

# Semantic Theory

## Lecture 6: Underspecification

M. Pinkal / A. Koller  
Summer 2006

### Scope ambiguities

- Some sentences have more than one possible semantic representation:

Every student presents a paper.

$$(a) \forall x[\textit{student}'(x) \rightarrow \exists y[\textit{paper}'(y) \wedge \textit{present}'(x,y)]]$$

$$(b) \exists y[\textit{paper}'(y) \wedge \forall x[\textit{student}'(x) \rightarrow \textit{present}(x,y)]]$$

Every student didn't pay attention.

$$(a) \forall x[\textit{student}'(x) \rightarrow \neg \textit{pay-attention}'(x)]$$

$$(b) \neg \forall x[\textit{student}'(x) \rightarrow \textit{pay-attention}'(x)]$$

## Scope ambiguities

- The number of readings of a sentence with scope ambiguities grows with the number of NPs:

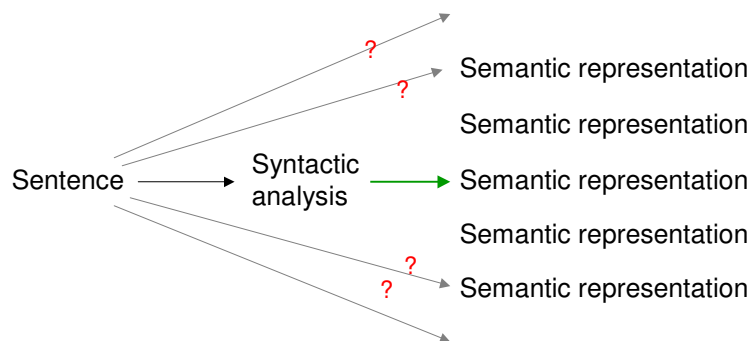
Every researcher of a company saw some sample.

- $\forall x(\text{res}'(x) \wedge \exists y(\text{cp}'(y) \wedge \text{of}'(x,y)) \rightarrow \exists z(\text{spl}'(z) \wedge \text{see}'(x,z))$
- $\exists z(\text{spl}'(z) \wedge \forall x(\text{res}'(x) \wedge \exists y(\text{cp}'(y) \wedge \text{of}'(x,y)) \rightarrow \text{see}'(x,z))$
- $\exists y(\text{cp}'(y) \wedge \forall x(\text{res}'(x) \wedge \text{of}'(x,y)) \rightarrow \exists z(\text{spl}'(z) \wedge \text{see}'(x,z))$
- $\exists y(\text{cp}'(y) \wedge \exists z(\text{spl}'(z) \wedge \forall x(\text{res}'(x) \wedge \text{of}'(x,y)) \rightarrow \text{see}'(x,z))$
- $\exists z(\text{spl}'(z) \wedge \exists y(\text{cp}'(y) \wedge \forall x(\text{res}'(x) \wedge \text{of}'(x,y)) \rightarrow \text{see}'(x,z))$

Every researcher of a company saw some samples of most products.

etc.

