

# Semantic Theory

## Underspecification

Manfred Pinkal  
Stefan Thater

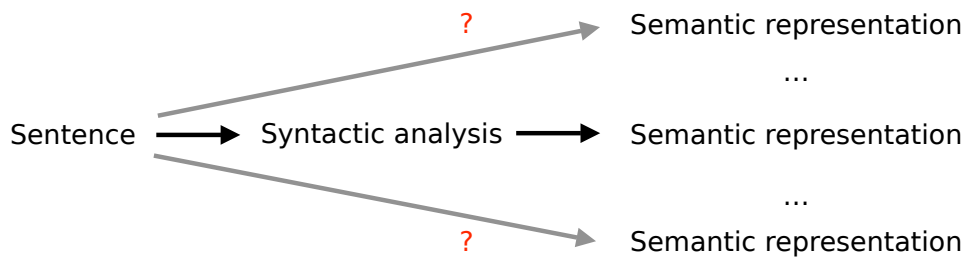
2008-05-20

## Scope ambiguities

- Sentences with two or more scope bearing operators such as quantifiers, negations, ... are often ambiguous:
- “Every student presents a paper.”
  - $\forall x(\text{student}'(x) \rightarrow \exists y(\text{paper}'(y) \wedge \text{present}'(x,y)))$
  - $\exists y(\text{paper}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{present}'(x,y)))$
- “Every student didn't pay attention.”
  - $\forall x(\text{student}'(x) \rightarrow \neg \text{pay-attention}'(x))$
  - $\neg \forall x(\text{student}'(x) \rightarrow \text{pay-attention}'(x))$

# Scope Ambiguities: Problem #1

- Compositional semantic construction: the readings are determined by the syntactic structure.
- How can we derive more than one reading if the sentence has only one syntactic structure?



3

# Nested Cooper Storage

- “Every student presents a paper.”
  - $(\text{present}^*(x_2)(x_1), \{$ 
    - $\langle \lambda P \forall x[\text{student}'(x) \rightarrow P(x)], \emptyset \rangle_1,$  [ =  $\langle \text{ES}, \emptyset \rangle_1$  ]
    - $\langle \lambda Q \exists y[\text{paper}'(y) \wedge Q(y)], \emptyset \rangle_2$  } ] [ =  $\langle \text{AP}, \emptyset \rangle_2$  ]
- Retrieval:
  1.  $\text{ES}(\lambda x_2(\text{AP}(\lambda x_1(\text{present}^*(x_2)(x_1))))$   
 $\Rightarrow_{\beta} \exists y(\text{paper}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{present}^*(y)(x)))$
  2.  $\text{AP}(\lambda x_1(\text{ES}(\lambda x_2(\text{present}^*(x_2)(x_1))))$   
 $\Rightarrow_{\beta} \forall x(\text{student}'(x) \rightarrow \exists y(\text{paper}'(y) \wedge \text{present}^*(y)(x)))$

4

## Nested Cooper Storage

- Storage techniques like Nested Cooper Storage allow to derive several distinct readings on the basis of a single syntactic analysis.
  - Problem #1 solved (... to a certain extent, see below)
- But note that Nested Cooper Storage has its own problems:
  - Non-determinism: storage vs. application at NP-nodes, retrieval at S-nodes.
  - For certain types of sentences it is not possible to derive all readings (e.g., "every student did not pay attention.")

## Scope Ambiguities: Problem #2

- Combinatorial explosion of readings: the number of readings can grow exponentially with the number of scope bearing operators.
  - (1) *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)
  - (2) *But that would give us all day Tuesday to be there.* (ca. 65000 readings according to the ERG)

## Enumeration of Readings is sometimes not 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 of Readings is 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.”

## Enumeration of Readings is not always necessary

- Sentences can contain “spurious ambiguities”
  - “We quickly put up the tents in the lee of a small hillside and cook for the first time in the open.”
  - 480 readings according to the English Resource Grammar ...
  - but only 2 equivalence classes, characterised by the relative scope of “the lee of” and “a small hillside”

## Disambiguating Factors

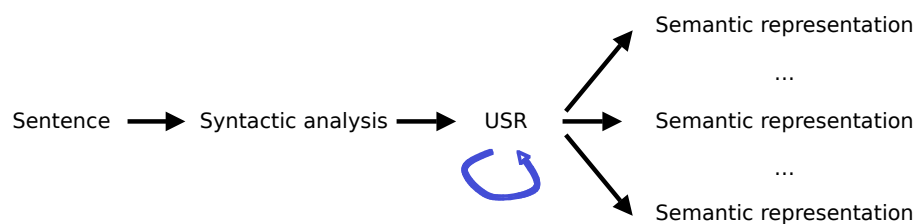
- World knowledge can exclude some readings:
  - “A rabbit is in every hat.”
  - “She has a finger in every pie.”
- Preferences, such as
  - Word order
  - Intonation
  - Choice of determiners: “a search engine for every subject” vs. “a search engine for each subject”
  - (from Language Log: A quantifier for every season)

## So where do we stand?

- By using storage techniques, we can compute the readings of scopally ambiguous sentences compositionally.
- But ...
  - the number of readings can grow exponentially with the number of scope-bearing elements.
  - enumerating all readings can thus take a long time.
  - most of this time is wasted.

