A Relational Syntax-Semantics Interface Based on Dependency Grammar

Ralph Debusmann[1]

Denys Duchier[3]

Alexander Koller[2]

Marco Kuhlmann[1]

Gert Smolka[1]

Stefan Thater[2]

[1] Programming Systems Lab, Saarbrücken, Germany

[2] Computational Linguistics, Saarbrücken, Germany

[3] LORIA, Nancy, France

Overview

1. *Background and Motivation*
2. Extensible Dependency Grammar
3. A Relational Syntax-Semantics Interface
4. Summary and Future Work

Background

- the traditional perspective on the syntax-semantics interface is *functional*, i.e. semantic representations are obtained from the syntax tree by structural induction
- but some phenomena (e.g. scope, anaphora) are *not functional*: *one* syntax tree has *several* readings

Some Approaches

- *Categorial Grammar* recasts *semantic ambiguity as syntactic ambiguity* (Montague 1974, Steedman 1999, Moortgat 2002)
- *GB* assumes a *non-deterministic mapping* from syntax to semantics (“Logical Form”) (Chomsky 1986)
- *LFG* makes use of *functional uncertainty* to allow for a restricted form of relationality (Bresnan/Kaplan 1982, Kaplan/Maxwell III 1988)
- *Underspecification* restores functionality by making the semantics less ambiguous, e.g. *MRS*, *CLLS* (Copestake et al. 2004, Egg et al. 2001)

This talk

- we present a *completely relational syntax-semantics interface*
- formalized using *Extensible Dependency Grammar* (XDG)
- the *XDG solver* for parsing supports the *concurrent* flow of possibly partial information such that syntax and semantics can *mutually constrain each other*

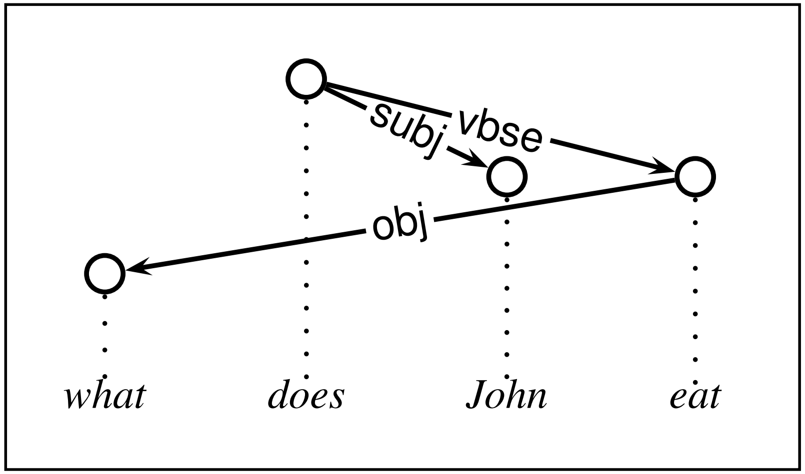
Overview

1. Background and Motivation
2. *Extensible Dependency Grammar*
3. A Relational Syntax-Semantics Interface
4. Summary and Future Work

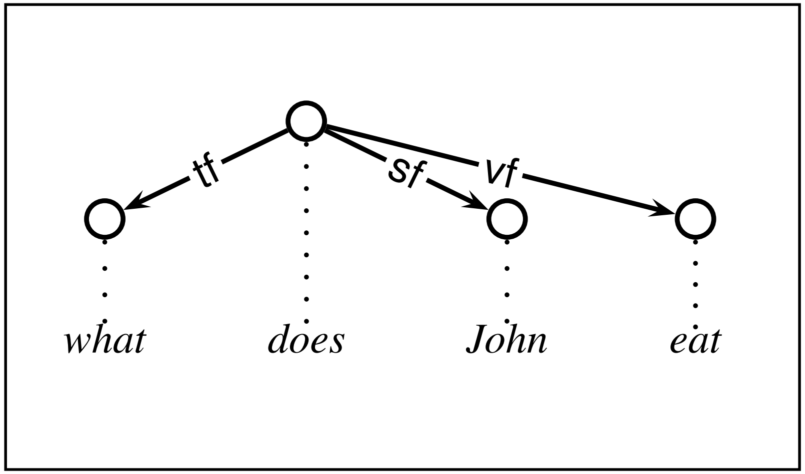
Extensible Dependency Grammar

- XDG is a *graph description language* designed for the dependency-based modeling of natural language, based on Topological Dependency Grammar (TDG) (Duchier/Debusmann 2001)
- an XDG *analysis* involves arbitrary many *graph dimensions* sharing the same set of nodes, but having different edges
- XDG is *strongly lexicalized*, and has a *powerful lexicon language* supporting e.g. lexical inheritance a la HPSG

An Example Analysis



Immediate Dominance (ID)

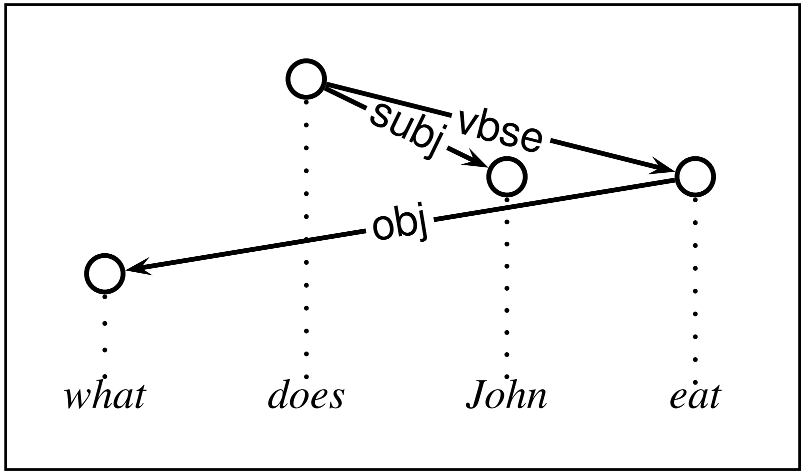


Linear Precedence (LP)

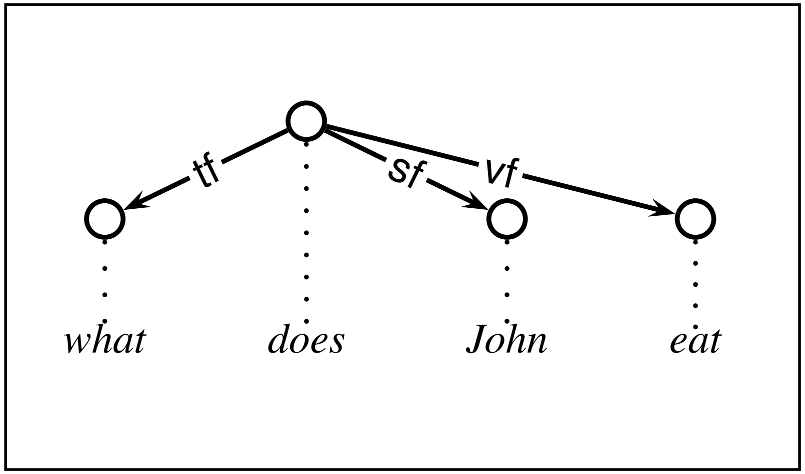
Principles

- *principles* determine the *well-formedness conditions* of XDG analyses, constraining:
  - global properties of graphs (e.g. treeness)
  - local properties of nodes (e.g. valency)
  - structural relations between graphs (e.g. climbing)
- the latter is done by *multi-dimensional principles*, as opposed to *one-dimensional principles*