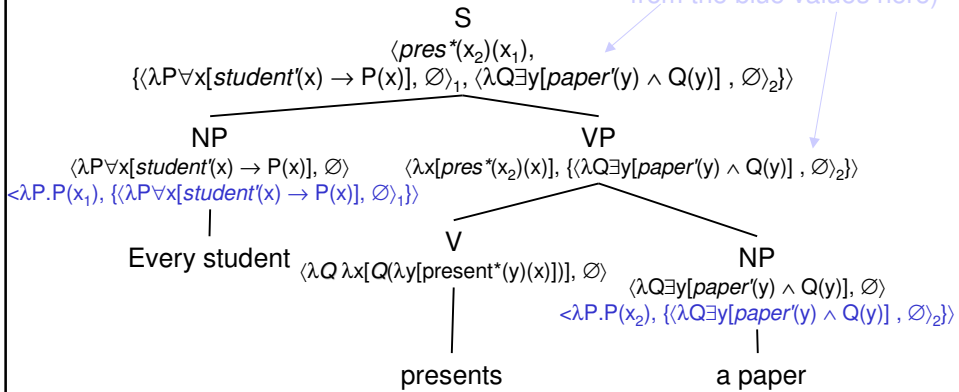
## Semantic ambiguity: A picture



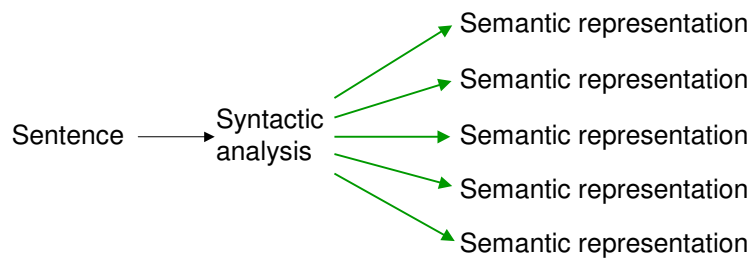
## Nested Cooper Storage: Example

*Every student presents a paper.*

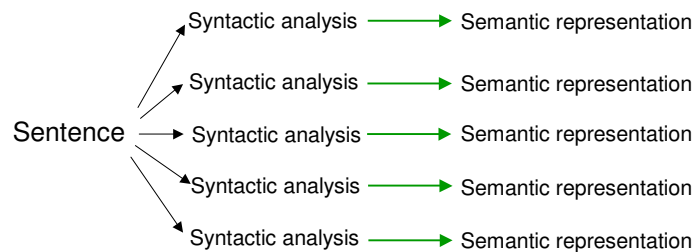
(only showing the results from the blue values here)



## (Nested) Cooper Storage: Schema



## Montague Grammar: Schema



- Montague 1974: "A proper treatment of quantification in ordinary English"
- Analyse scope ambiguity as a syntactic ambiguity.

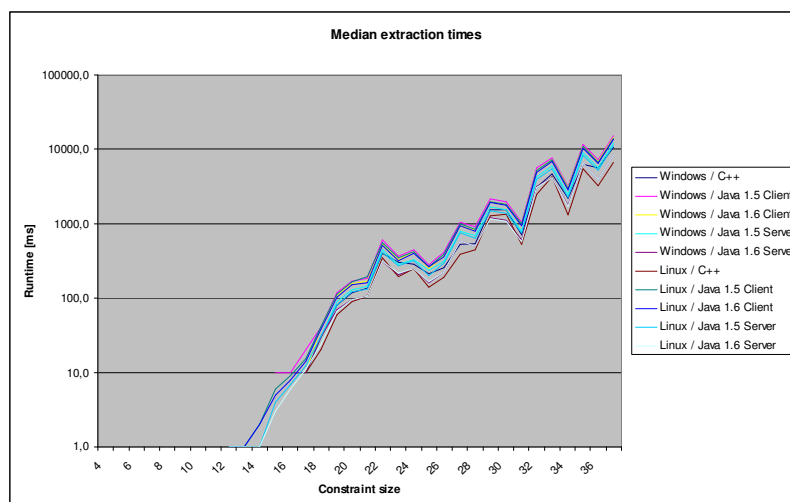
## So where do we stand?

- The Good News:
  - We can compute the readings of a scope ambiguity compositionally.
- The Bad News:
  - The number of readings grows exponentially with the number of scope-bearing elements.
  - Enumerating them takes a long time.
  - Most of this time is wasted.

## 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)
- Record: One sentence in Rondane Treebank has 2.4 trillion ( $10^{12}$ ) scope readings according to ERG.

## Enumeration can get expensive



## Enumeration is not always necessary

- Some sentences can be evaluated semantically without having to commit to one scope reading:

In Saarbrücken, many scientists at several institutes are working on numerous interesting research problems in different areas of semantics.

Every student must speak two foreign languages. This is definitely too much.

## Immediate enumeration not always necessary

- The disambiguation to one reading can occur naturally as the discourse progresses:

Every student must speak two foreign languages. These languages are taught at our department.

Every student must speak two foreign languages. Appendix 1 of the Studienordnung lists the twenty admissible languages.

## Disambiguating factors

- World knowledge can exclude some readings:  
Every woman gave birth to a baby.
- Preferences, such as
  - word order
  - intonation
  - choice of determiners:  
"a search engine for every subject" vs. "a search engine for each subject"

## Underspecification: The big picture

Sentence → Syntax

semantic repres. 1

semantic repres. 2

semantic repres. 3

semantic repres. 4

## Underspecification: The big picture


Sentence → Syntax → **USR**

semantic repres. 1  
semantic repres. 2  
semantic repres. 3  
semantic repres. 4

- Derive a single **underspecified semantic representation (USR)** from the syntactic analysis.