11

## 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.

12

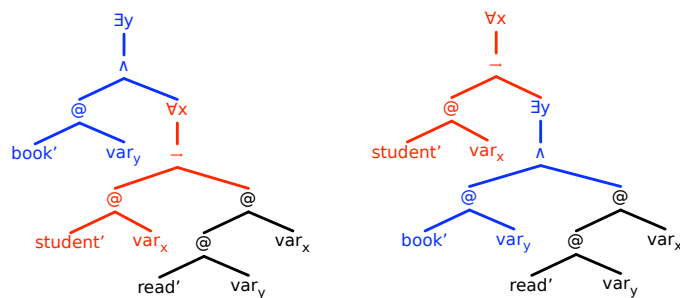
# Scope Underspecification

- Basic observation:
  - The readings of scopally ambiguous sentences are made up of the same set of constants, connectives and variables, and differ only in their structure
  - “Every student reads a book.”
    - $\forall x(\text{student}'(x) \rightarrow \exists y(\text{book}'(y) \wedge \text{read}'(x,y)))$
    - $\exists y(\text{book}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{read}'(x,y)))$
- Basic idea:
  - Consider semantic representations as trees
  - Describe sets of trees using dominance graphs

13

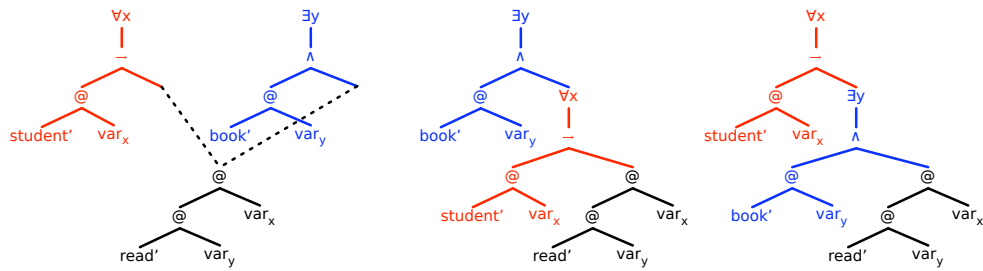
# Scope Underspecification: the Idea

- “Every student reads a book”
  - $\forall x(\text{student}'(x) \rightarrow \exists y(\text{book}'(y) \wedge \text{read}'(y)(x)))$
  - $\exists y(\text{book}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{read}'(y)(x)))$
- Readings as trees:



14

# Scope Underspecification: the Idea



(USR)

(Readings as trees)

## Outline

- Terms as trees
- Dominance graphs as descriptions of sets of trees
- Semantics construction with dominance graphs
- Things you can do with dominance graphs



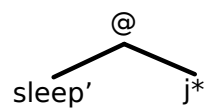
## Terms as Trees

- 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 correspond to leaf labels
  - Variables  $x$  correspond to leaves with label  $\text{var}_x$ .
    - (Alternatively: binding edges, see slides at the end)

17

## Terms as Trees

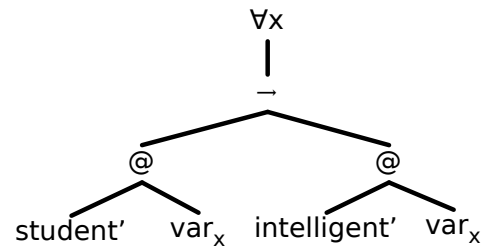
- $\text{sleep}'(j^*)$



18

## Terms as Trees

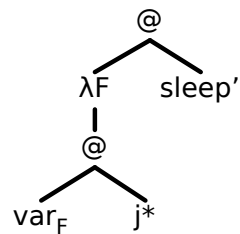
- $\forall x(\text{student}'(x) \rightarrow \text{intelligent}'(x))$



