[redacted]

Motivation

- Adaptable NLP: improve performance through adaption to already observed situations
- Efficient uniform strategies for parsing and generation
- Relationship of HPSG and tree-based grammars

Starting points

- Explanation-based learning EBL:

  keep track of problems solved in the past and replay those solutions to solve new but somewhat similar problems in the future

- EBL and parsing, e.g.:
  - Rayner, Samuelsson: CLE, patr-like formalism
  - Srinivas, Joshi: LTAGS and FST
  - Neumann: HPSG, parsing *and* generation

Tree-based approaches

Basic building blocks: trees with depth >= 1

- Data oriented parsing (R. Bod)
  use of annotated corpus as stochastic grammar

- Tree adjoining grammars
  manually specified, competence-based

- Compilation from HPSG to TAGs

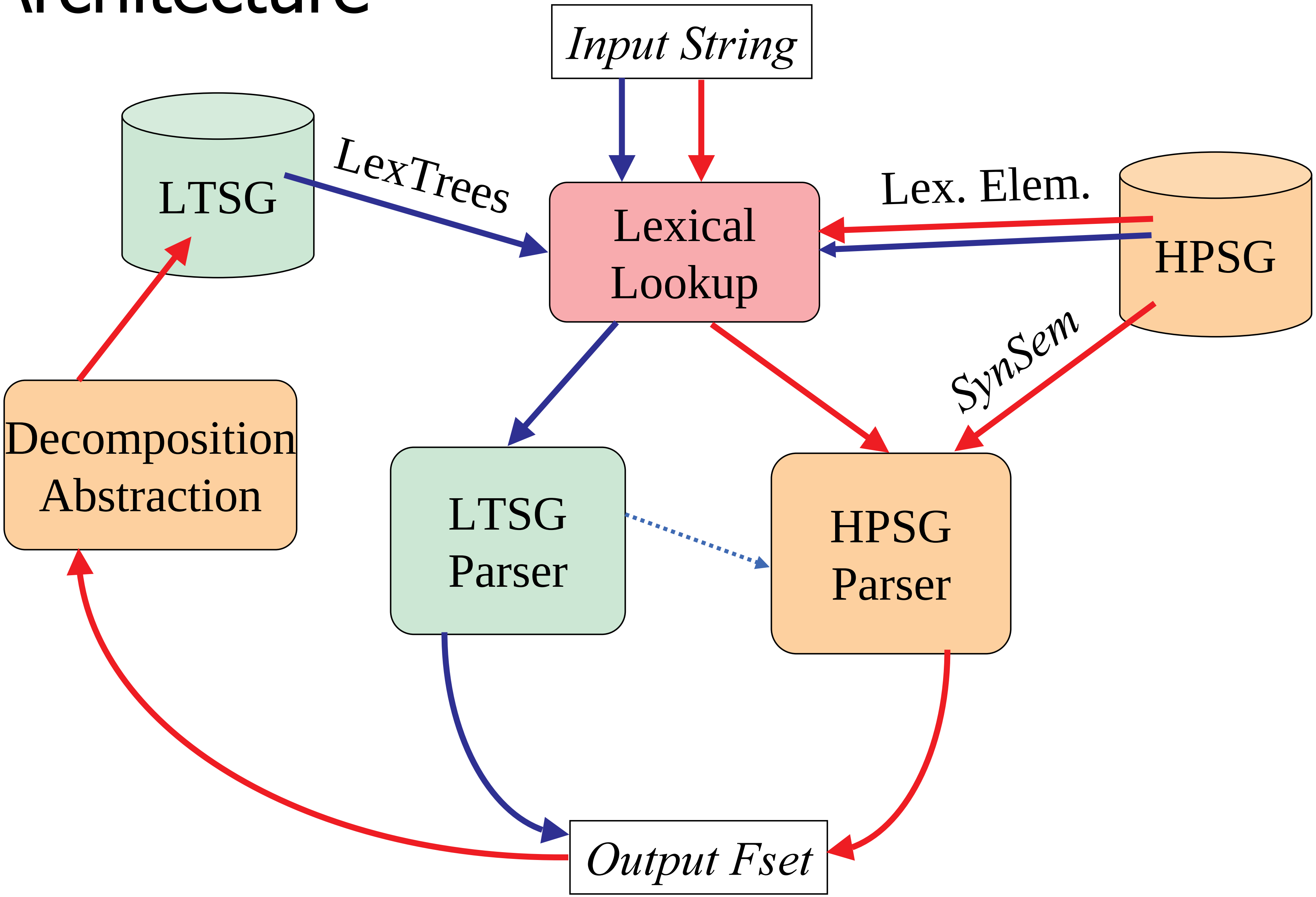
Data-driven extraction of tree-based grammar from HPSG