## Underspecification: The big picture

Sentence → Syntax → **USR**

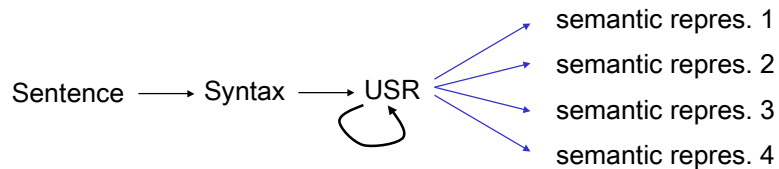


semantic repres. 1  
semantic repres. 2  
semantic repres. 3  
semantic repres. 4

- Derive a single underspecified semantic representation (USR) from the syntactic analysis.
- Perform **inferences** on USR to eliminate readings excluded by the context.



## Underspecification: The big picture



- Derive a single underspecified semantic representation (USR) from the syntactic analysis.
- Perform **inferences** on USR to eliminate readings excluded by the context.
- **Enumerate** readings by need.

## Underspecification

- Main points:
  - define USRs
  - show how to do semantics construction
  - show how to enumerate readings from USR (next week)
- Quantifiers stores of NCS can be seen as proto-USRs, but nondeterministic procedural element remains.
- USRs will be completely declarative.

## Underspecification with Dominance Graphs

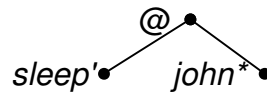
- Basic idea:
  - Consider semantic representations as trees.
  - Describe sets of trees using **dominance graphs**.
  - Special mechanisms for variable binding.
- Equivalent to **normal dominance constraints**, a logical formalism interpreted over trees (Egg et al. 2001, etc.).

## Terms as trees

- Terms (and formulas) of type theory have a natural reading as trees:
  - Application  $M(N)$  is the tree  $@(M,N)$
  - Abstraction  $\lambda x.M$  is the tree  $\text{lam}(M)$ ; quantifiers analogously.
  - Constant symbols become leaf labels
  - All variables become leaves with label  $\text{var}$ .
- Every node label has an **arity** that determines the number of children in the tree.  $@$  has arity 2; abstraction, quantifiers have arity 1; constants, variables have arity 0.

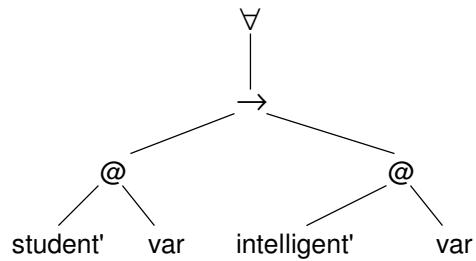
## Terms as trees

Tree representation of the formula  $\text{sleep}'(\text{john}^*)$ :



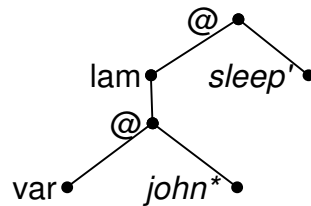
## Terms as trees

Tree representation of the formula  
 $\forall x.\text{student}'(x) \rightarrow \text{intelligent}'(x)$ :



## Terms as trees

Tree representation of the formula  
 $(\lambda F.F(\textit{john}^*))(\textit{sleep}')$ :



Notice: This tree is different from the one for  $\textit{sleep}'(\textit{john}^*)$ , because these are different (albeit equivalent) expressions.

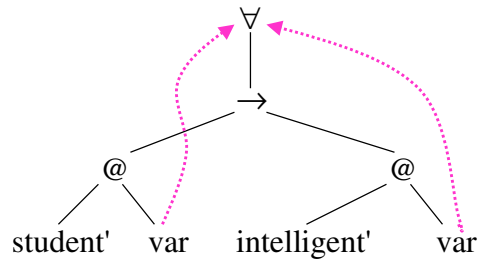
## Representing variable binding

- A **lambda structure**  $L = (t, \lambda)$  is a pair of a tree  $t$  and a partial function  $\lambda$  (the **binding function**) that maps nodes of  $t$  to nodes of  $t$ .
- The function  $\lambda$  maps variables to their binders; binders must dominate variables.
- Every expression of type theory corresponds to a unique lambda structure.
- We will see later that variable names are no longer sufficient to indicate variable binding in an underspecification context.

