Introduction

Ralph Debusmann

and

Denys Duchier

Programming Systems Lab, Saarland University, Saarbrücken, Germany

and

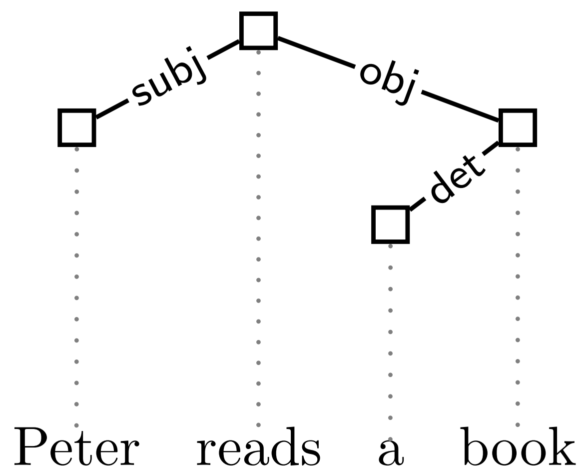
Équipe Calligramme, LORIA, Lille, France

Purpose of this course

- a methodology for modeling language (XDG)
  - *constraint-based (model theoretic syntax)*
  - *dependency-based*
  - *multiple dimensions*
  - *lexicalized*
  - *principles governing well-formedness and interactions*
  - *macroscopic phenomena are emergent*
- how to cook your own DG formalism (XDK)
- relate techniques and architectural principles to what can be found elsewhere

