

# Computational Linguistics

## Algorithms for Scope Underspecification

Stefan Thater

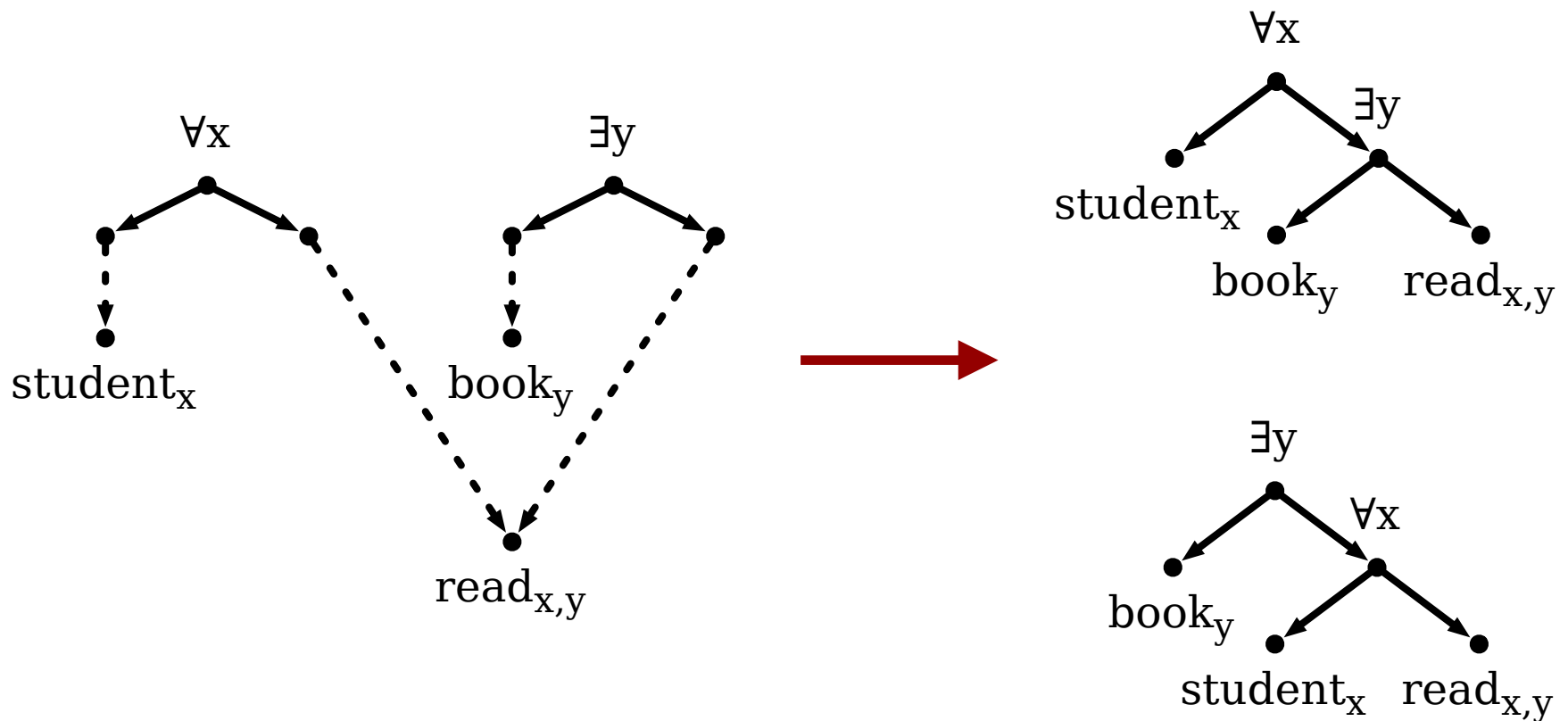
FR 4.7 Allgemeine Linguistik (Computerlinguistik)

Universität des Saarlandes

Summer 2011



# What this lecture is about



# Overview

- Some basic assumptions about sentence meaning
- Scope ambiguities
- Modelling scope ambiguities with dominance graphs
- An algorithm for solving dominance graphs