## Terms as lambda structures

Tree representation of the formula

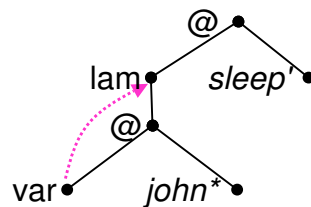
$\forall x.\text{student}'(x) \rightarrow \text{intelligent}'(x)$ :



## Terms as trees

Tree representation of the formula

$(\lambda F.F(\text{john}^*))(\text{sleep})$ :



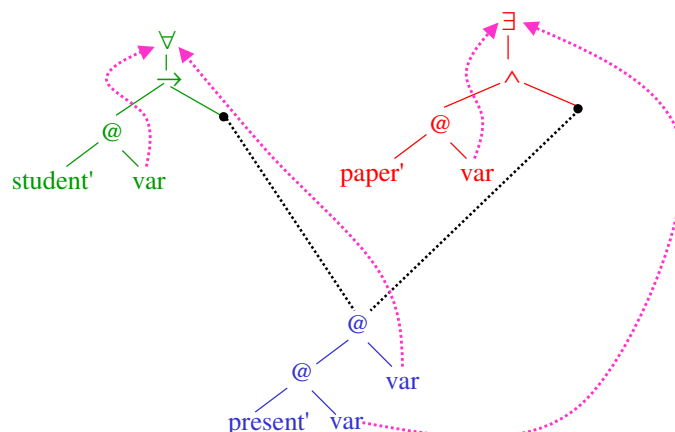
## Dominance graphs

- A (normal) dominance graph is a directed graph with node labels and three kinds of edges: tree edges, dominance edges, and binding edges.
- The graph restricted to only the tree edges is a collection of trees.
- All unlabelled nodes are leaves of these trees. These nodes are called holes.
- Each labelled node has as many children over tree edges as the arity of the label demands.
- Each tree contains at least one non-hole.
- The dominance edges all start at holes.

Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

27

## A dominance graph



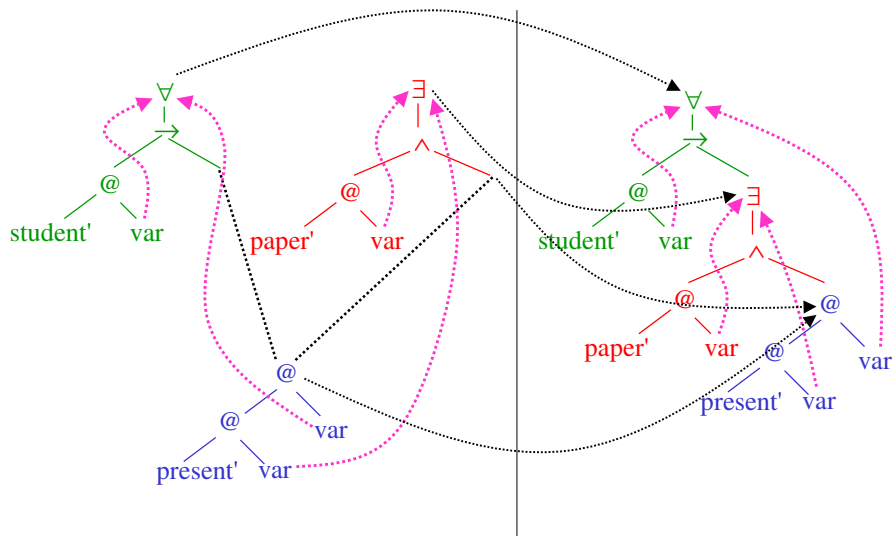
Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