Treeness Principle



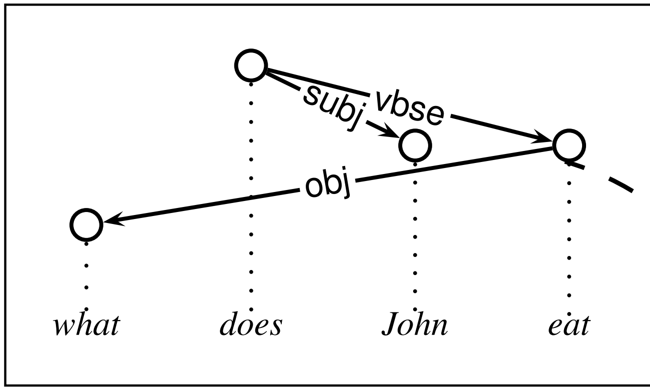
Immediate Dominance (ID)



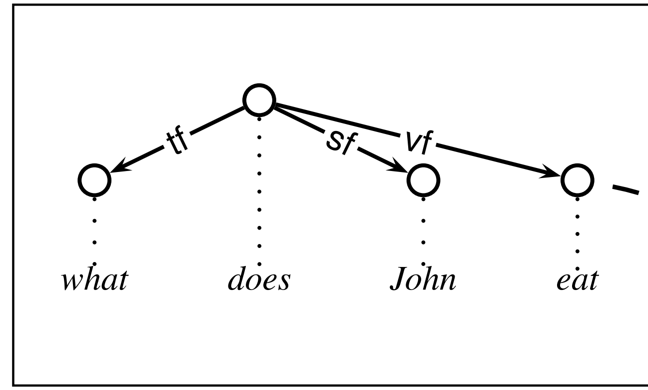
Linear Precedence (LP)

- both graphs must be trees

Valency Principle



Immediate Dominance (ID)



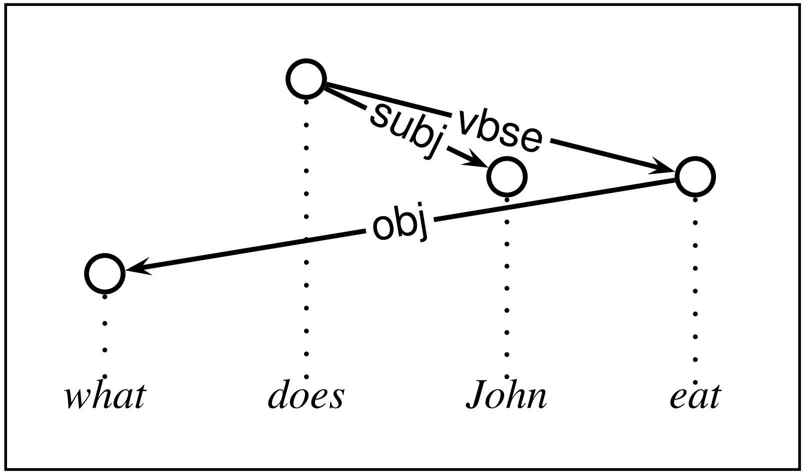
Linear Precedence (LP)

[in: {vbse?}
out: {obj!}]

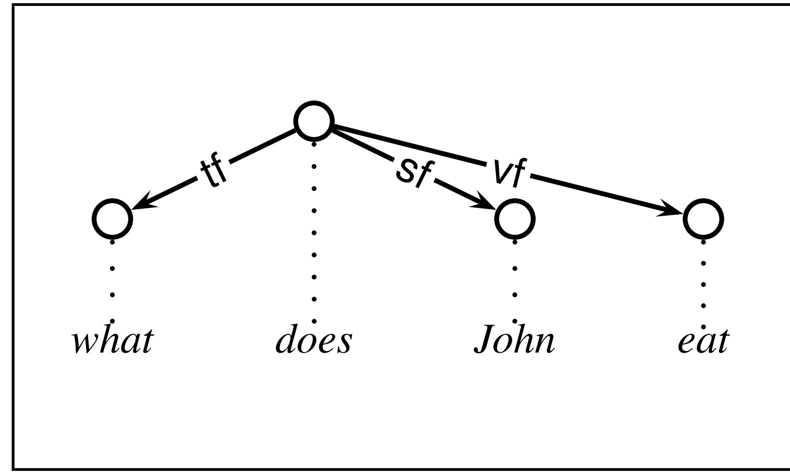
[in: {vf?}
out: {}]

- both graphs must satisfy the in and out specifications in the lexicon

Order Principle



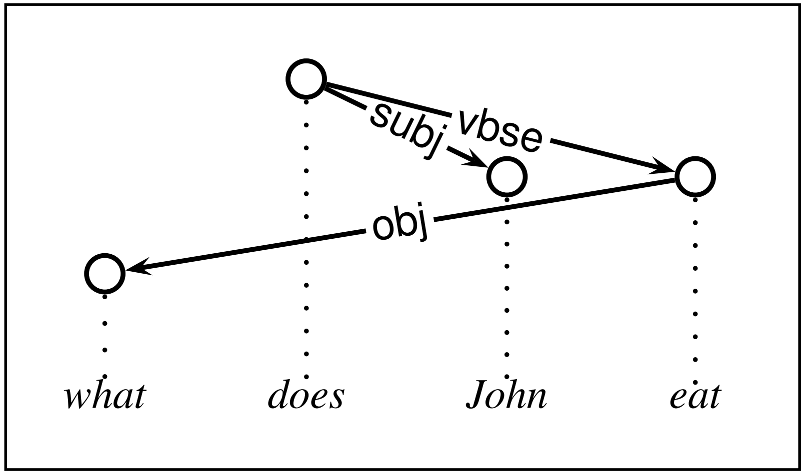
Immediate Dominance (ID)



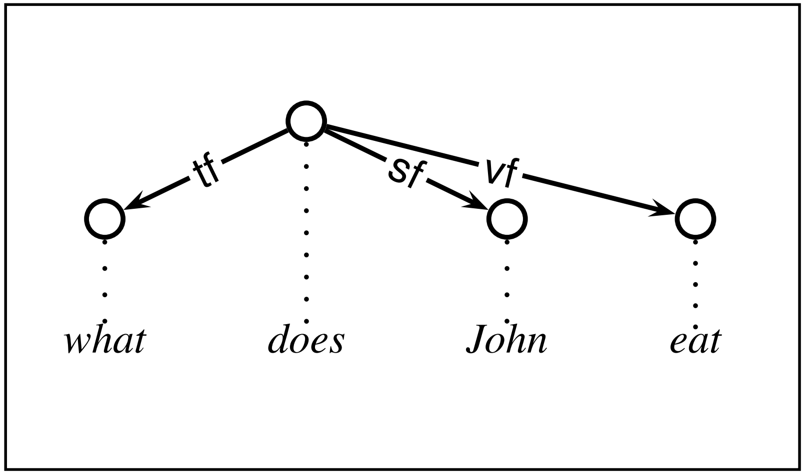
Linear Precedence (LP)

- the LP tree is ordered and projective (the ID tree is unordered)
- here: $tf \prec sf \prec vf$

Climbing Principle



Immediate Dominance (ID)



Linear Precedence (LP)

- the LP tree must be a flattening of the ID tree
- Also called *lifting* or *emancipation* (Kahane et al. 1998, Gerdes/Kahane 2001)

