A concurrent syntax-semantics interface for dependency grammar: A first sketch

Ralph Debusmann
Universität des Saarlandes
Programming Systems Lab
rade@ps.uni-sb.de

Thursday, May 23rd 2002

Adding semantics to TDG

- starting point: TDG (Topological Dependency Grammar) grammar formalism (Diplomarbeit 2001, Duchier/Debusmann ACL 2001)
- so far: only syntax and word order, but no semantics
- goals of my dissertation:
  - extend the grammar formalism
  - develop a concurrent syntax-semantics interface (to CLLS-semantics)

Concurrent semantics construction

- vision: syntax-semantics interface for TDG shall be *concurrent*

- concurrent means bi-directional: while parsing, information from syntax can be used to disambiguate semantics and vice versa

- provides the ideal basis for the integration of preferences

Overview of this talk

1. TDG summary

2. First ideas for a syntax-semantics interface

3. First ideas on how to incorporate preferences

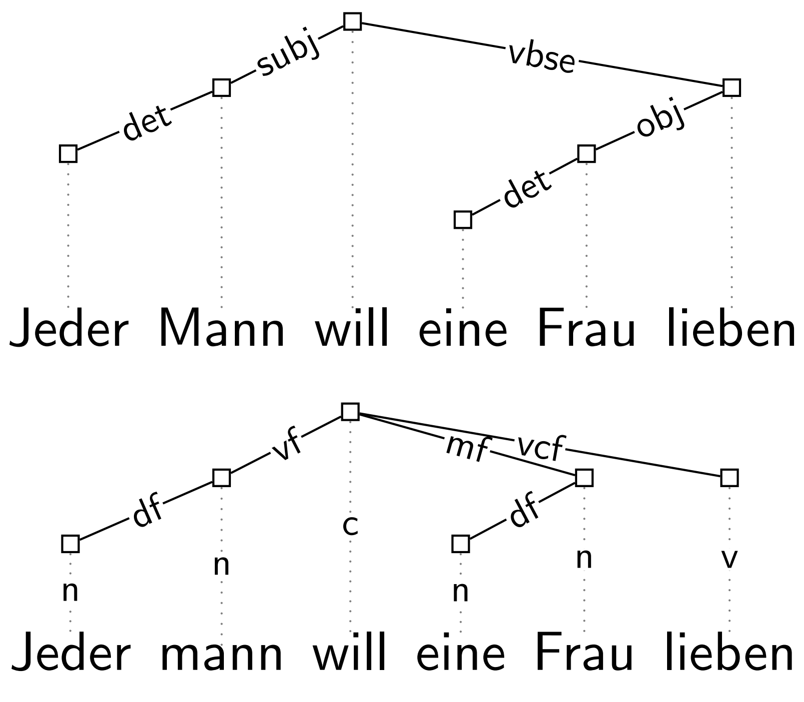
4. Demo

TDG summary

- dependency-based, lexicalized grammar formalism, efficient constraint-based parser implementation

- fundamental: lexicalized principles of *accepted labels* and *valency*

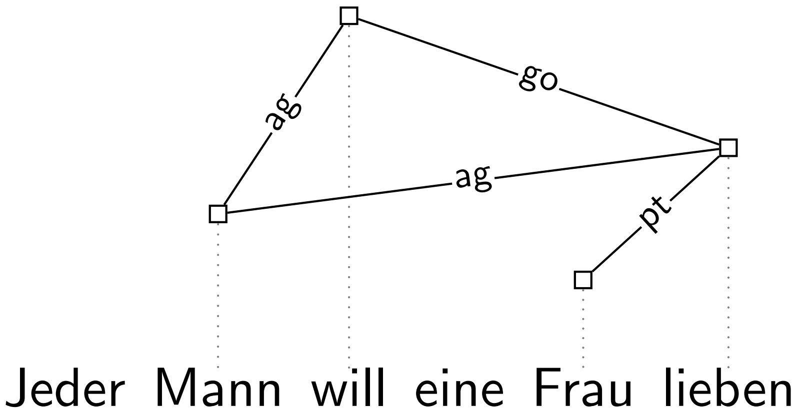
- two levels: dependency tree and topology tree:



A syntax-semantics interface for TDG: first ideas

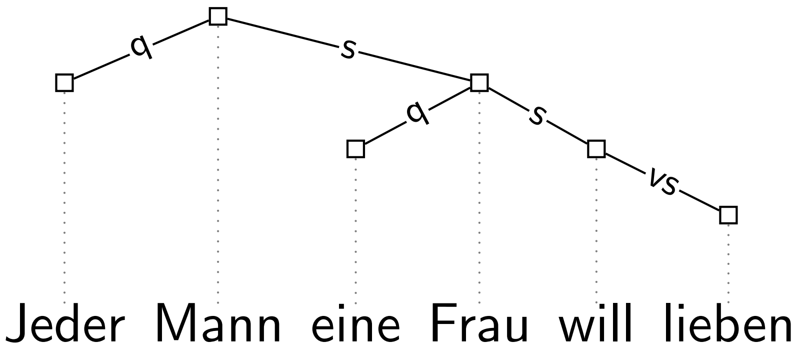
- goal: obtain a CLLS-constraint from a list of words
- two issues:
  1. need to recover the λ-bindings
  2. need to recover information on how to plug CLLS-fragments together
- idea investigated so far: add two additional levels of analysis:
  1. thematic graph
  2. CLLS-derivation tree

Thematic graph



- ag = agent, pt = patient, go = goal
- used to recover the λ-bindings
- accepted labels and valency are again the most important well-formedness conditions
- connected to syntax (dependency tree) by linking constraints theory (mapping e.g. agent to subject)
- similar to: a-structure (LFG), HPSG: done in the syntax

CLLS-derivation tree

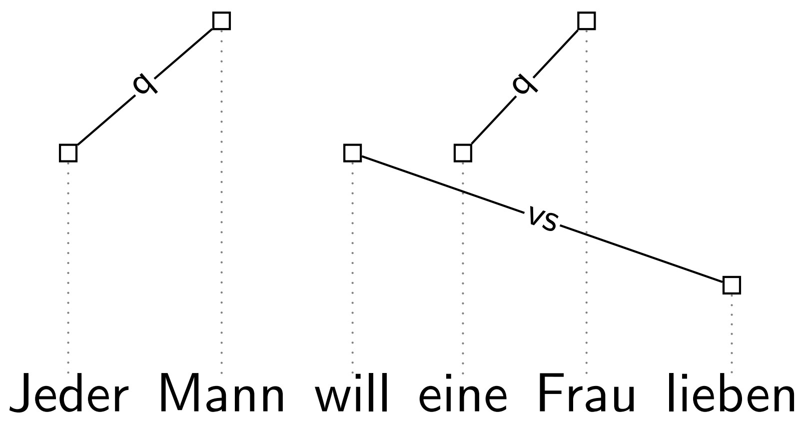


- q = quantifier, r = restriction, s = scope
- represents information on how to plug CLLS-fragments together (a la TAG, substitution and adjunction)
- accepted labels and valency once again the most important well-formedness conditions
- connected to syntax by covariance constraints
- similar to: glue-structure (LFG), HPSG: MRS

Partially solved CLLS-derivation tree

- do not need and do not want to enumerate all possible CLLS-derivation trees

- if we do not enumerate, the partially solved CLLS-derivation trees we obtain precisely correspond to the underspecified semantic representations (=CLLS-constraints) we want, e.g.:

