Semantic Theory

Scope Underspecification

Manfred Pinkal
Stefan Thater

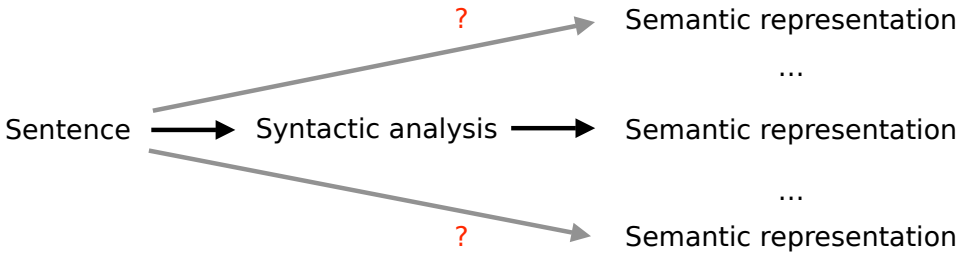
2009-07-07

Scope ambiguities

- Sentences with two or more scope bearing operators such as quantifiers or negations are often ambiguous:
- *Every student presents a paper.*
  - ∀x(student'(x) → ∃y(paper'(y) ∧ present'(x,y)))
  - ∃y(paper'(y) ∧ ∀x(student'(x) → present'(x,y)))
- *Every student didn't pay attention.*
  - ∀x(student'(x) → ¬pay-attention'(x))
  - ¬∀x(student'(x) → 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?



Nested Cooper Storage

- "Every student presents a paper."
  - $(\text{pres}^*(x_2)(x_1), \{ (\lambda P\, \forall x[\text{stud}'(x) \rightarrow P(x)], \emptyset)_1, \quad [= (ES, \emptyset)_1]$
    $(\lambda Q\, \exists y[\text{paper}'(y) \wedge Q(y)], \emptyset)_2 \quad \}) \quad [= (AP, \emptyset)_2]$
- Retrieval:
  - $ES(\lambda x_2(AP(\lambda x_1(\text{present}^*(x_2)(x_1)))))$
    $\Rightarrow_\beta \exists y(\text{paper}'(y) \wedge \forall x(\text{student}'(x) \rightarrow \text{present}^*(y)(x)))$
  - $AP(\lambda x_1(ES(\lambda x_2(\text{present}^*(x_2)(x_1)))))$
    $\Rightarrow_\beta \forall x(\text{student}'(x) \rightarrow \exists y(\text{paper}'(y) \wedge \text{present}^*(y)(x)))$

Nested Cooper Storage

- Nested Cooper Storage allows to derive distinct readings on the basis of a single syntactic analysis.
  - Problem #1 solved (… to a certain extent, see below)
- But Nested Cooper Storage has its own problems:
  - Non-determinism: storage vs. application at NP-nodes, retrieval at sentence 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

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

Always enumerate readings?

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

Always enumerate readings?

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

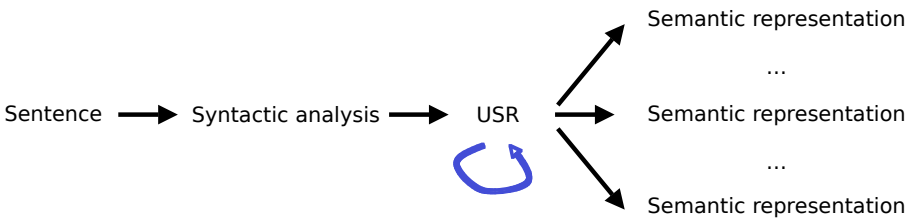
Equivalent readings

- *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”

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 often wasted.

