Ling/CSE 472: Introduction to Computational Linguistics

5/21/12
Unification, parsing with unification
Meaning representation

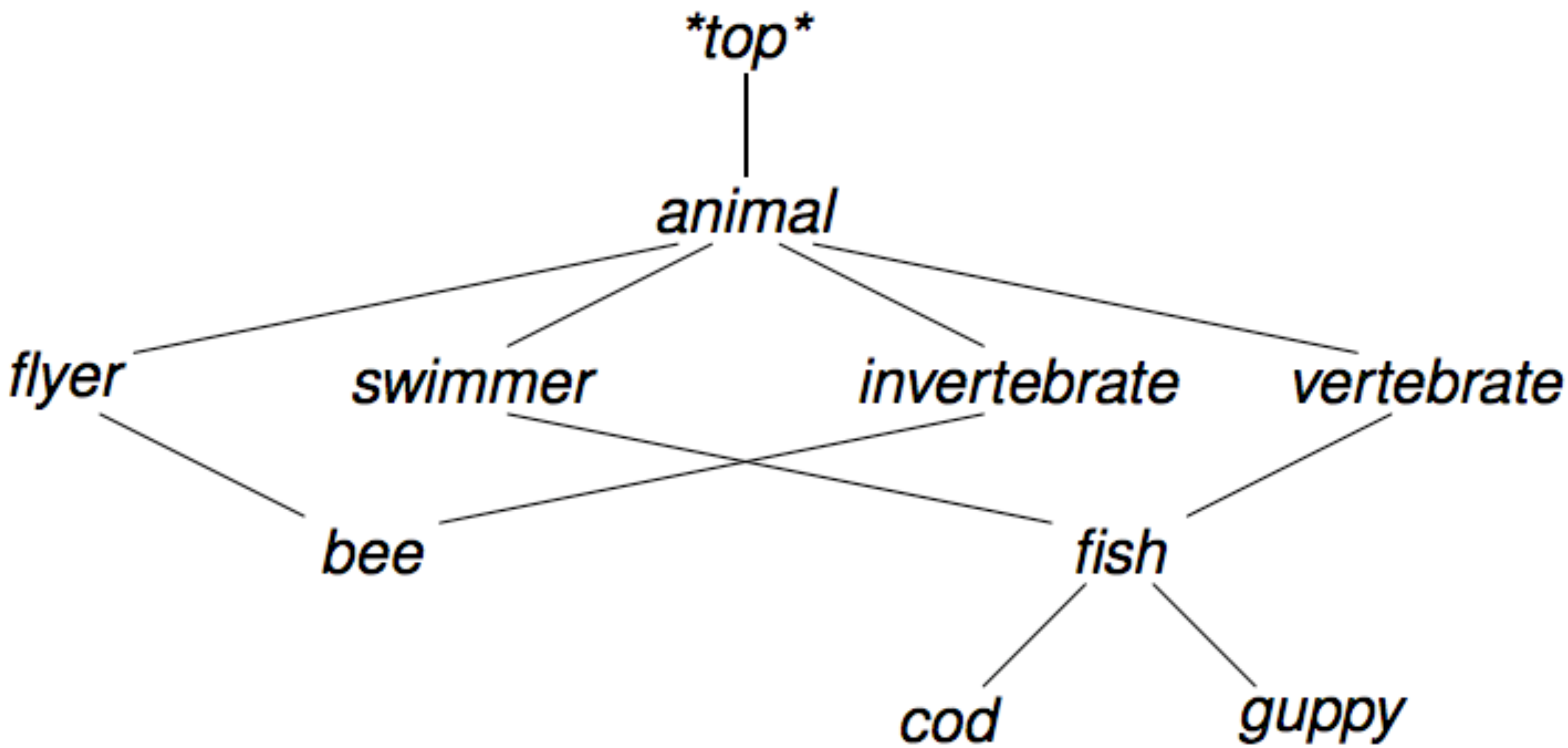
Overview

- Unification
- Unification algorithm
- Parsing with unification
- Representing meaning

Unification

- Input: Two feature structures (or two typed feature structures)
- Output: Failure signal or the unique most general feature structure containing all info from both inputs
  - Two feature structures unify if they contain no contradictory information
  - Two types unify if:
    - They are the same type
    - One is a supertype to the other
    - They share a mutual subtype

Unification of types example
(from Flickinger & Oepen)