28

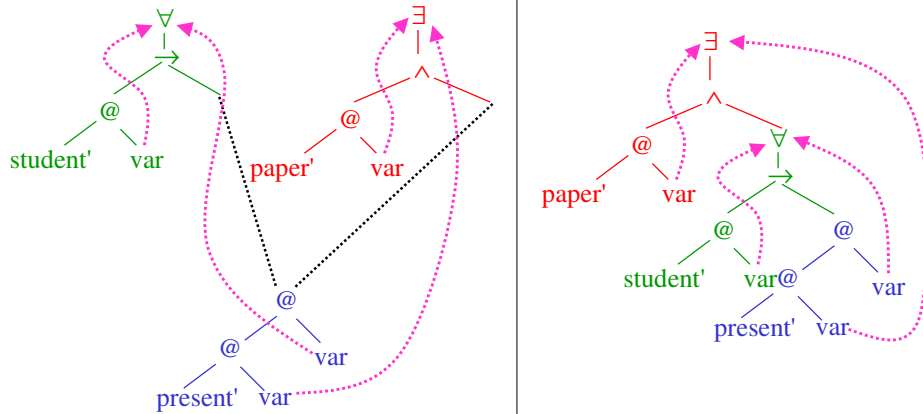
## Graphs describe lambda structures

- A lambda structure  $L$  is a **solution** of a dominance graph  $G$  iff there is a mapping  $\alpha$  of the nodes of  $G$  into the nodes of  $L$  such that
  - $\alpha$  maps no two non-holes to the same node in  $L$
  - for all labelled nodes  $v$  in  $G$ ,  $\alpha(v)$  has the same label as  $v$
  - if the tree-edge children of  $v$  are  $v_1, \dots, v_n$ , then the children of  $\alpha(v)$  are  $\alpha(v_1), \dots, \alpha(v_n)$
  - for each dominance edge  $(u, v)$  in  $G$ , there is a path from  $\alpha(u)$  to  $\alpha(v)$  in  $L$
  - for each binding edge  $(u, v)$  in  $G$ ,  $\lambda(\alpha(u))$  is defined and  $\lambda(\alpha(u)) = \alpha(v)$ .

## Solutions of dominance graphs



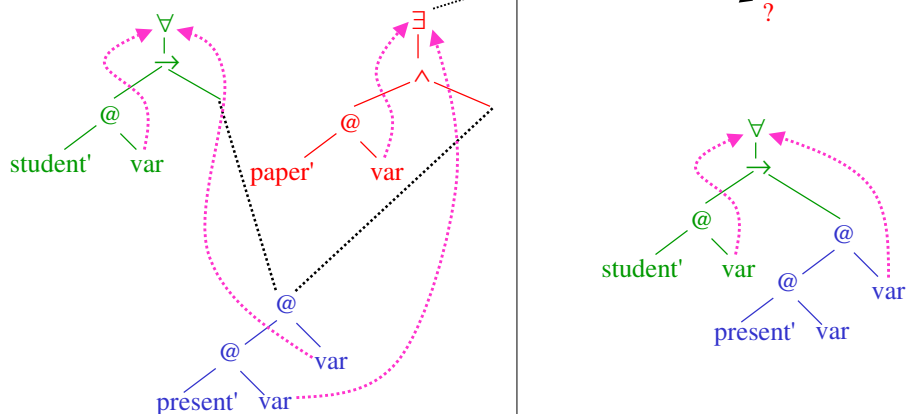
## Solutions of dominance graphs



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

31

## Not a solution

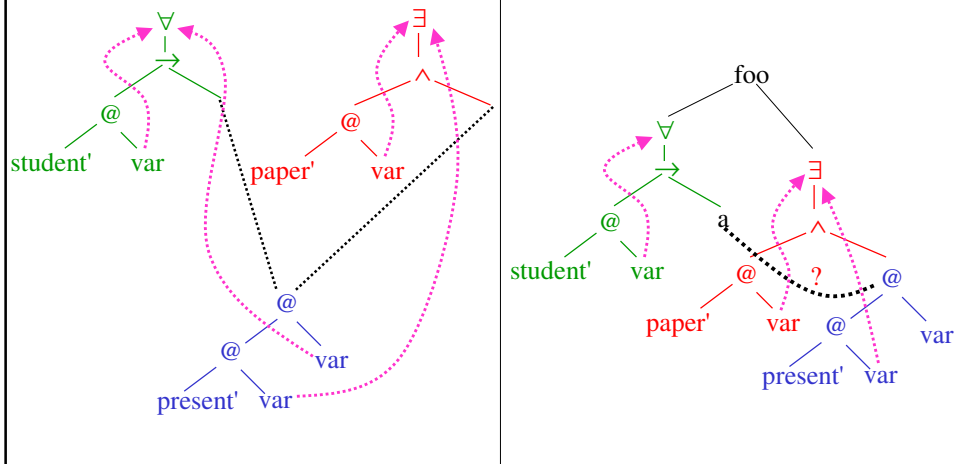


Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

32



## Not a solution



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

33

## What can we do now?

- Represent terms of type theory as lambda structures.
- Represent sets of terms of type theory (e.g. the semantic representations of a sentence) as the solutions of a dominance graph.
- Todo 1 (Semantics construction): How can we get a dominance graph for a sentence?
- Todo 2 (Enumeration): How can we compute the solutions of a dominance graph? (next week)

Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

34

## Semantics construction: Principles

- For every node in the syntax tree, we derive a dominance graph.
- Each syntax rule is associated with a semantics rule that combines dominance graphs.
- Each of these sub-dominance graphs has an **interface node** that is used to connect it with other subgraphs.
- The USR for the whole sentence is then the dominance graph associated with the root of the sentence.