Underspecification: the big picture



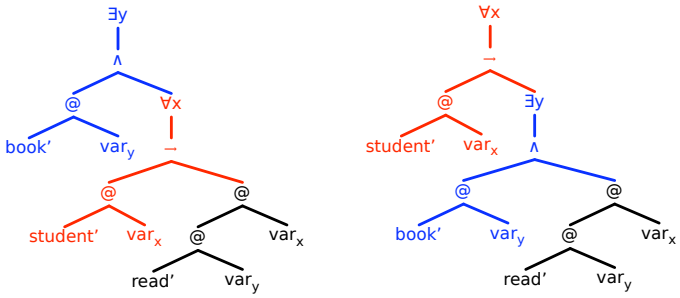
- Derive a single underspecified semantic representation (USR) from the syntactic analysis.
- Perform inferences on USR to eliminate readings excluded by the context.
- Enumerate readings by need.

Scope Underspecification

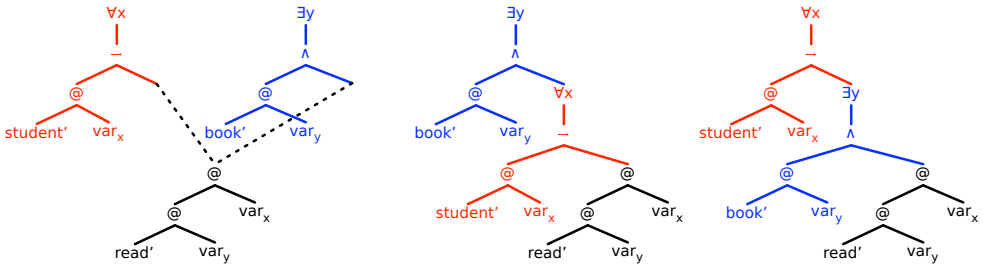
- Basic observation: The readings of scopally ambiguous sentences are made up of the same set of logical symbols, and differ only in their structure.
- *Every student reads a book.*
  - ∀x(student'(x) → ∃y(book'(y) ∧ read'(x,y)))
  - ∃y(book'(y) ∧ ∀x(student'(x) → read'(x,y)))
- Basic idea:
  - Consider semantic representations as trees
  - Describe sets of trees using dominance graphs

Scope Underspecification: Idea

- "Every student reads a book"
  - ∀x(student'(x) → ∃y(book'(y) ∧ read'(y)(x)))
  - ∃y(book'(y) ∧ ∀x(student'(x) → read'(y)(x)))
- Readings as trees:



Scope Underspecification: Idea



(USR) (Readings as trees)

Outline

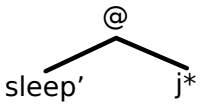
- Terms as trees
- Dominance graphs as descriptions of sets of trees
- Semantics construction with dominance graphs
- Things you can do with dominance graphs

Terms as Trees

- Terms (and formulas) of type theory have a natural reading as trees.
- Application M(N) is the tree @(M,N)
- Abstraction λx.M is the tree lam(M)
  - Quantifiers analogously
- Constant symbols correspond to leaf labels
- Variables x correspond to leaves with label var_x.
  - (Alternatively: binding edges, see slides at the end)

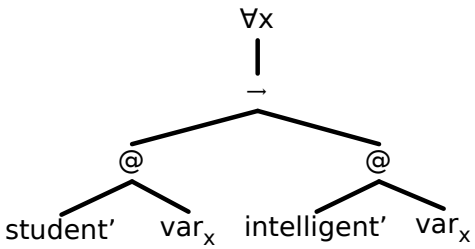
Terms as Trees

- sleep’(j*)



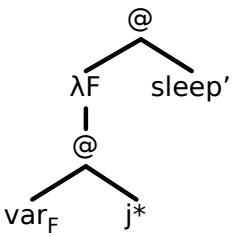
Terms as Trees

- ∀x(student’(x) → intelligent’(x))



Terms as Trees

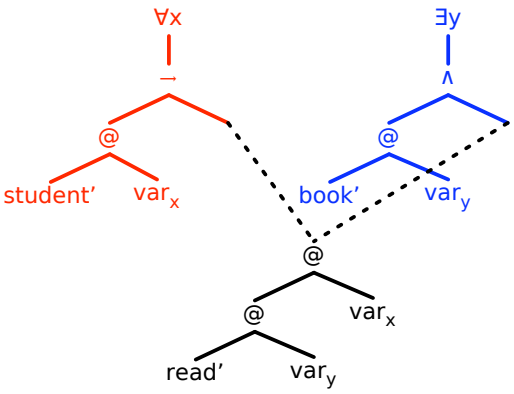
- (λF.F(j*))(sleep')



