Computational Semantics
Day 3: Scope and Underspecification

Aljoscha Burchardt
Alexander Koller
Stephan Walter

ESSLLI 2004, Nancy

Overview

- Scope ambiguities
- Montague's analysis of scope
- Underspecification
- Dominance graphs
- Representing dominance graphs in Prolog

Scope ambiguities

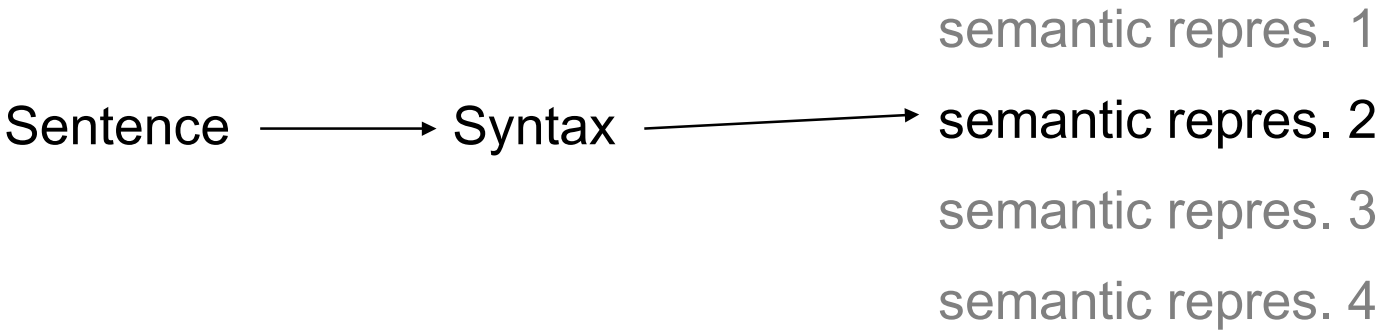
- Some sentences have more than one possible meaning.
- Example:
  "Every swimmer wants a medal."

1. ..., namely, the 200m Freestyle gold medal.
   $\exists y\ \text{medal}(y) \wedge (\forall x\ \text{swimmer}(x) \rightarrow \text{want}(x,y))$

2. ..., but not necessarily the same one.
   $\forall x\ \text{swimmer}(x) \rightarrow (\exists y\ \text{medal}(y) \wedge \text{want}(x,y))$

Semantic ambiguity: The problem

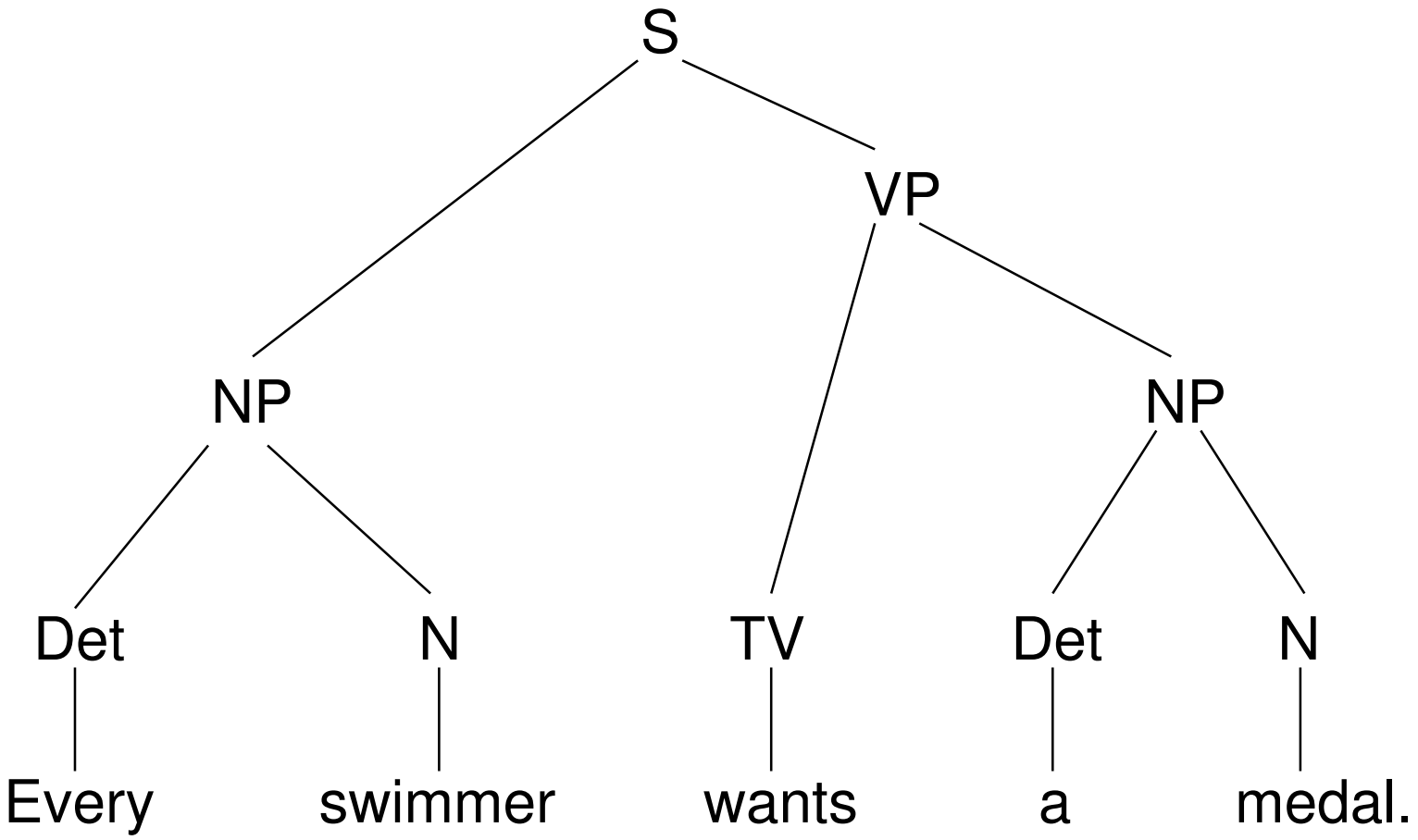
- But: The semantics construction process we have so far is incapable of computing more than one meaning representation for a sentence!



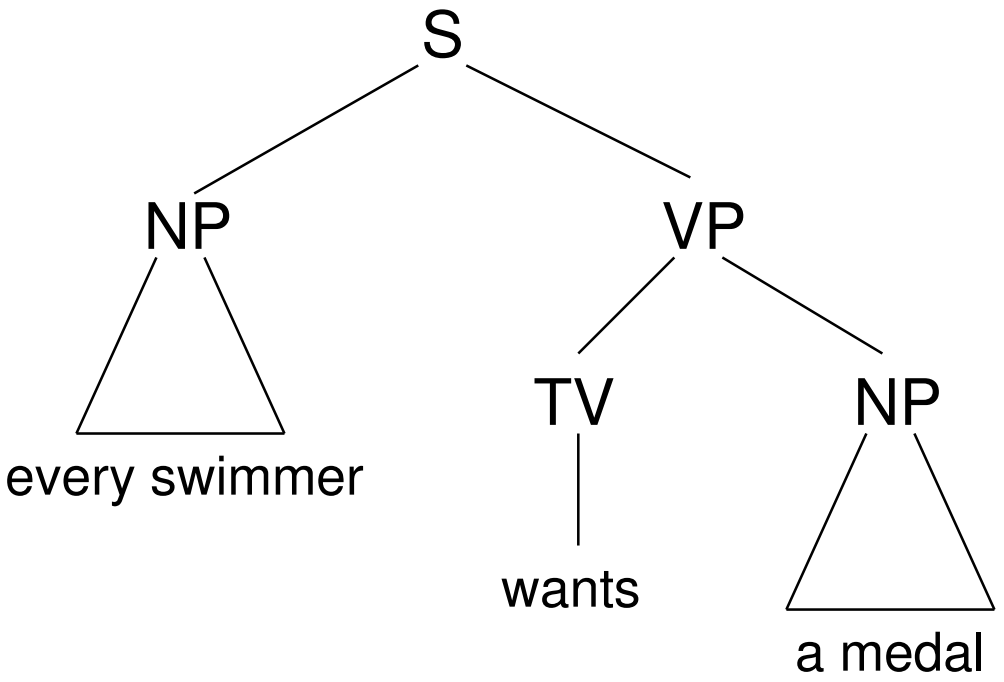
First solution: Montague 1974

- We can only construct one semantic representation per syntactic representation.
- So let's make more syntactic representations!
- New grammar rule ("quantifying-in") allows us to move NPs in the syntax tree.
- Change semantics construction rules so the two semantic representations can be derived from changed trees ("Montague's Trick").