19

## Terms as Trees

- $(\lambda F.F(j^*))(\text{sleep}')$



20

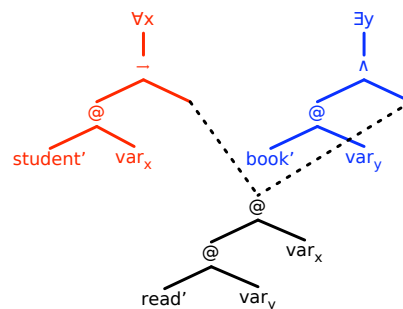
# Dominance Graphs

- Informally, a dominance graph is a directed graph which consists of trees (or “tree fragments”) which are connected by dominance edges.
- For modeling scope underspecification, we consider labeled dominance graphs, i.e. pairs of a dominance graph and a partial node labeling function  $L$ 
  - $L$  must be defined on all non-leaves of the tree fragments
  - Leaves may be unlabelled
- Terminology:
  - Unlabeled leaves are called “holes”

21

# An Example

- Three tree fragments that informally correspond to
  - $\forall x(\text{student}'(x) \rightarrow \dots)$
  - $\exists y(\text{book}'(y) \wedge \dots)$
  - $\text{read}(y)(x)$
- The two upper fragments each have one hole.
- The holes have outgoing dominance edges to the root of the lower fragment.



22

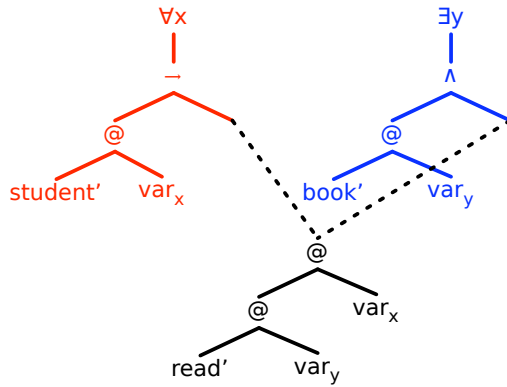
## Dominance Graphs

- More formally, a dominance graph is defined as a directed graph  $G = (V, E \cup D)$  where  $V$  is a set of nodes and
  - $E$  is a set of “tree edges” (solid edges)
  - $D$  is a set of “dominance edges” (dotted edges)
- The subgraph  $(V, E)$  must be a forest, i.e. it is acyclic and no node has more than one incoming (tree-) edge.
- Labelled dominance graphs:  $G = (V, E \cup D, L)$  where  $L$  is a partial labelling function mapping nodes in  $V$  to labels of some signature  $\Sigma$ .

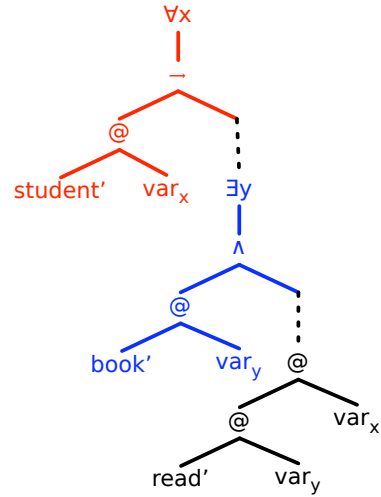
## Solved Forms

- A dominance graph  $G$  can be seen as a description of a set of trees into which  $G$  can be embedded.
- These trees can be represented by the solved forms of  $G$ .
- A dominance graph  $G_S$  is said to be **in solved form** if it is a forest, i.e. no node has more than one incoming dominance edge.
- $G_S$  is **a solved form of** some dominance graph  $G$  if
  - $G_S$  is in solved form
  - $G_S$  and  $G$  differ at most in their dominance edges, and
  - if nodes  $X$  and  $Y$  are connected by a dominance edge in  $G$ , then there is a directed path from  $X$  to  $Y$  in  $G_S$ .

# An Example

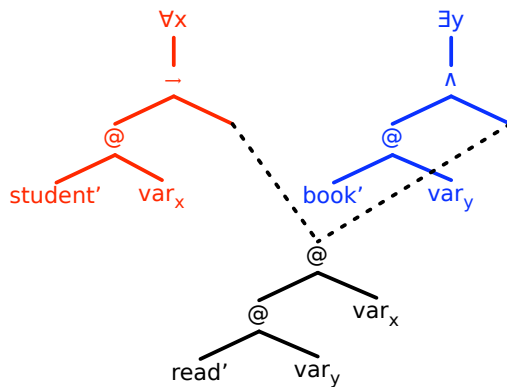


(dominance graph)

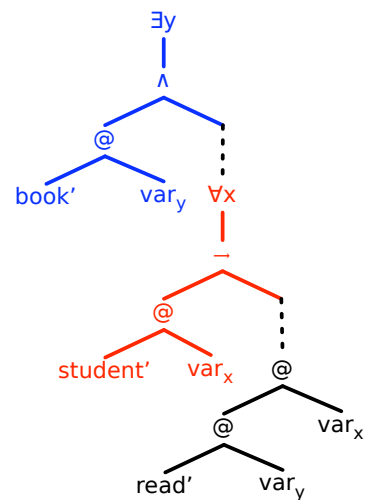


(solved form)