Dominance graphs

- A (normal) dominance graph G is a labelled directed graph with two kinds of edges: tree edges and dominance edges.
- The graph restricted to tree edges is a collection of trees.
- Unlabelled nodes must be leaves of these trees
  - Terminology: unlabelled node are called holes
- Each tree contains at least one labelled node (non-hole).
- Dominance edges start at holes and point to non-holes

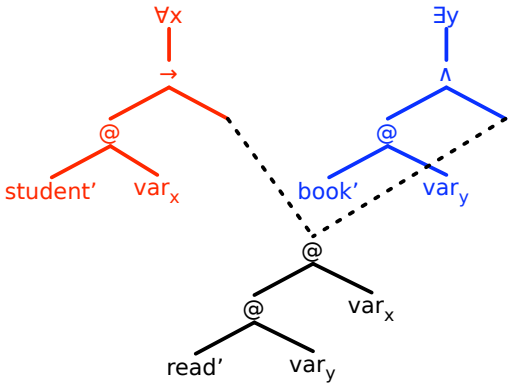
An Example



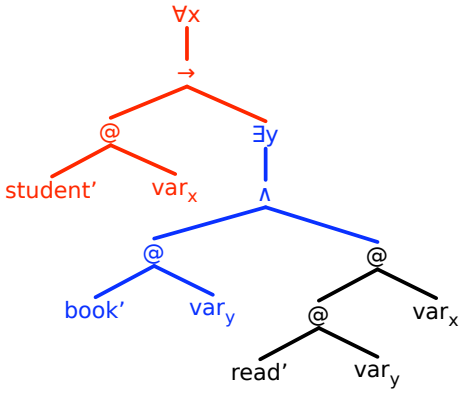
Readings ≈ "Pluggings"

- Dominance graphs can be seen as compact descriptions of sets of trees (i.e., readings).
- The readings described by a dominance graph can be obtained by "plugging" tree fragments into each other
  - identify holes and roots
  - result must be a tree.
- A plugging must respect the "dominance wishes" encoded by dominance edges:
  - if there is a dominance edge from node X to node Y in a dominance graph,
  - then node X must dominate (be an ancestor of) node Y in the resulting tree.

An Example

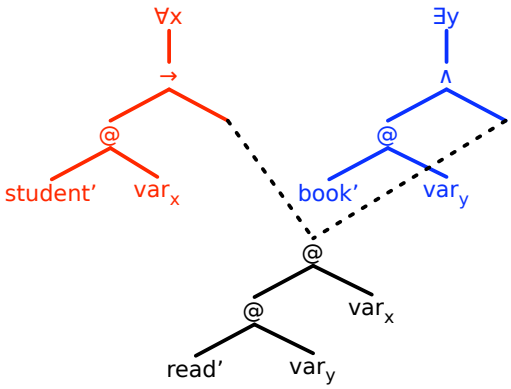


(dominance graph)

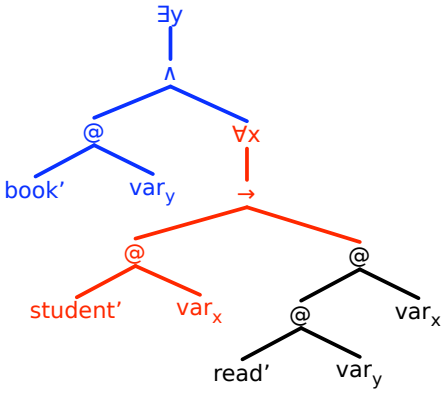


(reading #1)