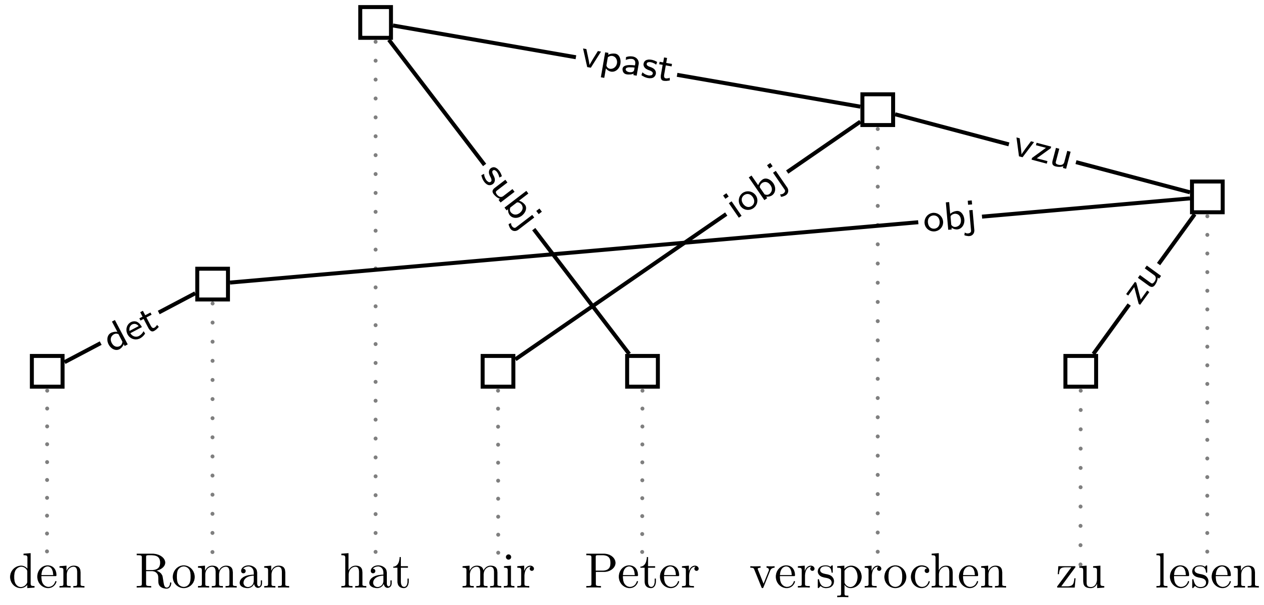
The notion of a dependency structure

- head/dependent asymmetry
- named relation



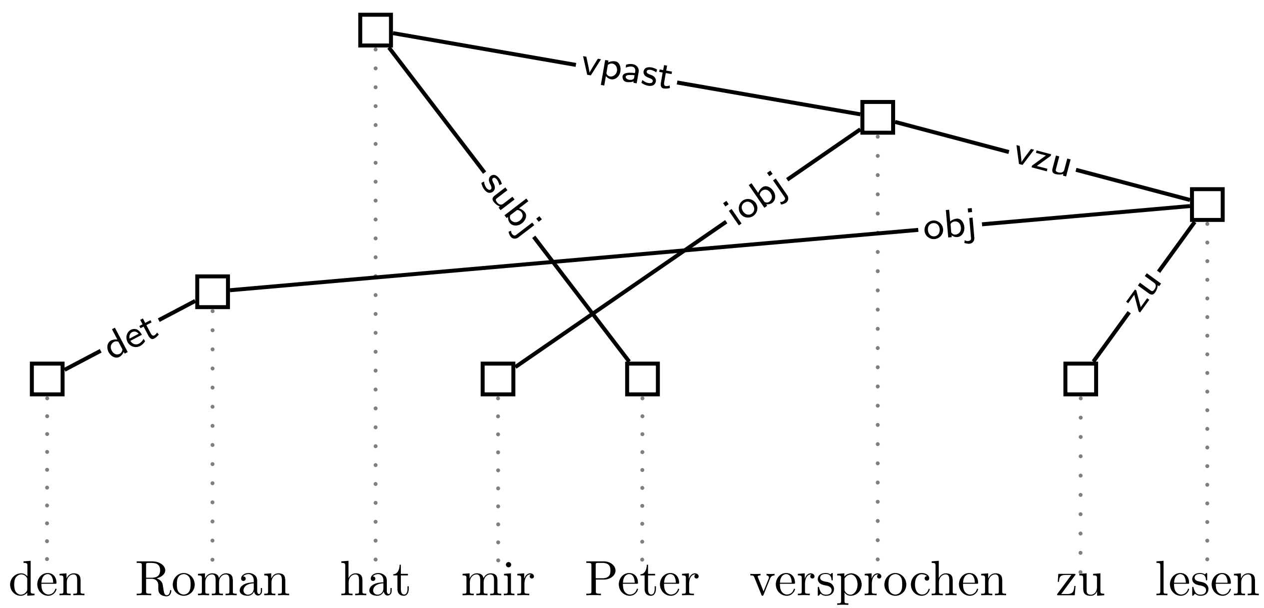
Non-projective analyses

- languages with free(r) word-order
- crossing branches (non-projectivity)
- discontinous constituents