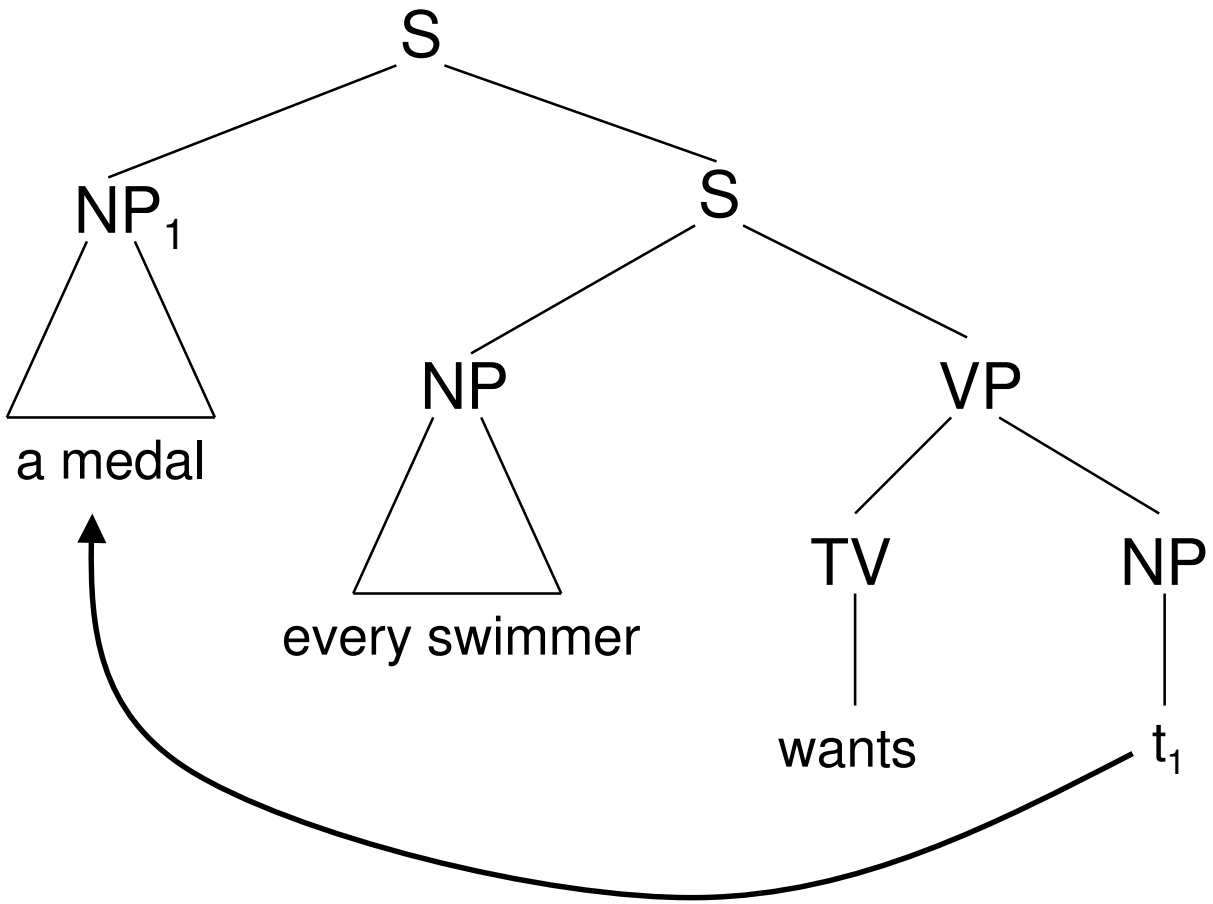
Montague: Basic syntactic analysis



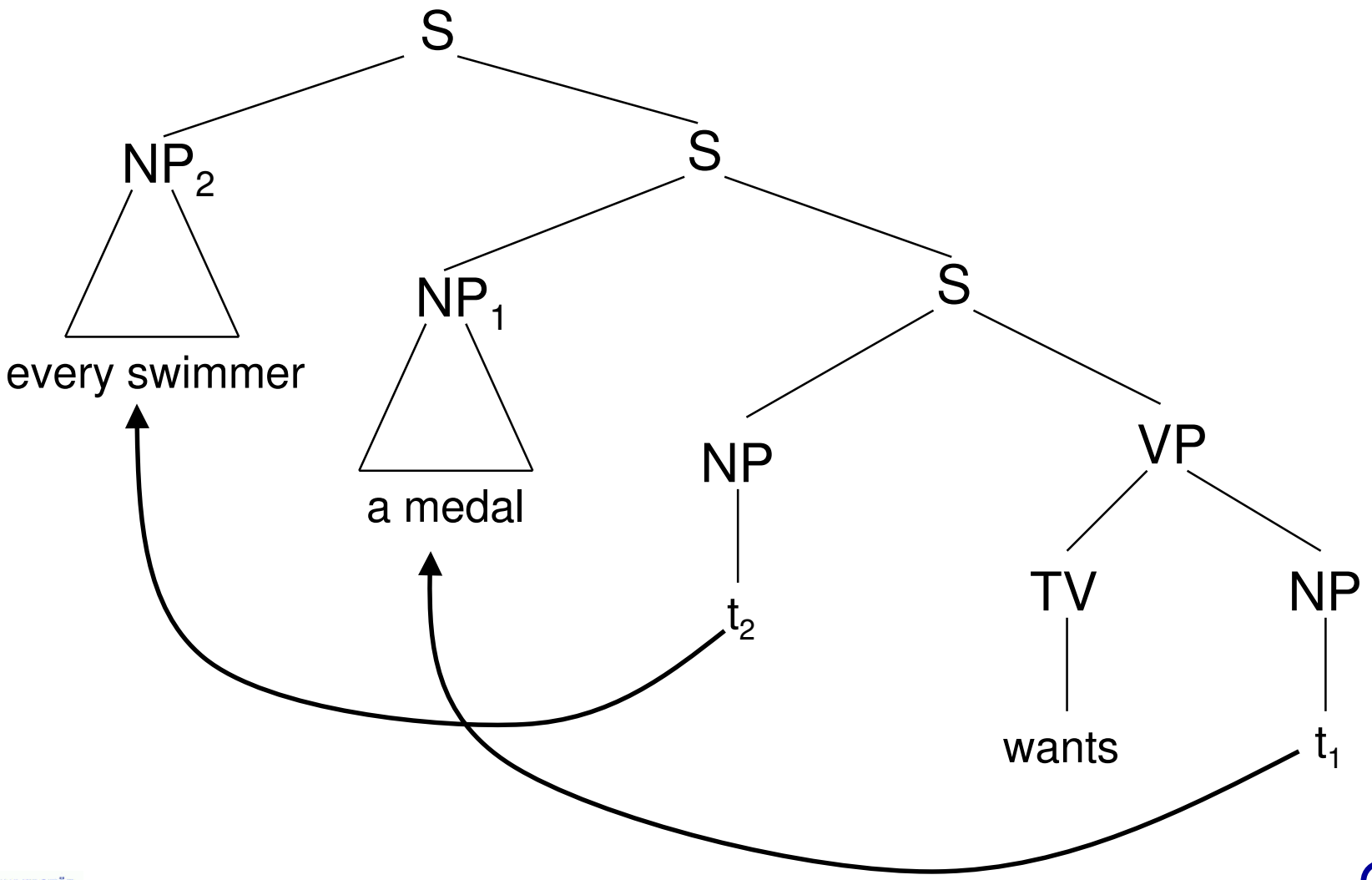
Montague: Quantifying In



Montague: Quantifying In



Montague: Quantifying In



Montague: Semantics Construction

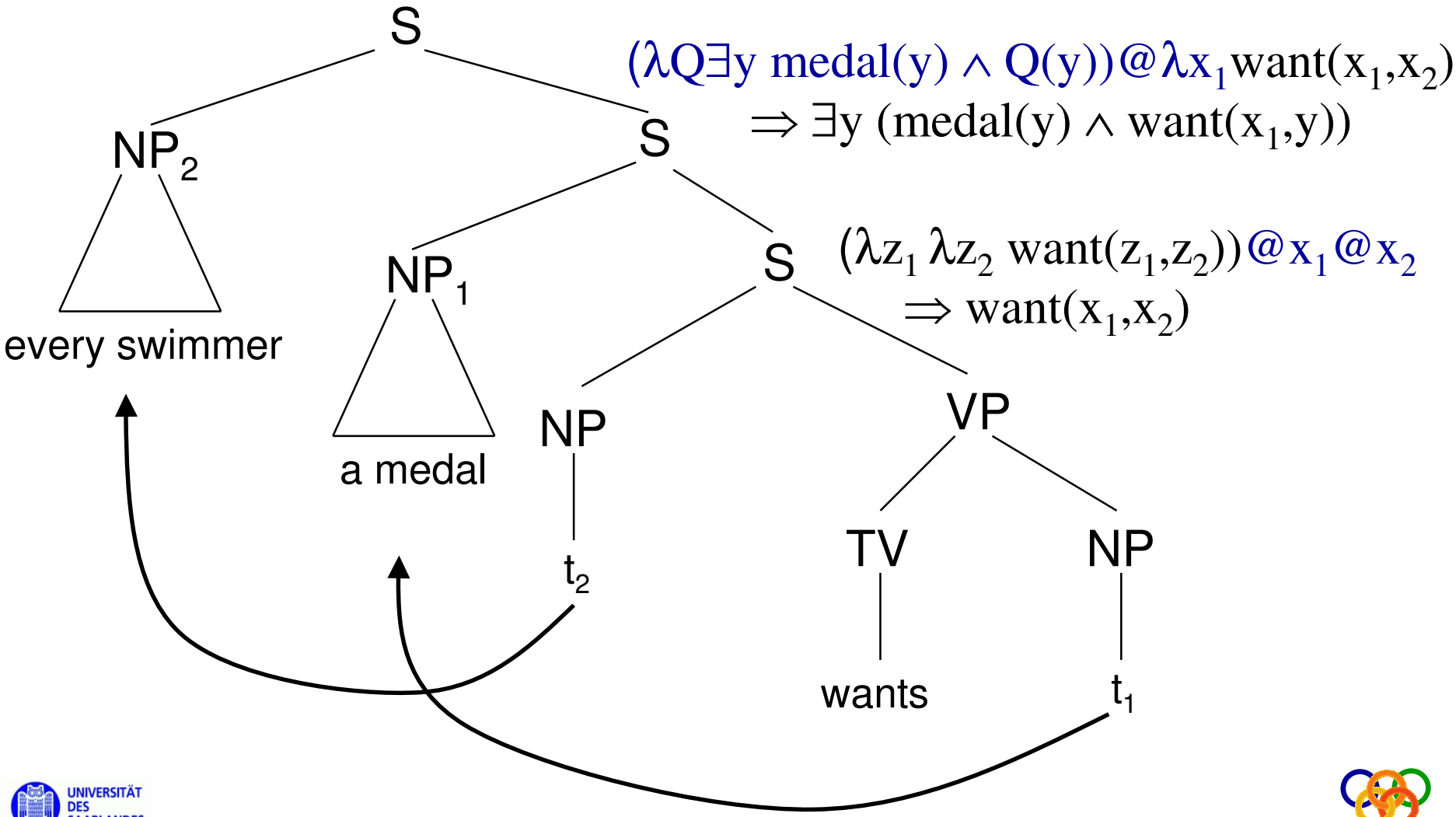
- Semantic representations for NPs as yesterday:
  - "every swimmer":
    $\lambda P \forall x.\text{swimmer}(x) \rightarrow P(x)$
  - "a medal":
    $\lambda Q \exists y.\text{medal}(y) \wedge Q(y)$
- Today, we use a more intuitive semantics for transitive verbs:
  - $\lambda z_1 \lambda z_2 \text{ want}(z_1, z_2)$

Montague's Trick

- The semantic representation of the transitive verb is now exactly the same as it was on Monday.
- The semantic representation for the trace t_i is the free variable x_i.