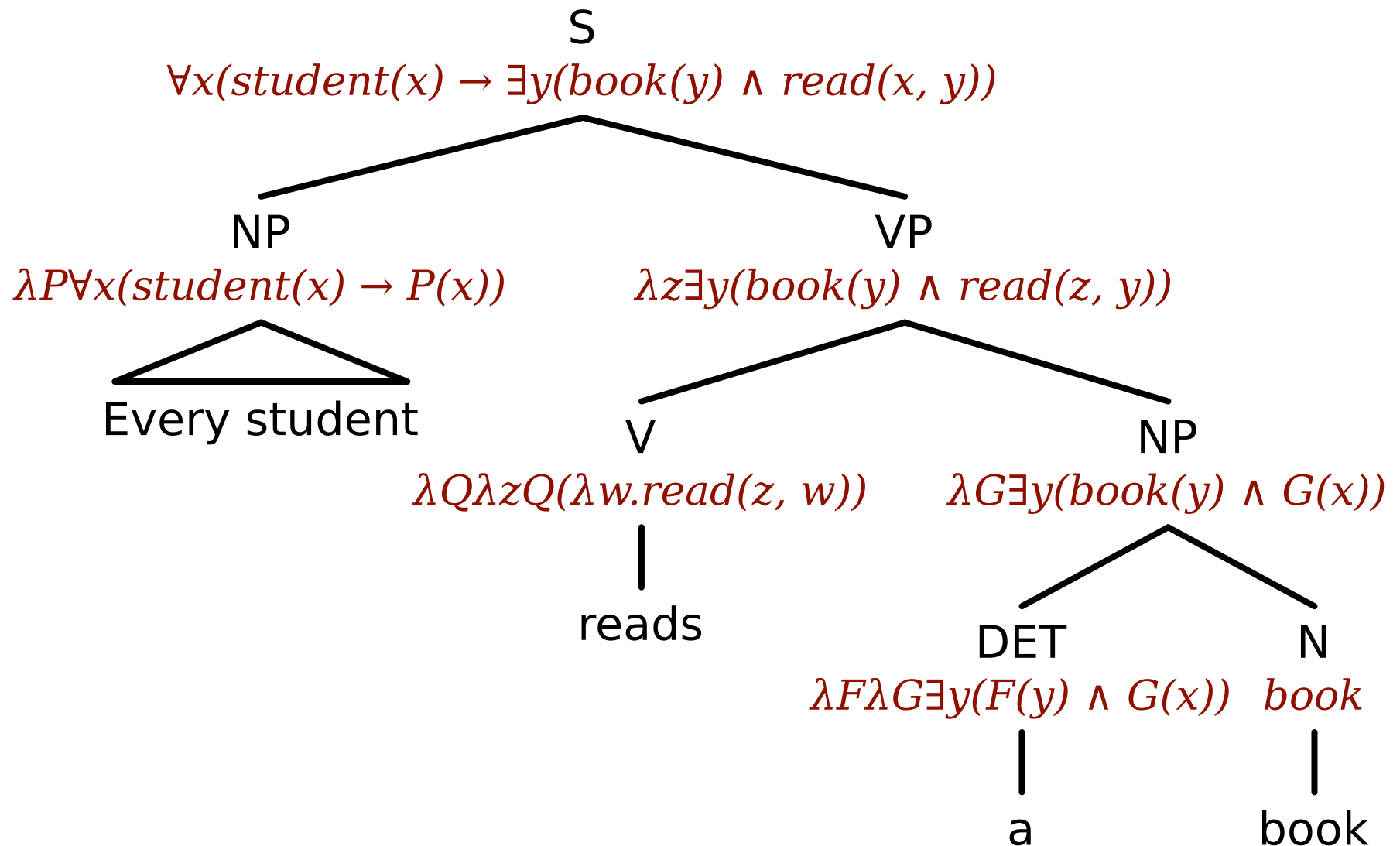
# Sentence meaning: Assumptions

- **Truth-functional interpretation:** The meaning of a declarative sentence is given by its truth conditions
- $\Rightarrow$  we can represent the meaning of natural language sentences by logical formula that “capture” the truth-conditions of the original sentence.
- *Every student works*
  - $\mapsto \forall x(\text{student}(x) \rightarrow \text{work}(x))$

# Sentence meaning: Assumptions

- **Compositionality:** The meaning of a complex expression is a function of the meanings of its parts and of the syntactic rules by which they are combined
- **Compositional semantic construction** based on the syntactic tree of the natural language expression
  - The semantic lexicon assigns meaning representations to lexical (leaf) nodes of the syntax tree.
  - The semantic representation of an inner node is computed by combining the representations of its child nodes.

# Compositional Semantics Construction



# Ambiguities

- Natural language is ambiguous: a sentence can have more than one interpretation (“reading”).
- **Lexical ambiguities**
  - *Iraqi head seeks arms*
- **Structural ambiguities**
  - *Enraged cow injures farmer with axe*
  - *The salesman sold the dog biscuits*
  - *Every student reads a book*

# Scope Ambiguities

- Scope ambiguities can arise when a sentence contains two or more quantifiers and/or other scope-taking operators (negations, modal expressions, etc.)
- *Every student reads a book*
  - $\forall x(\text{student}(x) \rightarrow \exists y(\text{book}(y) \wedge \text{read}(x, y)))$  [EA]
  - $\exists y(\text{book}(y) \wedge \forall x(\text{student}(x) \rightarrow \text{read}(x, y)))$  [AE]