## Lexicon access

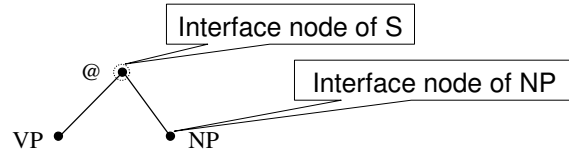
- Rule of lexical nodes:



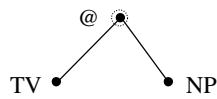
The semantic representation  $\beta$  for the word "a" is supplied by the lexicon.

## Semantics construction rules

- $S \rightarrow NP VP$



- $VP \rightarrow TV NP$



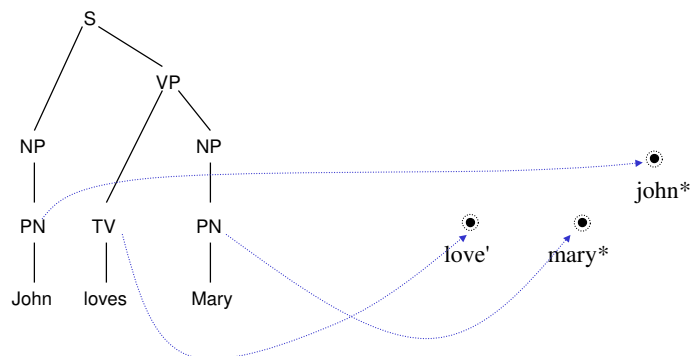
- $NP \rightarrow PN$



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

37

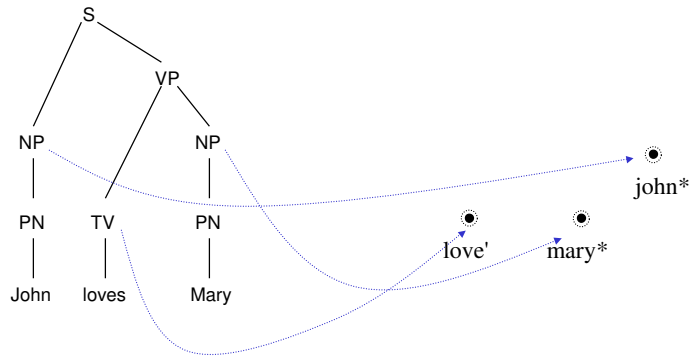
## A simple example



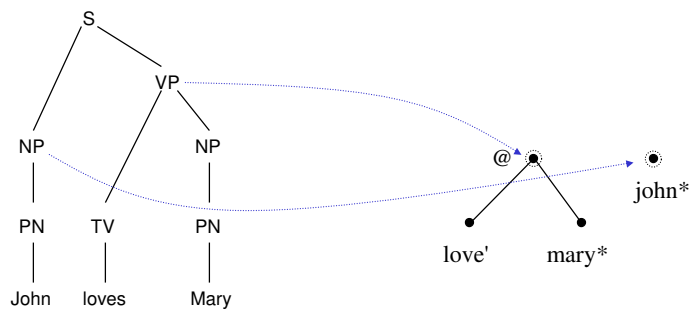
Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

38

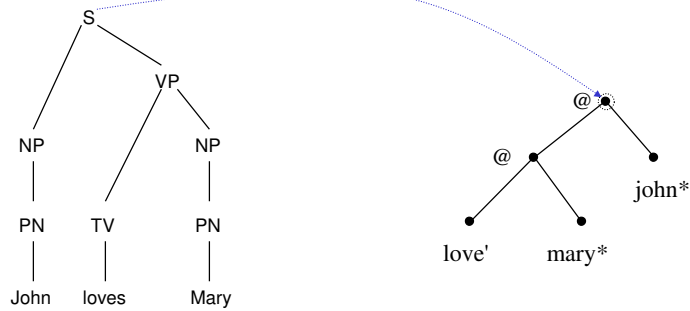
## A simple example



## A simple example



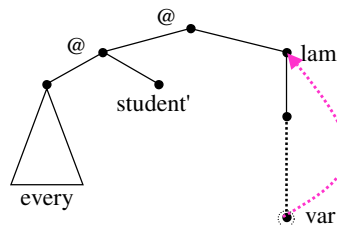
## A simple example



Semantic representation:  $\text{love}'(\text{mary}^*)(\text{john}^*)$

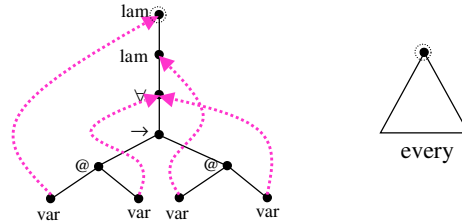
## Quantifiers

- The graph for a quantifier NP contains a variable node and its binder, linked by dominance and binding edges.
- The interface node of the graph is the variable node (representing a variable of type e)!

