Semantic Theory

Lecture 7: Advanced Underspecification

M. Pinkal / A. Koller
Summer 2006

Scope ambiguities

- Some sentences have more than one possible semantic representation:

Every student presents a paper.

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

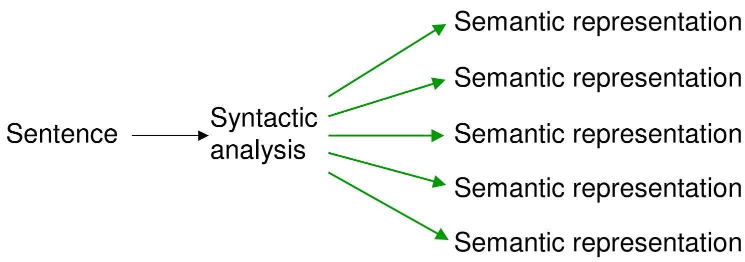
(b) $\exists y[\mathit{paper}'(y) \wedge \forall x[\mathit{student}'(x) \rightarrow \mathit{present}(x,y)]]$

Every student didn't pay attention.

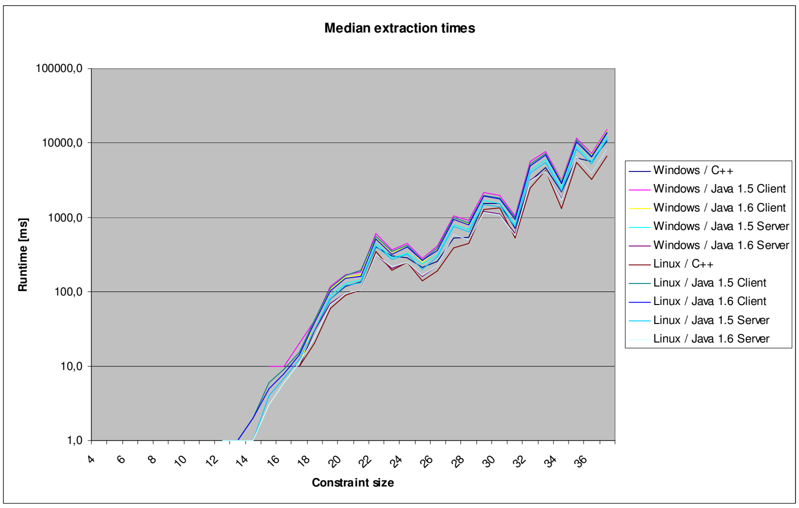
(a) $\forall x[\mathit{student}'(x) \rightarrow \neg \mathit{pay\text{-}attention}'(x)]$