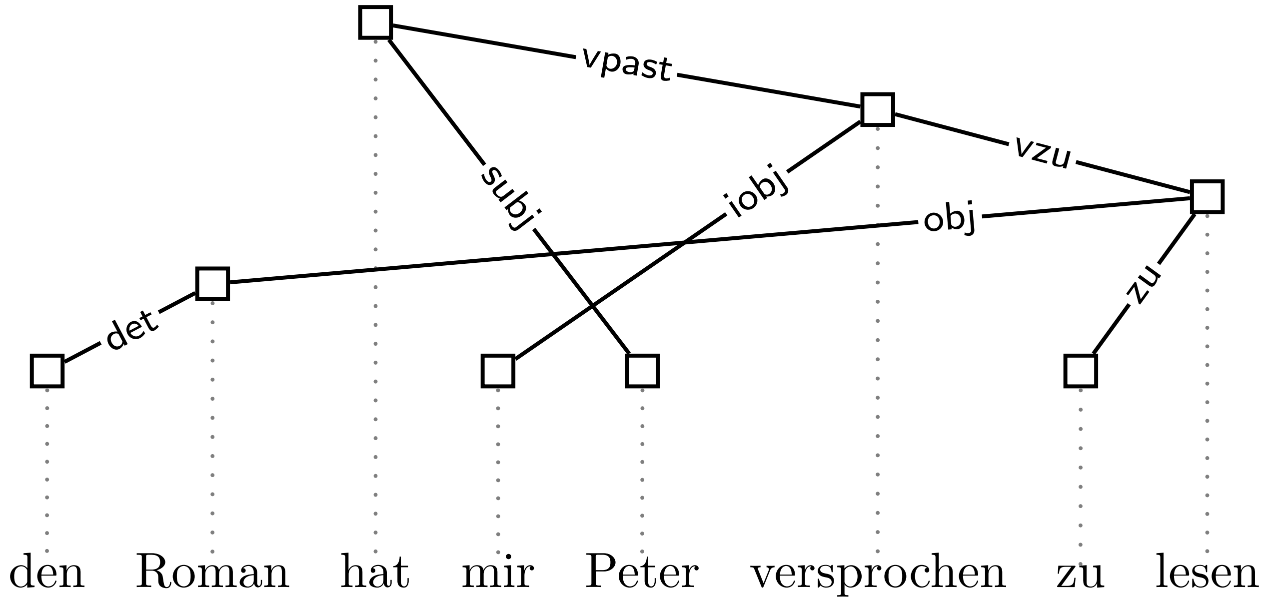
Model-theoretical view

- tree with edges labeled with grammatical relations
- must satisfy lexically assigned subcat frames
- must satisfy edge-specific agreement restrictions



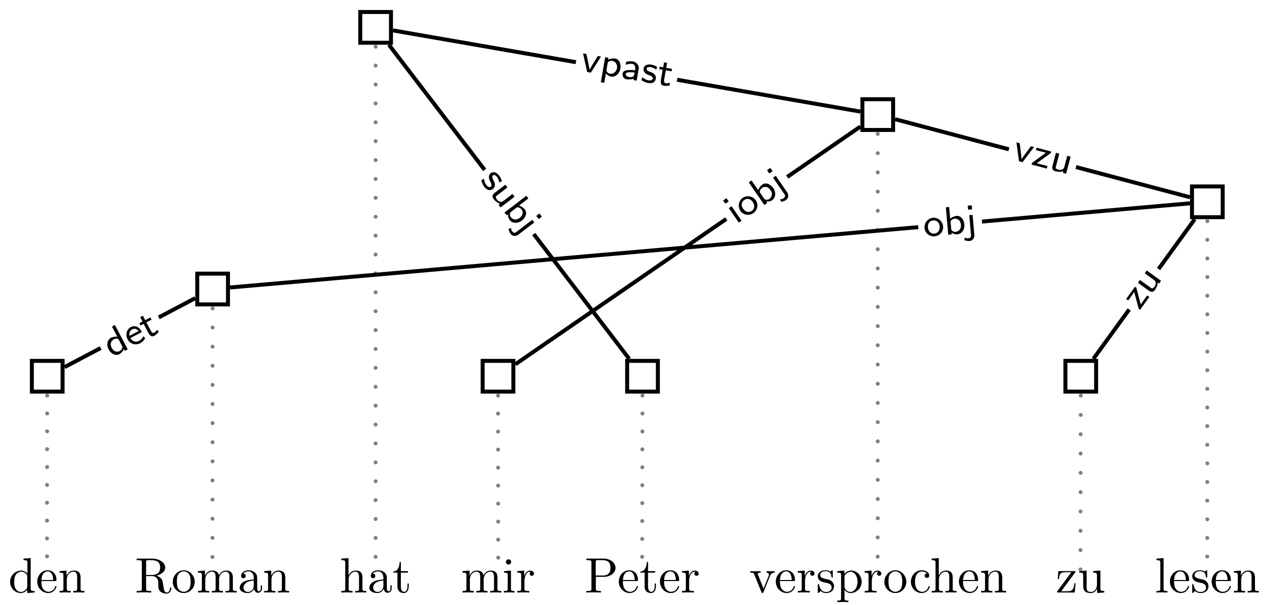
Constraint view

- this is a constraint satisfaction problem
- given n nodes: finitely many labeled trees
- pick one, check the constraints



Constraint propagation technique

- non-deterministic *generate and test* is inefficient
- use *constraint propagation* to prune the search space