# An Example



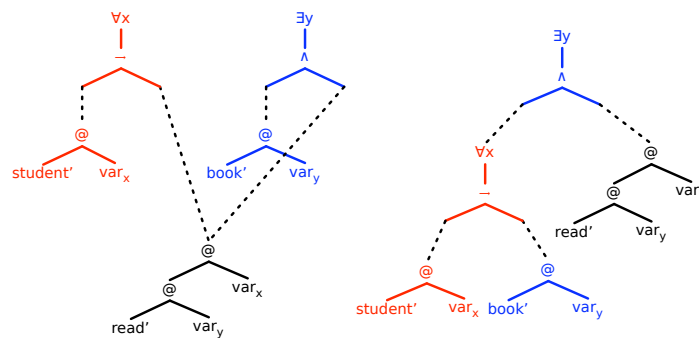
(dominance graph)



(solved form)

## Not a solved form of ...

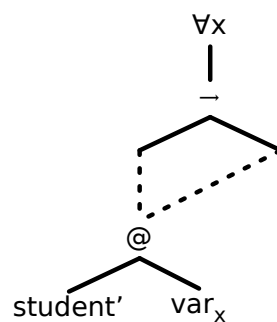
- The dominance graph on the right is in solved form, but it is not a solved form of the graph on the left
  - the dominance edge from the  $\forall$ -fragment to the read-fragment is not realised as reachability in the right graph



27

## An unsolvable graph

- Not all dominance graphs have a solved form



28

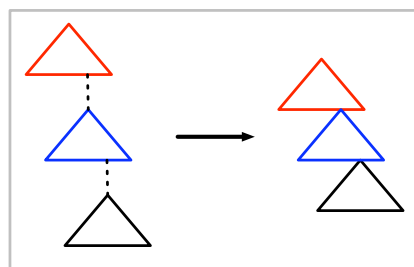
## Solved Forms – Remark #1

- As said earlier, the solved forms of a dominance graph  $G$  represent the trees into which  $G$  can be embedded. (these trees are the solutions of  $G$ )
- For modeling scope underspecification, we are usually interested in a particular class of solutions called constructive solutions.
- Not every solved form corresponds to a constructive solution, but recent studies indicate that the solved forms of all “linguistically relevant” graphs all correspond to constructive solutions.

29

## Solved Forms – Remark #1

- Basic idea: if each hole of a solved form has exactly one outgoing dominance edge, ...
- then one can obtain a constructive solution by identifying the two ends of each dominance edge.



30

## Solved Forms – Remark #2

- We can distinguish various sub-classes of dominance graphs, depending on which kinds of dominance edges are permitted
  - In [normal dominance graphs](#), dominance edges are only permitted between holes and roots.
  - [Weakly normal dominance graphs](#) additionally permit root-to-root dominance edges (but not hole-to-root edges)
- Note that for dominance graphs with hole-to-root dominance edges, we need a more general definition of a solved form.

## Where are we now?

- Formulas (readings of natural language sentences) can be seen as trees.
- These trees can be described by dominance graphs ...
- in the sense that the solved forms of a graph correspond to the readings of the underlying sentence.
- Next step: Semantic construction for dominance graphs.



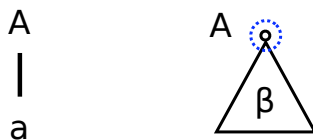
## Semantics Construction: Principles

- For every node in the syntax tree, we derive a dominance graph as follows:
  - 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.

33

## Lexicon access

- Rule of lexical nodes:



The semantic representation (sub-graph)  $\beta$  for a word "a" is supplied by the lexicon.