Processing

- the *XDG solver* implements an axiomatization of XDG as a *constraint satisfaction problem* (Duchier 1999, Duchier 2003)
- XDG solver can be used both for *parsing* and *generation*
- all dimensions are processed *concurrently*
- *partial analyses* can be extracted at each point during solving
- solving *efficient* for *small handcrafted grammars*
- solving of *large grammars work in progress*

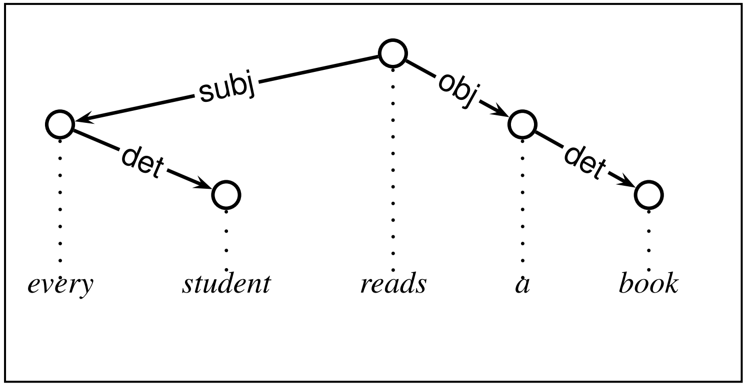
Overview

1. Background and Motivation
2. Extensible Dependency Grammar
3. *A Relational Syntax-Semantics Interface*
4. Summary and Future Work

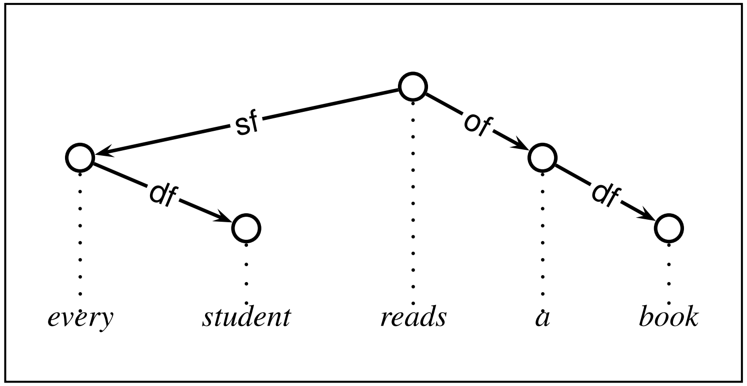
Ingredients

- *Immediate Dominance* tree (ID)
- *Linear Precedence* tree (LP)
- *Predicate-Argument* structure (PA)
  - models *variable binding*
  - resolves e.g. *raising and control constructions*
- *Scope tree* (SC)
  - models the *scopal relationships*, i.e. the structure of the reading
  - can be likened with the TAG *derivation tree*, reflecting how semantic building blocks are put together

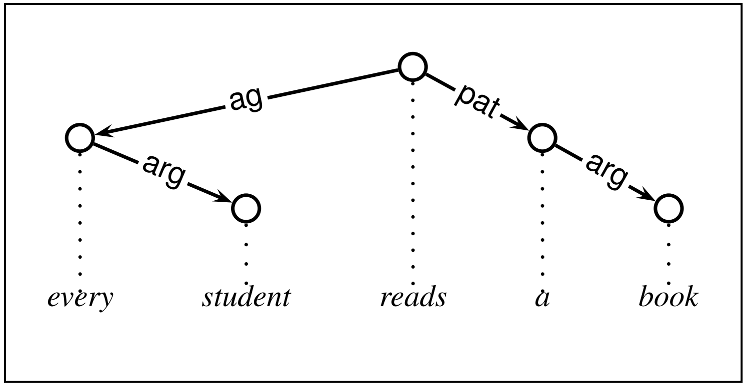
An Example



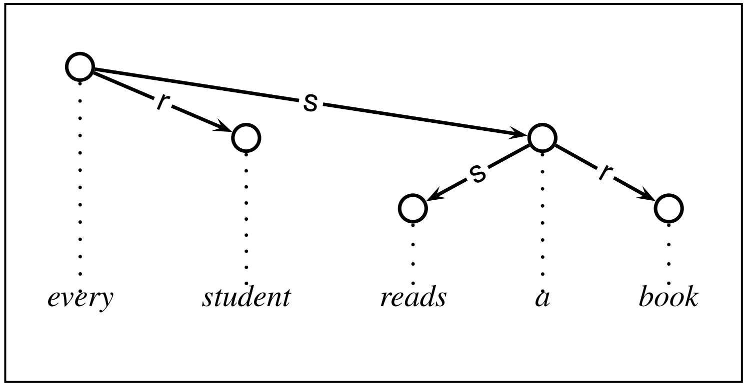
Immediate Dominance (ID)



Linear Precedence (LP)



Predicate−Argument (PA)

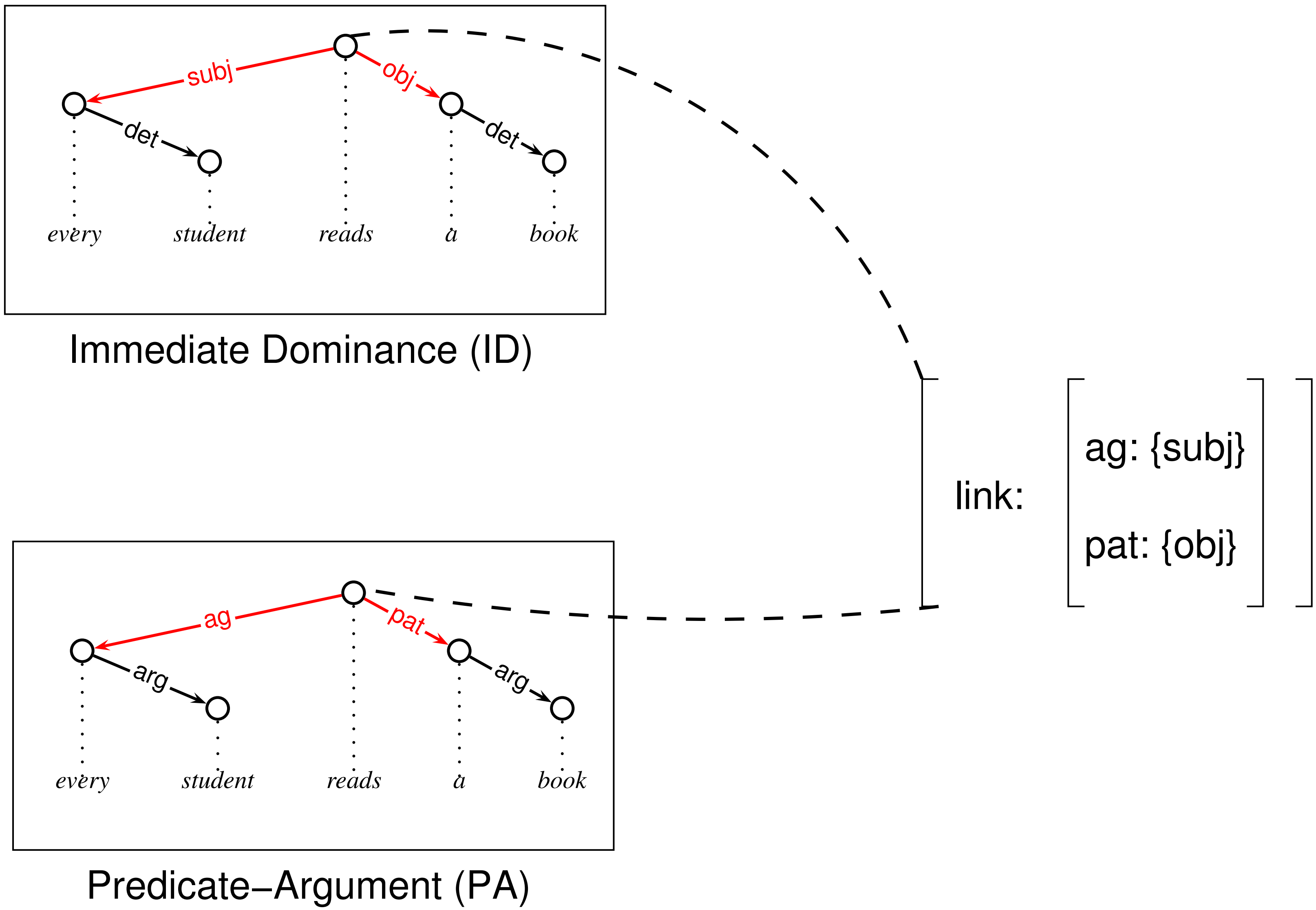


Scope (SC)