- Verb semantics is first applied to these variables.
- Then the variables are abstracted over as the NPs are recombines with the sentence.

Montague: Reading 1

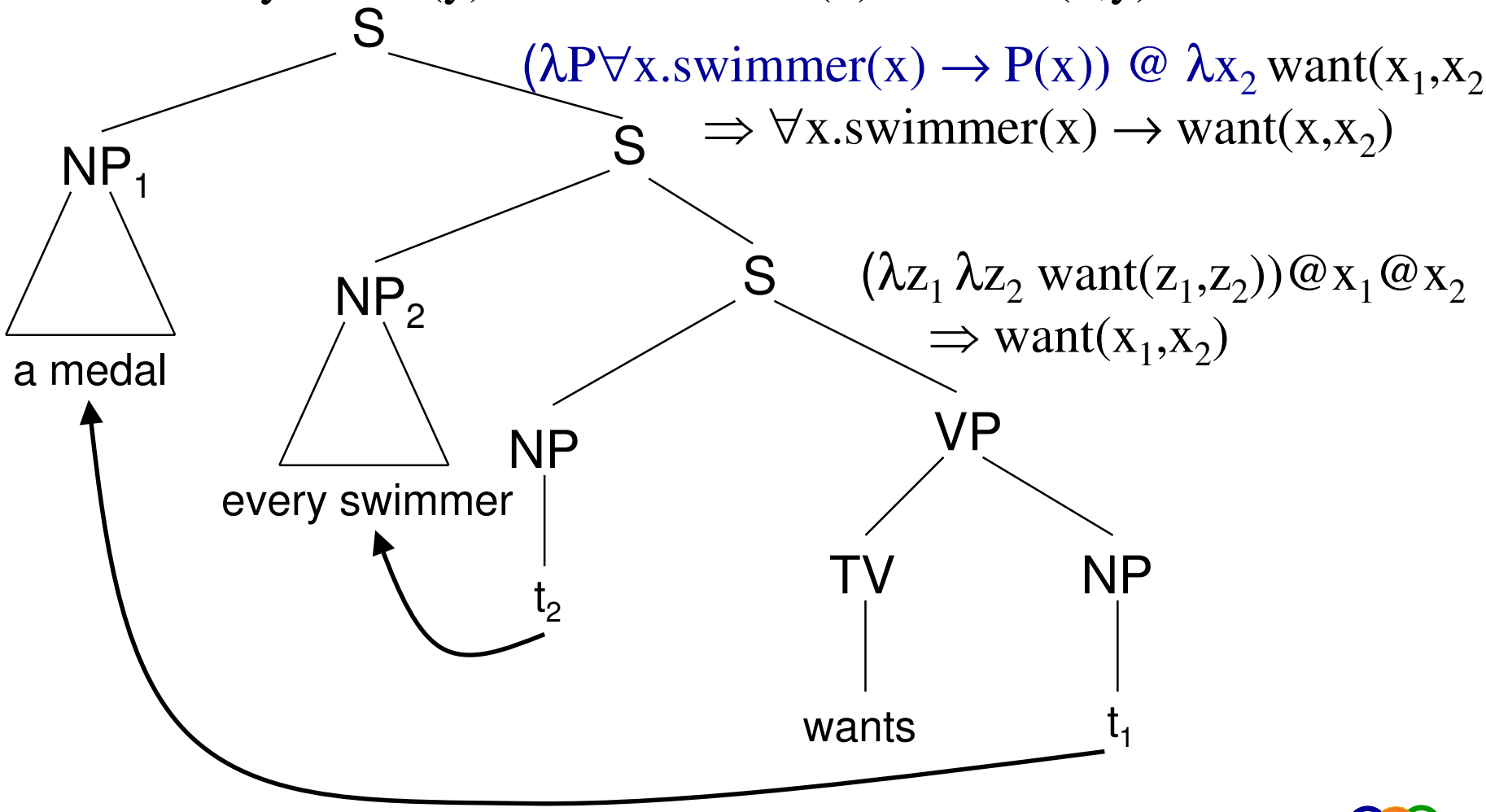
$(\lambda P \forall x.\text{swimmer}(x) \rightarrow P(x)) @ \lambda x_2 \exists y\ (\text{medal}(y) \wedge \text{want}(z_1,y))$
$\Rightarrow \forall x.\text{swimmer}(x) \rightarrow \exists y\ (\text{medal}(y) \wedge \text{want}(x,y))$



Montague: Reading 2

$(\lambda Q \exists y\ medal(y) \wedge Q(y)) @ \lambda x_1 \forall x.swimmer(x) \rightarrow want(x,x_2)$

$\Rightarrow \exists y\ medal(y) \wedge \forall x.swimmer(x) \rightarrow want(x,y)$



$(\lambda P \forall x.swimmer(x) \rightarrow P(x)) @ \lambda x_2\ want(x_1,x_2)$

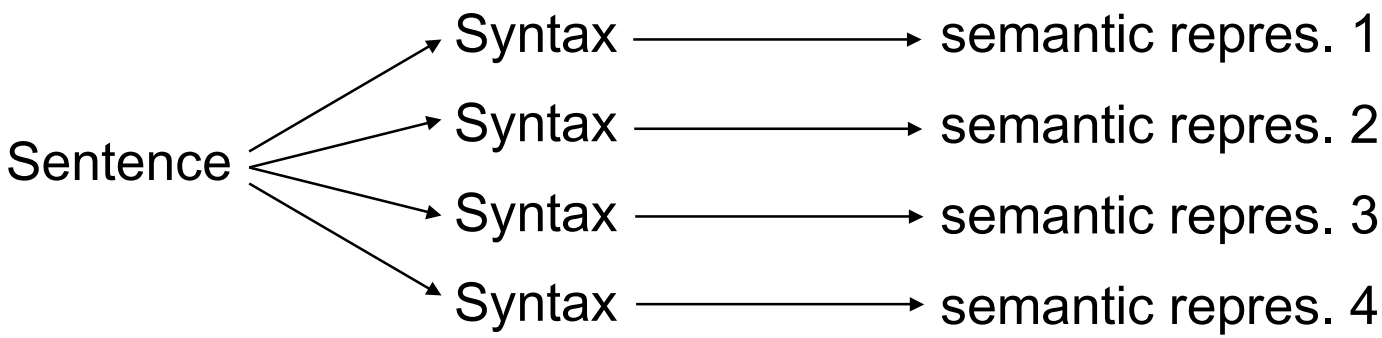
$\Rightarrow \forall x.swimmer(x) \rightarrow want(x,x_2)$