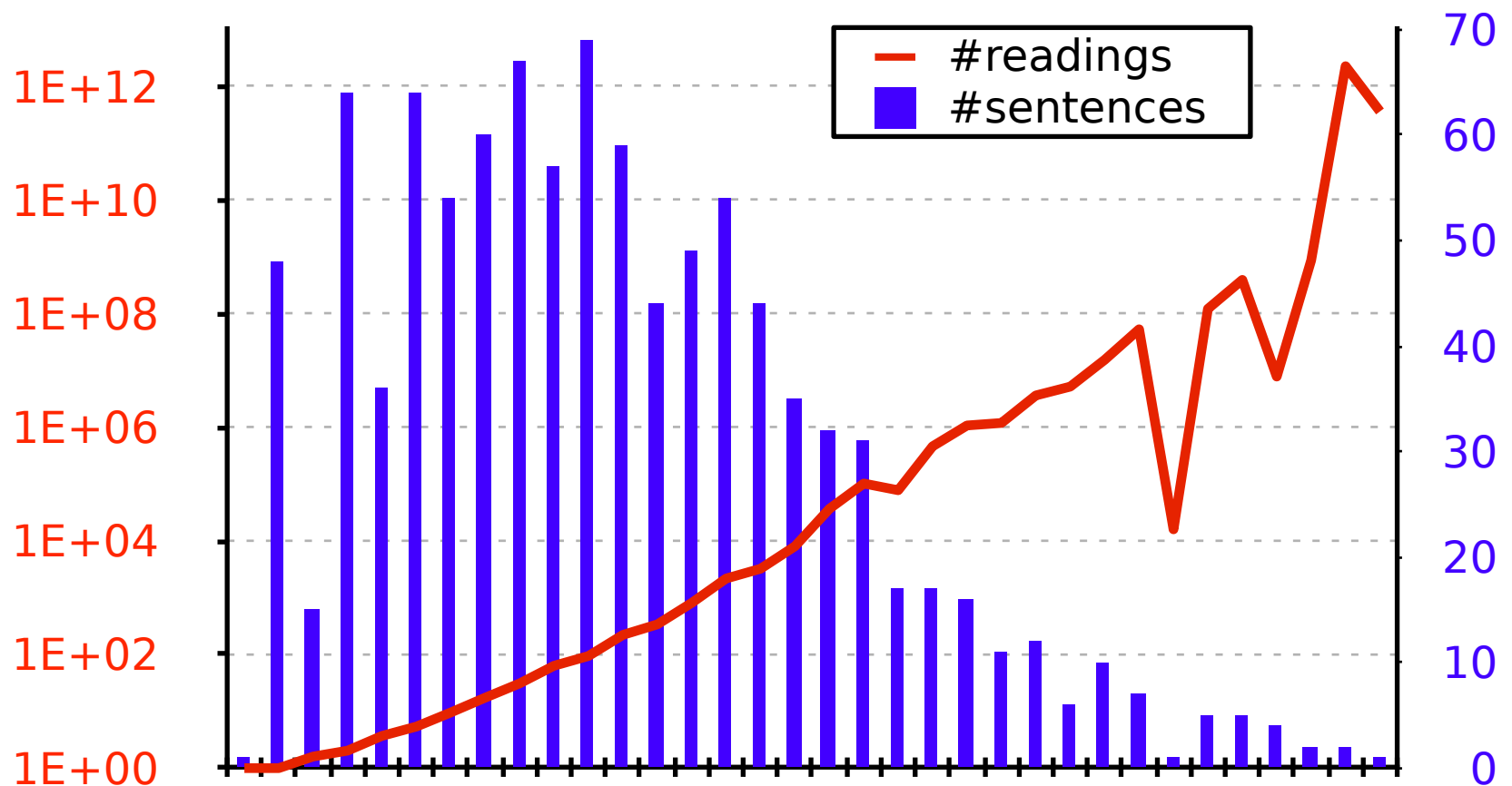
# Scope Ambiguities: Problem #2

- **Combinatorial explosion of readings:** The number of readings of a sentence can grow exponentially in the number of scope-taking operators it contains.
- *Every student reads a book* ⇒ 2 readings
- *Every student reads a book about an interesting topic* ⇒ 6 readings (some of which are logically equivalent)
- *We quickly put up the tents in the lee of a small hillside and cook for the first time in the open* ⇒ 480 readings\* (most of which are logically equivalent)

\* according to the English Resource Grammar

# Scope Ambiguities: Problem #2

- Number of readings in a “real-live” corpus according to the English Resource Grammar (Koller & al., ACL 2008)