Linking Principle

- *multi-dimensional*
- used to state how *semantic arguments* are *realized* in the *syntax*
- *lexicalized*, i.e. capable of handling alternations

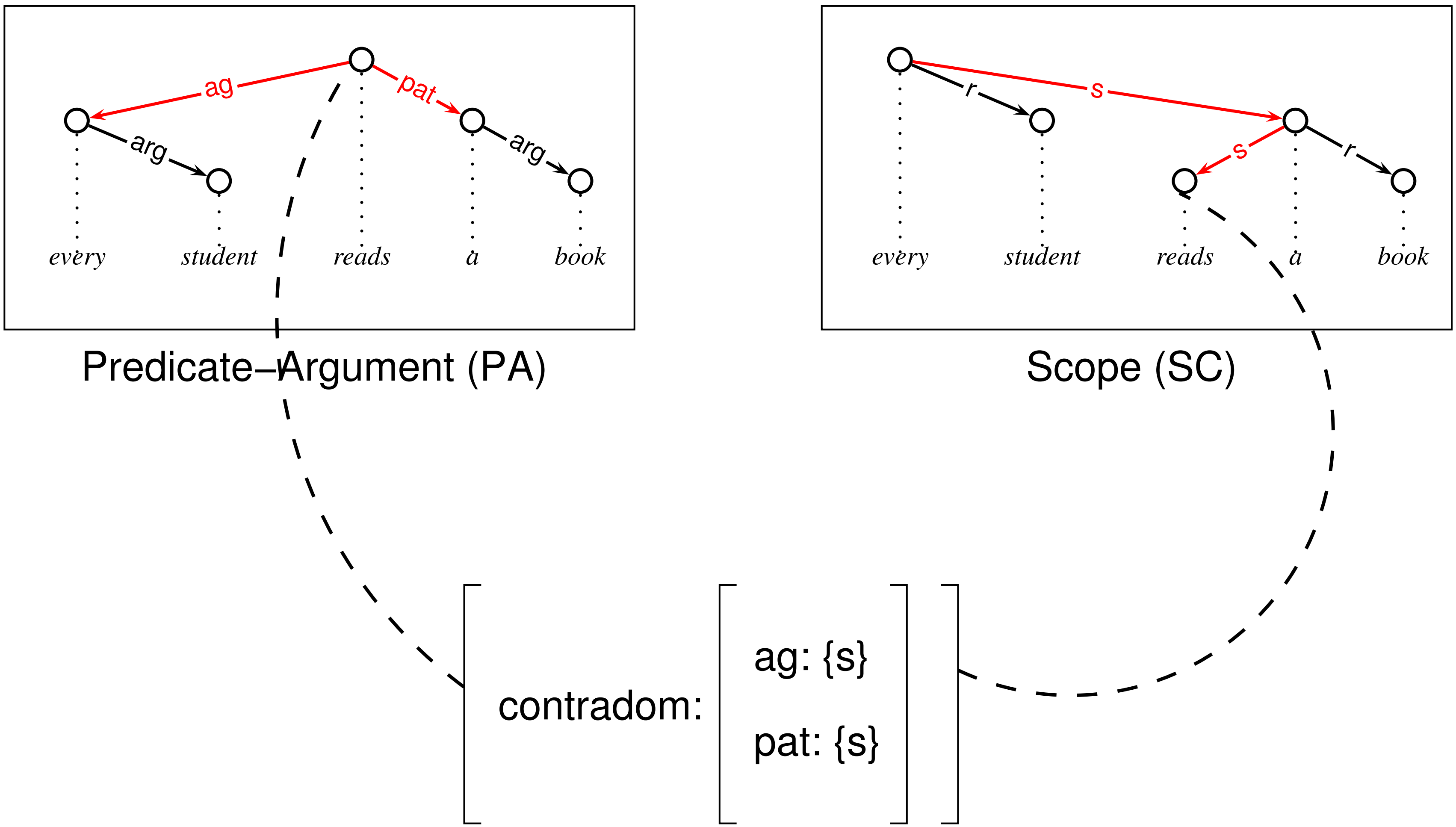
Linking Example



Contra-Dominance

- *multi-dimensional*
- used to constrain the relation between the Predicate-Argument structure and the Scope tree
- also lexicalized

Contra-Dominance Example



Interpretation

- we can translate *XDG analyses* into standard *type-theoretical expressions* (Montague 1974)
- the *Predicate-Argument structure* determines *variable binding*
- the *Scope tree* determines the *structure* of the reading

Interpretation functions

- $L(v)$ *lexical semantic value* of node v
- $P(v)$ *phrasal semantic value* of the entire subtree rooted at v

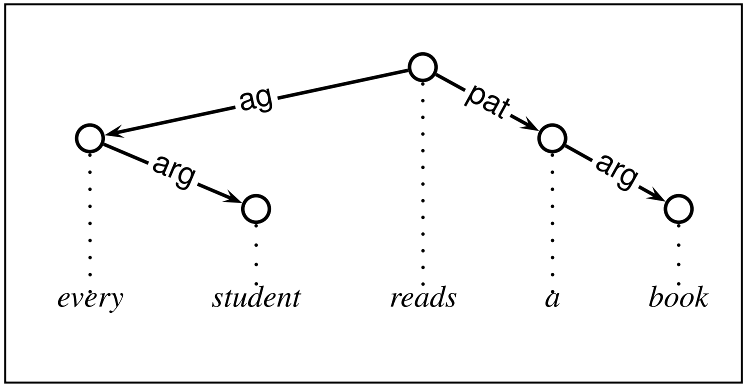
An Example Semantic Lexicon

- $\mathcal{L}(\text{“}\textit{every}\text{”}) = \lambda P \lambda Q \lambda e. \forall x (\mathcal{P}(x) \rightarrow Q(x)(e))$
- $\mathcal{L}(\text{“}\textit{a}\text{”}) = \lambda P \lambda Q \lambda e. \exists x (\mathcal{P}(x) \wedge Q(x)(e))$
- $\mathcal{L}(\text{“}\textit{student}\text{”}) = \textit{student'}$
- $\mathcal{L}(\text{“}\textit{book}\text{”}) = \textit{book'}$
- $\mathcal{L}(\text{“}\textit{reads}\text{”}) = \textit{read'}(\downarrow \textit{pat})(\downarrow \textit{ag})$