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

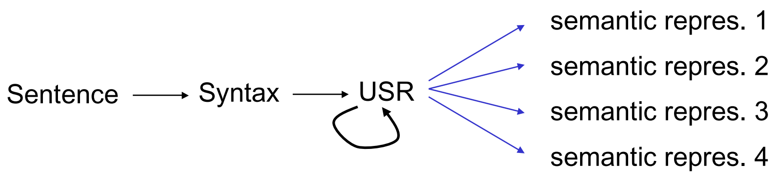
(Nested) Cooper Storage: Schema



Enumeration can get expensive



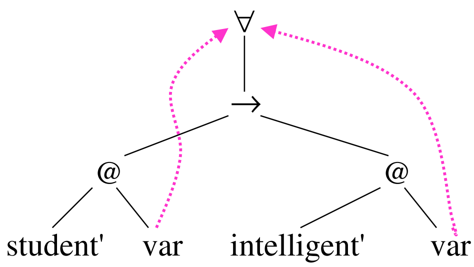
Underspecification: The big picture



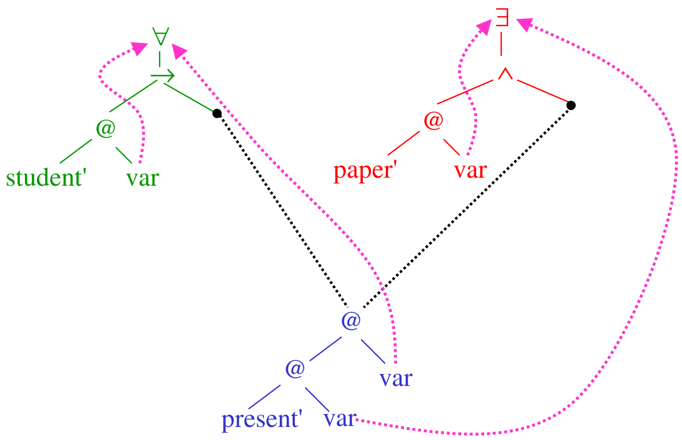
- Derive a single underspecified semantic representation (USR) from the syntactic analysis.
- Perform inferences on USR to eliminate readings excluded by the context.
- Enumerate readings by need.

Terms as lambda structures

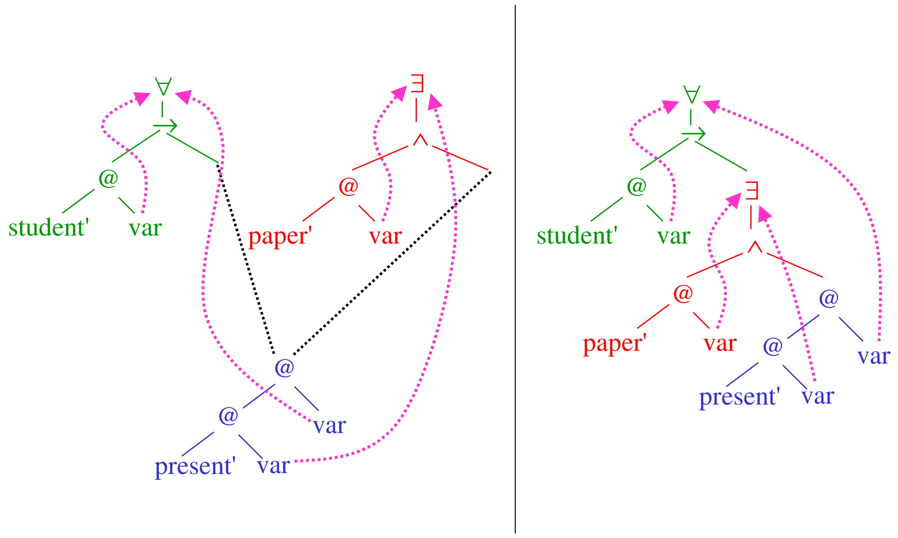
Tree representation of the formula
$\forall$x.student'(x) $\rightarrow$ intelligent'(x):



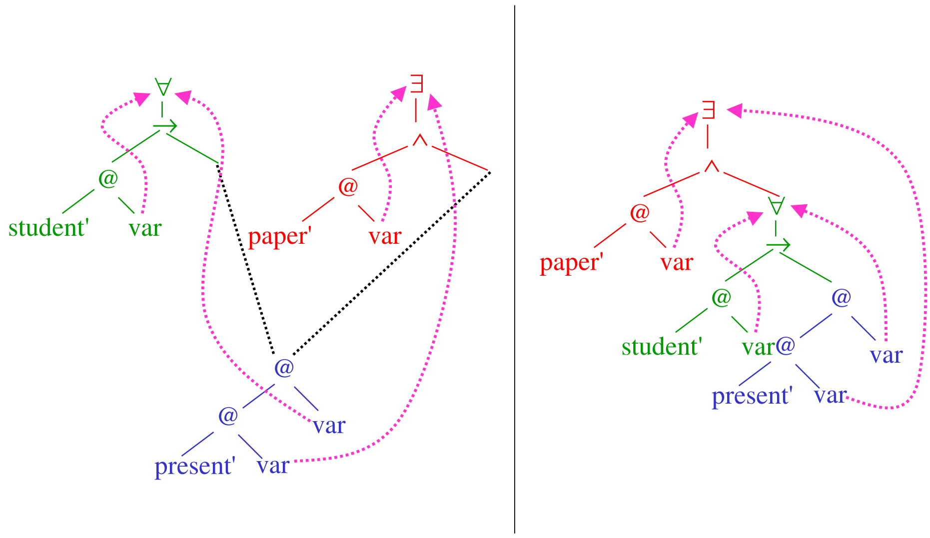
A dominance graph



Solutions of dominance graphs



Solutions of dominance graphs



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? (last week)
- Todo 2 (Enumeration): How can we compute the solutions of a dominance graph? (now)