$(\lambda z_1\ \lambda z_2\ want(z_1,z_2)) @ x_1 @ x_2$

$\Rightarrow want(x_1,x_2)$

Montague: The big picture

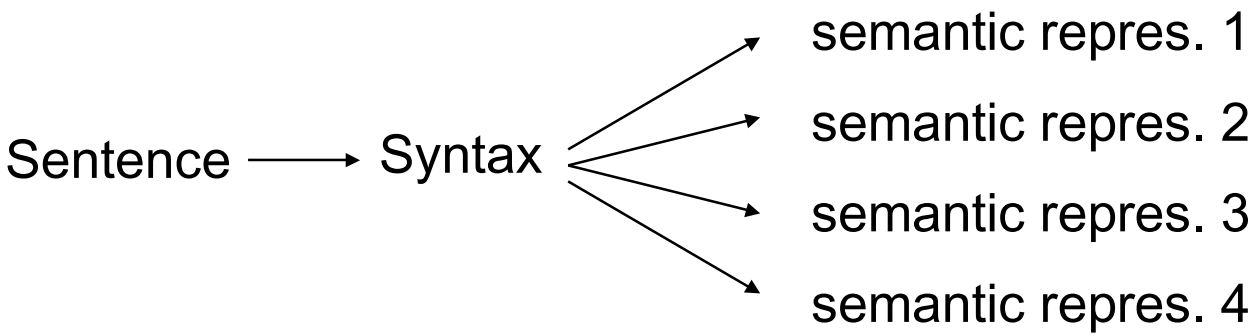
- So: We get the two semantic representations by introducing a syntactic ambiguity and applying Montague's Trick.



Montague: Does it work?

- We can now have more than one semantic representation per sentence.
- But: We have to stipulate that scope ambiguities are syntactic ambiguities. Do we want that?
- Intermediate formulas have free variables that are systematically captured.

Fixing Montague



- Montague can be modified so the semantic representations can be computed from a single syntactic analysis:
  - Cooper 1983 / Keller 1988
  - Hobbs & Shieber 1987

Explosion of Readings

- A sentence with more than one scope ambiguity can have an enormous number of readings:

  Most politicians can fool most voters on most issues most of the time, but no politician can fool every voter on every single issue all of the time.
  (ca. 600 readings, Hobbs)

- Modern large-scale grammars predict a lot of scope readings even for harmless-looking sentences:

  But that would give us all day Tuesday to be there.
  (ca. 65.000 readings, according to ERG grammar)

- In general, scope ambiguities contribute a number of readings exponential in the number of quantifiers (and other scope bearers).

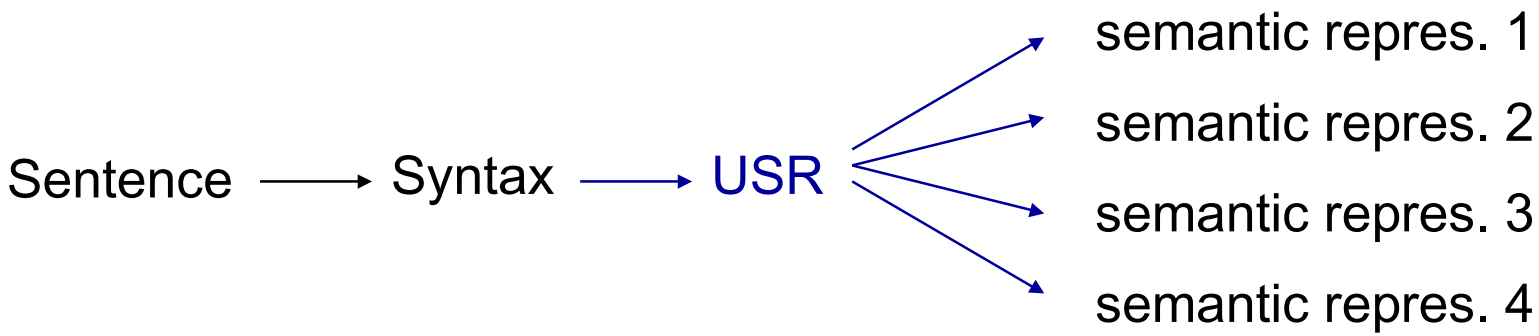
Enumerating readings is expensive

- We'd like to avoid enumerating these many readings.
- Most of the readings were not meant by the speaker.
- Do people enumerate readings of a scope ambiguity?
- With all approaches we have seen so far, we must enumerate all readings before we can do anything else.

Scope Underspecification

- So let's avoid enumerating the readings for as long as we can.
- Take a single syntactic analysis and derive a single underspecified semantic representation from it.
- Possibly perform inferences on underspecified descriptions to remove unwanted readings.
- Then enumerate readings from description by need.

Scope Underspecification: The big picture



Some underspecification formalisms

- QLF (Alshawi & Crouch 92)
- UDRT (Reyle 93)
- Hole Semantics (Bos 96)
- MRS (Copestake et al. 99)
- Dominance constraints/graphs (Egg et al. 98, etc.)

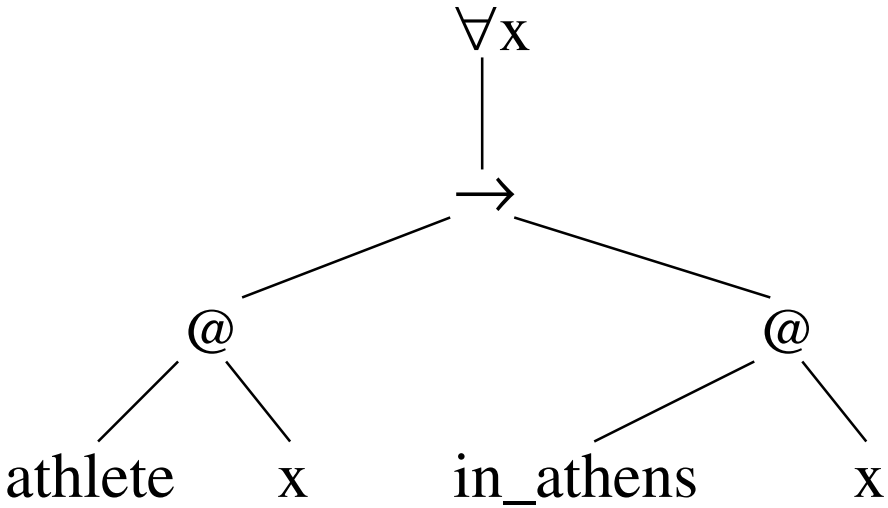
Dominance graphs: The basic idea

- Read FOL formulas (including lambdas) as trees.
- Describe these trees using graphs.
- Use special edges to represent variable binding.

Formulas as trees

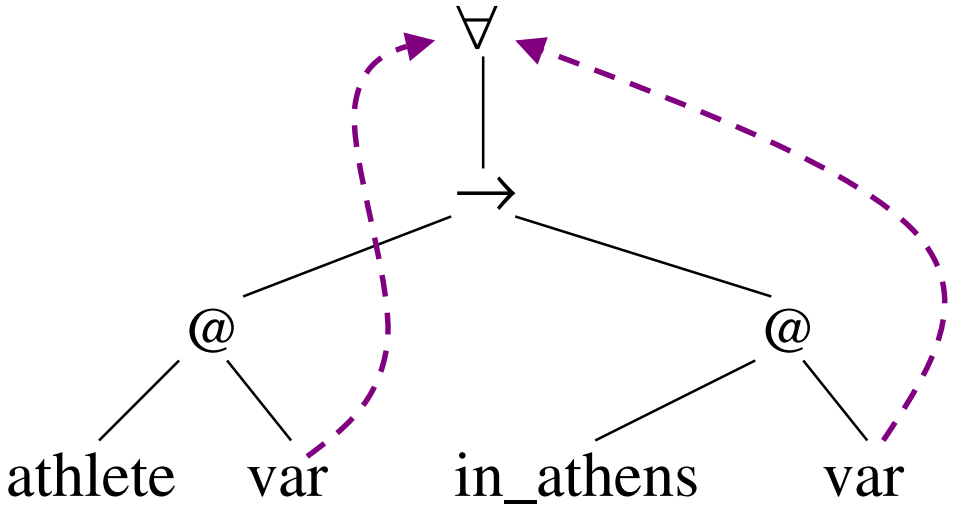
"Every athlete is in Athens."

$\forall x\ \text{athlete}(x) \rightarrow \text{in\_athens}(x)$



Trees + Binding Edges = Lambda Structures

- We use binding edges to indicate variable binding, rather than variable names.