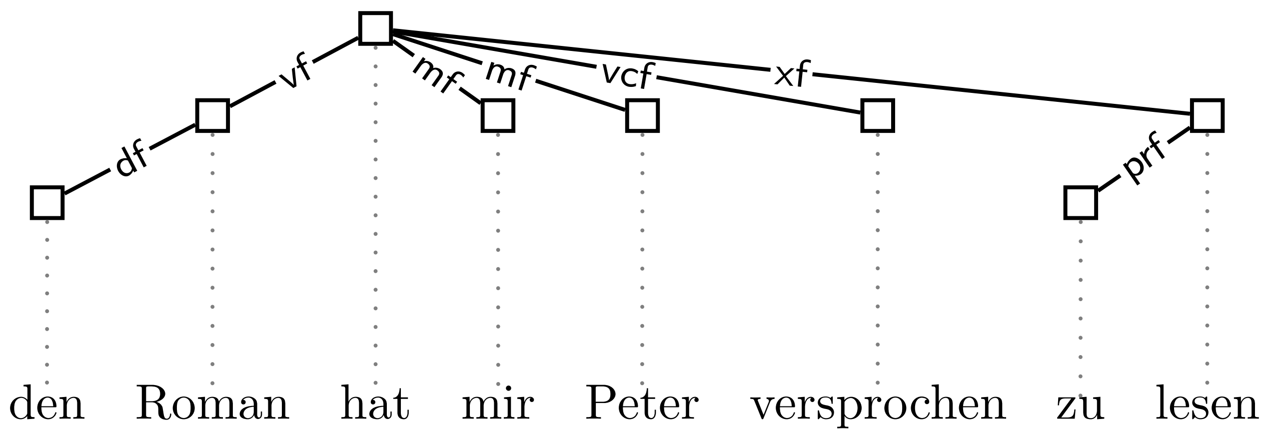


Word-order

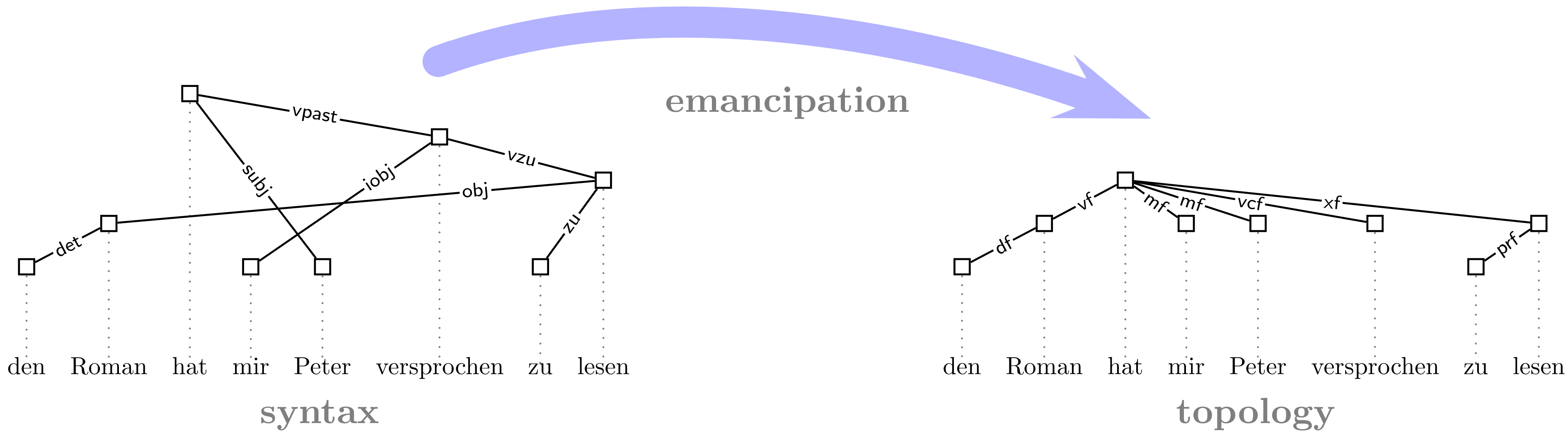
- $*$ *den hat lesen mir Peter Roman versprochen zu*
- tradition of German descriptive syntax: topological fields

