# Semantic Theory

Summer 2005

Underspecification

M. Pinkal / A. Koller

## 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

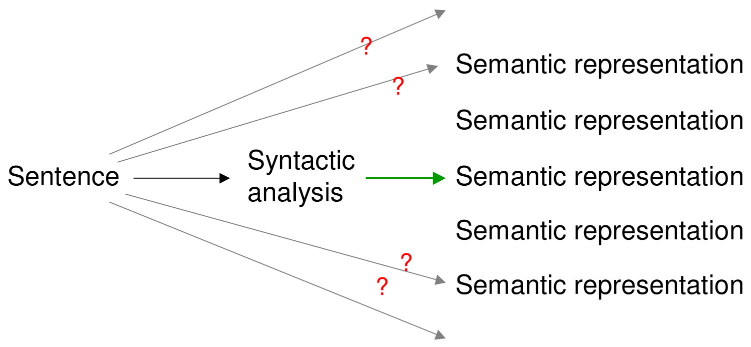
- The number of readings of a sentence with scope ambiguities grows with the number of NPs:

  Every researcher of a company saw some sample.
  1. $\forall x(res'(x) \wedge \exists y(cp'(y) \wedge of'(x,y)) \rightarrow \exists z(spl'(z) \wedge see'(x,z))$
  2. $\exists z(spl'(z) \wedge \forall x(res'(x) \wedge \exists y(cp'(y) \wedge of'(x,y)) \rightarrow see'(x,z))$
  3. $\exists y(cp'(y) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow \exists z(spl'(z) \wedge see'(x,z))$
  4. $\exists y(cp'(y) \wedge \exists z(spl'(z) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow see'(x,z))$
  5. $\exists z(spl'(z) \wedge \exists y(cp'(y) \wedge \forall x(res'(x) \wedge of'(x,y)) \rightarrow see'(x,z))$

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

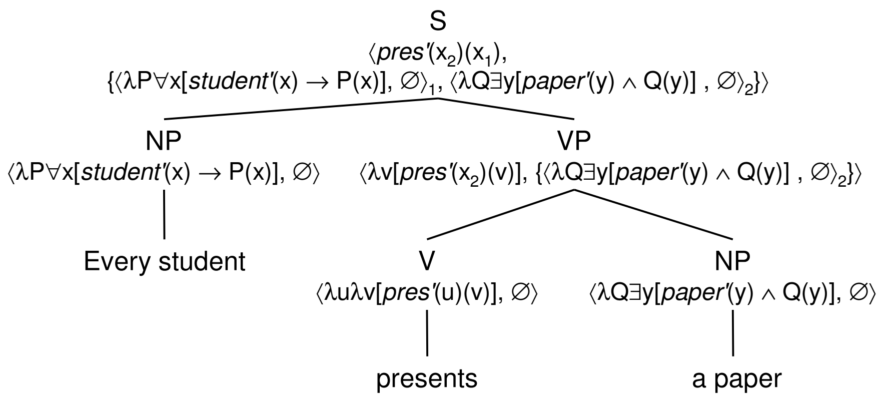
  etc.