An Example

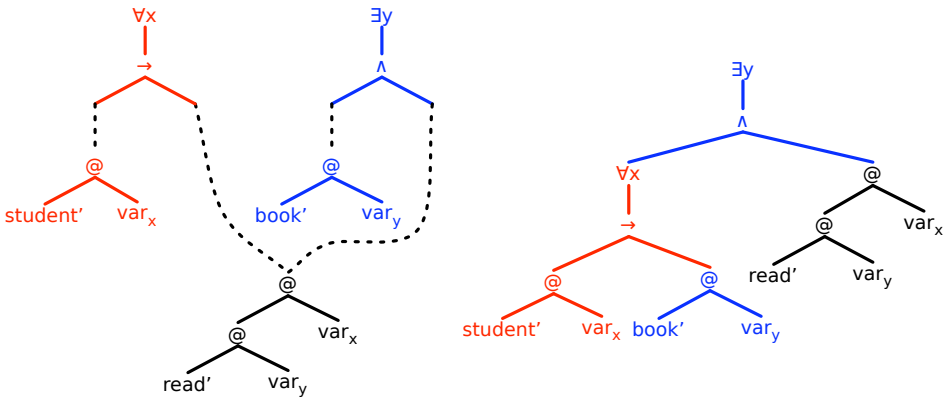


(dominance graph)

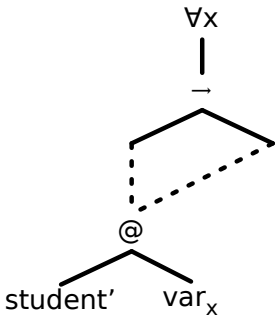


(reading #2)

Not a reading ...



An unsolvable graph (no reading)

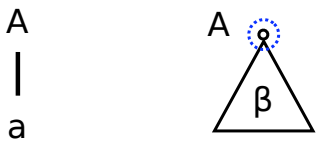


Semantics Construction

- 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.

Lexicon access

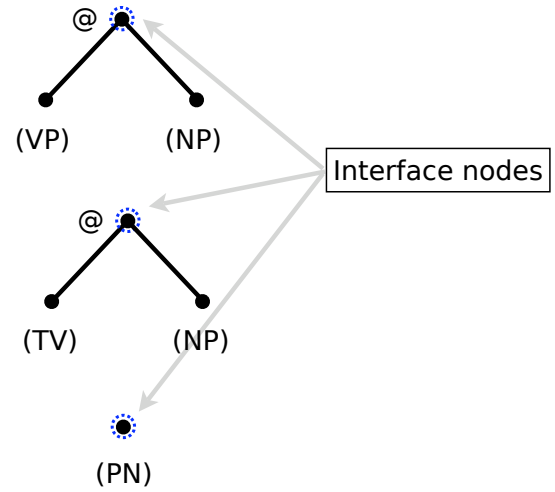
- Rule of lexical nodes:



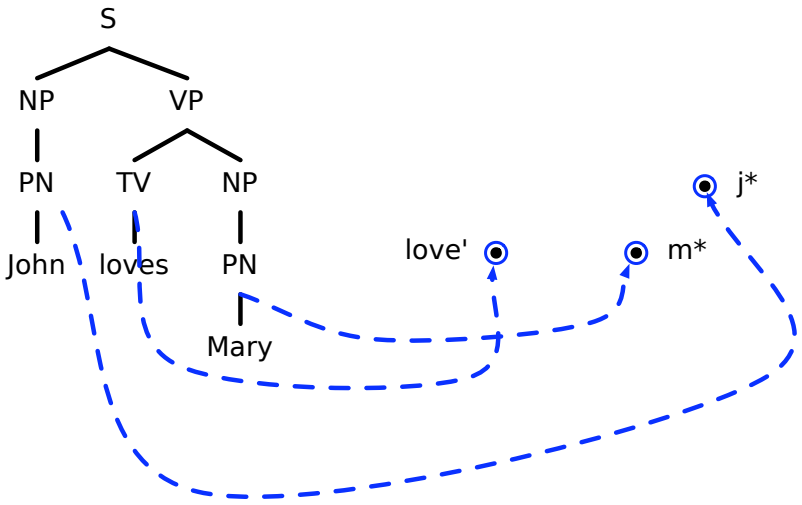
The semantic representation (sub-graph) β for a word “a” is supplied by the lexicon.