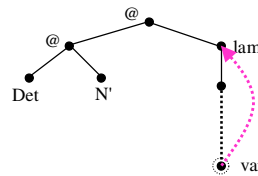


## Building graphs for quantifiers

- Lexicon entry for determiners (here "every"):



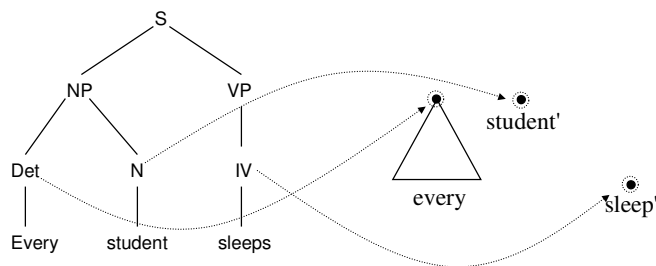
- Syntax rule  $NP \rightarrow Det N'$ :



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

43

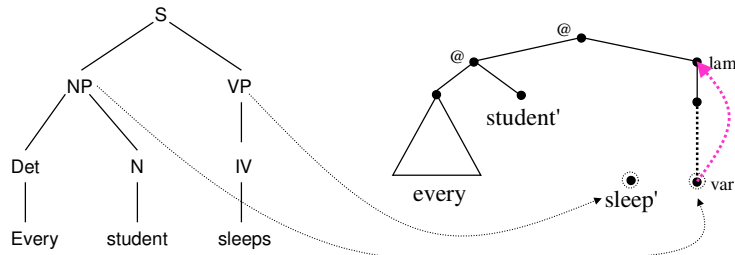
## An example with determiners



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

44

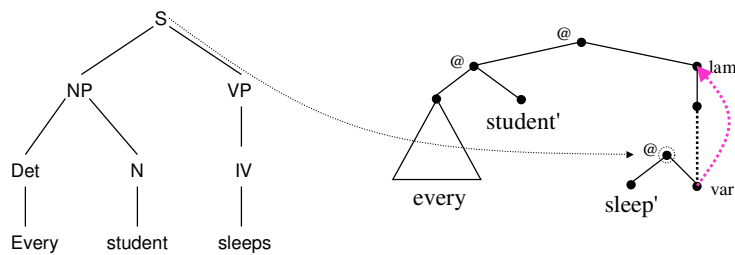
## An example with determiners



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

45

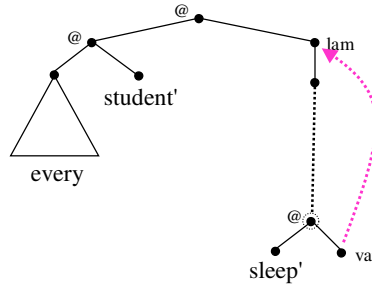
## An example with determiners



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

46

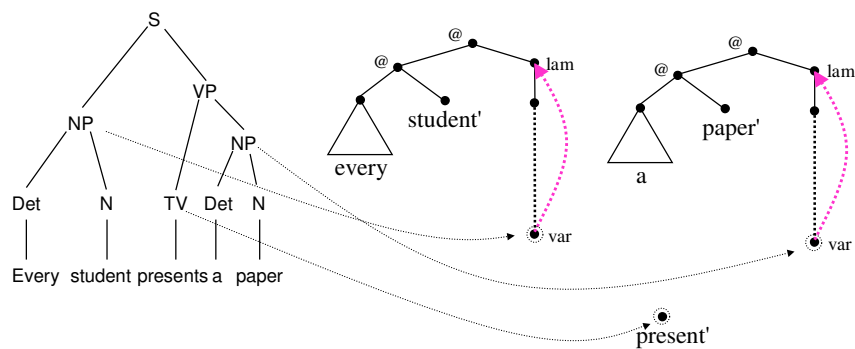
## Draw a little more nicely



$$\lambda P \lambda Q \forall x. (P(x) \rightarrow Q(x))(student')( \lambda y. sleep'(y) )$$