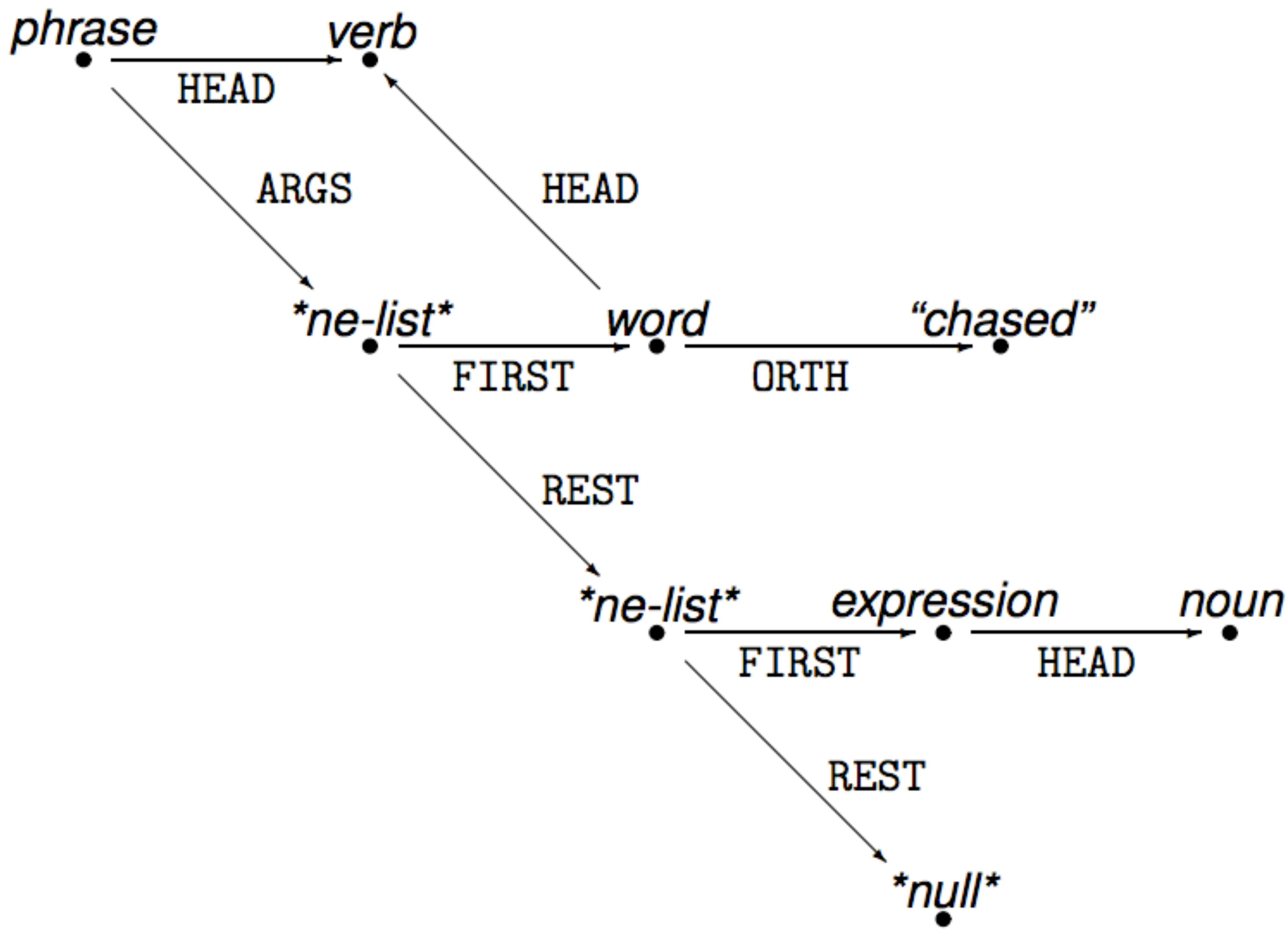


Unification algorithm

- Use graphs to represent feature structures: nodes labeled with types or values, arcs labeled with features.
- Augment these structures with another layer, adding features ‘pointer’ and ‘content’
- Use pointers to merge the graphs representing the two input feature structures (why?)
- Unification is recursive (why?)
- Unification is destructive (why?)

Feature structure as graph example

(from Flickinger & Oepen)



Unification example

$$\begin{bmatrix} \text{A} & \boxed{1}\begin{bmatrix} \text{B} & \text{c} \end{bmatrix} \\ \text{D} & \begin{bmatrix} \text{E} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{D} & \begin{bmatrix} \text{E} & \begin{bmatrix} \text{F} & \text{g} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- Represent each feature structure as a DAG
- Add the ‘pointer’ and ‘contents’ features
- Step through the unification algorithm to produce the result
- How would we have to alter this to handle typed feature structures?