Semantics construction rules

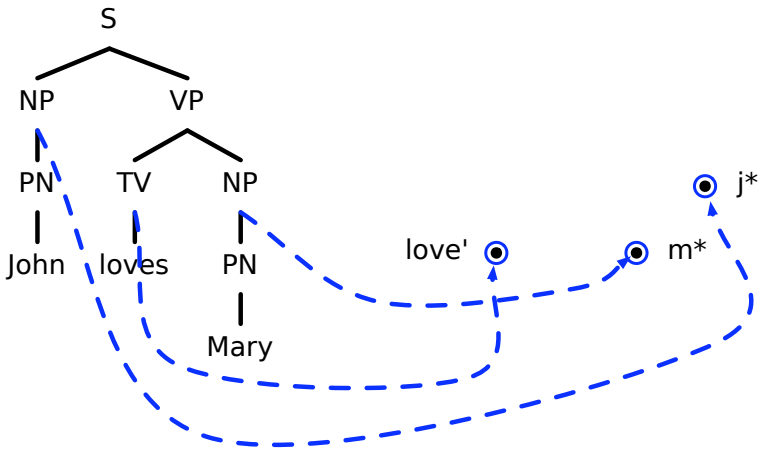
- S → NP VP
- VP → TV NP
- NP → PN



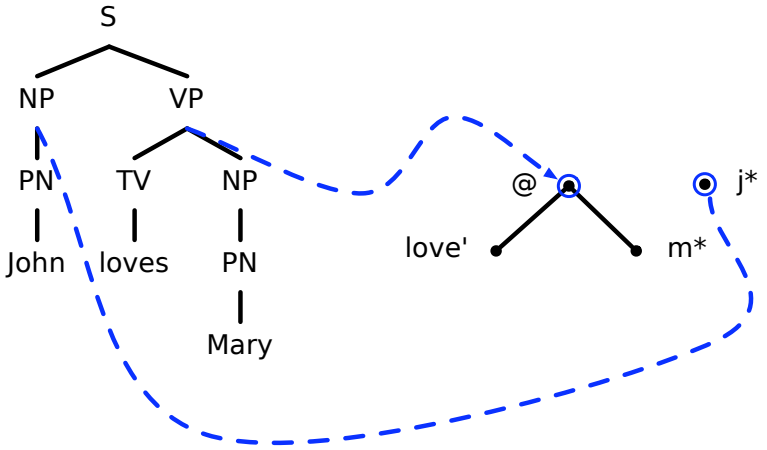
An Example



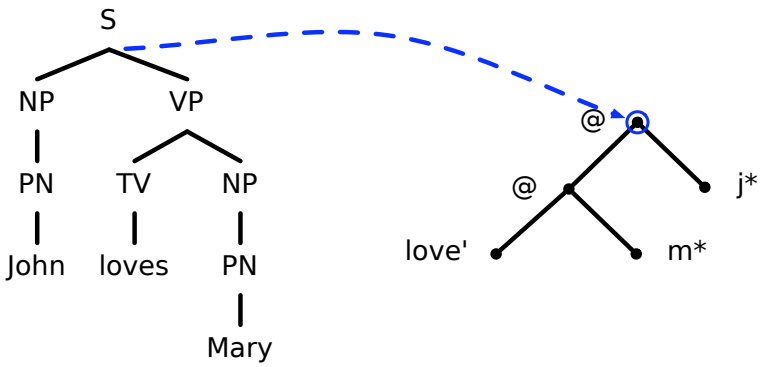
An Example



An Example



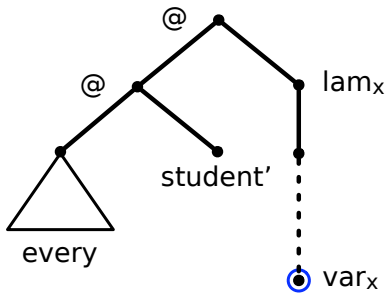
An Example



Semantic representation: love'(m*)(j*)