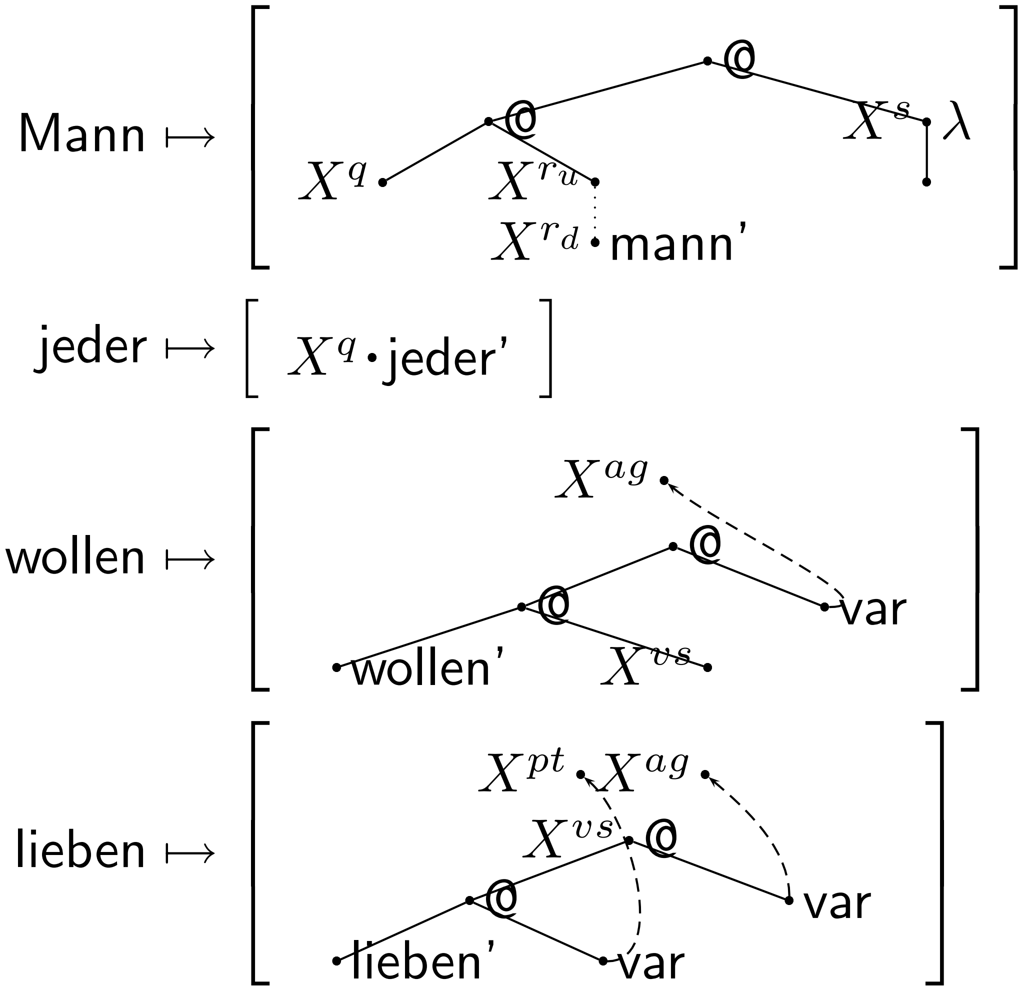


Reading off CLLS-constraints

- conjecture: there is an algorithm to read off CLLS-constraints from (fully solved) thematic graphs and (partially solved) CLLS-derivation trees, where:
  - the thematic graph provides the λ-binding information
  - the CLLS-derivation tree provides plugging information

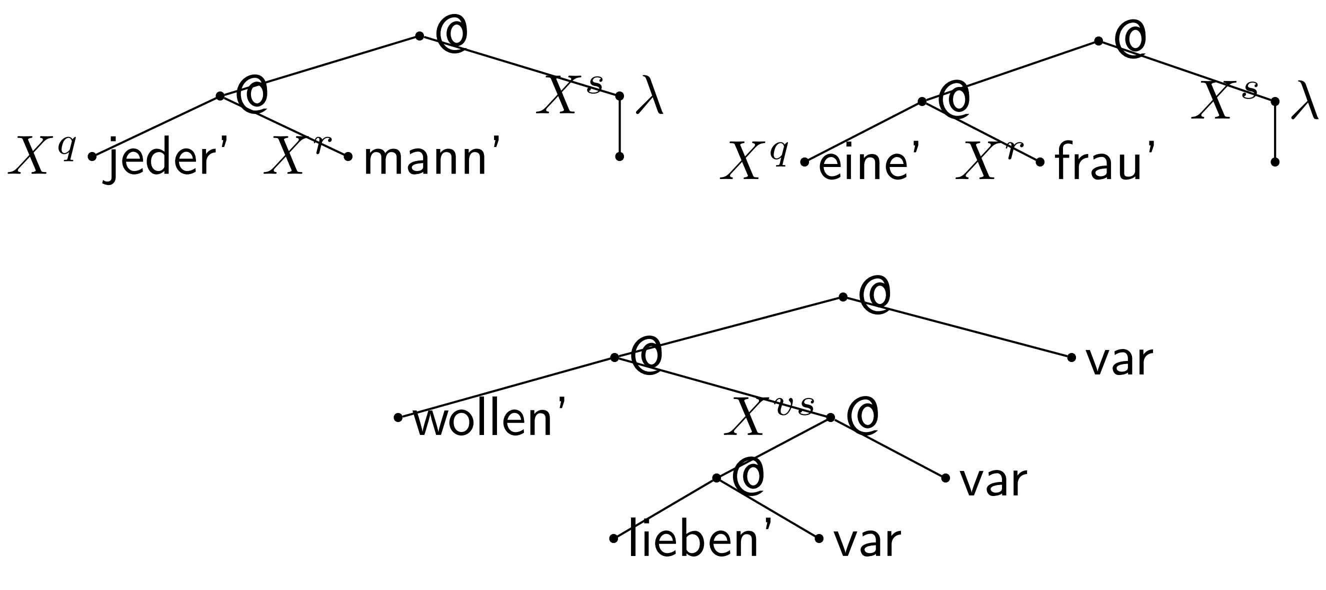
Reading off CLLS-constraints: an example

