the input
- What could it mean for a part to have a meaning?

Example

- *Franco likes Frasca.*

$$\exists e Liking(e) \wedge Liker(e, Franco) \wedge Liked(e, Frasca)$$

Compositional Analysis



Augmented Rules

- We'll accomplish this by attaching semantic formation rules to our syntactic CFG rules
- Abstractly

$$A \rightarrow a_1 \ldots a_n \quad \{f(a_1.sem, \ldots a_n.sem)\}$$

- This should be read as the semantics we attach to A can be computed from some function applied to the semantics of A's parts.

Example

- Easy parts…
  - NP -> PropNoun
  - PropNoun -> Frasca
  - PropNoun -> Franco
- Attachments

  {PropNoun.sem}

  {Frasca}

  {Franco}

Example

- S -> NP VP
- VP -> Verb NP
- Verb -> serves

- {VP.sem(NP.sem)}
- {Verb.sem(NP.sem)
- ???

$$\lambda x \lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, x)$$

Lambda Forms

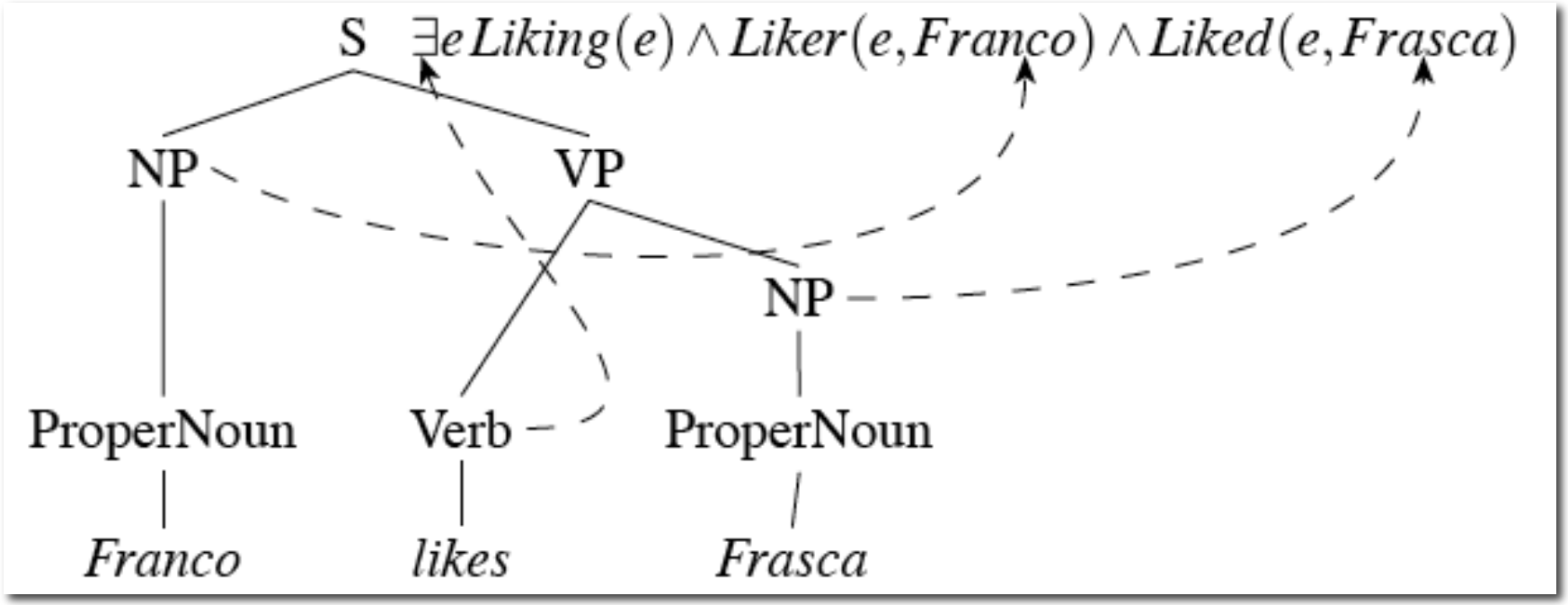
- A simple addition to FOL
  - Take a FOL sentence with variables in it that are to be bound.
  - Allow those variables to be bound by treating the lambda form as a function with formal arguments

$$\lambda x P(x)$$

$$\lambda x P(x)(Sally)$$

$$P(Sally)$$

Compositional Semantics by Lambda Application



Lambda Applications and Reductions

$$\lambda x \lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, x)(Frasca)$$

$$\lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, Frasca)$$

$$\lambda y \exists e Liking(e) \wedge Liker(e, y) \wedge Liked(e, Frasca)(Franco)$$

$$\exists e Liking(e) \wedge Liker(e, Franco) \wedge Liked(e, Frasca)$$

Complications

- Of course, that's the simplest possible example. Making it work for harder cases is more involved...
  - Mismatches between the syntax and semantics
    - Displaced arguments
    - Complex NPs with quantifiers