## Semantic ambiguity: A picture

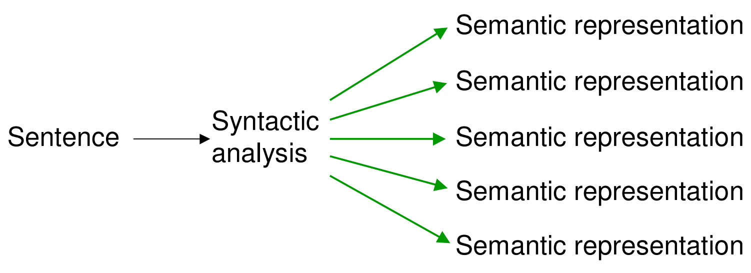


## Nested Cooper Storage: Example

*Every student presents a paper.*


S
$\langle \mathit{pres}'(x_2)(x_1),$
$\{\langle \lambda P \forall x[\mathit{student}'(x) \rightarrow P(x)], \varnothing\rangle_1, \langle \lambda Q \exists y[\mathit{paper}'(y) \wedge Q(y)] , \varnothing\rangle_2\}\rangle$
NP
$\langle \lambda P \forall x[\mathit{student}'(x) \rightarrow P(x)], \varnothing\rangle$
VP
$\langle \lambda v[\mathit{pres}'(x_2)(v)], \{\langle \lambda Q \exists y[\mathit{paper}'(y) \wedge Q(y)] , \varnothing\rangle_2\}\rangle$
Every student
V
$\langle \lambda u \lambda v[\mathit{pres}'(u)(v)], \varnothing\rangle$
NP
$\langle \lambda Q \exists y[\mathit{paper}'(y) \wedge Q(y)], \varnothing\rangle$
presents
a paper

## (Nested) Cooper Storage: Schema



## Montague Grammar: Schema

$$\begin{tikzcd}
 & \text{Syntactic analysis} \arrow[r] & \text{Semantic representation} \\
 & \text{Syntactic analysis} \arrow[r] & \text{Semantic representation} \\
\text{Sentence} \arrow[ruu] \arrow[ru] \arrow[r] \arrow[rd] \arrow[rdd] & \text{Syntactic analysis} \arrow[r] & \text{Semantic representation} \\
 & \text{Syntactic analysis} \arrow[r] & \text{Semantic representation} \\
 & \text{Syntactic analysis} \arrow[r] & \text{Semantic representation}
\end{tikzcd}$$

- 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 enumerate the readings of a scope ambiguity.
- 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 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 computational linguist 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.

## Underspecification: The big picture

Sentence $\longrightarrow$ Syntax

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

## Underspecification: The big picture

$$\begin{tikzcd}
\text{Sentence} \arrow[r] & \text{Syntax} \arrow[r] & \text{USR}
\end{tikzcd}$$

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

$$\begin{tikzcd}
\text{Sentence} \arrow[r] & \text{Syntax} \arrow[r] & \text{USR} \arrow[loop below]
\end{tikzcd}$$

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

$$\begin{tikzcd}
 & & & \text{semantic repres. 1} \\
 & & & \text{semantic repres. 2} \\
\text{Sentence} \arrow[r] & \text{Syntax} \arrow[r] & \text{USR} \arrow[loop below] \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \text{semantic repres. 3} \\
 & & & \text{semantic repres. 4}
\end{tikzcd}$$

- 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 (Thursday)
- Quantifiers stores of NCS can be seen as proto-USRs, but nondeterministic procedural element remains.
- USRs will be completely declarative.

## Underspecification with Dominance Graphs

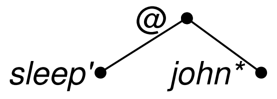
- Basic idea:
  - Consider semantic representations as trees.
  - Describe sets of trees using dominance graphs.
  - Special mechanisms for variable binding.
- Equivalent to normal dominance constraints, a logical formalism interpreted over trees (Egg et al. 2001, etc.).

## Terms as trees

- Terms (and formulas) of type theory have a natural reading as trees:
  - Application M(N) is the tree @(M,N)
  - Abstraction λx.M is the tree lam(M); quantifiers analogously.
  - Constant symbols become leaf labels
  - All variables become leaves with label var.