Outline

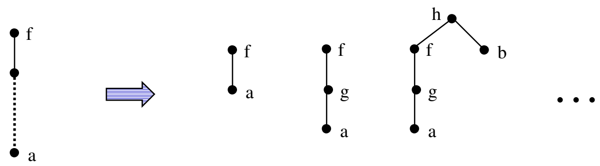
- The solvability and enumeration problems.
- An enumeration algorithm for dominance graphs.
- Hypernormally connected dominance graphs.
- Inferences on dominance graphs.

Solutions

- Question:
  How many solutions does a solvable dominance graph have?

Solutions

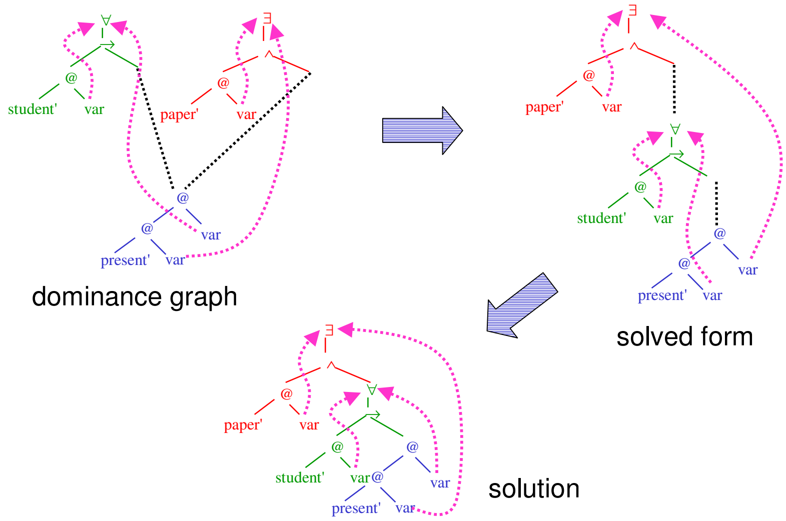
- Question:
  How many solutions does a solvable dominance graph have?
- Answer:
  An infinite number of solutions!



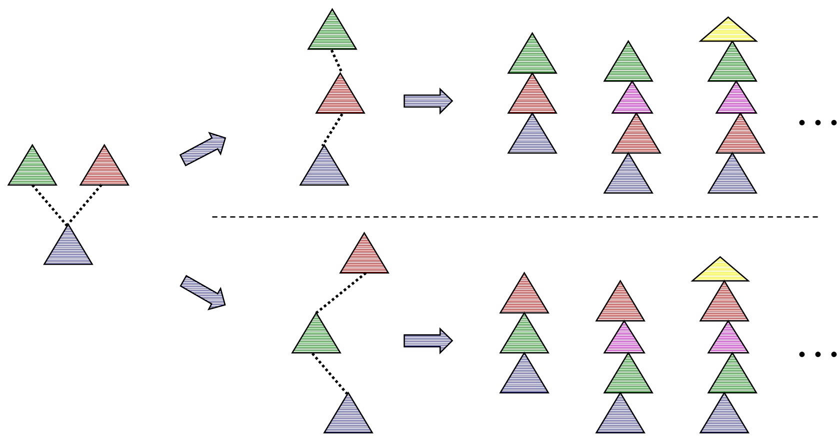
Solved Forms

- Enumerating all solutions of a graph is therefore hopeless (and not useful).
- Thus, we aim at enumerating all solved forms of a dominance graph and not all solutions.
- A dominance graph in solved form is a graph whose tree and dominance edges form a forest.
- A graph G' is a solved form of G iff G' is in solved form, G and G' have the same tree and binding edges, and whenever there is a path from u to v in G (over tree and dominance edges), there is also a path from u to v in G'.