- Use *competence-based* HPSG to get „annotated corpora"

- Apply linguistic-oriented *tree decomposition* principles

- Use resulting *performance-based* tree grammar for parsing new examples

Architecture



Quelle: GN

The Training Phase

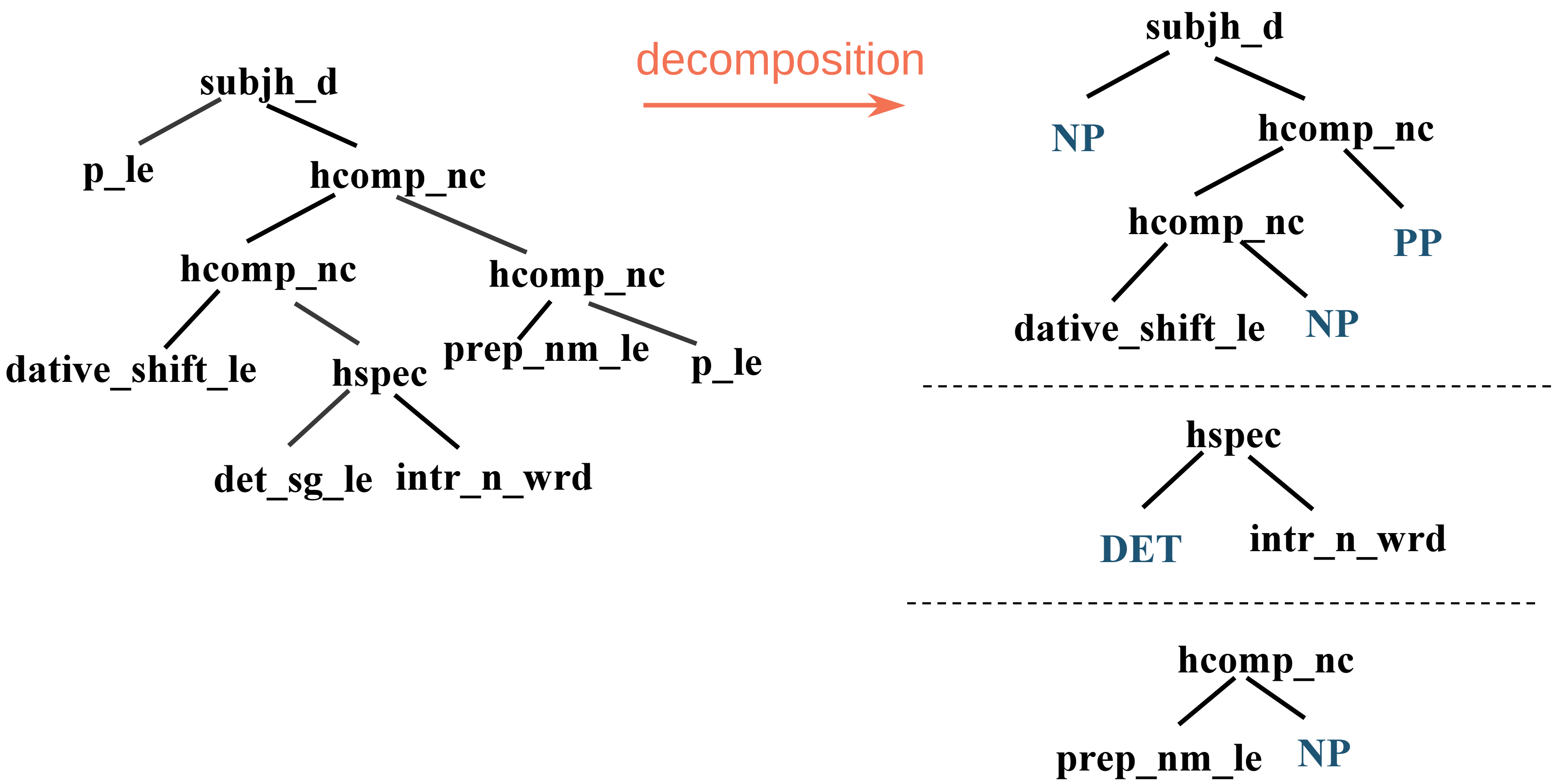
Input: TrainSet, a set of sentences and LTSG, an empty set

- For each Fset in parse(Sent,Source-HPSG) do:

- Apply **head-driven-decomposition-principle**(Deriv(Fset))
  - recursively cut out all non-headed subtrees
    (eventually apply relevance test on current subtree)
  - apply category abstraction on cutting points

- Add each resulting **lexical-typed anchored skeleton** to LTSG

Example: *Sandy gave a book to Kim*



Quelle: GN

Category abstraction is performed on the root node‘s fstruct (see Flickinger, CSLI grammar)

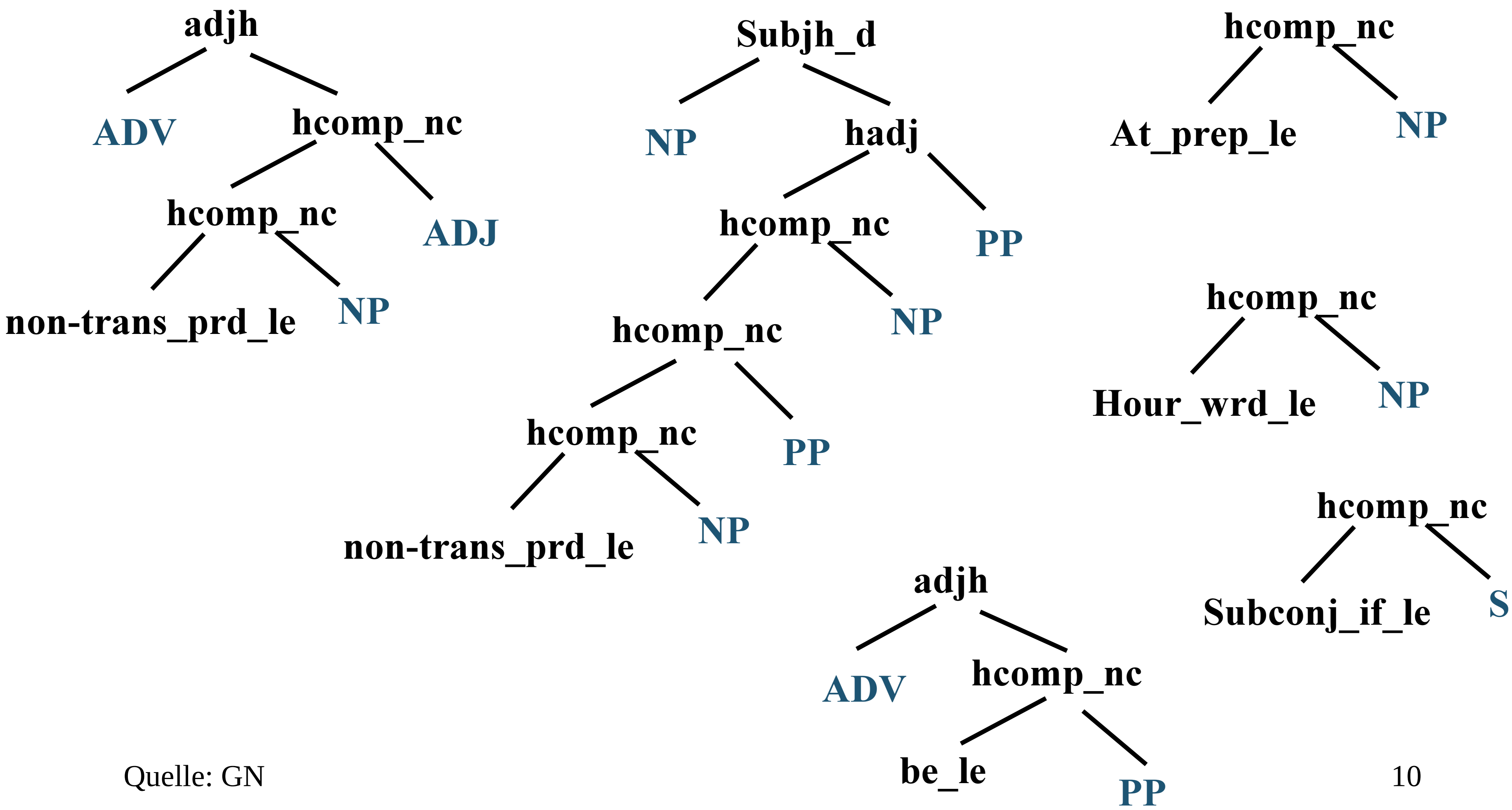
```
NP := [LOCAL.CAT [HEAD    noun,
                  VALENCE [ SUBJ   none,
                            SPR < synsem > ]]]

VP := [LOCAL.CAT [HEAD    verbal,
                  VALENCE [ SUBJ synsem,
                            COMPS *olist* ]]]

PP := [LOCAL.CAT [HEAD    prep,
                  VALENCE.COMPS *cons* ]]]
```