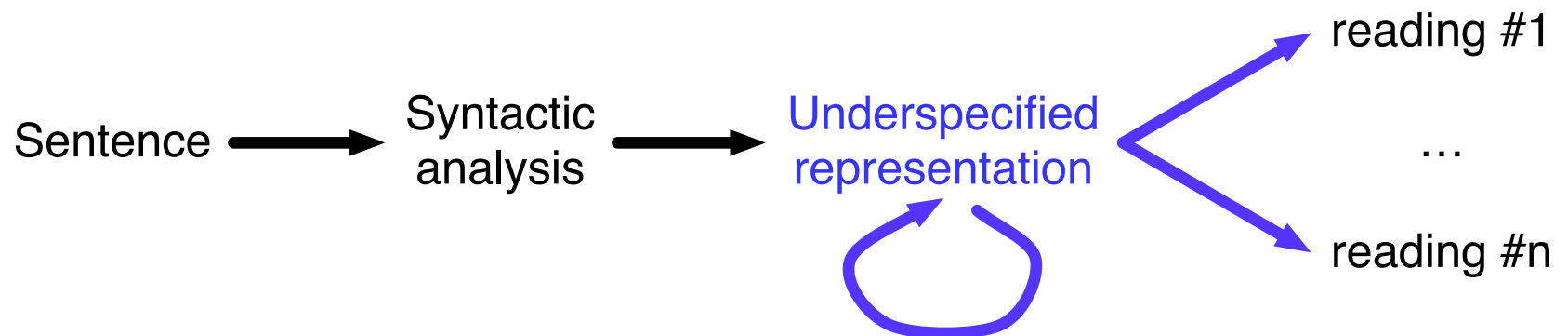


# Coping with scope: Options

- 1)** Ignore scope ambiguities
  - for instance, always compute the “surface scope” reading
- 2)** Enumerate all readings and then select the “right” one
  - we need more complex semantics construction rules and a method to choose the “right” reading
  - computationally very expensive, since sentences can easily have millions of readings
- 3)** Use scope underspecification

# Scope Underspecification

- Don't explicitly enumerate readings
- Instead, represent all readings of a sentence by a single compact underspecified representation (USR).
- The individual readings can be enumerated from the underspecified representation if needed (this lecture).
- We can perform inferences directly on the level of underspecified representations (next lecture).



# A notational change

- *Every student reads a book*
  - $\forall x(\text{student}(x), \exists y(\text{book}(y), \text{read}(x, y)))$
  - $\exists y(\text{book}(y), \forall x(\text{student}(x), \text{read}(x, y)))$
- Abbreviations:
  - $\forall x(X, Y)$  abbreviates  $\forall x(X \rightarrow Y)$
  - $\exists x(X, Y)$  abbreviates  $\exists x(X \wedge Y)$

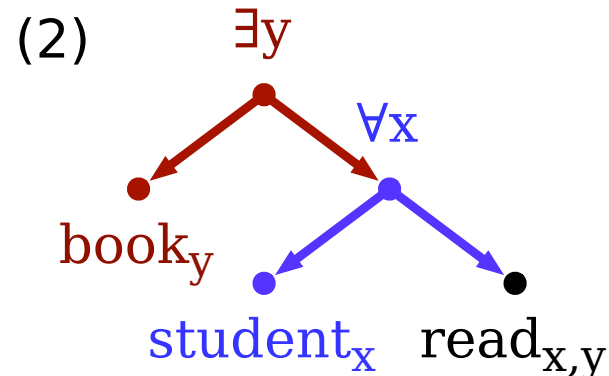
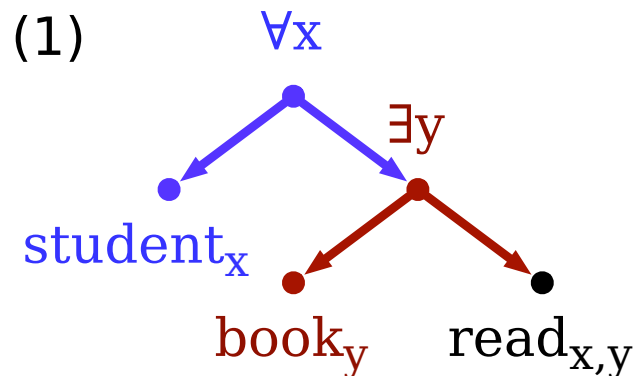
# Readings = Trees

- *Every student reads a book*

(1)  $\forall x(\text{student}(x), \exists y(\text{book}(y), \text{read}(x, y)))$

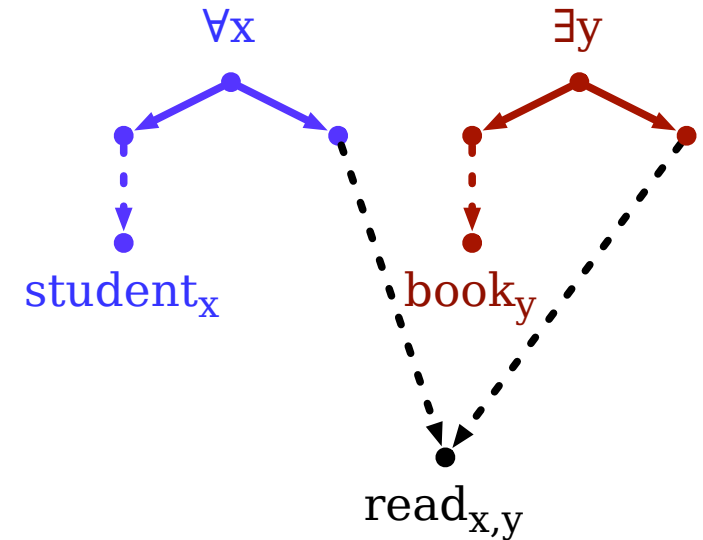
(2)  $\exists y(\text{book}(y), \forall x(\text{student}(x), \text{read}(x, y)))$

- Represent readings as trees:



# Dominance Graphs (informal)

- Dominance graphs consist of
  - tree “fragments” connected by
  - dominance edges
- Tree fragments (solid lines) specify the “semantic material” common to all readings
- Dominance edges (dotted lines) specify constraints: The upper node must dominate the lower one.
- Normal dominance graphs: dominance edges always go from leaves (“holes”) to roots.