Solutions and solved forms



Solved Forms and Solutions



Solved forms and solutions

- We can consider solved forms as representatives of classes of solutions that only differ in "irrelevant details".
- Every graph in solved form without binding edges has a solution.
- Every solution of a graph is also a solution of one of its solved forms.
- We will completely ignore binding edges when solving dominance graphs. The solver can be easily extended to deal with binding edges as they are generated e.g. by last week's grammar.

Computational Questions

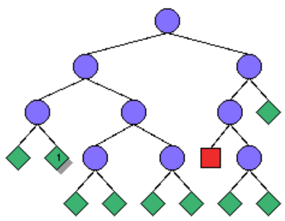
- Two computational questions arise in the context of dominance graphs.
  - The solvability problem: Does a given dominance graph have any solutions?
  - The enumeration problem: Enumerate the (minimal) solved forms of a dominance graph.
- The two questions are closely related.

Solving dominance graphs

- A solver for dominance graphs is an algorithm that solves the solvability and enumeration problems.
- There is a variety of different solvers for dominance graphs.
- The algorithm presented here is not the fastest one, but it is easiest to explain.

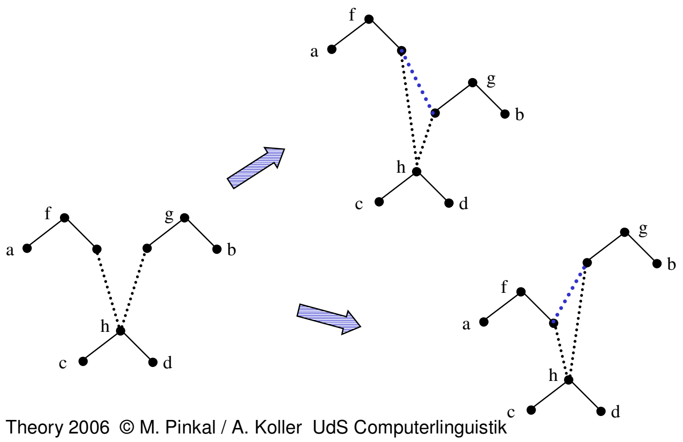
The solver: General architecture

- The solver is a search algorithm:
  - It recursively generates (simpler) new graphs by applying three simplification rules.
  - If none of the rules are applicable, it tests whether the graph is solvable.



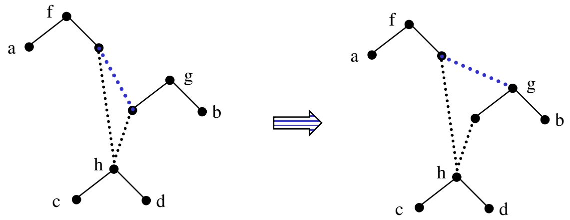
The Choice Rule

- Driving force behind solver is the Choice rule: Which of two trees comes on top?



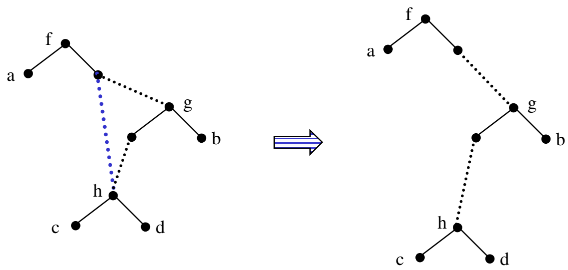
Cleaning Up I: Parent Normalisation

- Parent Normalisation changes a dominance edge (u,v) into a dominance edge (u,w), where w is the parent of v over a tree edge.



Cleaning Up II: Redundancy Elimination

- Redundancy Elimination deletes an edge (u,v) whenever there is a path from u to v that doesn't use this edge.