  $[\text{den Roman}]_{\text{VF}}[\text{hat}]_{\text{V2}}[\text{mir Peter}]_{\text{MF}}[\text{versprochen}]_{\text{VC}}[\text{zu lesen}]_{\text{NF}}$

idea: topological structure as a dependency tree

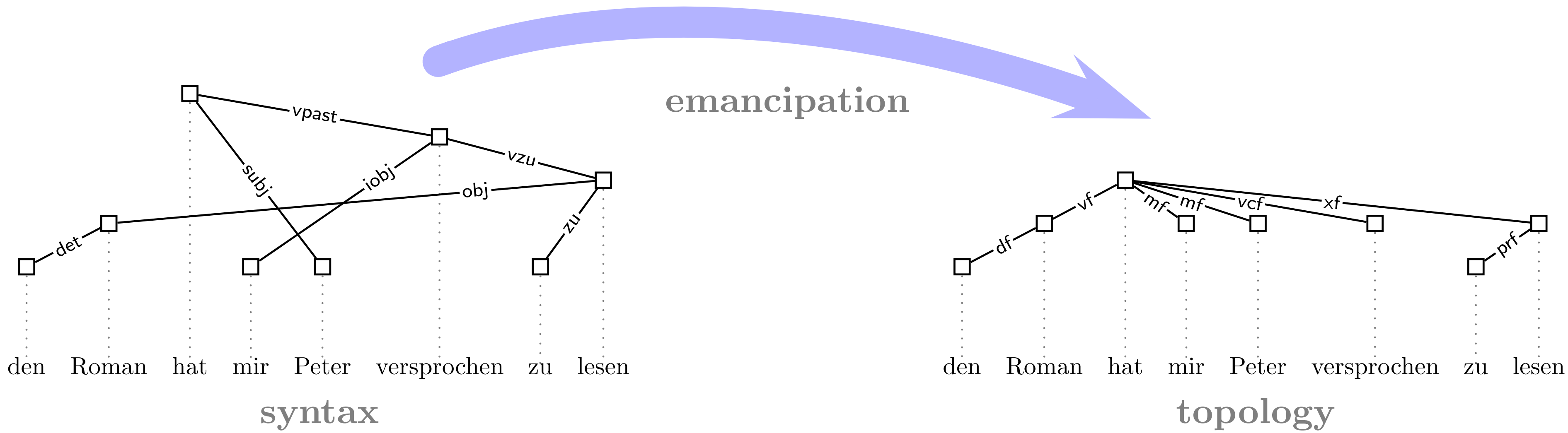


Topological dependency grammar (TDG)



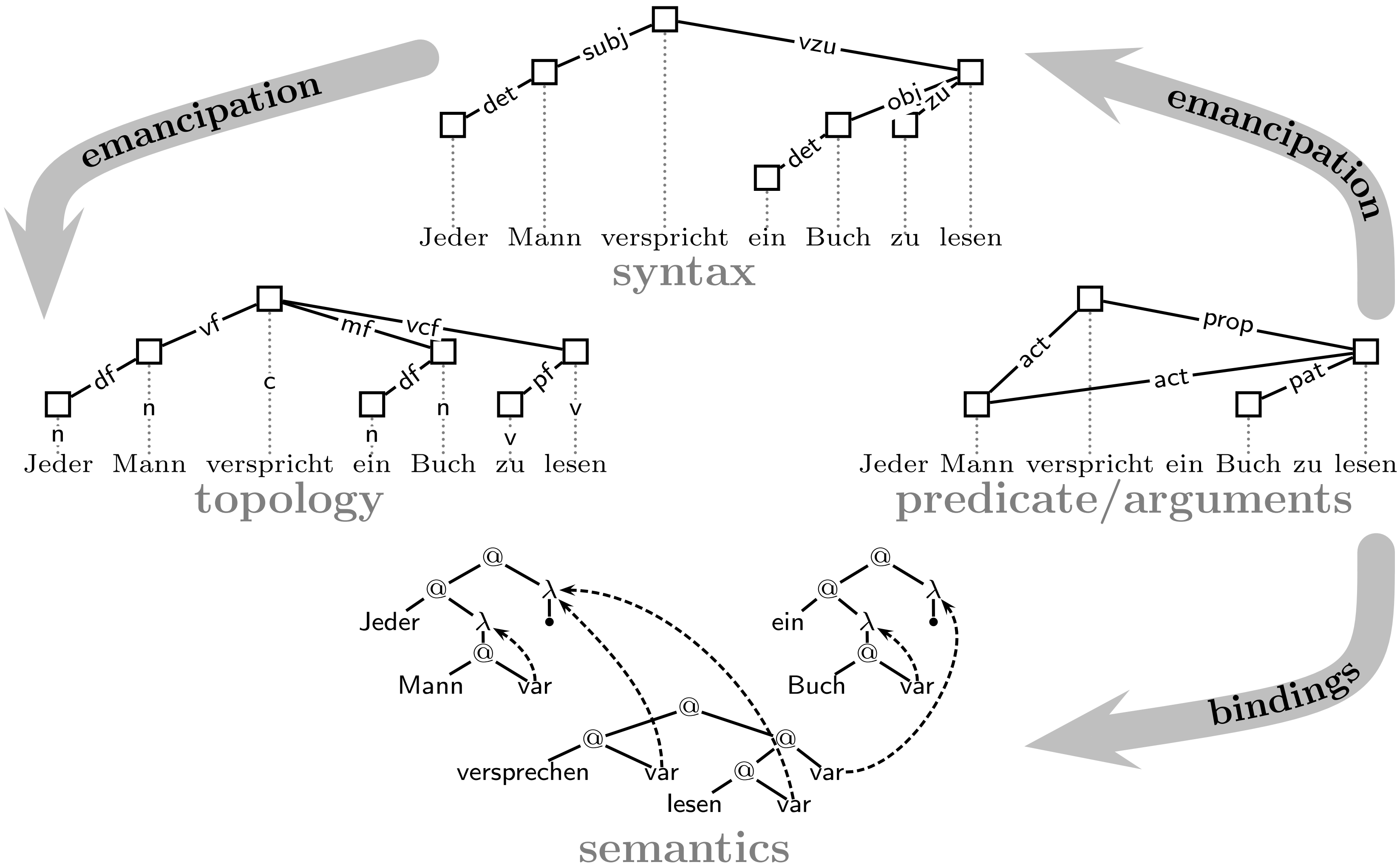
- a TDG analysis has 2 dimensions
- tree of syntactic dependencies (non-ordered)
- tree of topological dependencies (ordered & projective)