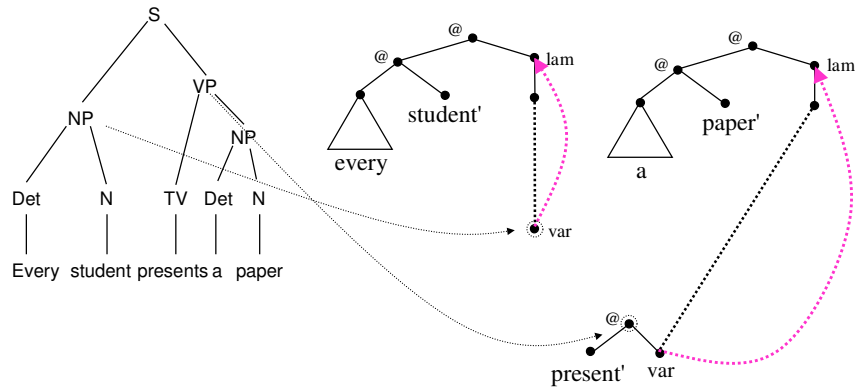
$$\Leftrightarrow_{\beta} \forall x. (student'(x) \rightarrow sleep'(x))$$

## Scope ambiguities





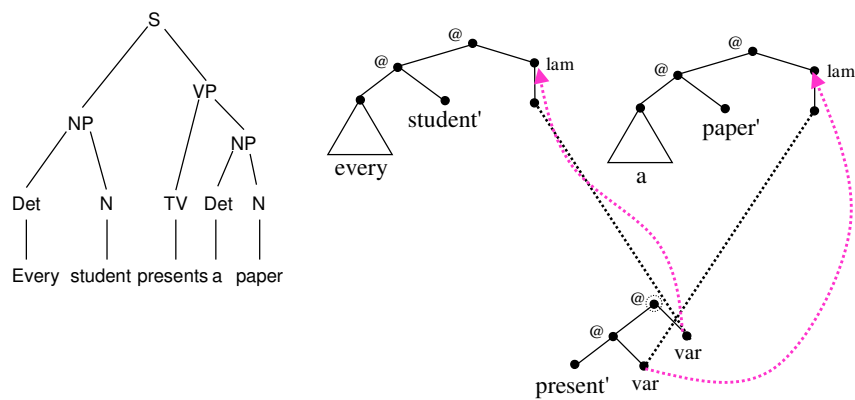
## Scope ambiguities



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

49

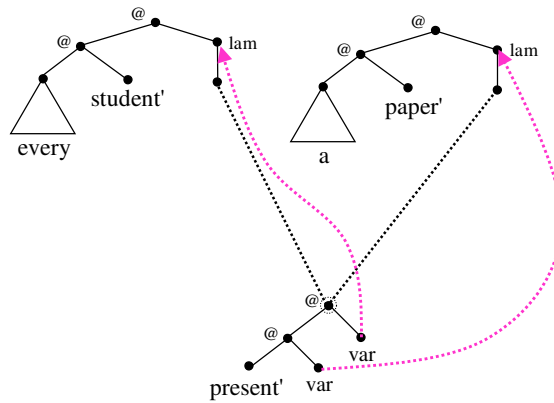
## Scope ambiguities



Semantic Theory 2006 © M. Pinkal / A. Koller UdS Computerlinguistik

50

## Draw a little more nicely



## An observation

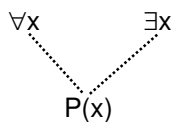
- We still use type theory as the object language, i.e. the language of semantic representations.
- However, types no longer drive the construction process.
- We use far fewer lambdas for "construction bookkeeping"; we replace this by plugging USRs into each other directly.
- This makes us more flexible in our choice of semantic representations:
  - can use  $\text{john}^*$  of type  $e$  for proper names
  - can use  $\text{present}^*$  of type  $\langle e, \langle e, t \rangle \rangle$  for transitive verbs

## An observation about noun phrases

- The quantifier representation is split into two parts:
  - a variable of type  $e$  which the verb is applied to; this is like the  $x_i$  that is introduced in the NCS Storage rule.
  - a fragment containing a quantifier representation of type  $\langle\langle e,t\rangle,t\rangle$ , which is applied at some point to what would be the "semantic content" in NCS.
- The two components are connected by binding and dominance edges.
- The variable binding is introduced together with the variable and the binder; no need for "variable capturing".

## Why binding edges?

- In an underspecification context, variable names aren't always sufficient to indicate the binder for each variable:



- Problem could be solved by requiring that variables are named apart, but this breaks down for extensions of dominance graphs.
- Binding edges are a clean and simple way of doing it.

## Conclusion

- Enumerating all readings is typically a waste of time.
- Underspecification: Enumerate only by need.
- Dominance graphs: Encode readings as trees; use graphs as underspecified semantic representations.
- Simple semantics construction that combines sub-dominance graphs.
- Each syntactic combination rule is associated with a semantic combination rule.