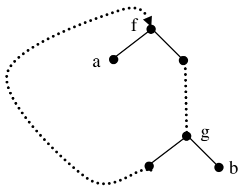


Correctness of the solver

- The rules are correct:
  - Every solved form of the original graph is a solved form of exactly one of the two results of Choice.
  - The original graph and the result of PN or RE have exactly the same solved forms.
- Every application of Choice (plus some applications of PN and RE) arranges the parents of one node.
- Eventually there will be no more nodes with two incoming edges left; so the algorithm terminates.

Detecting unsolvability

- It remains to check whether the end results are solvable or not.
- A dominance graph in which no node has two incoming edges is either a tree, or it has a cycle.
  - If it's a tree, then the graph is in solved form.
  - If it has a cycle, then it is unsolvable.

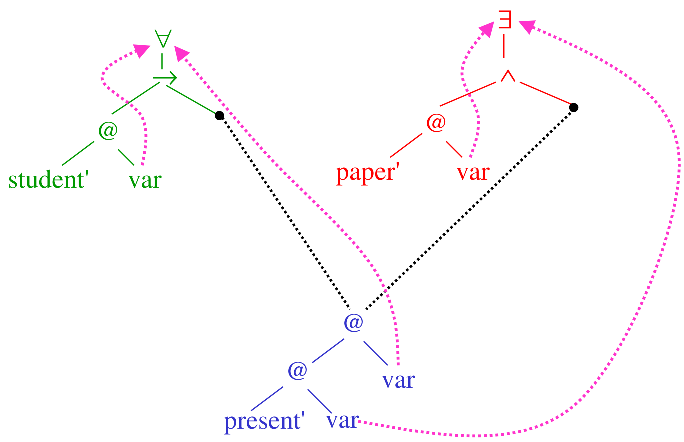


The complete solver

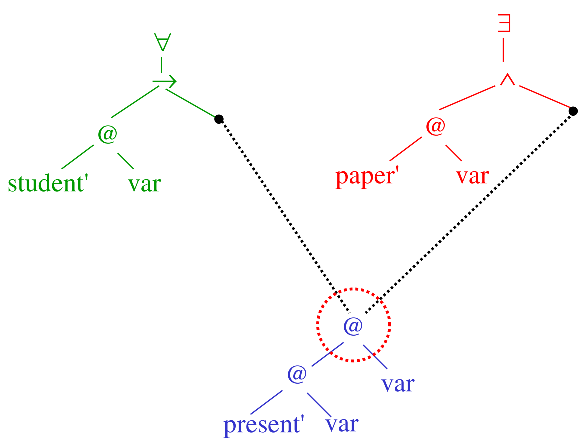
solve(G):

1. Apply Parent Normalisation and Redundancy Elimination exhaustively to G.
2. If there is a node v in G with two incoming dominance edges:
   apply Choice once; this gives new graphs H_1 and H_2
   solve(H_1)
   solve(H_2)
3. If there is no such node v, and if G has no cycle, then report G as a solved form of the original graph.