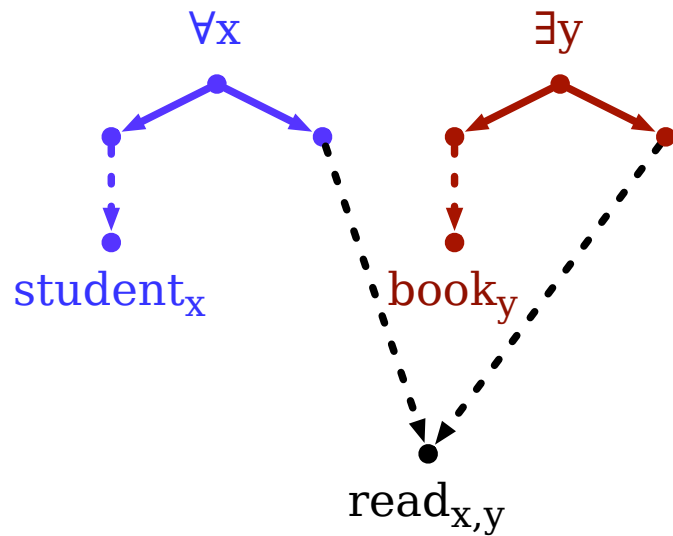
# Normal Dominance Graphs

- **A normal dominance graph** is a graph  $G = \langle V, E \uplus D \rangle$  such that
  - the subgraph  $\langle V, E \rangle$  is a collection of node disjoint trees where the height of each tree is  $\leq 1$   
we call the roots in  $\langle V, E \rangle$  **roots**, and all other node **holes**
  - if  $\langle v_1, v_2 \rangle \in D$ , then  $v_1$  is a hole and  $v_2$  a root
  - every hole has at least one outgoing dominance edge.
- **A labelled dominance graph** is a graph  $\langle V, E \uplus D, L \rangle$  such that  $\langle V, E \uplus D \rangle$  is a (normal) dominance graph and
  - $L$  is a labelling function that assigns a node  $v$  a label with arity  $n$  iff  $v$  is a root with  $n$  outgoing tree edges

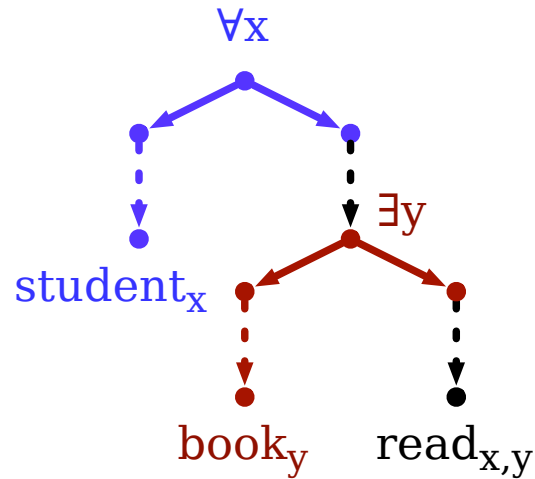
# Solved Forms

- A dominance graph  $G$  **is in solved form** if  $G$  is a forest
- A dominance graph  $G$  **is a solved form of** another graph  $G'$  if  $G$  is in solved form and the reachability relation of  $G$  extends that of  $G'$ 
  - That is: whenever  $v_1$  and  $v_2$  are connected by some dominance edge in  $G'$ , there must be a directed path from  $v_1$  to  $v_2$  in  $G$ .
- Note that dominance graphs and their solved forms differ only in their sets of dominance edges.
- Note also that the solved forms of connected dominance graphs are always trees.