Parsing with unification

- Associate feature structure constraints with rules
- Associate feature structures with edges
- Could just check after CFG parsing is done, but this is inefficient (why?)
- Instead: invoke unification when combining edges (COMPLETER)
- When deciding whether an edge to add is redundant, test is now *subsumption* (rather than identity): don't add edges that are *subsumed* by something already in the chart

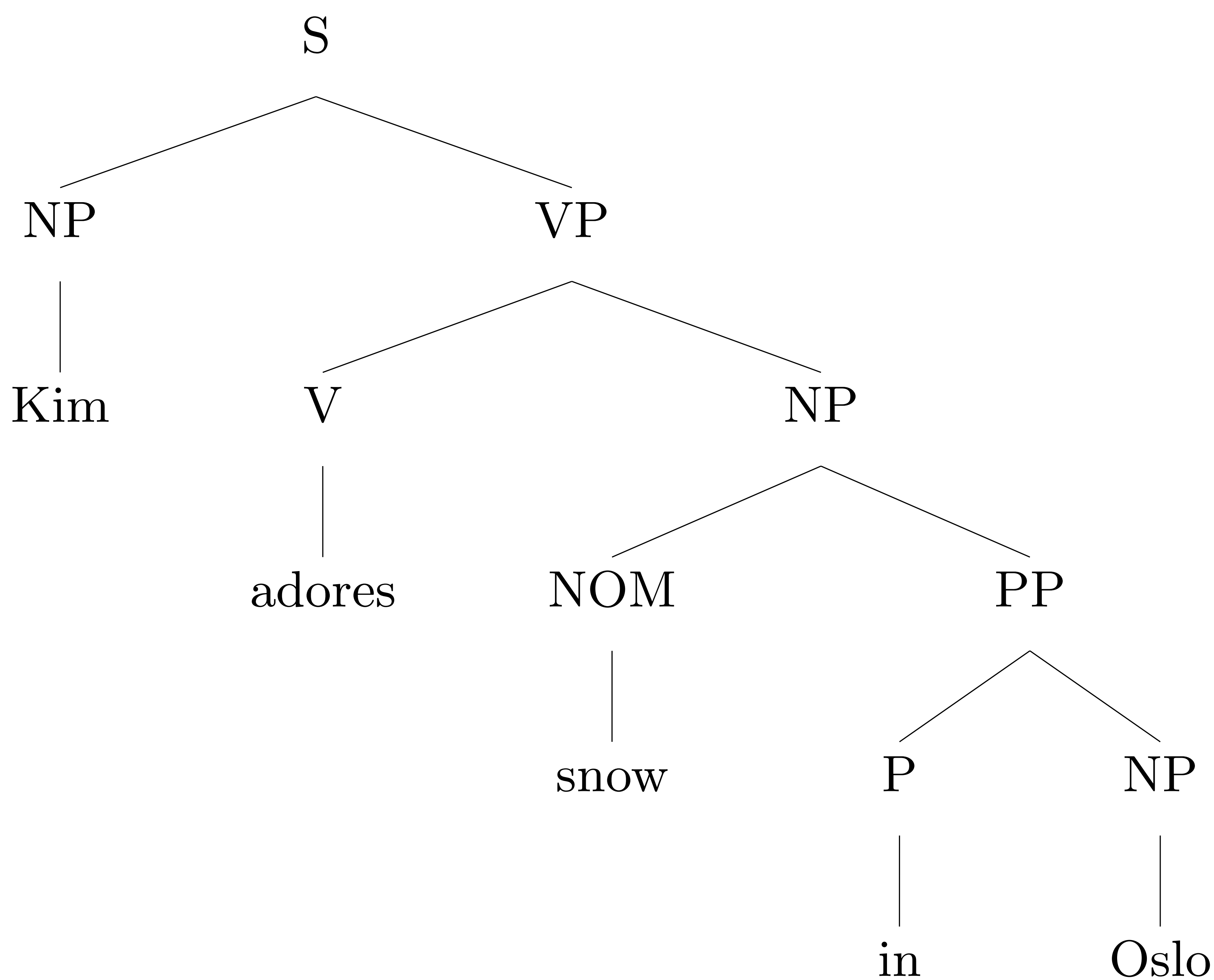
Evaluation slide

- What are we looking for when we evaluate unification algorithms?
- What are we looking for when we evaluate parsing algorithms that use unification?
    - What’s the gold standard?
    - What’s the baseline?
    - What are the metrics?

Overview

- Unification
- Unification algorithm
- Parsing with unification
- Representing meaning

Parsing makes explicit inherent structure. So, does this tree represent meaning?