Can and should be made more specific in order to allow for more selective abstraction process

Some more examples



The Application Phase

- Parsing of new sentences only with LTSG

- Basic steps
  - lexical lookup
  - selection of type-compatible trees from LTSG
  - tree composition and unification

- Result is fully specified fstruct wrt HPSG

Chart-parser for LTSG

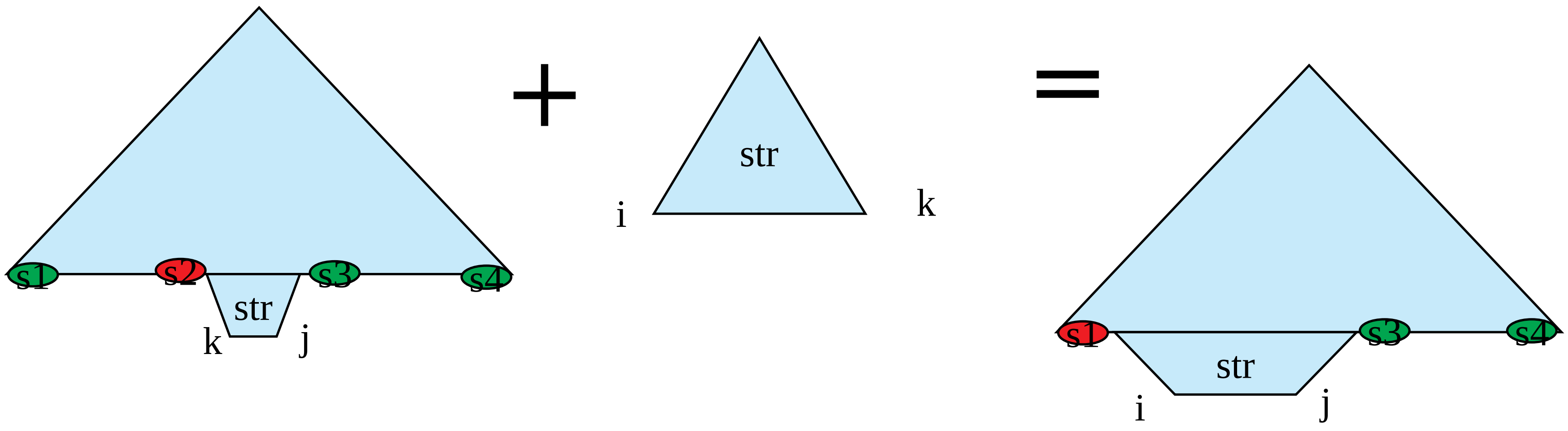
- LexLookup
  - fstruct(s) of each input word (lexical passive items Pi)
- Agenda initialization
  - get all type-compatible anchored trees of Pi
  - expand them by deterministic application of corresponding HPSG-constraints (active items Ai)
  - add Pi and Ai to agenda
- Iteration
  - combine passive and active items by means of tree substitution followed by unification (yields passive or active items)