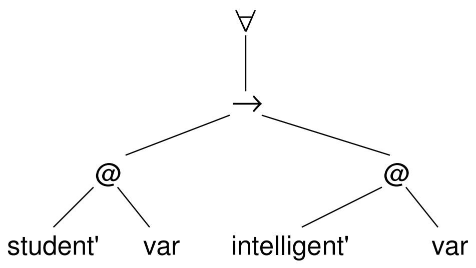
## Terms as trees

Tree representation of the formula sleep'(john*):



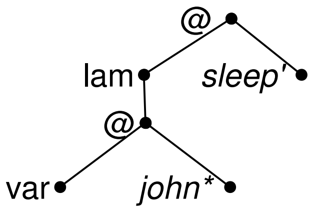
## Terms as trees

Tree representation of the formula
∀x.student'(x) → intelligent'(x):



## Terms as trees

Tree representation of the formula
(λF.F(*john**))(*sleep'*):

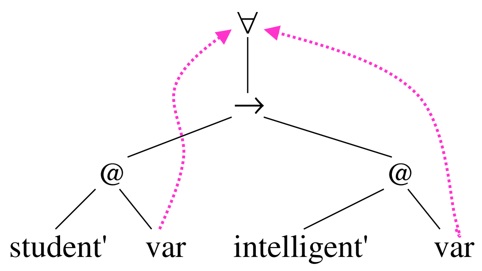


## Representing variable binding

- A lambda structure L = (t,λ) is a pair of a tree t and a partial binding function λ that maps nodes of t to nodes of t.
- The function λ maps variables to their binders; binders must dominate variables.
- Lambda structures represent λ-terms uniquely up to α-equivalence.
- We will see later that variable names are no longer sufficient to indicate variable binding in an underspecification context.

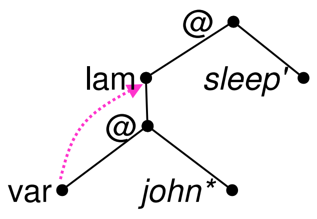
## Terms as lambda structures

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



## Terms as trees

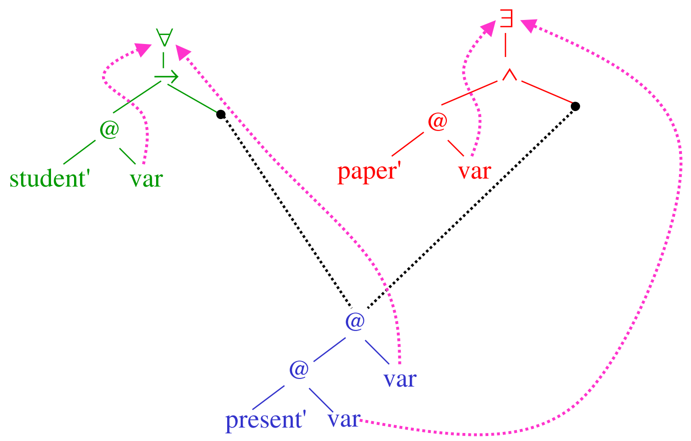
Tree representation of the formula
($\lambda$F.F(*john**))(*sleep'*):



## 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 tree contains at least one non-hole.
- The dominance edges all start at holes.