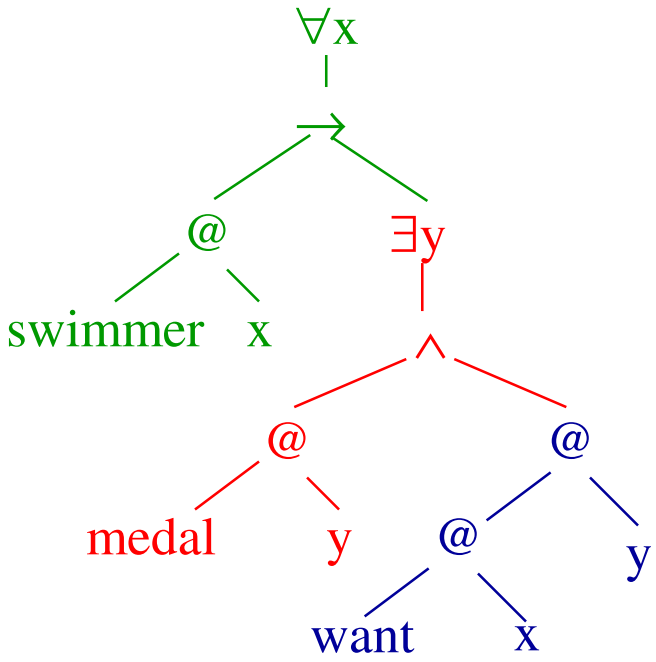
... but we still draw variable names on the slides because they're easier to read.

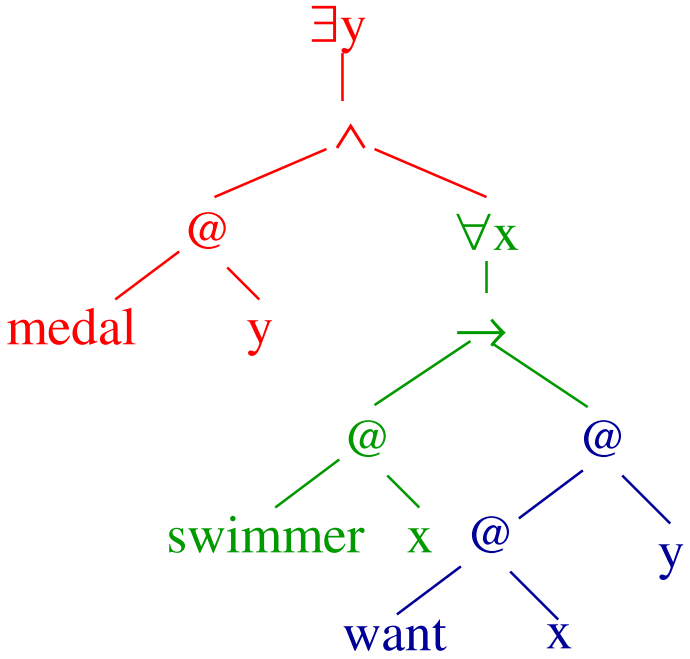
Describe Trees Using Graphs

"Every swimmer wants a medal."

$\exists y\ medal(y) \wedge \forall x.swimmer(x) \rightarrow want(x,y)$

$\forall x.swimmer(x) \rightarrow \exists y\ medal(y) \wedge want(x,y)$



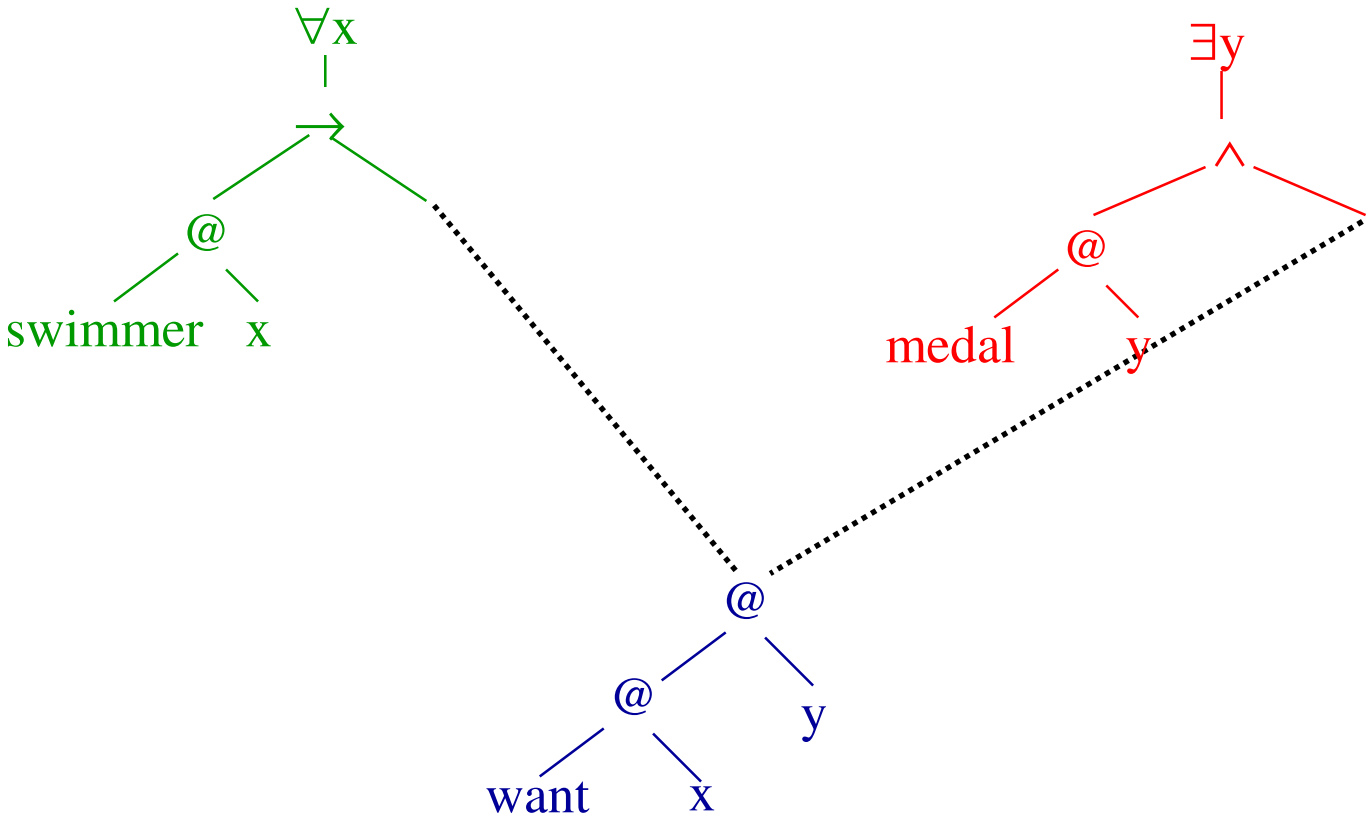


Describe Trees Using Graphs

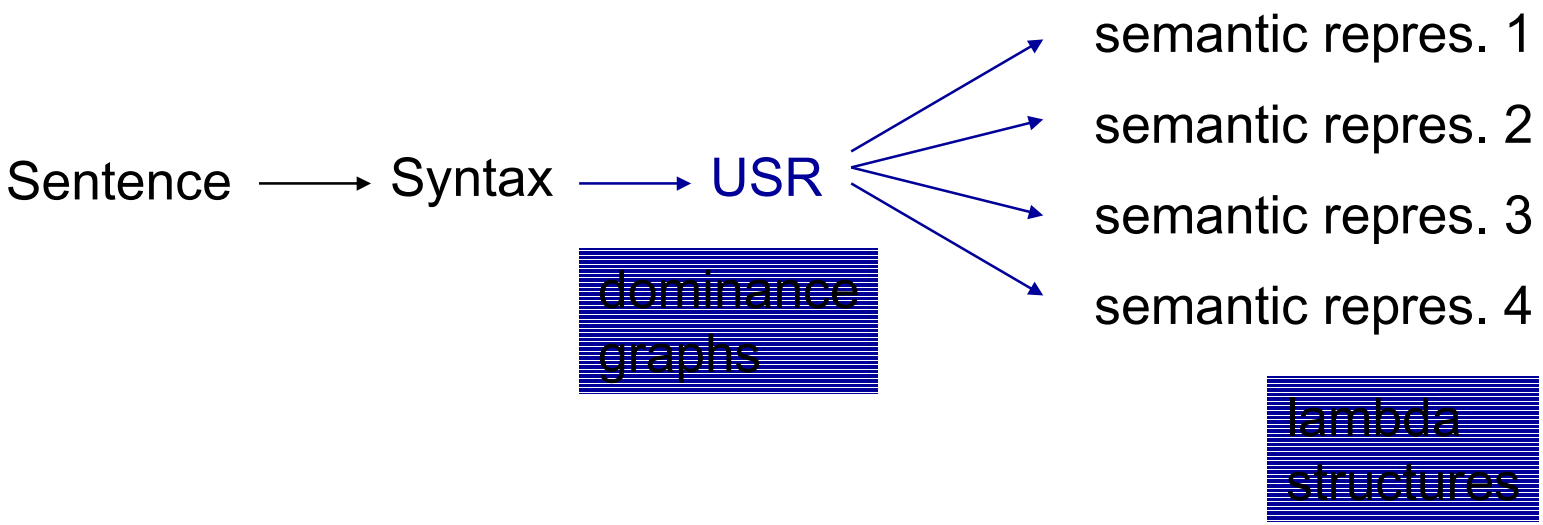
"Every swimmer wants a medal."