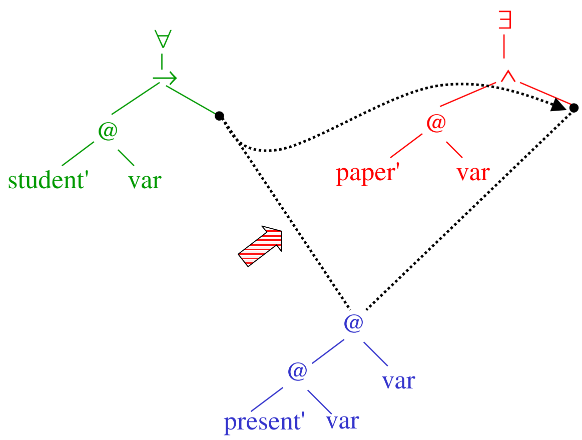
An example run of the solver



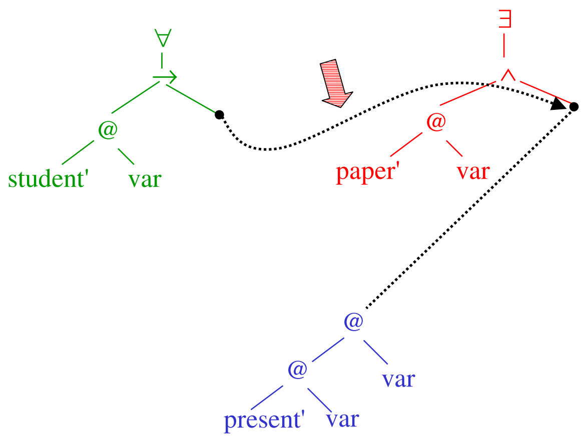
An example run of the solver



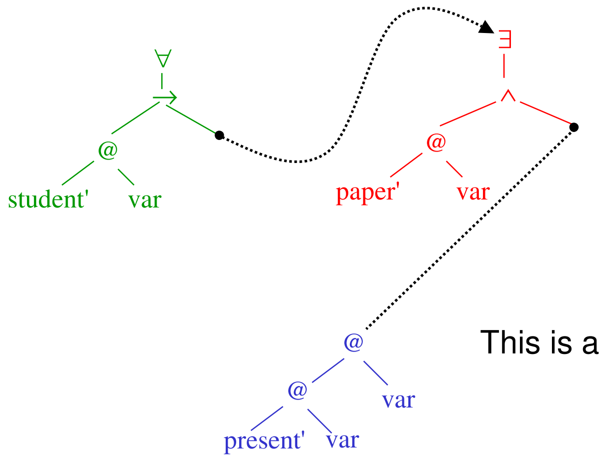
Choice 1



After Redundancy Elimination

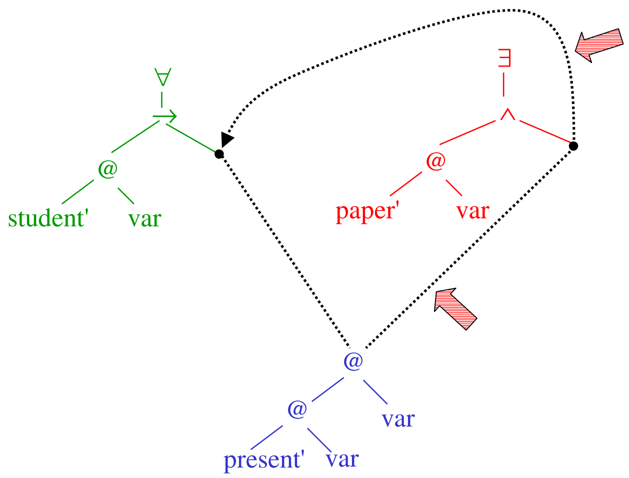


After Parent Normalisation (2 steps)

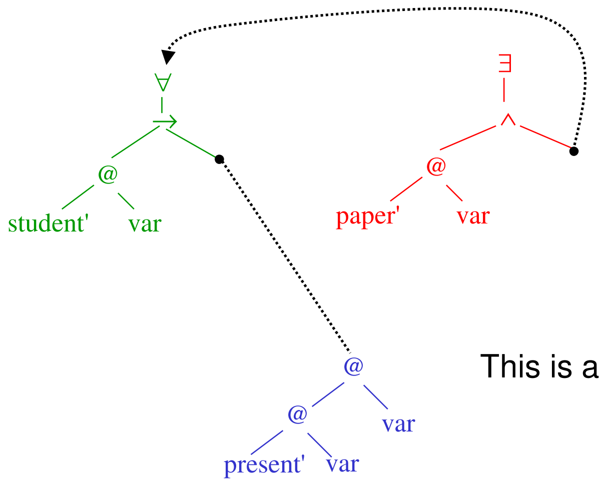


This is a solved form!