Quelle: GN

Tree composition

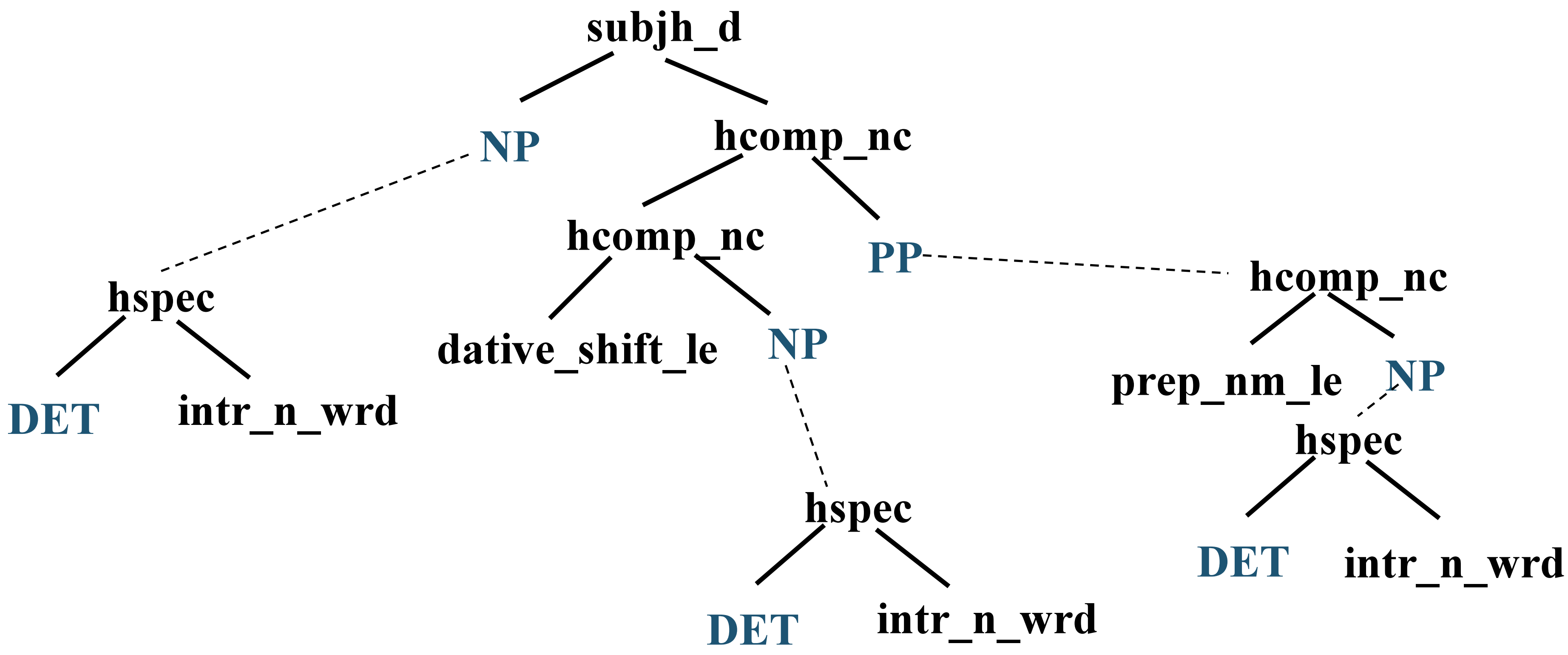
- Fundamental rule: active item Ai and passive item Pi combined
  if selected-element(Ai) subsumes root(Pi)
  then unify(fstruct(Pi), fstruct(selected-element(Ai));
  for new item Ai`: determine new selected element (i.e., substitution node)

- Selection function (left-completion rule):
  from anchor to all left nodes and then from anchor to all right



Quelle: GN

Example: *A woman gave the book to a man*



Two ways of combining active and passive trees

- **Downward** if current item Ci is active, apply fundamental rule to
  - left passive items Pi:
    chart(n, Ci(start)), 0 <= n
  - right passive items Pi:
    chart(Ci(end), n)), n <= length(input)

- **Upward** if Ci is passive
  - left active items Ai:
    chart(Ci(start), n), n <= length(input)
  - right active items Ai:
    chart(n, Ci(end)), 0 <= n

Quelle: GN

Some properties

- Incremental: combined left-to-right/anchor-driven
- Substitution by means of subsumption
- Left-completeness condition allows deletion of chart items
- Method is correct
- Finds valid but non-trained readings
- No restriction wrt. length of trained sentence
- Can be used stand-alone and interleaved with HPSG-parser

Implementation

- First version using page system and CSLI English grammar
- First test on small Verbmobil corpus (25 very different complex sentences)
  - 75 trees extracted
  - performance: 1 to 6 cpu seconds
    (all readings, > factor 30)

The new method can and will be improved in several ways

- Application and evaluation on large corpora

  trade-off between increased coverage and performance effort expected
- More fine-grained category abstraction
- More specific lexical information for anchor
- Use of statistically based prio-function for agenda
- Guidance of data-driven selection of candidate trees by means of phrasal tagging and/or reachability trees
- LTSG generation

LTSG generation: outline

- Use same LTSG also for generation (reversibility)
- Input is MRS representation of NL expression
- Output: fstruct of the NL expression
- Basic bottom-strategy:
  - lexical lookup using relation names
  - LTSG lookup using lexical type
  - use semantic information as chart-index (Neumann:94)
- Disadvantge:
  - lexical lookup only with semantic information

Mixed strategy

- incremental, bottom-up/top-down (~SHDGA)
  - select sem-head from input MRS
  - perform lexical lookup and then retrieval of anchored tree
  - use constraints from selected element to get next element
    - from input MRS (all elements with identical index) or
    - from passive items with compatible sem. information
- advantage:
  - traversal of input MRS and lexical lookup can be improved by means of top-down information from partial parse tree