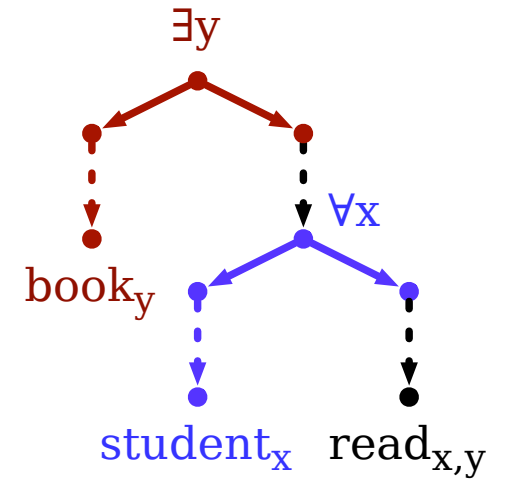
# Solved forms: Example



dominance graph

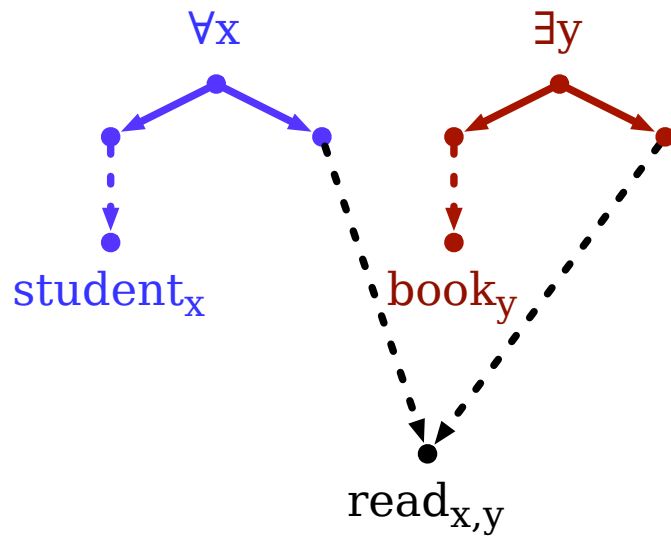


solved form #1

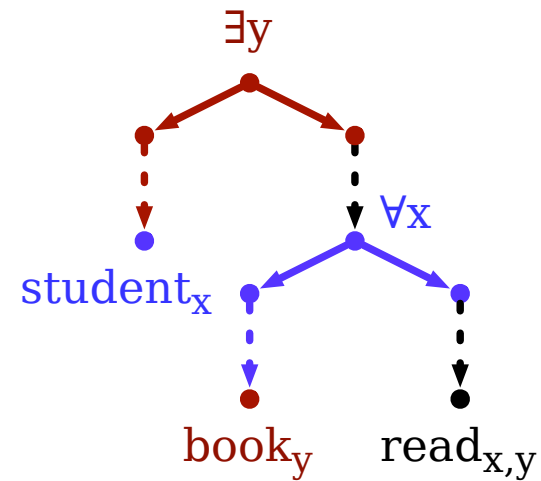


solved form #2

# Not a solved form



dominance graph  $G$



not a solved form of  $G$

# Computational Questions

- **The solvability problem:**  
Does a given dominance graph have any solved forms?
- **The enumeration problem:**  
Given a dominance graph, enumerate all its (minimal) solved forms.
- We will discuss the algorithm by Bodirsky & al. (2004)
- To keep things simple, we restrict the presentation to *connected* dominance graphs.

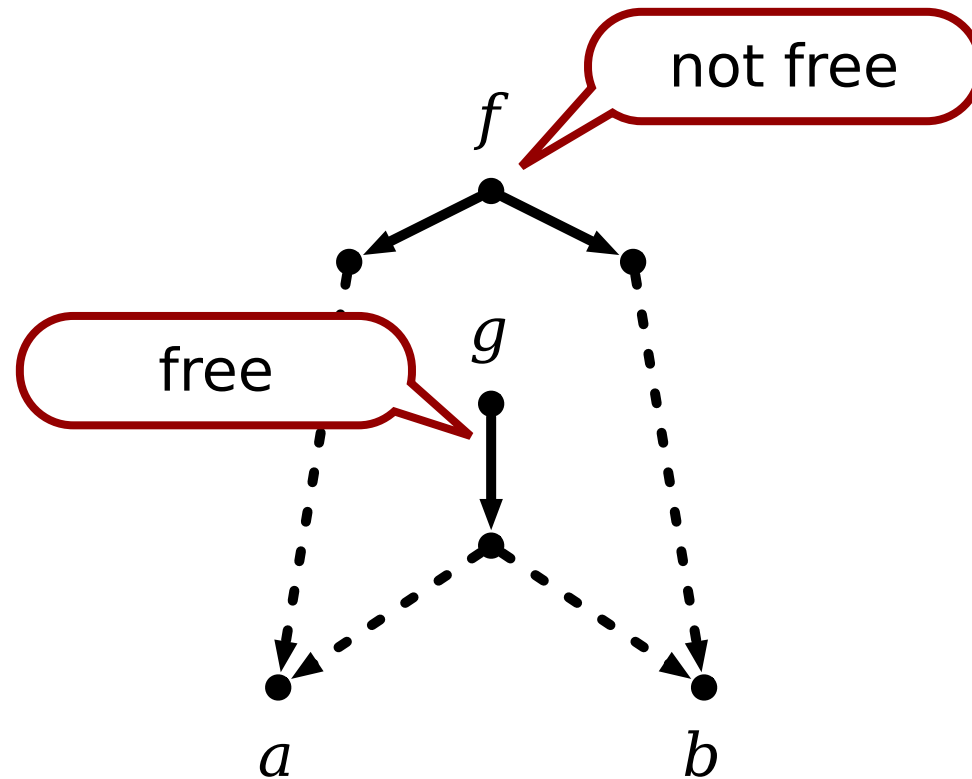
# Solving dominance graphs

- The algorithm of Bodirsky &al. (2004) constructs a solved form of a dominance graph  $G$  as follows:
  1. nondeterministically choose a “free fragment”  $F$  from  $G$
  2. remove  $F$  from  $G$ ; this decomposes the graph  $G$  into weakly connected components  $G_1, \dots, G_k$
  3. recursively compute a solved form for  $G_1, \dots, G_k$
  4. attach the solved form of  $G_i$  under the corresponding hole of the free fragment  $F$  (for  $1 \leq i \leq k$ )