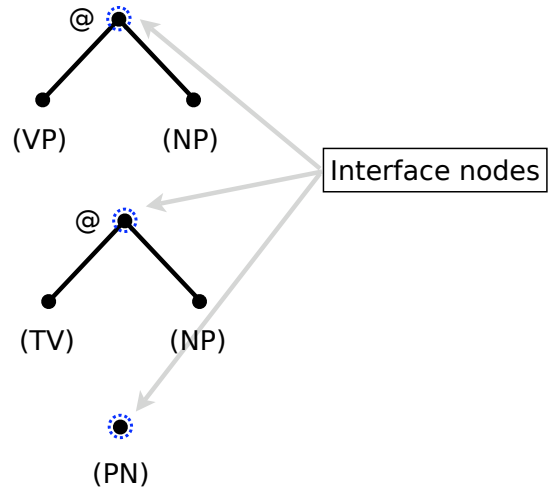
34

## Semantics construction rules

- $S \rightarrow NP VP$

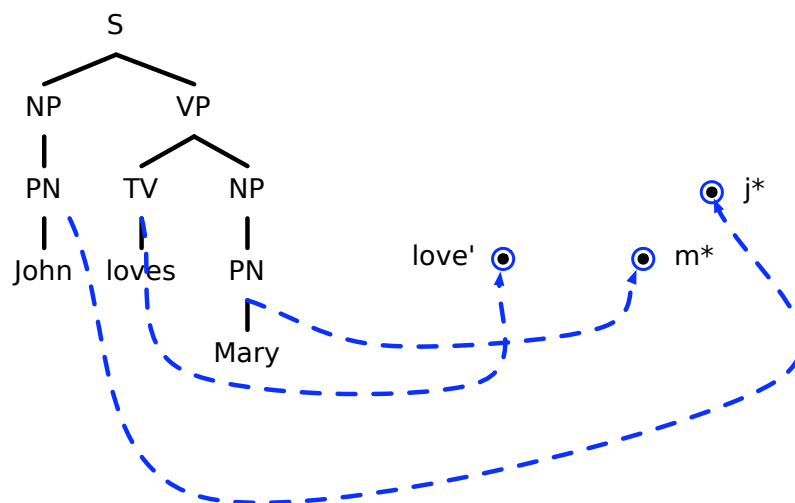
- $VP \rightarrow TV NP$

- $NP \rightarrow PN$



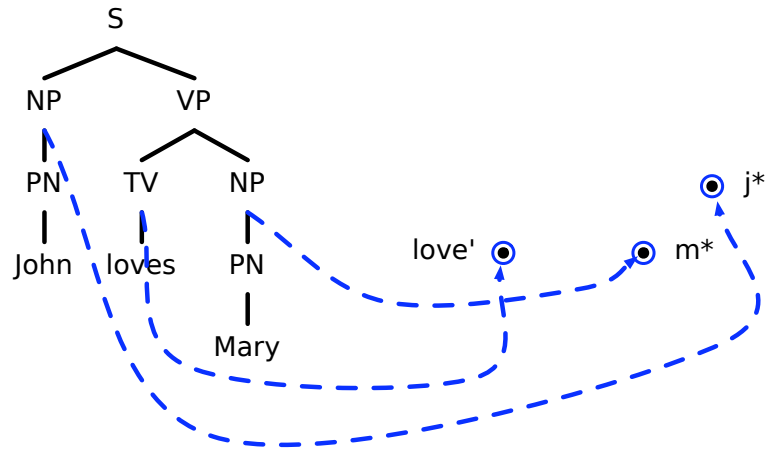
35

## An Example



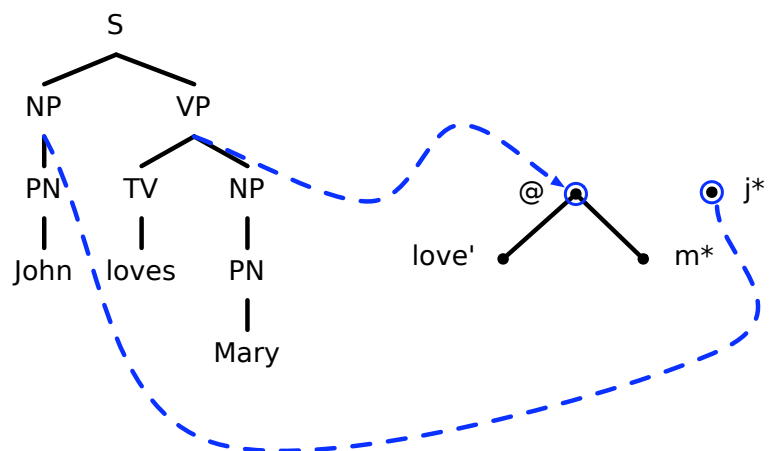
36

# An Example



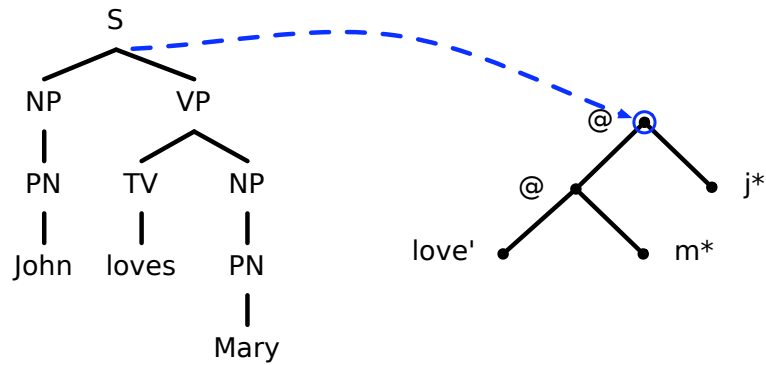
37

# An Example



38

## An Example

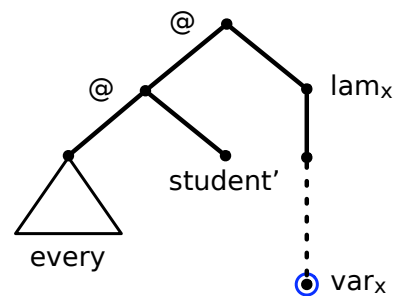


Semantic representation:  $\text{love}'(m^*)(j^*)$