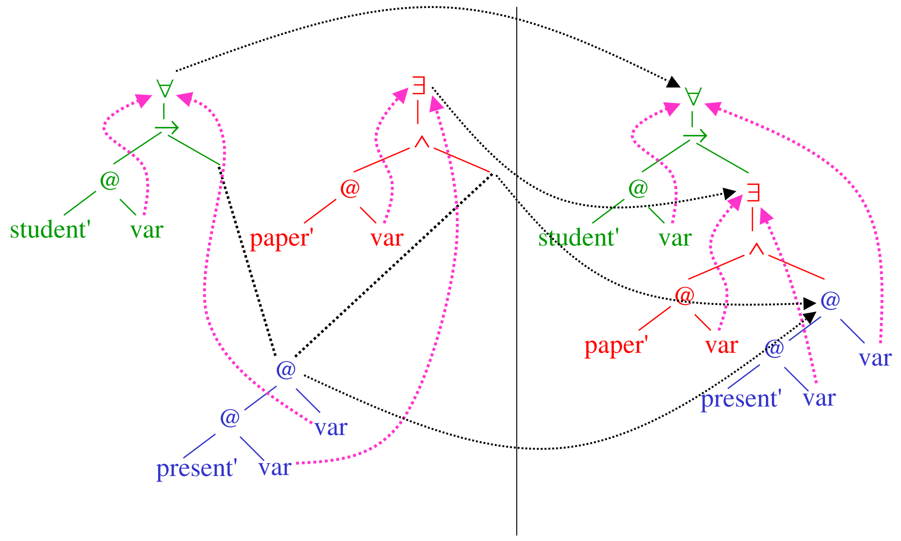
## A dominance graph



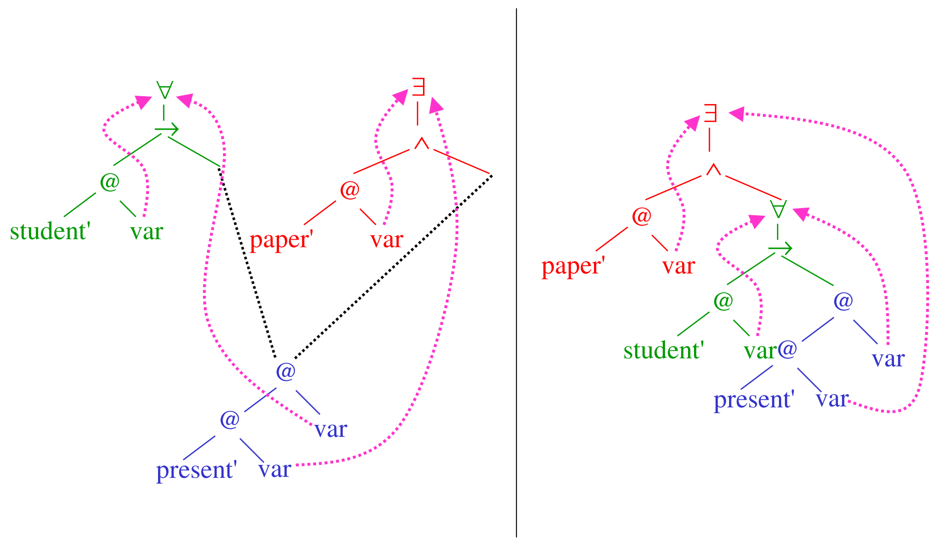
## Graphs describe lambda structures

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

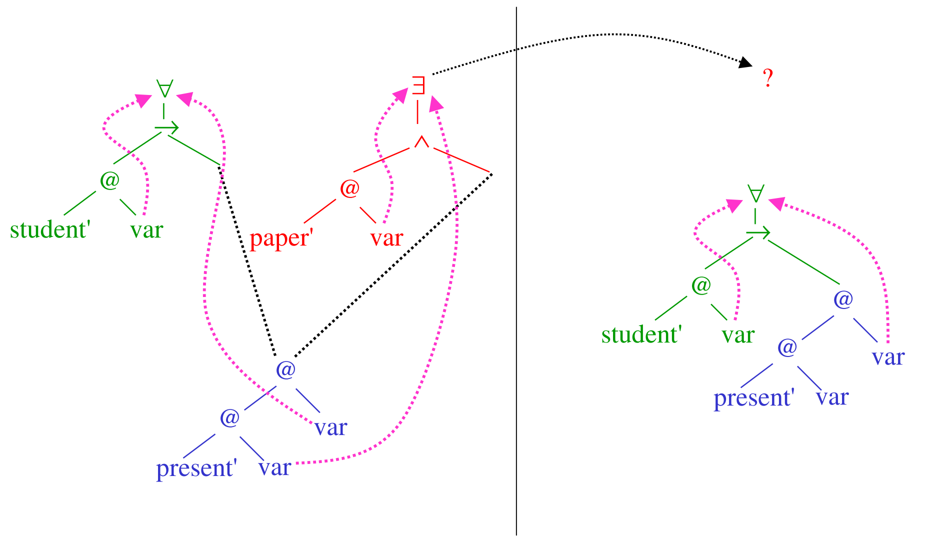
## Solutions of dominance graphs



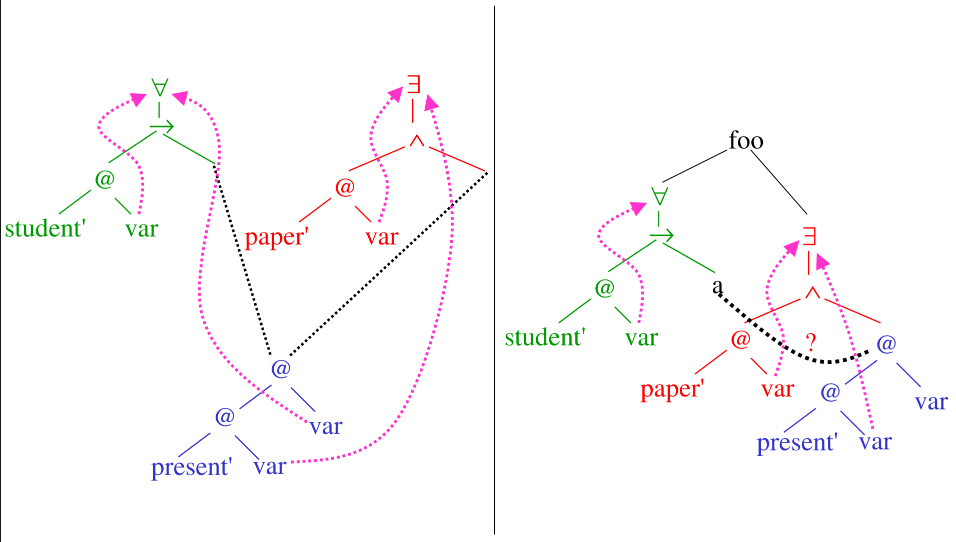
## Solutions of dominance graphs



## Not a solution



## Not a solution



## What can we do now?

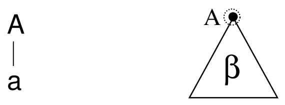
- Represent terms of type theory as lambda structures.
- Represent sets of terms of type theory (e.g. the semantic representations of a sentence) as the solutions of a dominance graph.
- Todo 1 (Semantics construction): How can we get a dominance graph for a sentence?
- Todo 2 (Enumeration): How can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

## 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

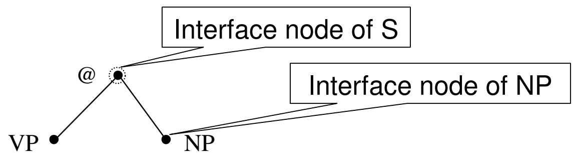
- Rule of lexical nodes:



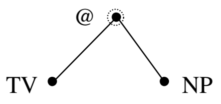
The semantic representation β for the word "a" is supplied by the lexicon.