Topological dependency grammar (TDG)



- dimensions are not independent
- coupled by the lexicon:
  - syntax: assignment of a subcat frame
  - topology: assignment of a topological frame
- coupled by a **relation** of emancipation
  - syntax and topology are mutually constraining

Towards a syntax/semantics interface



Multi-dimensional dependency analyses

- models: dependency structures (tree or dag)
- lexicon: subcat/topological/valency frames
- interactions: relation of emancipation

observations:

- new language modeling methodology
- needs convenient support:
    - principled way to introduce new dimensions
    - and to state their interactions

Extensible dependency grammar (XDG)

- support the declarative specification of a grammar instance
- arbitrarily many dimensions
- dependency structures as models
- well-formedness conditions (principles library)
- dimensions coupled by:
  - each lexical entry simultaneously constrains all dimensions *(e.g. subcat+topology+valency frames)*
  - inter-dimensional constraints (principles library) *(e.g. emancipation)*

- macroscopic phenomena are emergent properties of configurational interactions

XDG Development Kit (XDK)

- declarative specification of grammar instances
- static typing
- extensible library of parametric principles
- metagrammar facilities
  - organize and structure the lexicon
  - abstraction, inheritance, composition, alternation
- automatic computational support
  - constraint-based parser
  - GUI

Generative vs. model-theoretic syntax

- generative syntax: traditional
- model-theoretic syntax: term coined by Rogers (1996), see also Pullum's ESSLLI 2003
- broadly characterize and constrast them
- generative vs elimnative parsing

2 perspectives on logic

- **syntactic perspective:**
  - how to derive expressions from other expressions
  - proof theory
  - example: modus-ponens

- **semantic perspective:**
  - interpret expressions over models
  - state when a model satisfies an expression
  - example: dominance

2 perspectives on syntax

- **generative syntax:**
  - a grammar is a device for recursively enumerating sets of expressions
  - example: production rules

- **model-theoretic syntax:**
  - we assume a universe of expressions
  - example: typed feature structures
  - state (universal) conditions (constraints) that they must satisfy to be deemed grammatical
  - example: LFG (Bresnan&Kaplan 82, Kaplan 95)

Model-theoretic syntax is **not** generative syntax with constraints (Pullum)