39

## Quantifiers

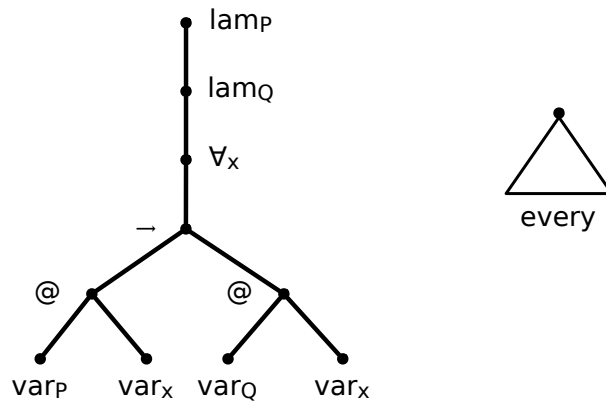
- The graph for a quantifier noun phrase contains a variable node and its binder.
- The interface node of the graph is the node that represents the variable (of type e)



40

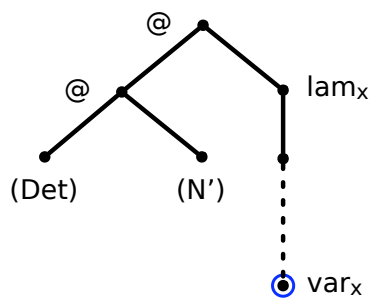
# Constructing Graphs for Quantifiers

- Lexicon entry for determiners (here “every”):

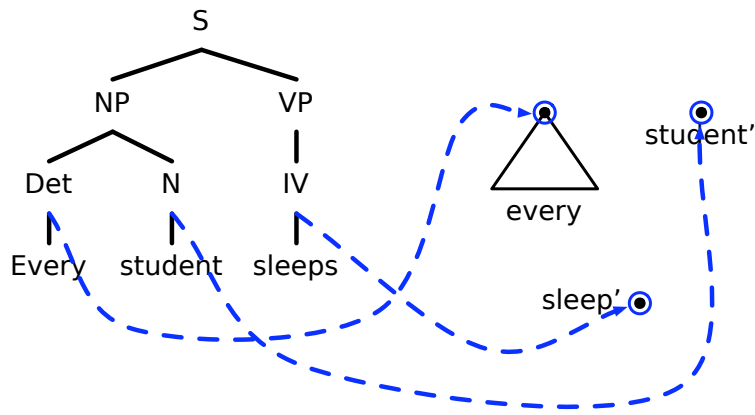


# Constructing Graphs for Quantifiers

- Syntax rule: NP → Det N'