1. Nodes in the derivation tree correspond to CLLS-fragments, and these fragments correspond to CLLS-constraints:



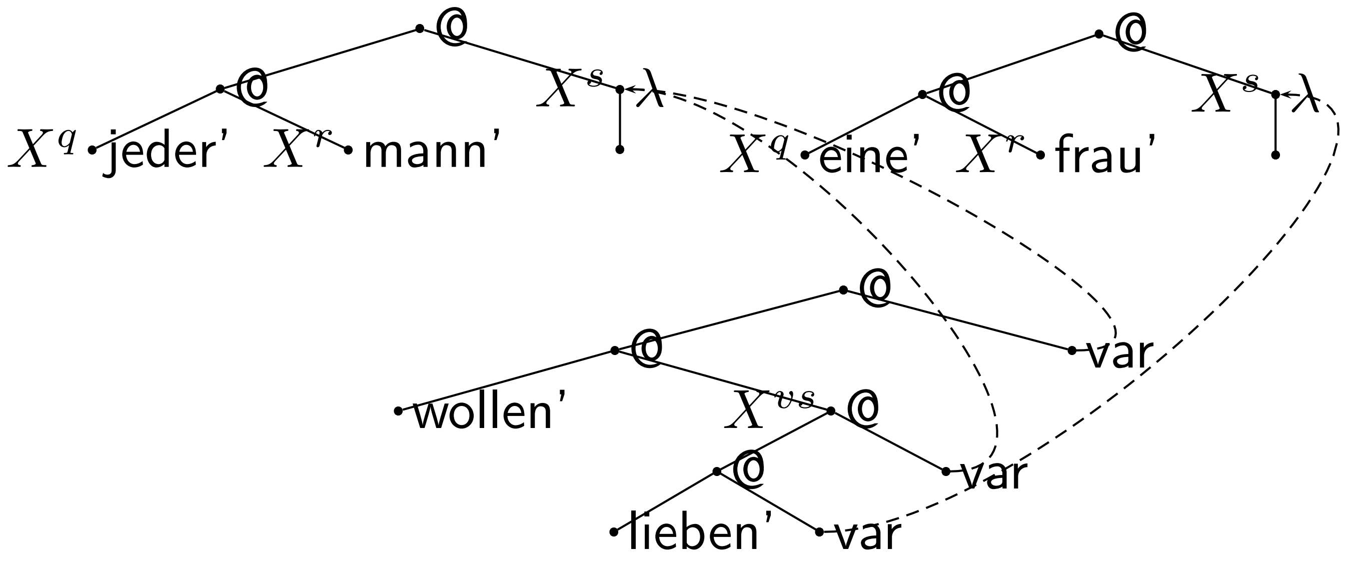
Reading off CLLS-constraints: an example

2. We arrange the CLLS-constraints according to the information contained in the CLLS-derivation tree:



Reading off CLLS-constraints: an example

3. Finally, we add the λ-binding constraints according to the information contained in the thematic graph:



How to incorporate preferences

- idea: after all deterministic inferences have been drawn, we export partial parse information to the preferences module

- the preferences module acts as an oracle, predicting e.g. where a PP can be attached, and feeds this information back into the dependency parser to resolve ambiguities

- process very similar to what Brandts/Duchier already did for dependency parsing (1999): distinguish edges that are determined and those that are candidates, then let an "oracle" rank the candidates, and then first try the highest ranked ones

Demo - state of the art

- the old TDG grammar development system was rather inflexible:
  - difficult to add e.g. new levels (thematic graph, derivation tree)
  - difficult to extend the grammar formalism (e.g. add lexical rules)

- to be more flexible, we are reimplementing the grammar development system from scratch

- state-of-the-art: parsing using all four analysis structures (dependency tree, topology tree, thematic graph, CLLS-derivation tree)

- yet missing: construction of CLLS-constraints, lexical rules and other additions to the grammar formalism, comprehensive GUI interface (can mostly be taken over from the old demo)