# Free Fragments

- The workhorse of the algorithm is the notion of a “free fragment.”
- We say that a fragment **F is free in a** normal dominance graph **G** iff
  - the root of F has no incoming dominance edges, and
  - no distinct holes of F are connected by an undirected path in the graph  $G'$  obtained from G by removing the root of F
- It can be shown that the following statements are equivalent if G is solvable (normal) dominance graph:
  - F is a free fragment in G
  - G has a solved form with top-most fragment F

# Free Fragments: An Example



# Solving dominance graphs

**solve**( $G = \langle V, E \uplus D \rangle$ ) = <sup>(\*)</sup>

**choose** a free fragment  $F$  of  $G$  **else fail**

let  $G_1, \dots, G_k$  be the WCC's of  $G \setminus F$

let  $\langle V_i, E_i \uplus D_i \rangle = \text{solve}(G_i)$

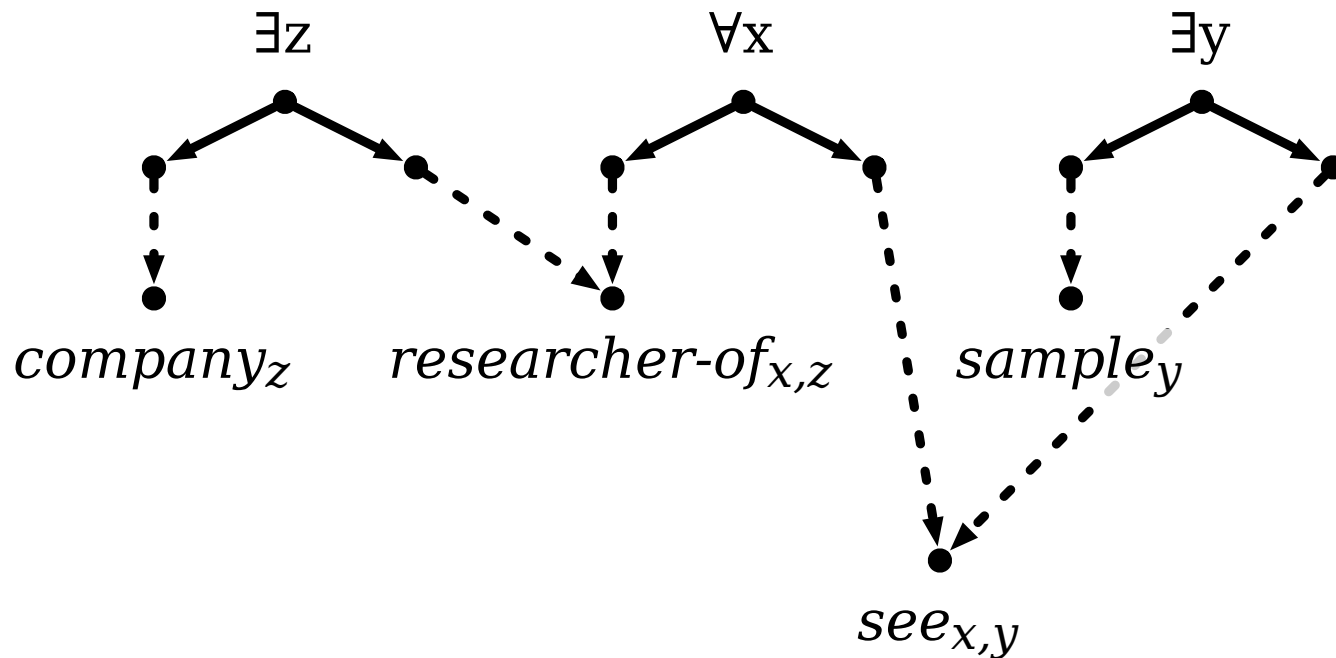
**return**  $\langle V, E \uplus D_1 \uplus \dots \uplus D_k \uplus D' \rangle$

where  $D'$  are dominance edges that connect the holes of  $F$  with the solved form of one of the corresponding  $G_i$