## Semantics construction rules

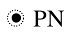
- S → NP VP



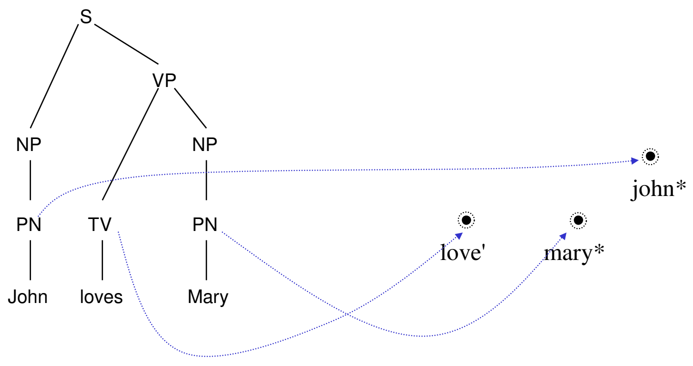
- VP → TV NP



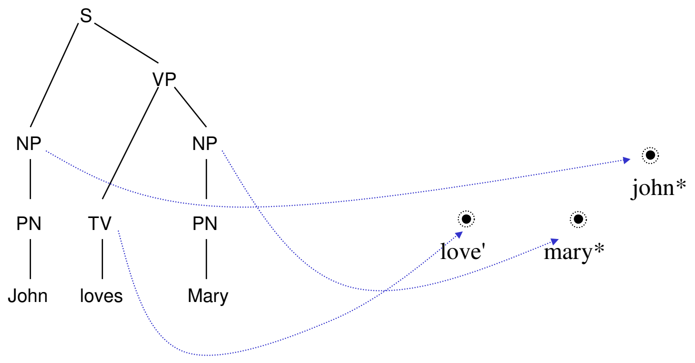
- NP → PN



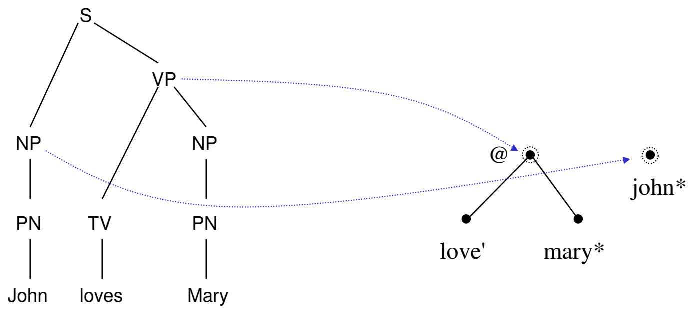
## A simple example



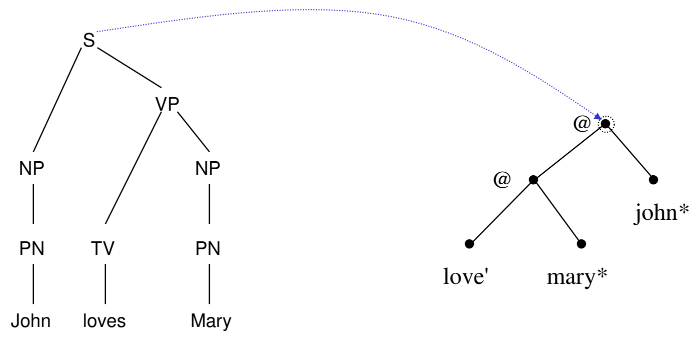
## A simple example



## A simple example



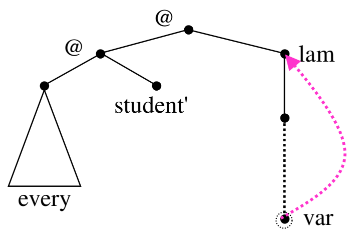
## A simple example



Semantic representation: love'(mary*)(john*)

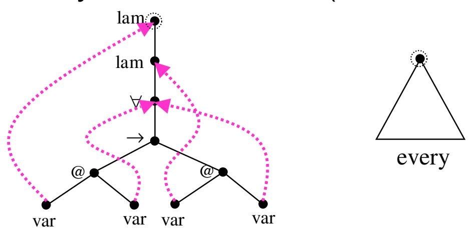
## Quantifiers

- 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)!

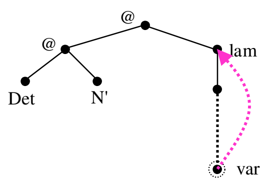


## Building graphs for quantifiers

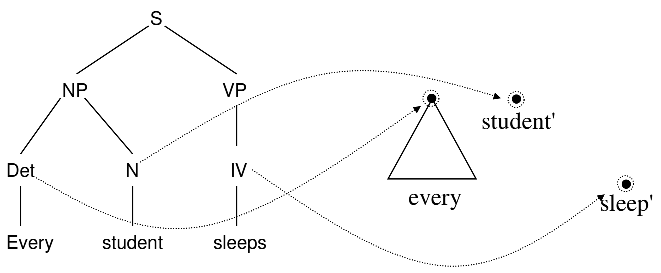
- Lexicon entry for determiners (here "every"):