$\exists y\ medal(y) \wedge \forall x.swimmer(x) \rightarrow want(x,y)$

$\forall x.swimmer(x) \rightarrow \exists y\ medal(y) \wedge want(x,y)$



Dominance graphs in the big picture



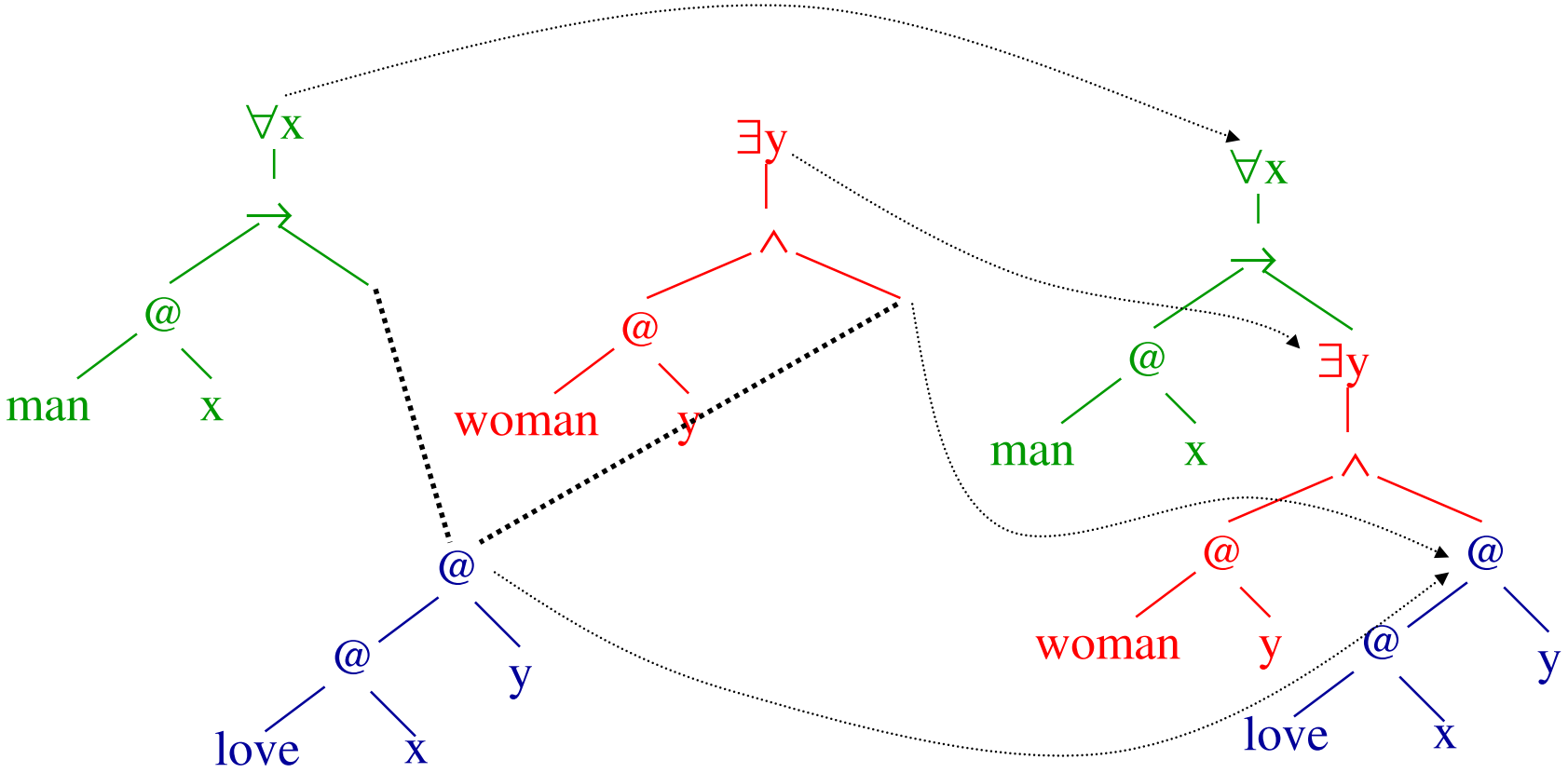
Dominance graphs, more formally

- A dominance graph is a directed graph with node labels.
- There are three kinds of edges:
  - tree edges (solid)
  - dominance edges (dotted)
  - binding edges (dashed arrows)
- Tree edges form a collection of trees.
- Dominance edges go from holes to roots.

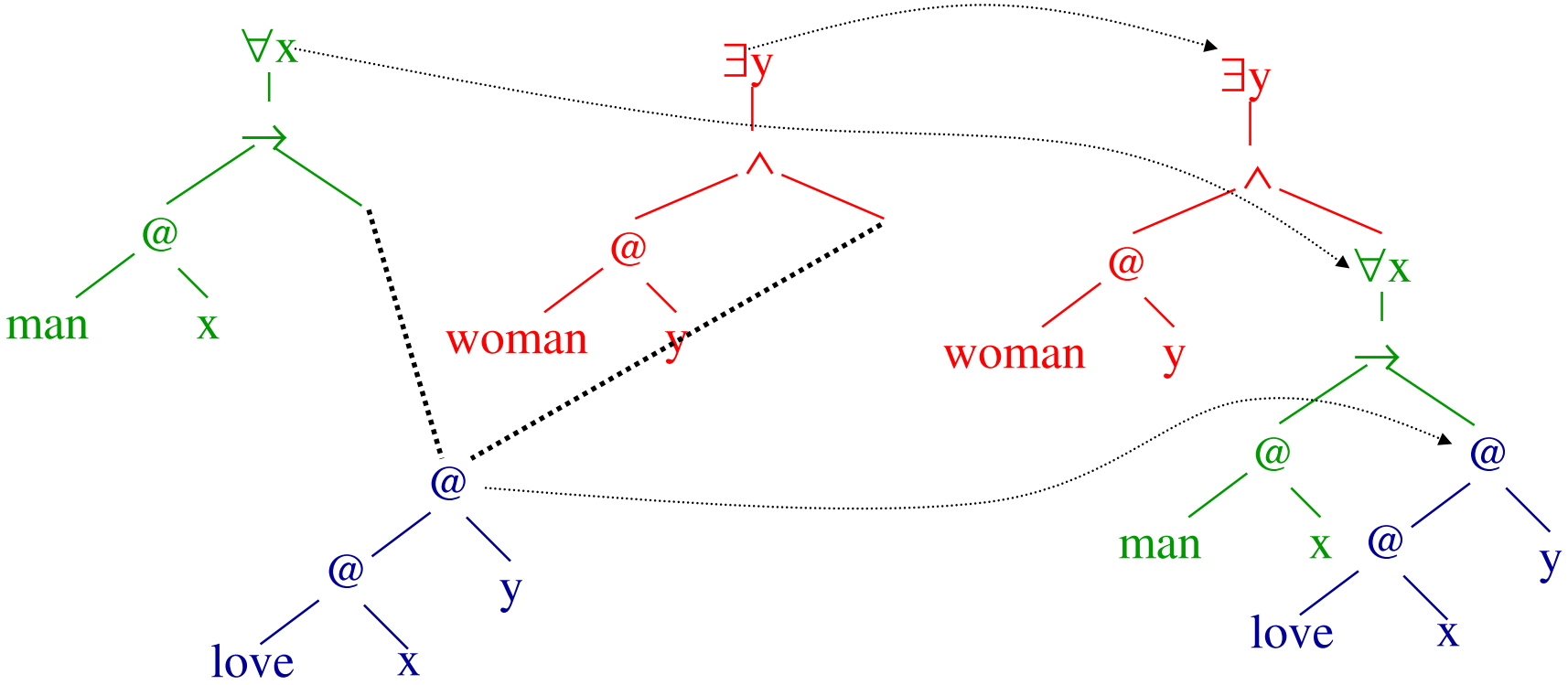
What does a dominance graph mean?

- A lambda structure satisfies a dominance graph if we can embed the graph into the tree such that
  - no two labelled graph nodes are mapped to the same tree node;
  - all labels, solid edges, and binding edges are present in the tree;
  - whenever (u,v) is a dominance edge in the graph, there is a path from u to v over solid edges in the tree.