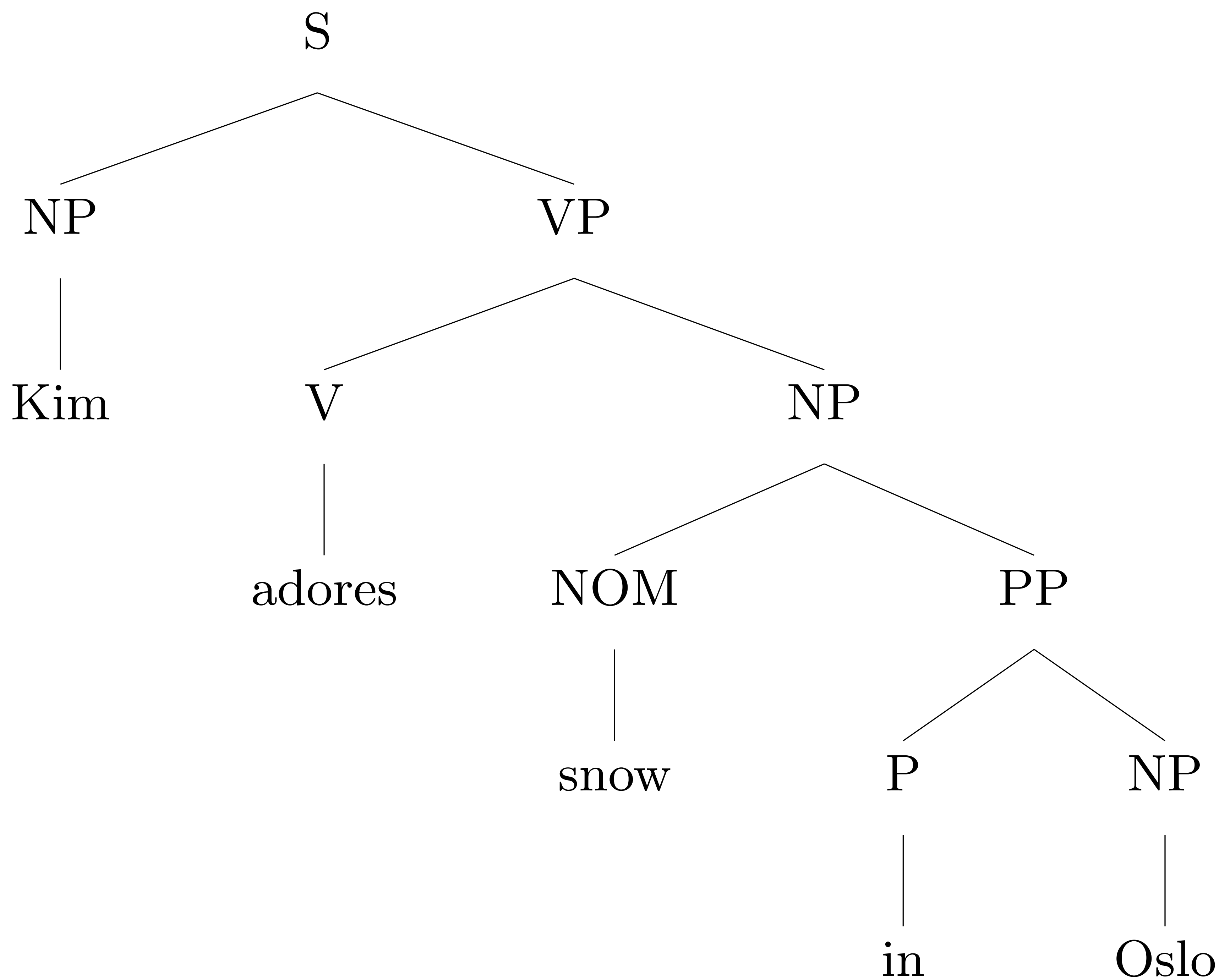


Why represent semantics?

- When “earlier” levels aren’t enough

- Bridge between linguistics and real world items/models

How could we put this tree in correspondence to a model of the world?