Pullum on generative syntax

- consider PP $\longrightarrow$ P NP
- does this say that Ps precede NPs?
- no, because we could also have PP $\longrightarrow$ NP P
- *everything depends on what the rest of the grammar says*
- *minor changes in a GES grammar can have catastrophic effects on the language that it generates*

Theoretical and practical consequences

- **linguistic modeling:**
  - GES: to cover more, you need to say more
  - MTS: to cover less, you need to say more
- **grammar engineering:**
  consider modular development. When you combine packages:
  - GES: you never get less
  - MTS: you never get more
- **processing:**
  - generative/constructive parsing
  - eliminative parsing

Generative/constructive parsing

$[\text{die}]_D \quad [\text{Frau}]_N \quad [\text{schreibt}]_V \quad [\text{einen}]_D \quad [\text{Roman}]_N$

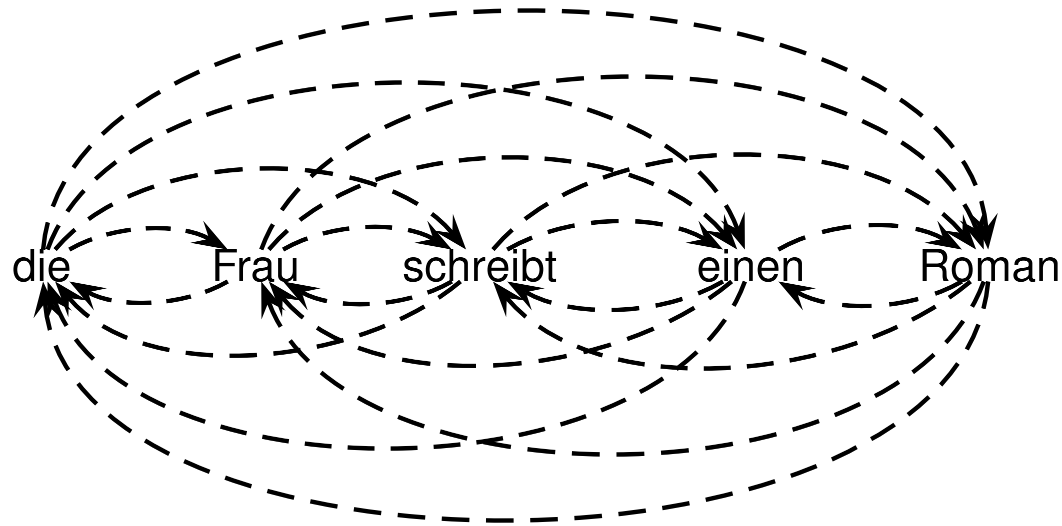
$[[\text{die}]_D\ [\text{Frau}]_N]_{NP} \qquad [[\text{einen}]_D\ [\text{Roman}]_N]_{NP}$

$[[\text{schreibt}]_V\ [[\text{einen}]_D\ [\text{Roman}]_N]_{NP}]_{VP}$

$[[[\text{die}]_D\ [\text{Frau}]_N]_{NP}\ [[\text{schreibt}]_V\ [[\text{einen}]_D\ [\text{Roman}]_N]_{NP}]_{VP}]_S$

- start with just the lexical items
- incrementally assemble items into larger fragments
- until a complete analysis is obtained