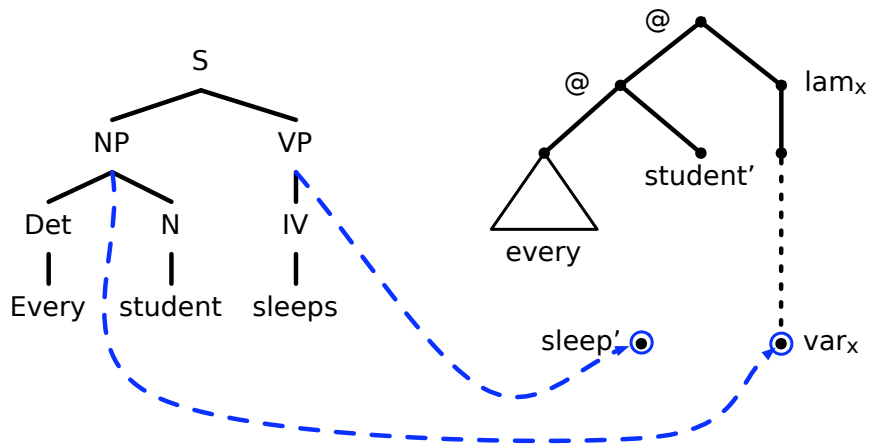


# An Example



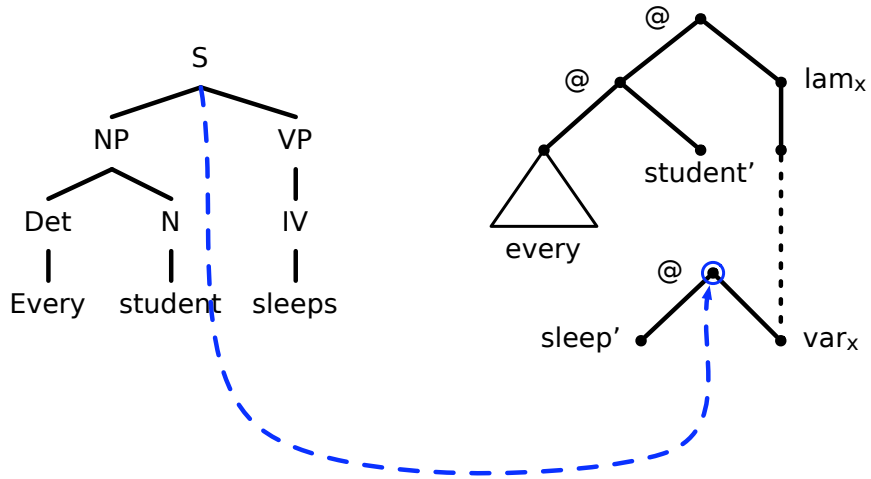
43

# An Example



44

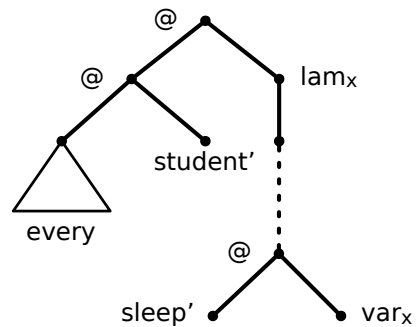
# An Example



45

## After "Normalisation"

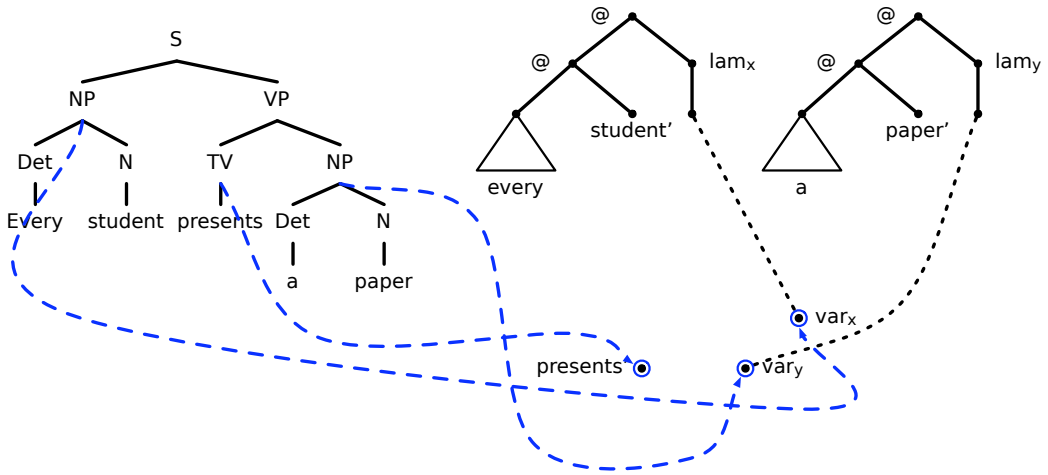
- In a final step, we replace dominance edges pointing into fragments by dominance edges pointing to the root of the fragment.



- Corresponding formula:
  - $(\lambda P \lambda Q \forall y [P(y) \rightarrow Q(y)])(student')(\lambda x sleep'(x))$
  - $\Rightarrow_{\beta} \forall y [student'(y) \rightarrow sleep'(y)]$