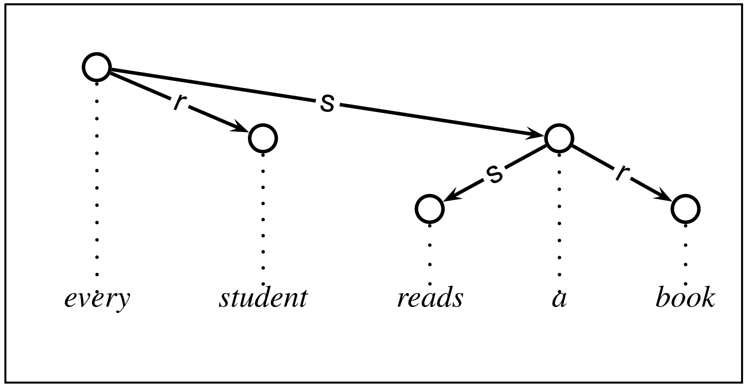
The Phrasal Semantic Value

- $\mathcal{P}(\text{“every”}) = \mathcal{L}(n)(\mathcal{P}(\downarrow \text{r}))(\lambda \downarrow n.\mathcal{P}(\downarrow \text{s}))$
- $\mathcal{P}(\text{“a”}) = \mathcal{L}(n)(\mathcal{P}(\downarrow \text{r}))(\lambda \downarrow n.\mathcal{P}(\downarrow \text{s}))$
- $\mathcal{P}(\text{“student”}) = \mathcal{L}(\text{“student”})$
- $\mathcal{P}(\text{“book”}) = \mathcal{L}(\text{“book”})$
- $\mathcal{P}(\text{“reads”}) = \mathcal{L}(\text{“reads”})$

An Example



Predicate–Argument (PA)



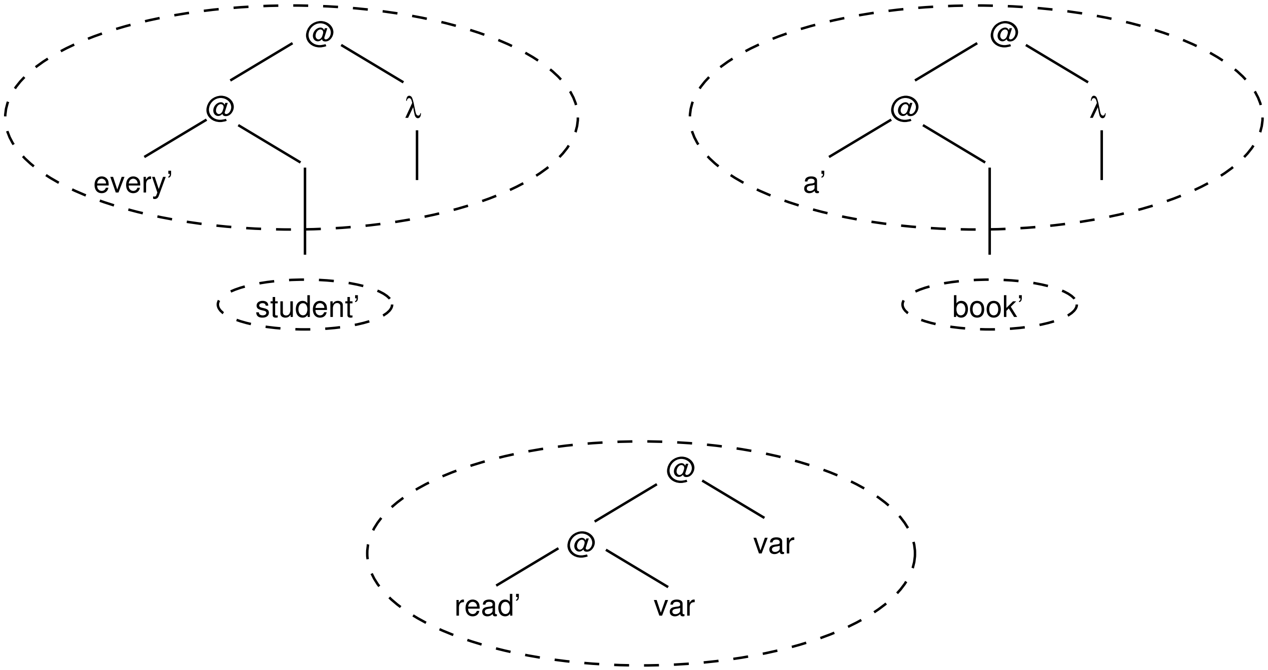
Scope (SC)

$$\mathcal{P}(\text{“every”}) = \ldots =$$
$$\mathcal{L}(\text{“every”})(\mathcal{L}(\text{“student”}))(\lambda x.\mathcal{L}(\text{“a”})(\mathcal{L}(\text{“book”}))(\lambda y.\text{“read’”}(y)(x))) = \ldots =$$
$$\lambda e.\forall x.\text{student'}(x) \rightarrow \exists y.\text{book'}(y) \wedge \text{read'}(y)(x)(e)$$

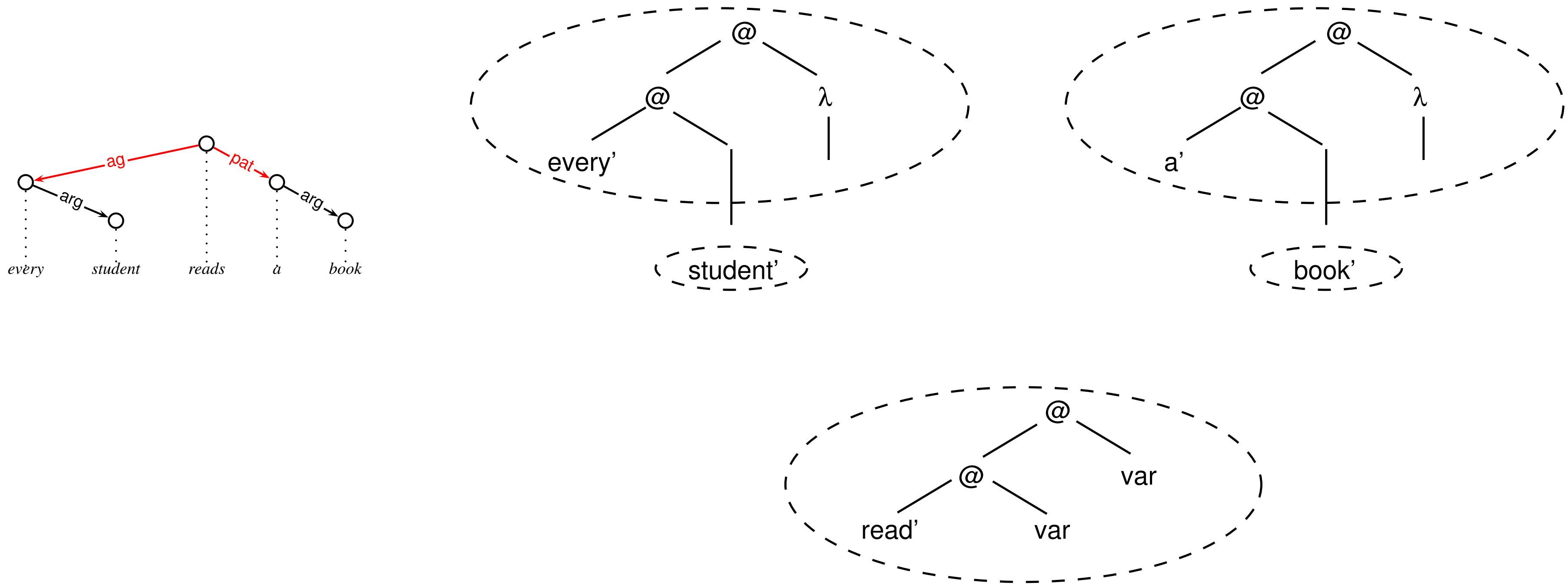
Underspecification

- the Montague-style interpretation presupposes *completely specified analyses*
- we can reformulate the interpretation to support an extraction of *underspecified semantic descriptions* from *partial analyses*
- idea: associate lexical entries with *partial tree descriptions* a la CLLS (Egg et al. 2001)
- the *Predicate-Argument* structure again contributes the variable bindings
- partial information from the *Scope tree* contributes *additional dominance edges*