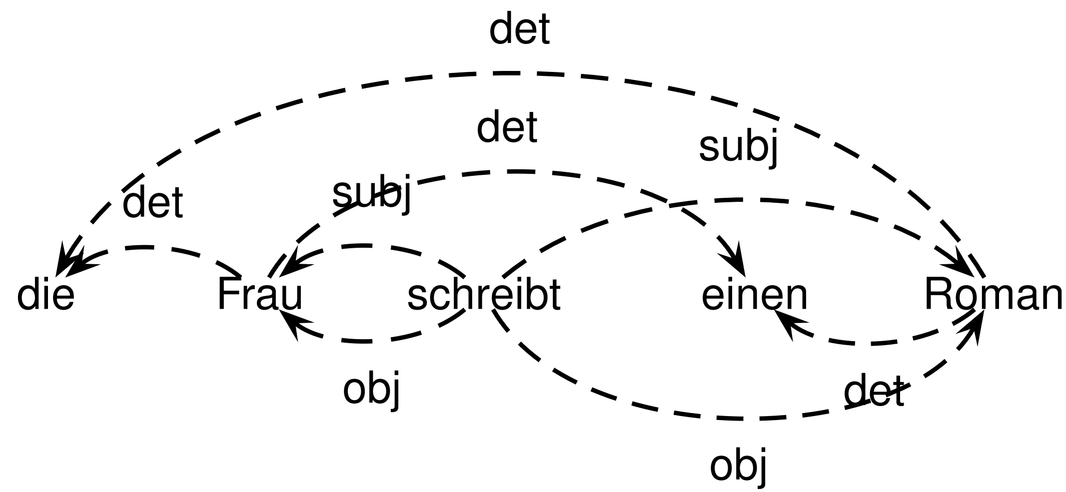
Eliminative parsing



subcat:	{ }	{det}	{subj, obj}	{ }	{det}
agr:	nom&fem	fem		acc&masc	masc

- start with ambiguous representation of all possible analyses
- incrementally disambiguate
- initial candidates are all possible arrows labeled in all possible ways

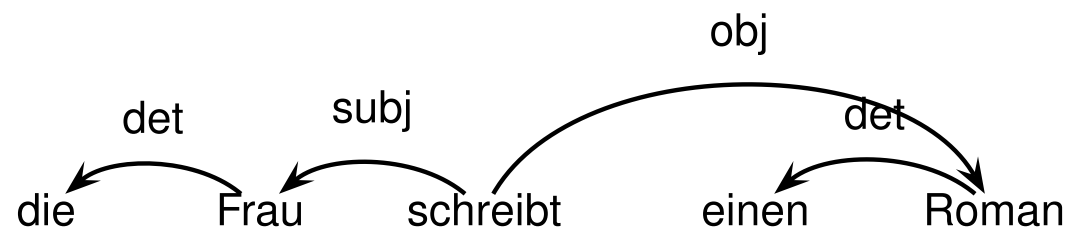
Eliminative parsing



subcat:	{ }	{det}	{subj, obj}	{ }	{det}
agr:	nom&fem	fem		acc&masc	masc

- subcat constraints eliminate many possibilities (model elimination)

Eliminative parsing



subcat:	{ }	{det}	{subj, obj}	{ }	{det}
agr:	nom&fem	fem		acc&masc	masc

- agreement constraints then suffice to determine the rest

Controlling valid analyses

- chosen class of models: labeled trees
- which ones are valid grammatical analyses?

characterization in terms of

- category
- subcategorization
- restriction / agreement

Controlling valid analyses

in a dependency-based approach, the fundamental constructor is the labeled edge $\stackrel{\ell}{\longrightarrow}$

category	$\text{—}\ell\text{⟶}D$	lexicalized (in)
subcategorization	$H\text{—}\ell\text{⟶}$	lexicalized (out)
restriction	$H\text{—}\ell\text{⟶}D \;\Rightarrow\; C(D)$	principle
agreement	$H\text{—}\ell\text{⟶}D \;\Rightarrow\; C(H,D)$	principle

Lexicalized subcategorization

- each word must have the right kinds of dependents
- i.e. each node must have the right kinds of out-going edges

lexicalized subcat descriptions

$$\textit{Roman} \mapsto \left[\ \textsf{out} \ :\ \{\texttt{det}!, \texttt{adj}*\}\ \ldots\ \right]$$

$$\textit{lesen} \mapsto \left[\ \textsf{out} \ :\ \{\texttt{zu}!, \texttt{obj}?\}\ \ldots\ \right]$$

det!	exactly one determiner
adj*	one or more adjectives
obj?	one optional object

Lexicalized category

- lexicalized subcat ensures that we have the right kinds of out-going edges, but they they can connect arbitrarily
- each word may only fill certain grammatical functions
- i.e. each node must have the right kind of in-coming edge

lexicalized (super)cat descriptions

$$\textit{Roman} \mapsto \left[\ \textsf{in} \ : \ \{\texttt{subj}?, \texttt{obj}?\} \ \ldots \ \right]$$

$$\textit{lesen} \mapsto \left[\ \textsf{in} \ : \ \{\texttt{vinf}?\} \ \ldots \ \right]$$