Semantics

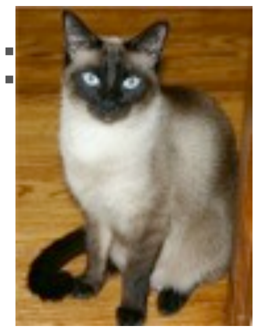
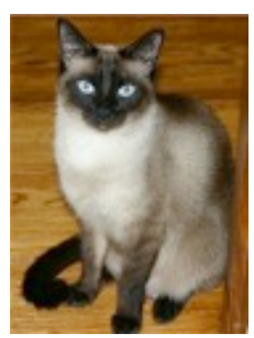
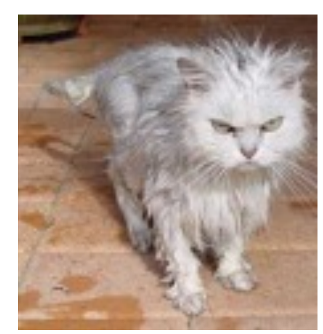
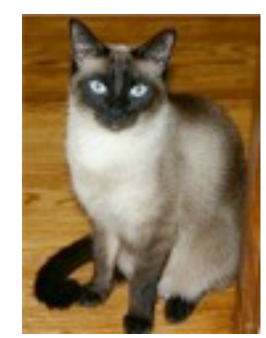
- Create representations which can be put in correspondence with models of the world

- ... and which can be built compositionally via parsing

Basic model-theoretic semantics

- Create a model of the world, consisting of elements, sets of elements and relations
- Create an interpretation function which maps linguistic elements (parts of the semantic structure) to parts of the model
- Simple propositions are interpreted by checking their truth in the model
- Define semantics for “logical vocabulary”: *and, or, not, if, every, some*, ...

Model theoretic semantics example

- Entities: Joey:  Fluffy:  Tiger: 
- Properties: calm: {  ,  }; angry: {  }
- Relations: knows: { <  ,  > , <  ,  > }

Model theoretic semantics example: denotations

- [[Fluffy]] = 
- [[angry]] = { x | x is angry } = {  }
- [[Fluffy is angry]] = True *iff* the entity denoted by *Fluffy* is in the set denoted by *angry*
- Compositionality: The process of determining the truth conditions of *Fluffy is angry* based on the denotations of its parts and its syntactic structure

Logical vocabulary gets special treatment

- *Fluffy is angry and Joey is not angry.*
  - What does *and* mean? (How does it affect the truth conditions of the whole?
  - What does *not* mean?
- *Every cat is angry.*
  - What does *cat* mean? (Is this a logical operator?)
  - What does *every* mean?
- Is the division into logical and non-logical vocabulary an inherent property of language or an artifact of the system of meaning representation?

More on quantifiers

- The semantic type of a quantifier is a relation between sets, called the *restriction* and *body* (or *scope*) of the quantifier

  - [[every]] { <P,Q> | P ⊆ Q}

  - [[every cat is angry]] is True *iff* { x | x is a cat } ⊆ { y | y is angry }

  - [[some]] { <P,Q> | P ∩ Q ≠ ∅}

  - [[some cat is angry]] is True *iff* { x | x is a cat } ∩ { y | y is angry } ≠ ∅

- Where do those sets come from?

Why represent semantics?

- When “earlier” levels aren’t enough
- Bridge between linguistics and real world items/models

Semantics in NLP

- Construct knowledge base or model of the world
- Extract meaning representations from linguistic input
- Match input to world knowledge
- Produce replies/take action on the basis of the results

- In what other cases might semantic representations be useful?

Semantics in NLP

- In what other cases might semantic representations be useful?
  - Transfer-based MT
  - Building a knowledge base by “reading” the web (or wikipedia or...)
  - Generation

Semantic representations: Desiderata (Jurafsky & Martin)

- Verifiability: We must be able to compare the representation to a knowledge base
- Lack of ambiguity: A semantic representation should have just one interpretation
- Canonical form: A given interpretation should have just one representation
  - Does Maharani have vegetarian dishes?
  - Do they have vegetarian food at Maharani?
  - Are vegetarian dishes served at Maharani?
  - Does Maharani have vegetarian fare?
  - But not: Can vegetarians eat at Maharani?
- Expressiveness: Must be able to adequately represent a wide range of expressions

Semantic Representations: Desiderata (Copestake et al 2005)

- Expressive Adequacy: The framework must allow linguistic meanings to be expressed correctly
- Grammatical Compatibility: Semantic representations must be linked clearly to other kinds of grammatical information (most notably syntax)
- Computational Tractability: It must be possible to process meanings and to check semantic equivalence and to express relationships between semantic representations straightforwardly
- Underspecifiability: Semantic representations should allow underspecification (leaving semantic distinctions unresolved), in such a way as to allow flexible, monotonic resolution of such partial semantic representations

Evaluation slide

- How would we evaluate a system of semantic representations?
- How would we evaluate a parsing system which produces semantic representations from input?
    - What’s the gold standard?
    - What’s the baseline?
    - What are the metrics?
    - What else might we need?