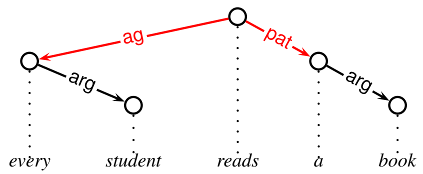
An Example

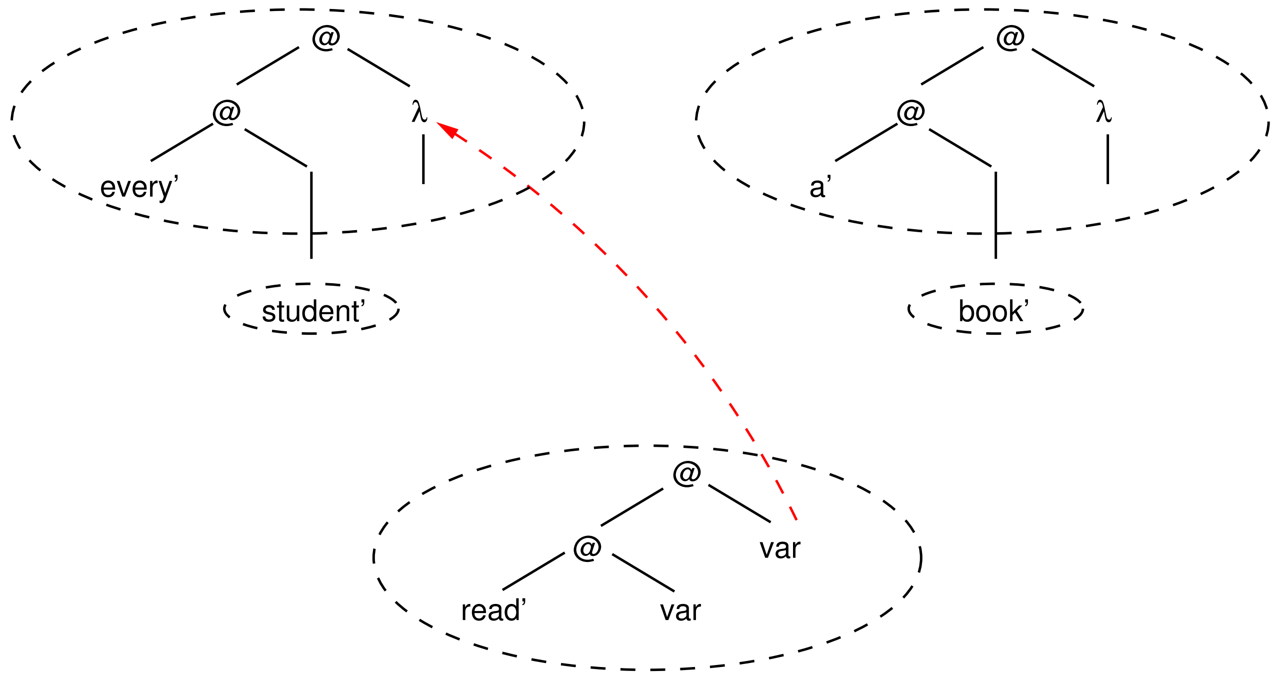


An Example

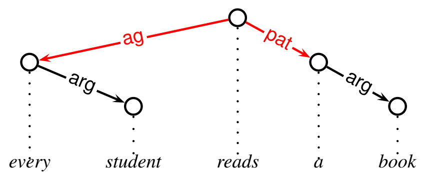

ag
pat
arg
arg
every
student
reads
a
book
@
@
every'
λ
student'
@
@
a'
λ
book'
@
@
var
read'
var