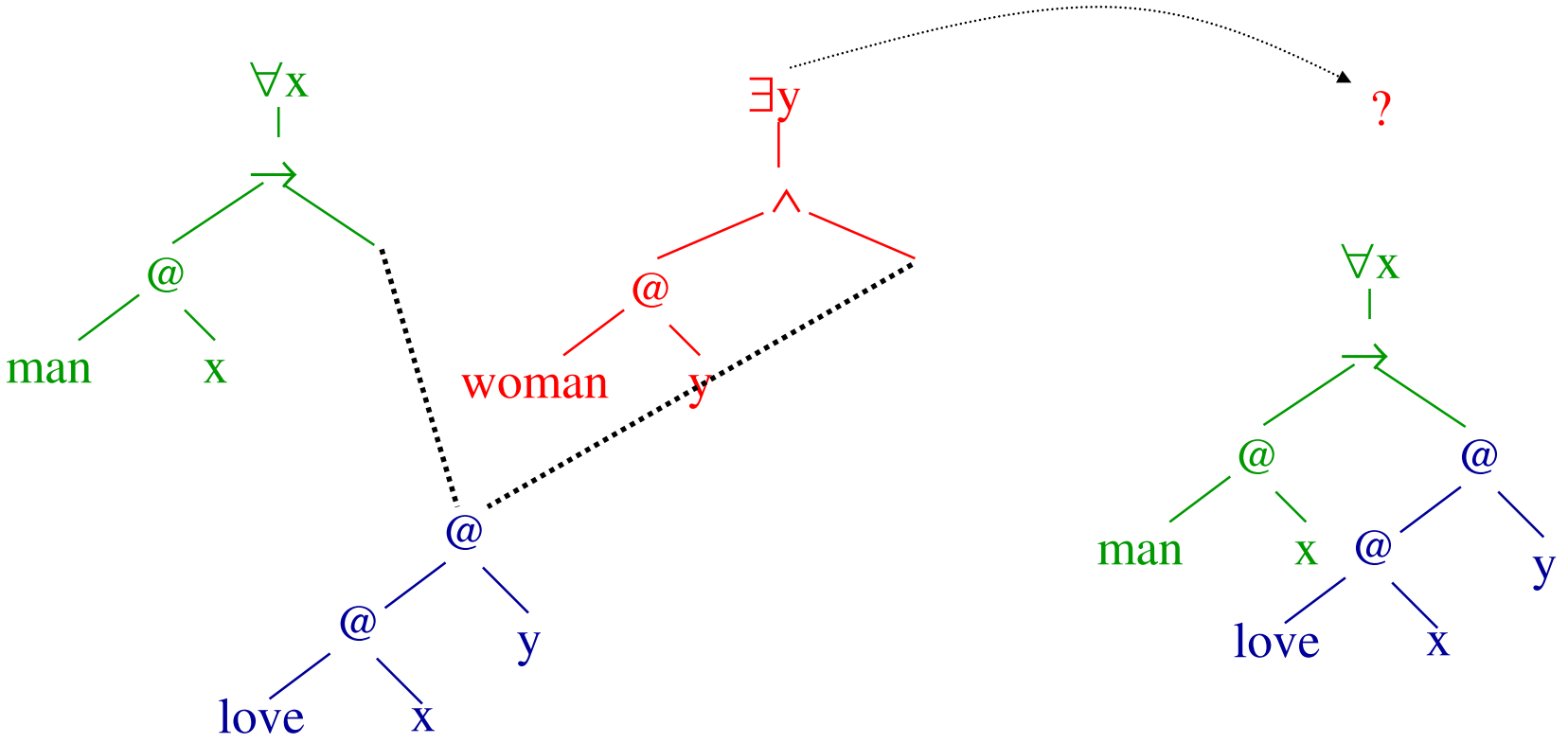
Solutions of dominance graphs



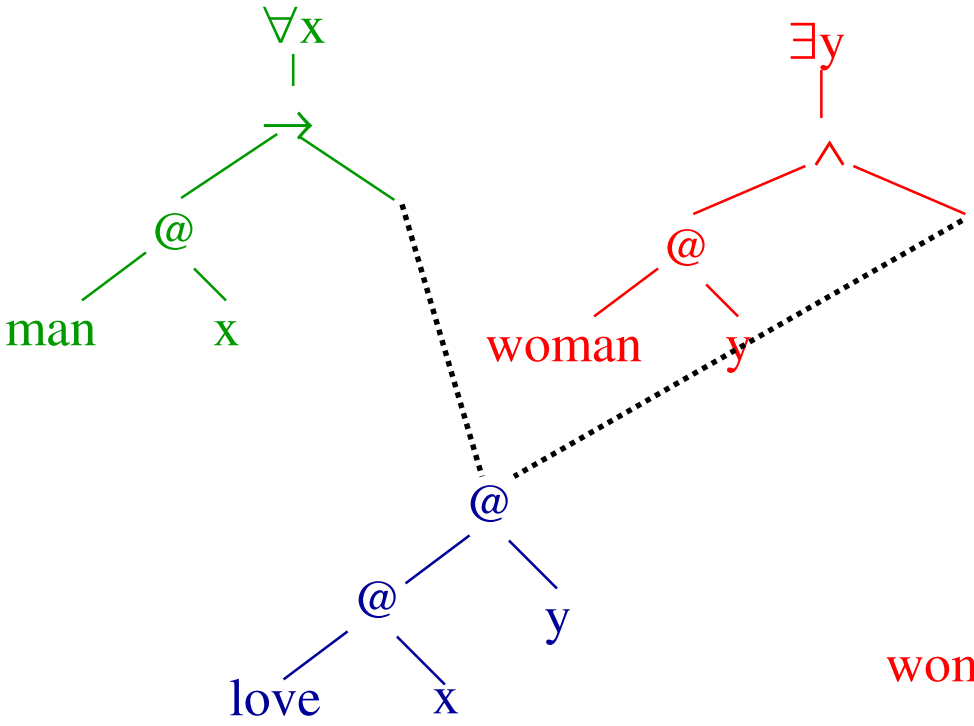
Solutions of dominance graphs

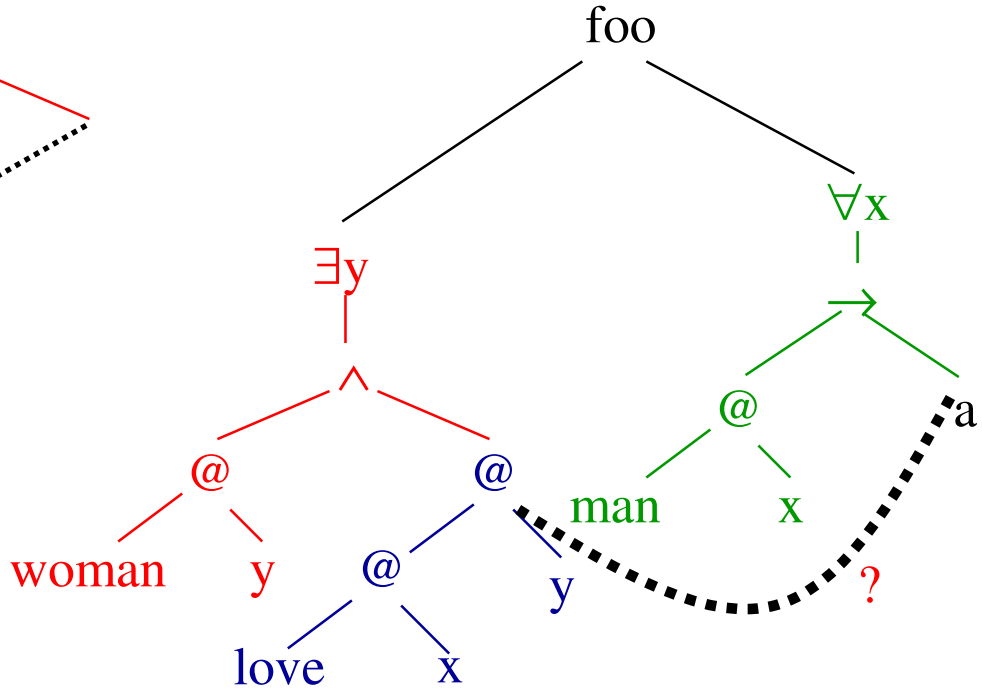


Not a solution

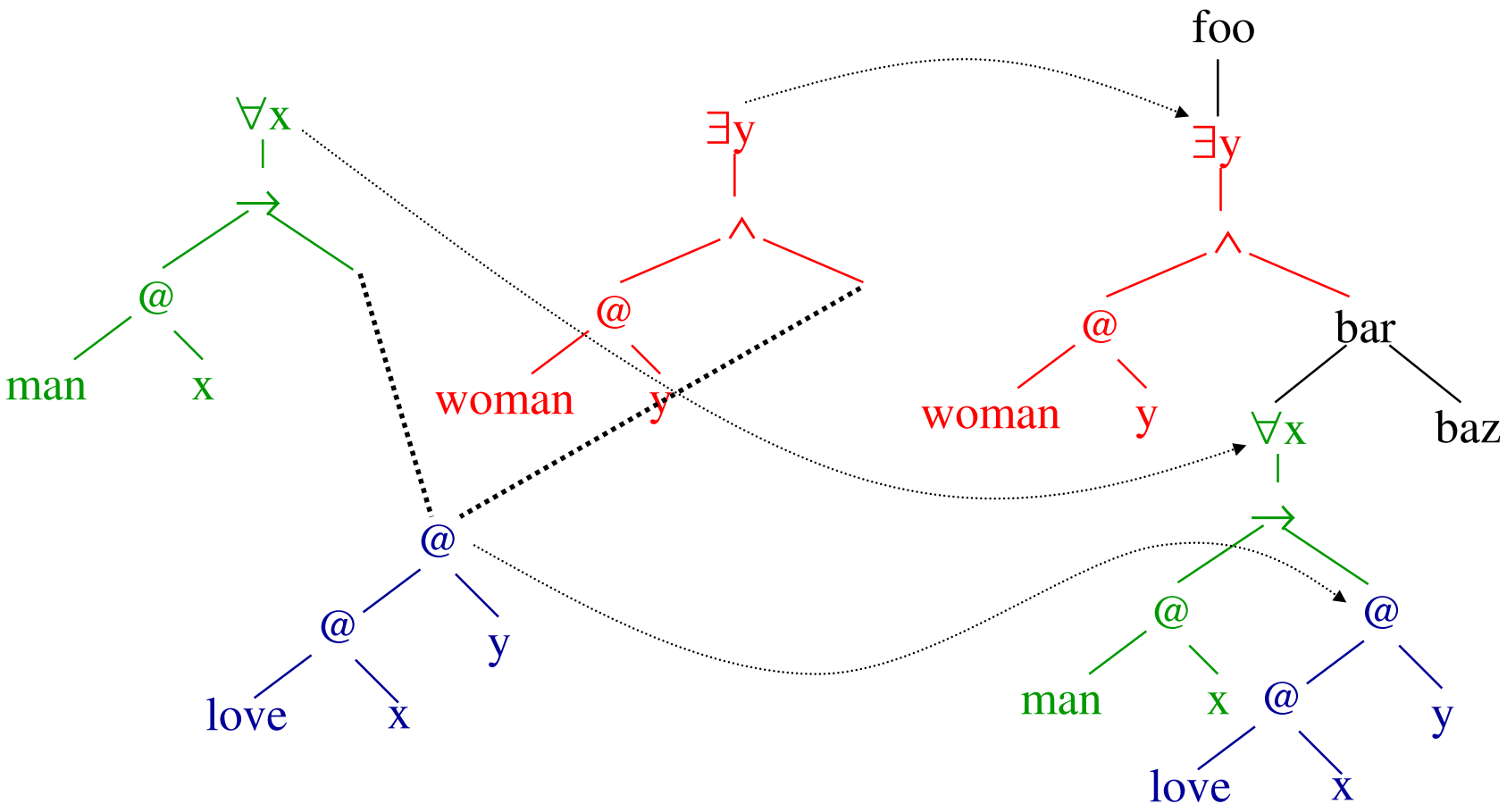


Not a solution either





Solutions can be larger than graph



(but: see Fuchss et al., ACL 04)

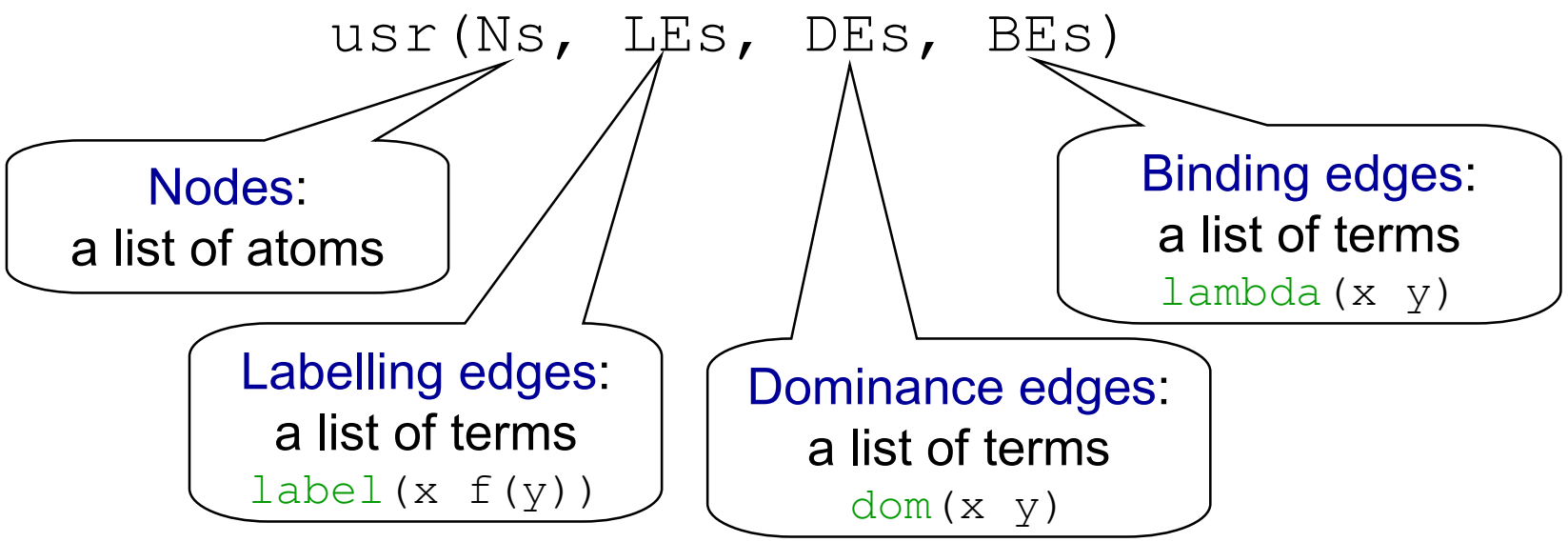
Dominance graphs and dominance constraints

- Dominance graphs can be seen as graph representations of normal dominance constraints.
- Dominance constraints are a logical language interpreted over trees.
- Can be extended to the Constraint Language for Lambda Structures (CLLS): Describe not just scope, but also anaphora and ellipsis.
- See Egg et al. (2001), JoLLI.

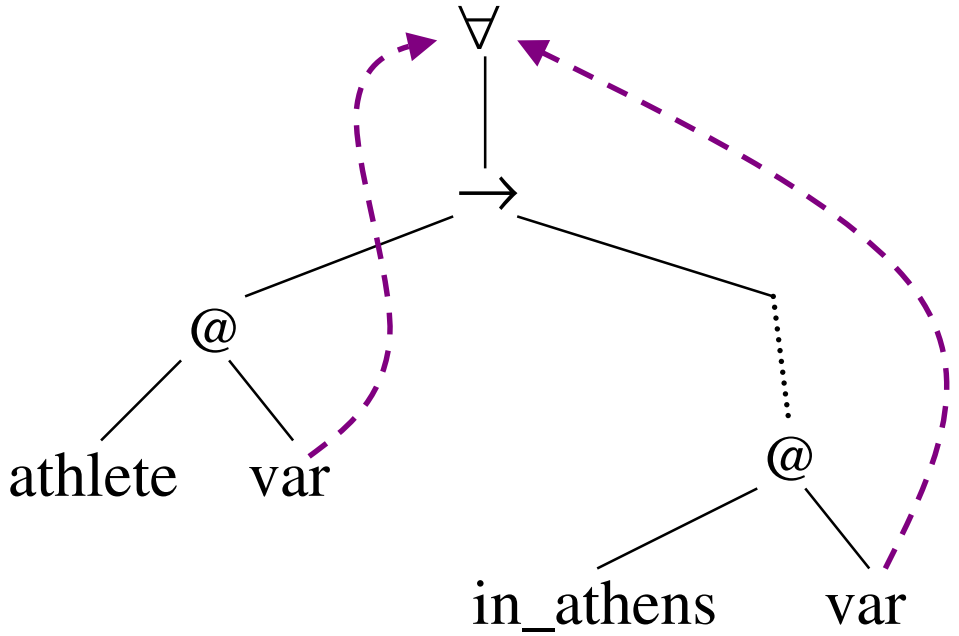
Dominance graphs in Prolog

- Need to represent nodes and edges of the graph.
- Nodes as atoms.
- Edges as terms whose arguments are atoms.

Dominance graphs in Prolog



An Example




```
usr([x1,x2,x3,x4,x5,x6,x7,x8,x9],
    [label(x1,forall(x2)), label(x5,athlete), ...],
    [dom(x4,x7)],
    [lambda(x9,x1)])
```

Summary

- Problem: How do we derive all readings of a semantically ambiguous sentence?
- Here: Scope ambiguity.
- Montague: Recast as syntactic ambiguity.
- Underspecification: Introduce additional layer of abstraction (underspecified description).
- Dominance graphs: See formulas as trees, describe them using graphs.

Towards the second round

- Solve the problem of semantic ambiguity by deriving a single dominance graph from a single syntactic analysis.
  - We will show tomorrow how this is done.
- When you want the actual semantic readings, enumerate them from the dominance graph.
  - We will show tomorrow how this is done.
- Modular system that factorises the ambiguity into a component of its own.