Choice 2

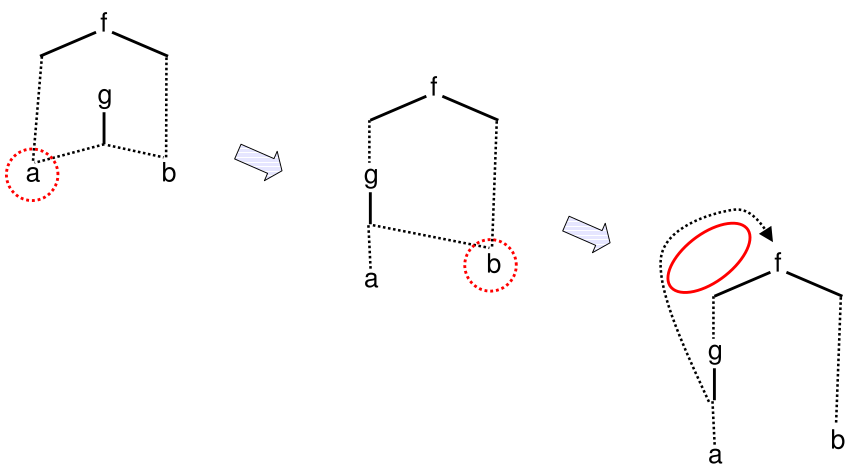


After RE and PN



This is a solved form!

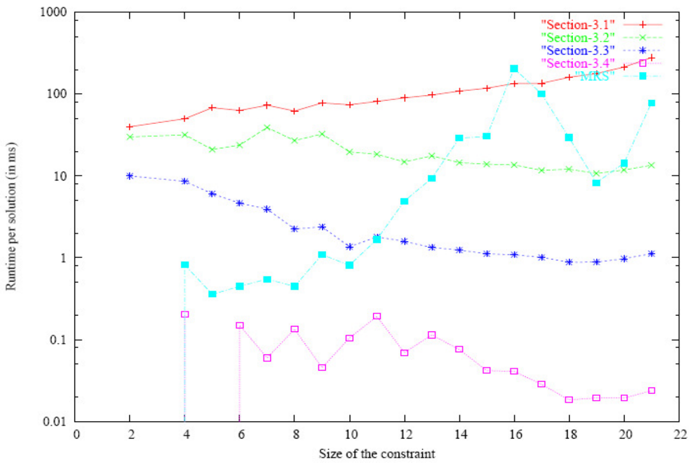
Running into unsolvability



The solver: Summary

- The solver is a search algorithm that computes a set of solved forms for a dominance graph.
- It doesn't enumerate all solved forms, but it does enumerate all minimal solved forms. Every solution of G solves exactly one minimal solved form of G.
- The algorithm may spend a lot of time trying to solve unsolvable graphs.
- This can be improved by a smarter unsolvability test.

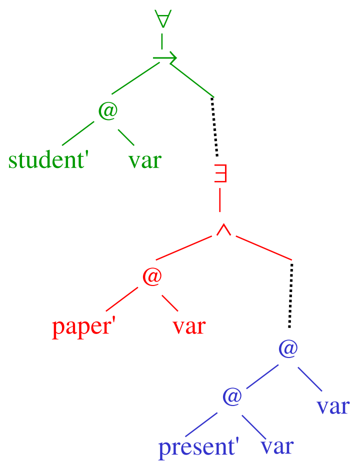
Comparison of different solvers



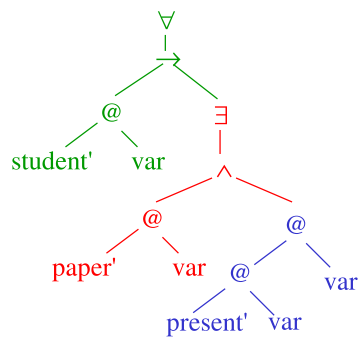
Constructive solutions

- Our initial idea was that solutions of a dominance graph should correspond to semantic representations.
- But now we know that there is generally an infinite number of solutions!
- We are really only interested in constructive solutions, i.e. solutions for which every node in the solution is the α-image of a non-hole (with a label).
- Can we always extract constructive solutions from solved forms?

Solved forms vs. constructive solutions

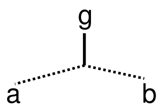


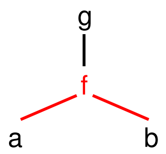
a graph in solved form ...



... and its unique constructive solution

Not all graphs *have* constructive solutions!