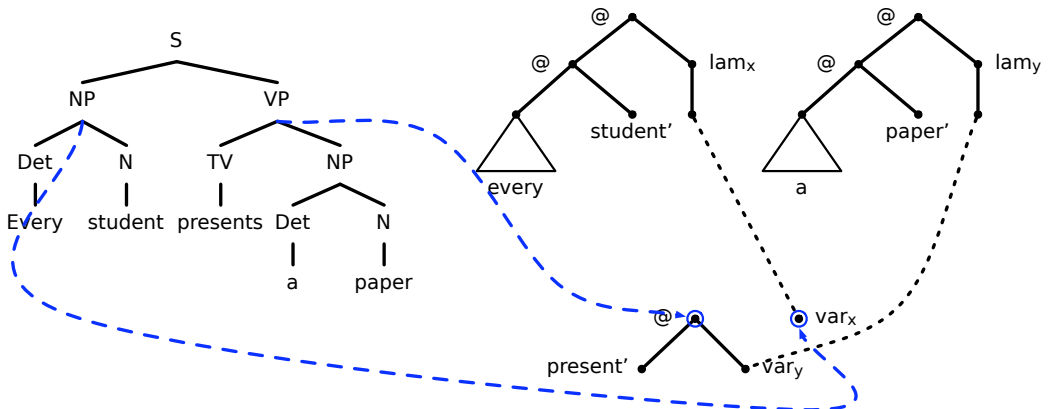
46

# Scope Ambiguities



47

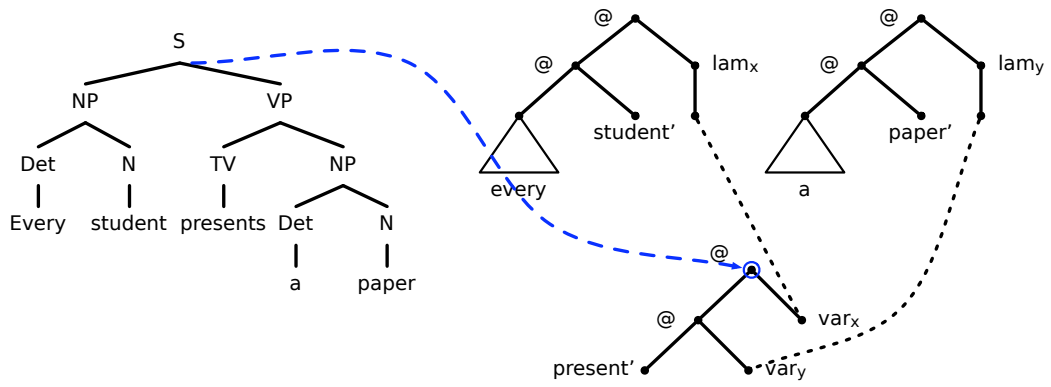
# Scope Ambiguities



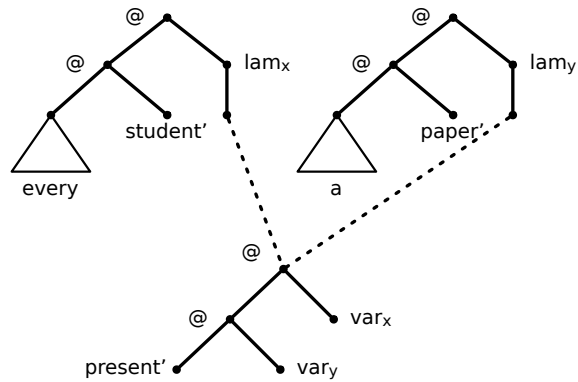
48



# Scope Ambiguities



# Scope Ambiguities



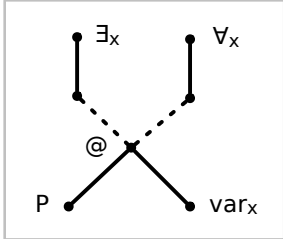
## An observation

- 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 NPs

- 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 Nested Cooper 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 Nested Cooper Storage.
- 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.”

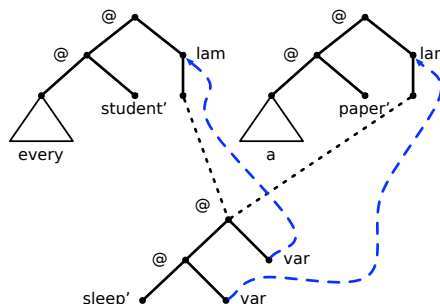
## Representing variable binding

- As in type theory, we use variable names to model the relation between a binder ( $\lambda$ ,  $\forall$ ,  $\exists$ ) and the variables bound by it.
- 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.
- Binding edges are a cleaner and simpler way of doing it.

53

## Using binding edges

- Assume a third type of edges: binding edges
- All variables have label "var," and labels representing lambda-binders as "lam" (quantifiers analogously)
- The graph for "every student presents a paper" with binding edges:



54

## Algorithms

- Deciding solvability
  - given a dominance graph  $G$ , has  $G$  as solved form?
- Enumerating solved forms
  - given a dominance graph  $G$ , enumerate the (minimal) solved forms of  $G$ .
- Eliminating redundant readings
  - Strengthen an USR  $G$  such that it has fewer readings, but still contains a representative for each equivalence class of  $G$ .

## 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.