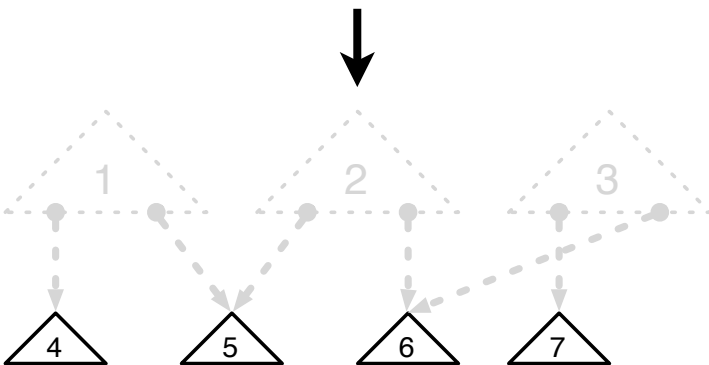
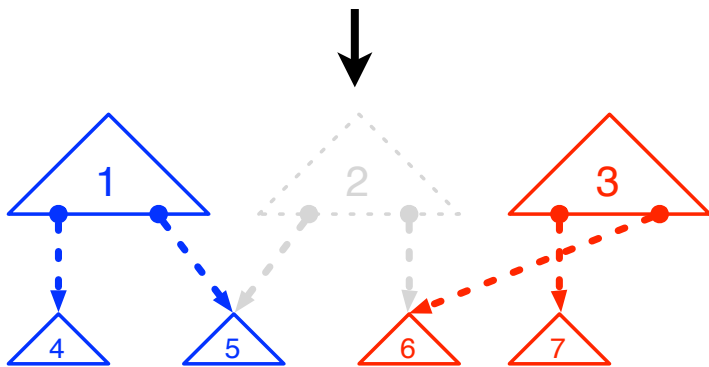
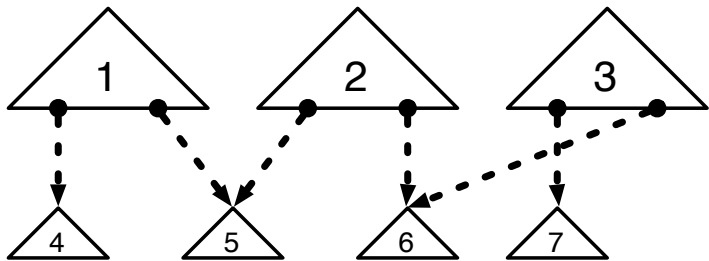
(\*) slightly simplified version, works only for connected normal dominance graphs

# An Example

- Underspecified representation for the sentence “every researcher of a company sees a sample:”



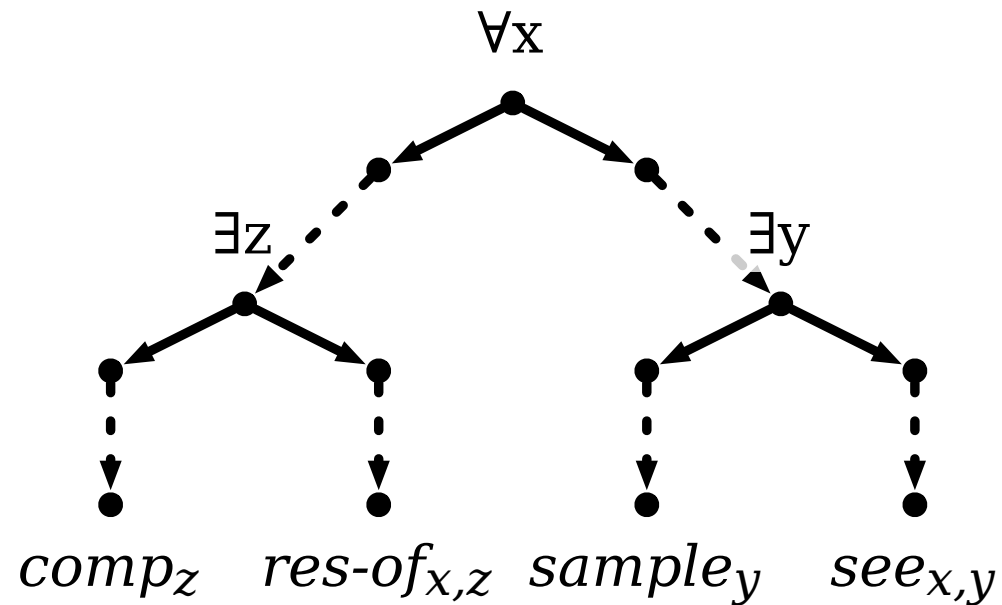
# An Example



| subgraph          | free | wccs                          |
|-------------------|------|-------------------------------|
| $\{1, \dots, 7\}$ | 2    | $\{1, 4, 5\}$ , $\{3, 6, 7\}$ |
| $\{1, 4, 5\}$     | 1    | $\{4\}$ , $\{5\}$             |
| $\{3, 6, 7\}$     | 3    | $\{6\}$ , $\{7\}$             |

# The final solved form

- $\forall x(\exists z(\text{comp}(z), \text{res-of}(x,z)), \exists y(\text{sample}(y), \text{see}(x, y)))$



# Properties of the solver

- It can be shown that the following statements are equivalent:
  - $\text{solve}(G)$  fails for some nondeterministic choice
  - $G$  is not solvable
  - $\text{solve}(G)$  fails for all nondeterministic choices

# Properties of the solver

- We can test whether a fragment is free in time  $O(n + m)$ 
  - where  $n$  is the number of nodes and
  - $m$  the number of edges in a dominance graph  $G$ .
- The overall running time of  $\text{solve}(G)$  is in  $O(n \cdot (n + m))$  per solved-form.

# References

- Alexander Koller, Manfred Pinkal, and Stefan Thater. Scope Underspecification with Tree Descriptions: Theory and Practice [\[PDF\]](#). In: Resource Adaptive Cognitive Processes. Ed. by Matthew Crocker and Jörg Siekmann. Cognitive Technologies Series. Berlin: Springer. 2009.
- Manuel Bodirsky, Denys Duchier, Joachim Niehren, and Sebastian Miele (2004). A new algorithm for normal dominance constraints [\[PDF\]](#). In the proceedings of the Symposium on Discrete Algorithms (SODA04), 59-67.