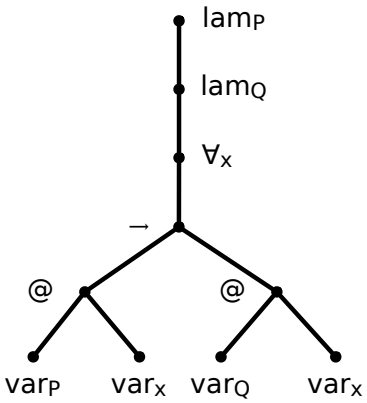
Quantifiers

- 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)



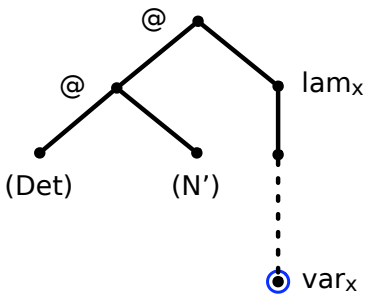
Constructing Graphs for Quantifiers

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

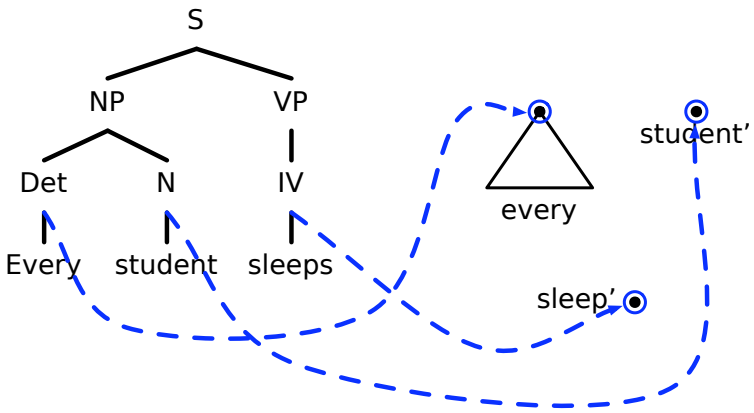


Constructing Graphs for Quantifiers

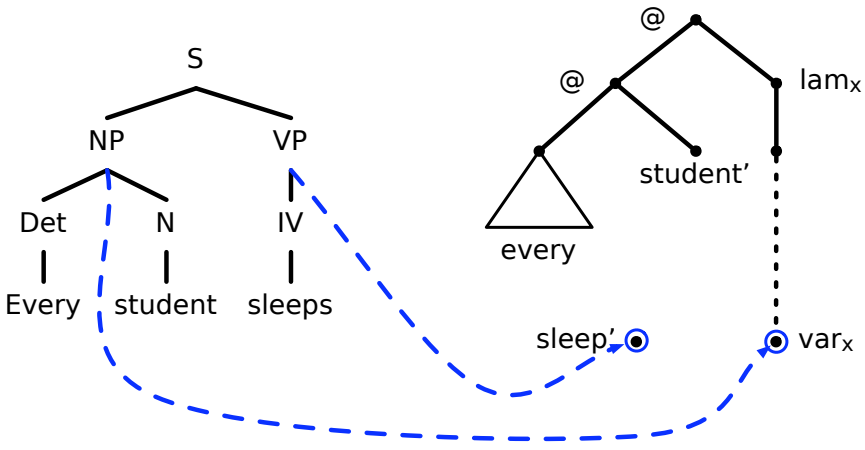
- Syntax rule: NP → Det N’