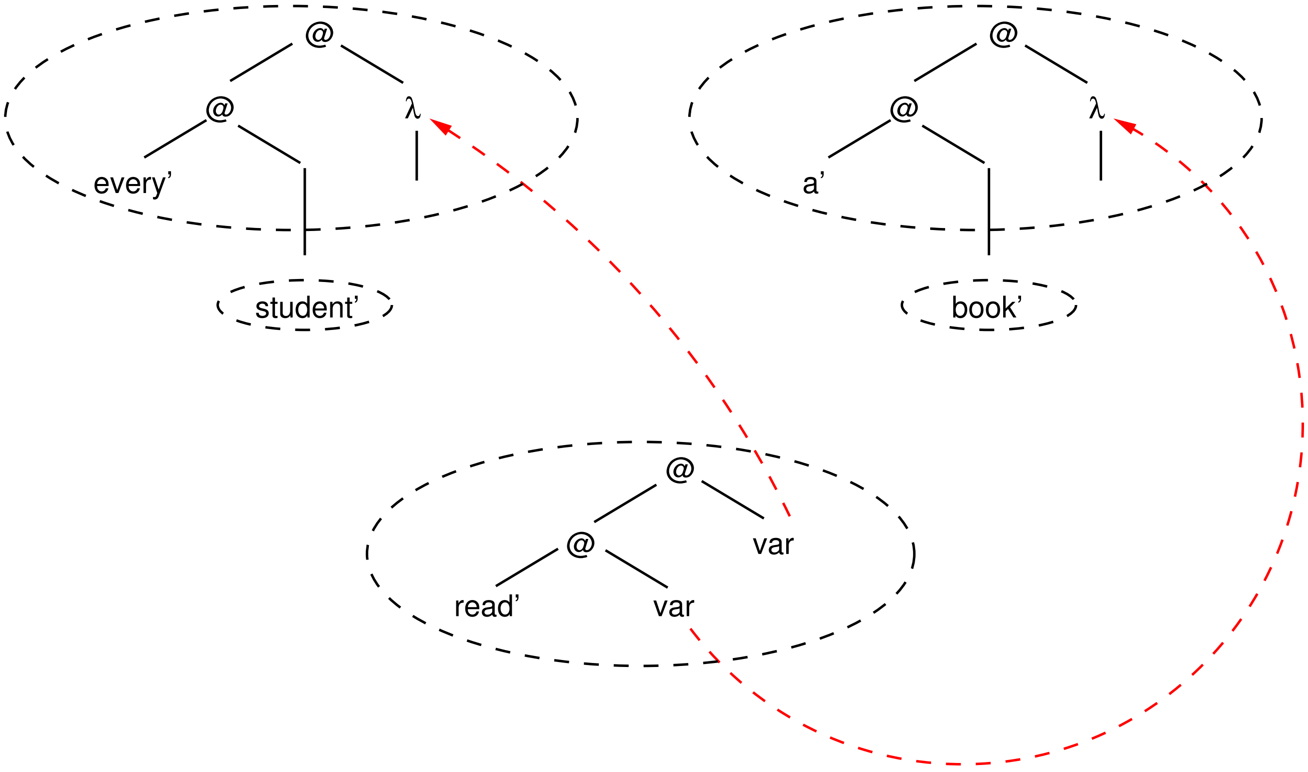
An Example


ag
pat
arg
arg
every
student
reads
a
book

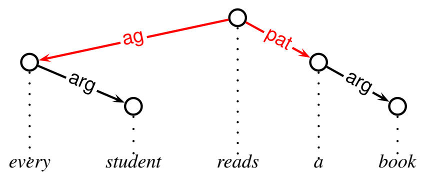

@
@
every'
λ
student'
@
@
a'
λ
book'
@
@
var
read'
var

An Example

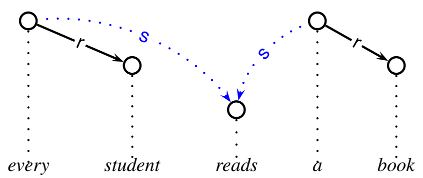

ag
pat
arg
arg
every
student
reads
a
book


@
@
λ
every'
student'
@
@
λ
a'
book'
@
@
var
read'
var

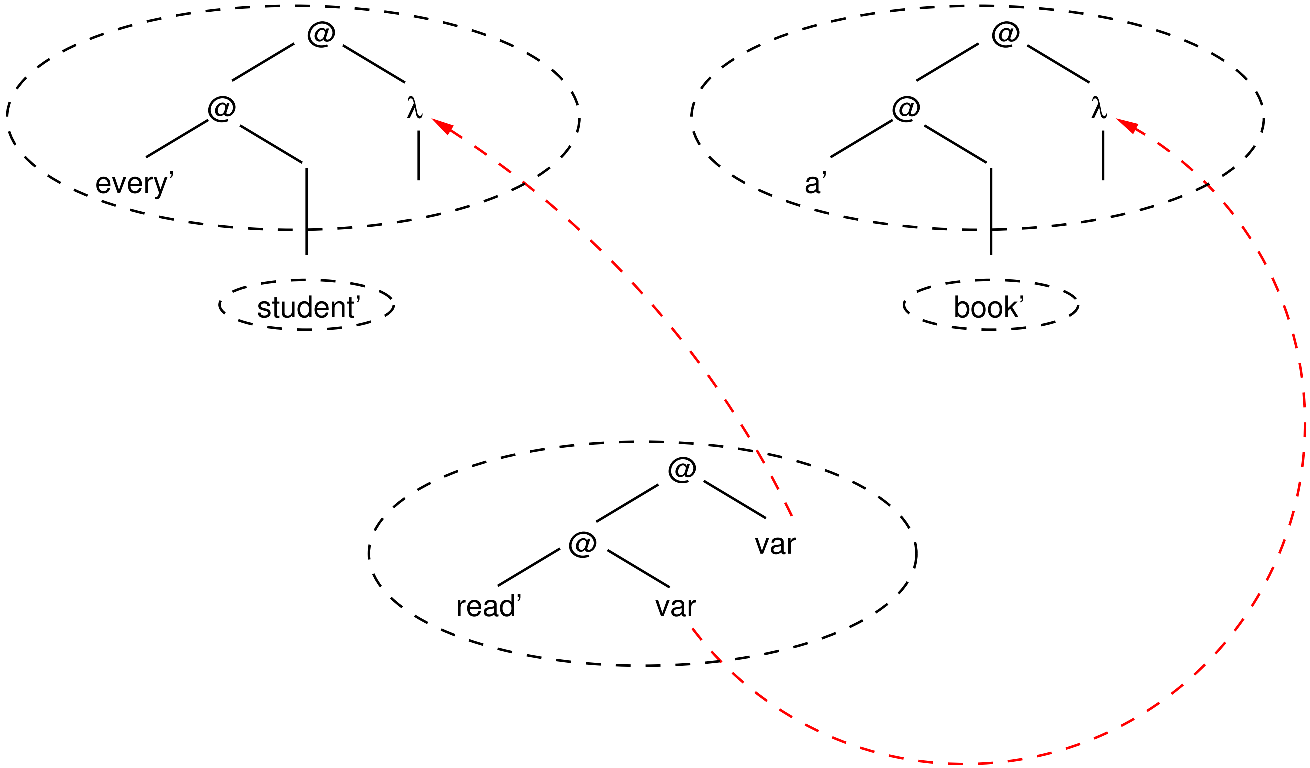
An Example



ag
pat
arg
arg
every
student
reads
a
book

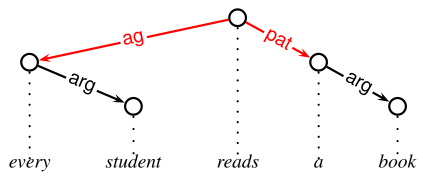


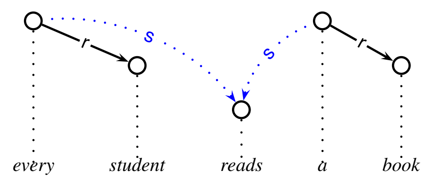
r
s
s
r
every
student
reads
a
book

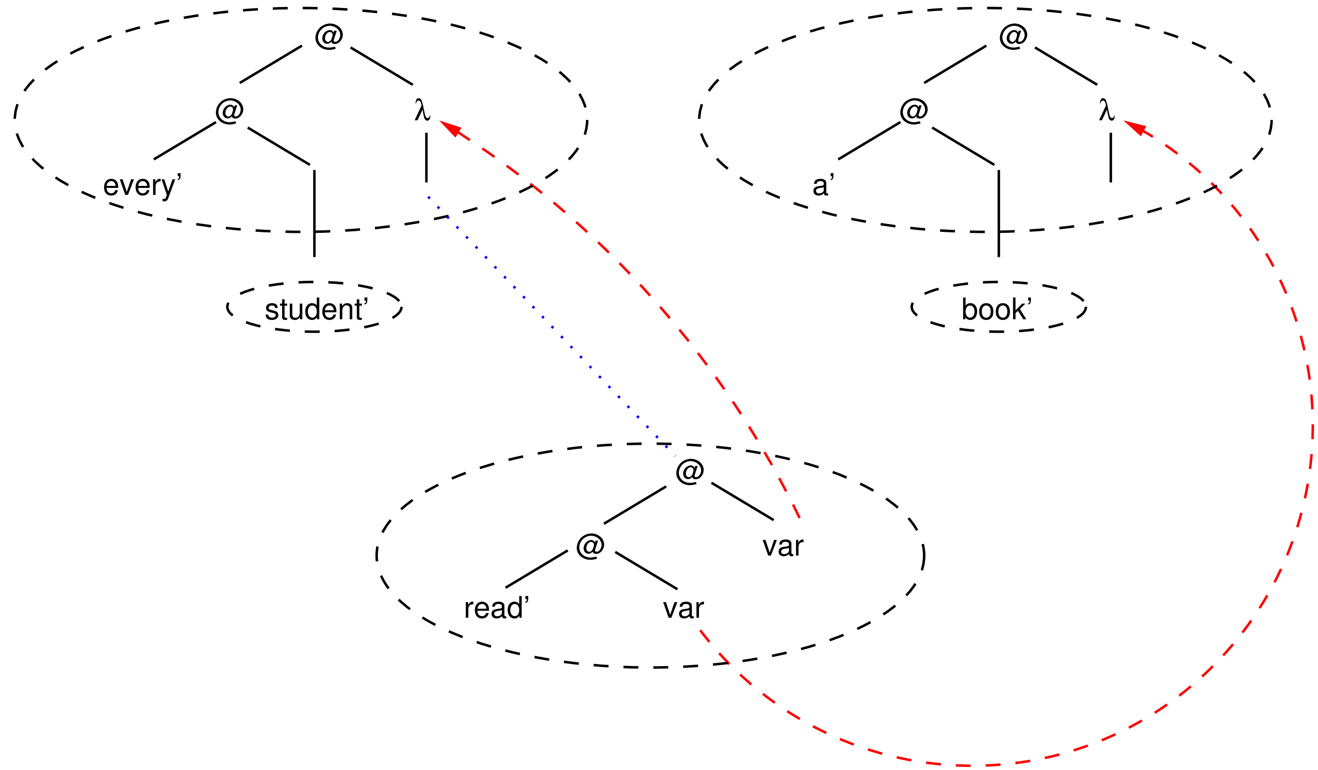


@
@
every'
λ
student'
@
@
a'
λ
book'
@
@
read'
var
var

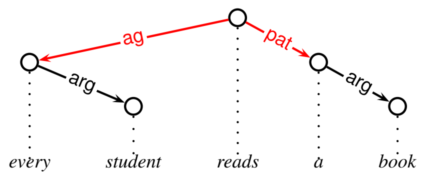