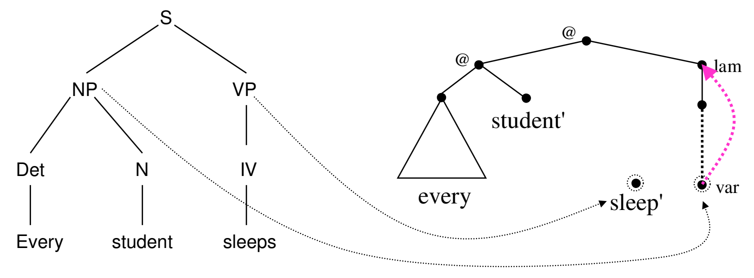
- Syntax rule NP → Det N':



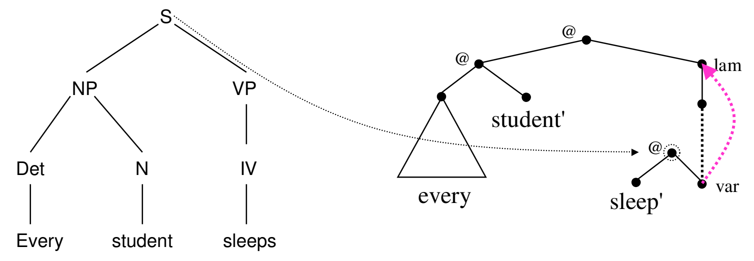
## An example with determiners



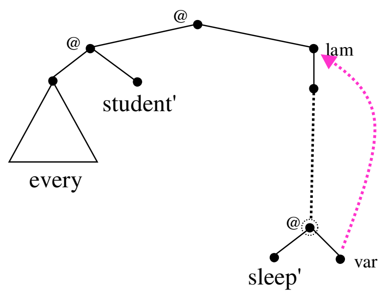
## An example with determiners



## An example with determiners

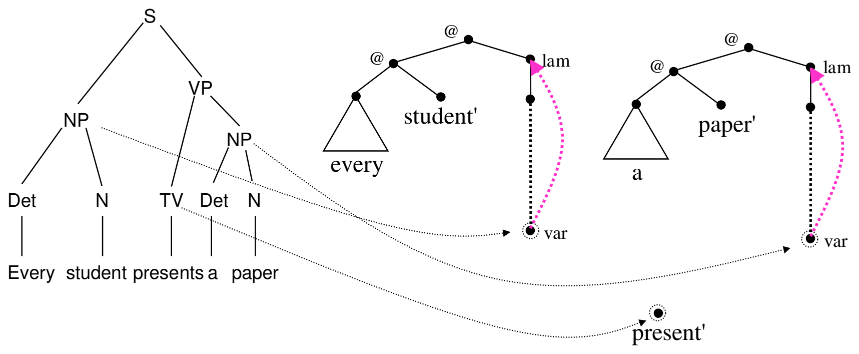


## Drawn a little more nicely

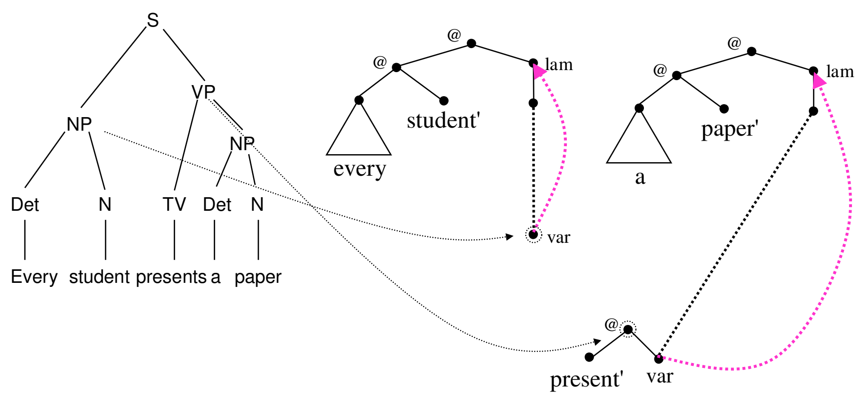


$$\lambda P\, \lambda Q\, \forall x.(P(x) \rightarrow Q(x))(\text{student}')(\lambda y.\text{sleep}'(y))$$
$$\Leftrightarrow_\beta \forall x.(\text{student}'(x) \rightarrow \text{sleep}'(x))$$

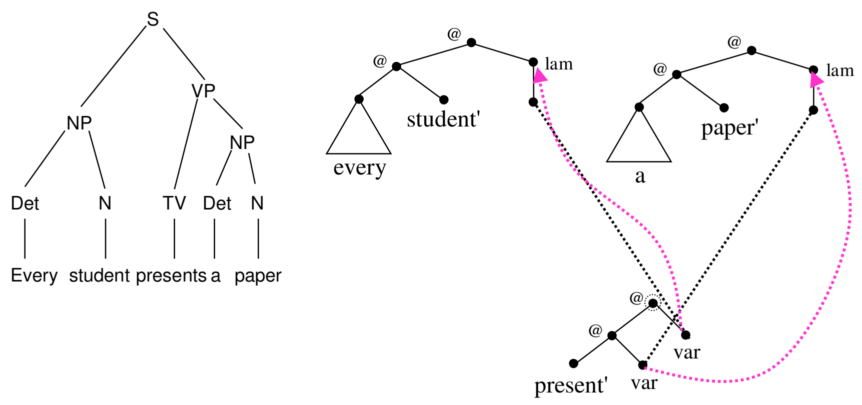
## Scope ambiguities



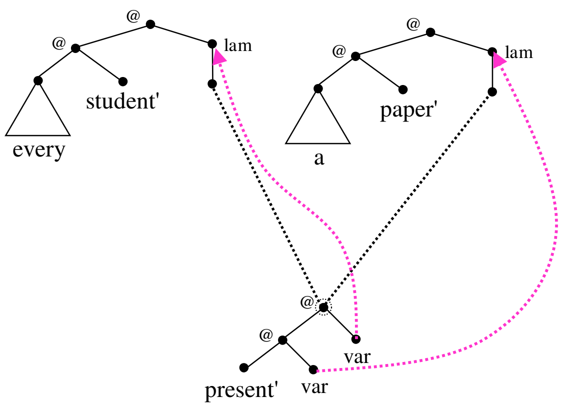
## Scope ambiguities



## Scope ambiguities



## Drawn a little more nicely



## An observation about semantics construction

- We can represent transitive verbs (and prepositions etc.) as terms of type <e,<e,t>> and apply them to variables of type e.
- 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 <<e,t>,t>, 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.

## Why binding edges?

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

$$\begin{tikzcd}
\forall x \arrow[dr, dotted, no head] & & \exists x \arrow[dl, dotted, no head] \\
& P(x) &
\end{tikzcd}$$

- 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.

## Scope ambiguities in the real world

- Scope ambiguities are not a very intuitive type of ambiguity, and are sometimes not seen as a serious problem for computational linguistics.
- In practice, they are often resolved by context, world knowledge, preferences, etc.
- We consider them here because they pose a fundamental challenge for semantics construction.
- If we want "deep" semantic representations that say something about scope, we must take scope ambiguities into account.

## Scope ambiguities in the real world

- Also, some large-scale grammars (e.g. the English Resource Grammar) compute semantic representations with scope.
- The ERG analyses all NPs as scope bearers to keep the grammar simple. (This is not necessarily correct: proper names, definites, etc.)
- Median number of scope readings in the Rondane corpus: 55.