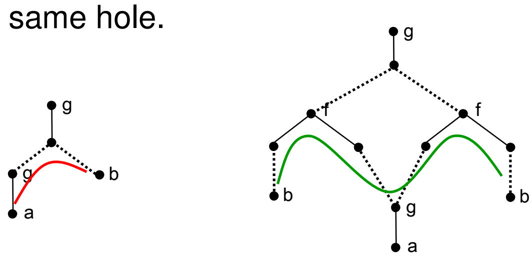
a graph in solved form ... ... and a smallest solution.

Constructive solvability

- In general, not all dominance graphs have constructive solutions.
- How can we tell which ones do?

Hypernormal paths

- A hypernormal path is an undirected path in a dominance graph that doesn't use two dominance edges out of the same hole.



- A dominance graph is hypernormally connected (or hnc, or a net) iff every pair of nodes is connected by a hypernormal path.

Simple solved forms

- A solved form is called simple iff every hole has exactly one outgoing dominance edge.
- Every graph in simple solved form has exactly one constructive solution.
- All solved forms of a hypernormally connected graph are simple.

- Thus: Every solved form of a hnc graph has exactly one constructive solution.

The usefulness of nets

- Hypernormal paths have a number of really useful properties:
  - All solved forms of a hnc graph are simple, i.e. solved forms correspond to readings.
  - USRs from other formalisms (Hole Semantics, MRS) can be translated into dominance graphs if the result is hnc.
  - A dominance graph is unsolvable iff its undirected version has a hypernormal cycle.

The usefulness of nets

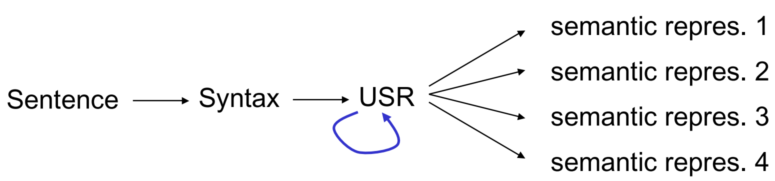
- The only question now is:
  - How useful is the fragment of hnc graphs?
  - Do we know that all graphs that we want to use in practice are in fact hnc?
- We believe: Yes!
  - This is called the Net Hypothesis.
  - An upper limit on scope flexibility.
  - Ongoing research.

The Net Hypothesis

- Can be proved for (an extension of) last week's grammar.
- Relationship to (Nested) Cooper Storage.
- Empirical verification (Flickinger et al., HPSG 2005):
  - Compute USRs for all 960 sentences in the Rondane Treebank using the English Resource Grammar.
  - Result: 90% are hypernormally connected.
  - The rest seem to be due to errors in grammar (but this is ongoing research).

What can we do with USRs?

- We know now how to enumerate readings from USRs, and that is good and important.
- But really, we wanted to use USRs as a platform for disambiguation.



- How can we do this?

Inference on USRs

- Direct deduction (Reyle, de Rijke, Jaspars, ..., 1990s): Infer from USR another USR that describes logical consequences of its readings.
- Use anaphora (Koller & Niehren 2000):
  Every linguist speaks two languages. These languages are taught at our department.
- Eliminate logical redundancy (Koller & Thater 2006):
  A researcher of some company saw a sample of a product. (14 readings, all logically equivalent)

Utool

- A fast implementation of a solver for dominance graphs is available online:
  - Utool, the Swiss Army Knife of Underspecification
    `http://www.coli.uni-saarland.de/projects/chorus/utool`
- Implements another graph algorithm (not the one presented here).
- Extra functionality:
  - support for other underspecification formalisms and file formats
  - redundancy elimination

Summary

- Solving means enumeration of solved forms (not solutions).
- Solving dominance graphs:
  - search algorithm that is driven by the Choice rule
  - detect unsolvability via cyclicity test

Summary

- Hypernormally connected graphs (nets):
  - guarantee that solved forms have constructive solutions
  - it seems that every graph used in underspecification is a net
- Some first results about inference on underspecified representations.