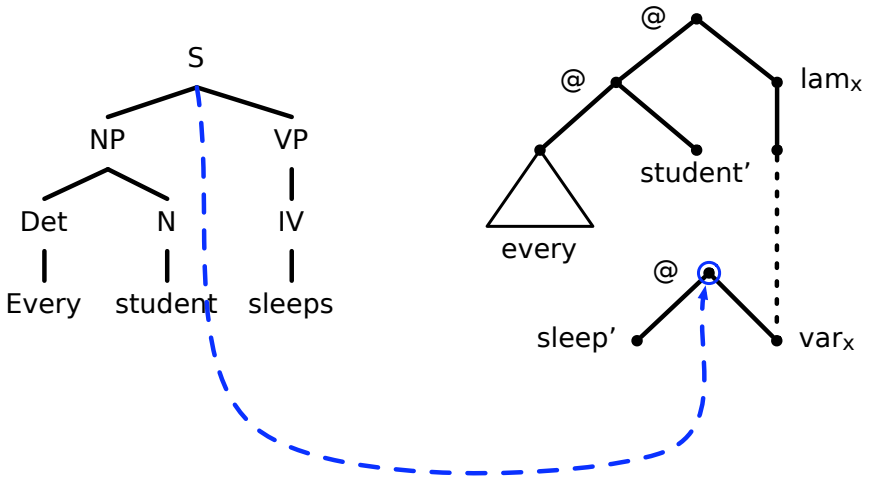
An Example



An Example

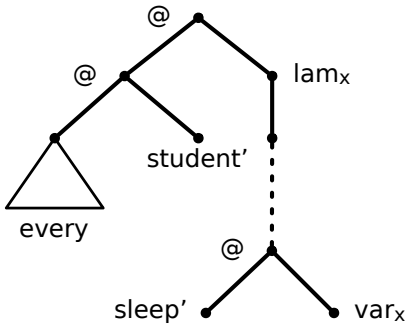


An Example



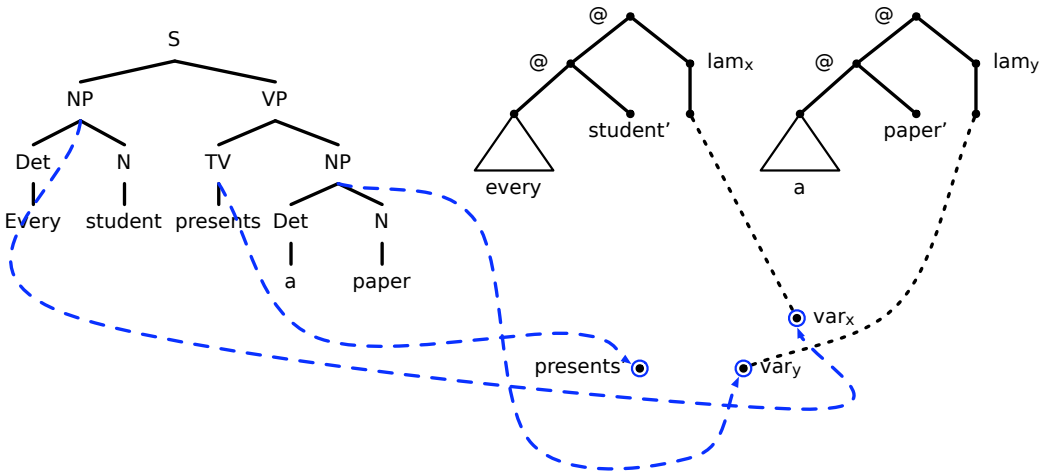
Result

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

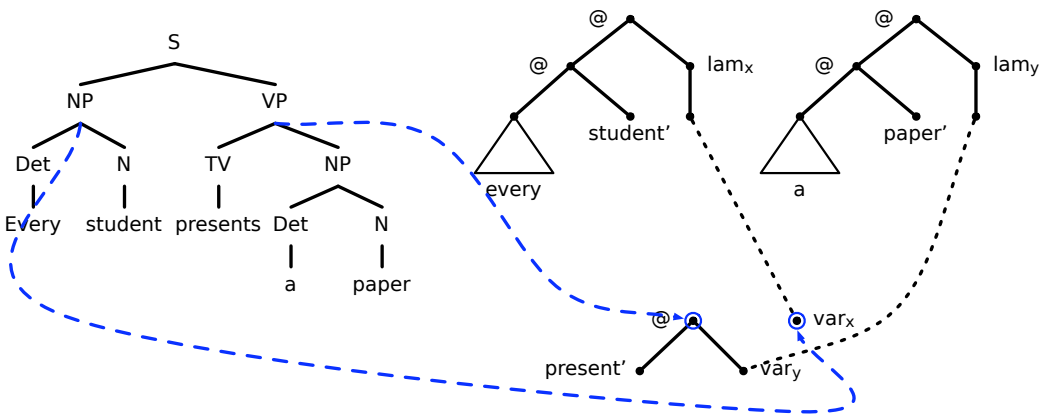


- Corresponding formula:
  - $(\lambda P \lambda Q \forall y[P(y) \rightarrow Q(y)])(\text{student'})(\lambda x\ \text{sleep'}(x))$
  - $\Rightarrow_\beta \forall y[\text{student'}(y) \rightarrow \text{sleep'}(y)]$

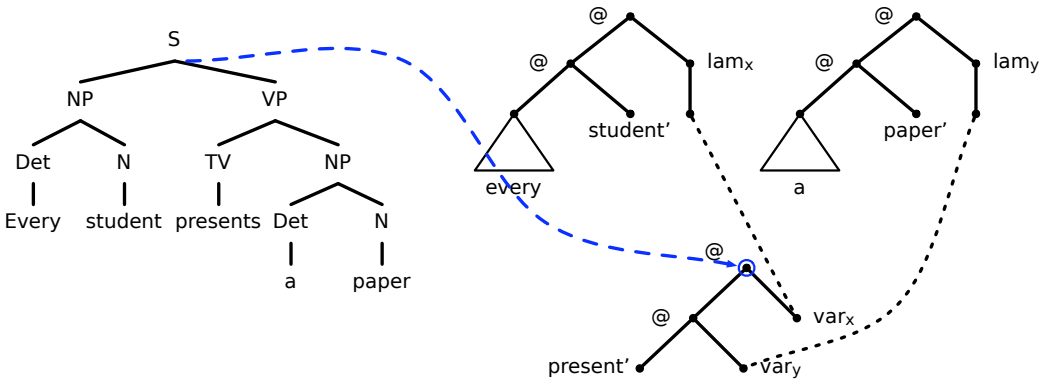
Scope Ambiguities



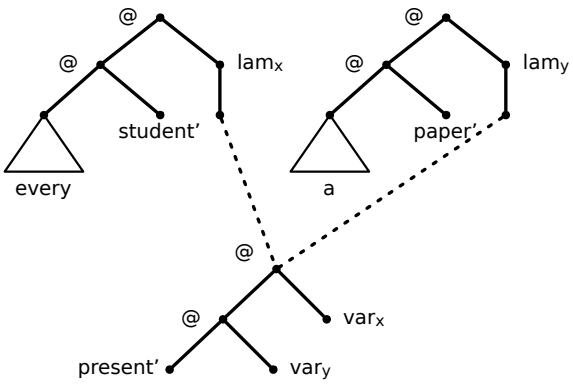
Scope Ambiguities



Scope Ambiguities



Scope Ambiguities



An observation

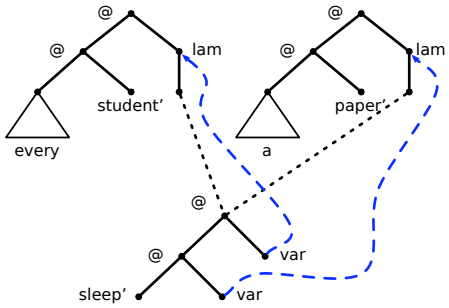
- We still use type theory as the object language, i.e. the language of semantic representations.
- We need fewer lambda operators for “construction bookkeeping.” Semantics construction is done mainly by plugging USRs into each other directly.
- This makes us more flexible in our choice of semantic representations:
  - can use e.g. bill* of type e for proper names
  - can use e.g. present* of type (e,(e,t)) 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 ((e,t),t), 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.

Using binding edges

- Avoid unintended “variable capturing”
- 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:



Algorithms

- Algorithms for dominance graphs don’t compute pluggings, but rather so-called “solved forms”
- Solved forms are “tree-shaped” dominance graphs that respect dominance egdes.

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 often not needed.
- 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.