An Example


ag
pat
arg
arg
every
student
reads
a
book

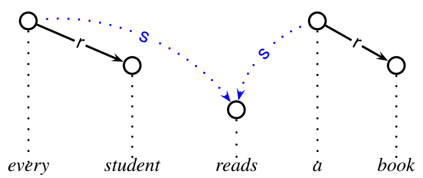

r
s
s
r
every
student
reads
a
book


@
@
λ
every’
student’
@
@
λ
a’
book’
@
@
var
read’
var

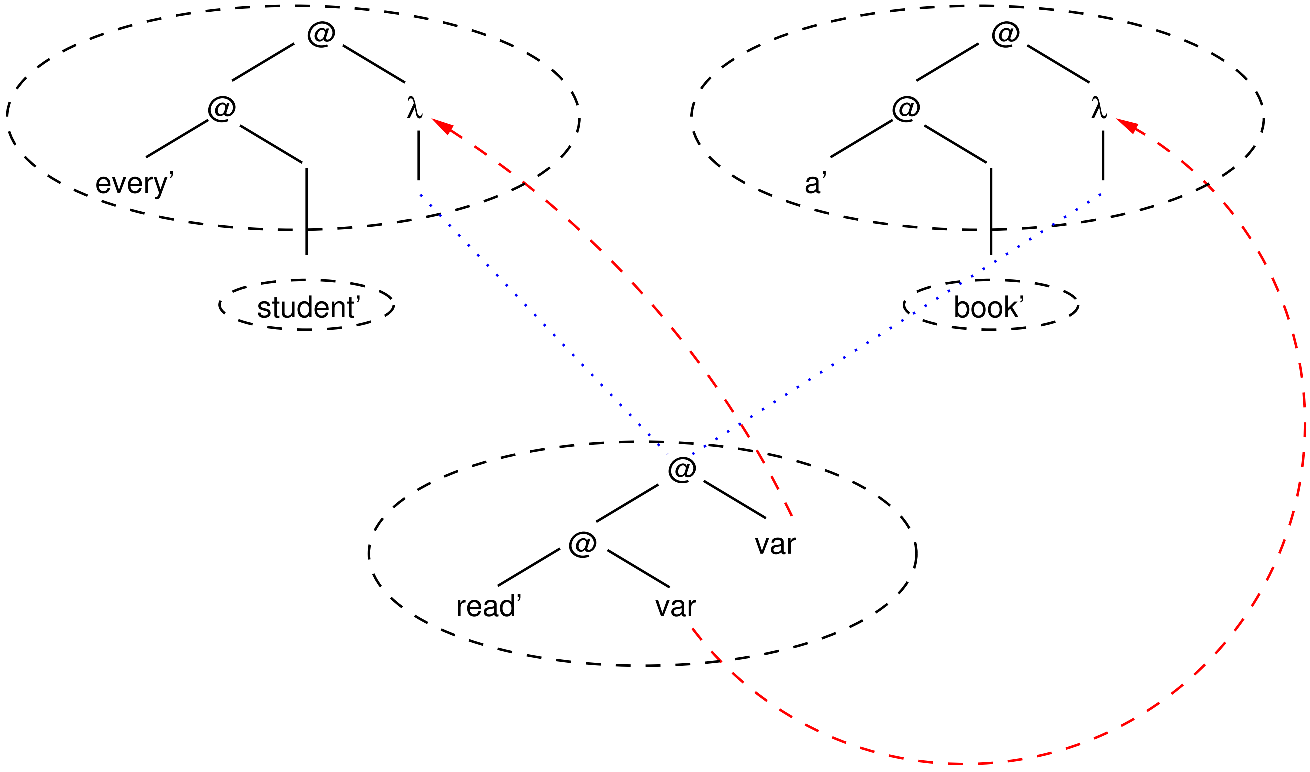
An Example



ag
pat
arg
arg
every
student
reads
a
book



r
s
s
r
every
student
reads
a
book

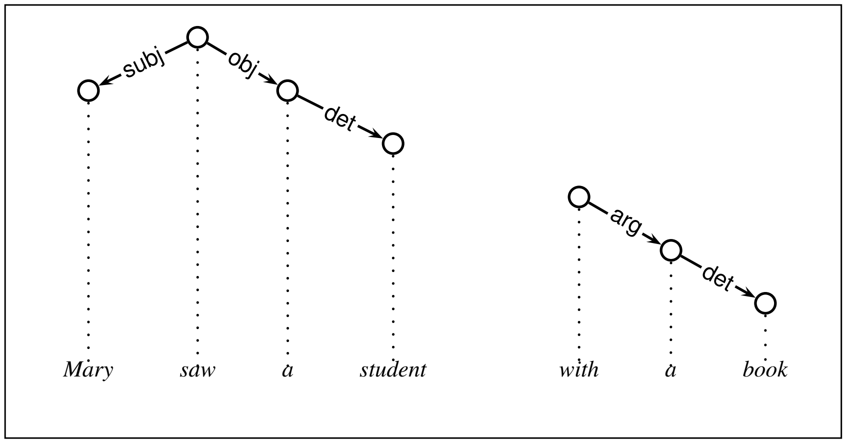


@
@
λ
every'
student'
@
@
λ
a'
book'
@
@
var
read'
var

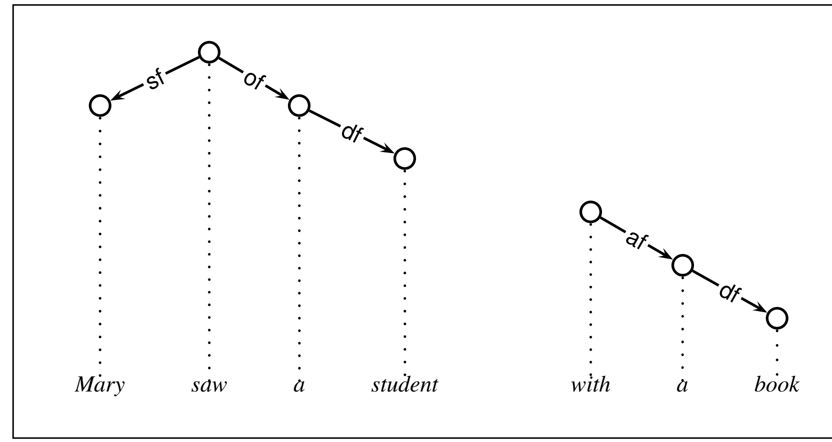
Interaction of Syntax and Semantics

- the relational syntax-semantics interface allows for *inferences from the syntax* to *disambiguate semantics*
- and also vice versa, i.e. *inferences from semantics* can *disambiguate syntax*

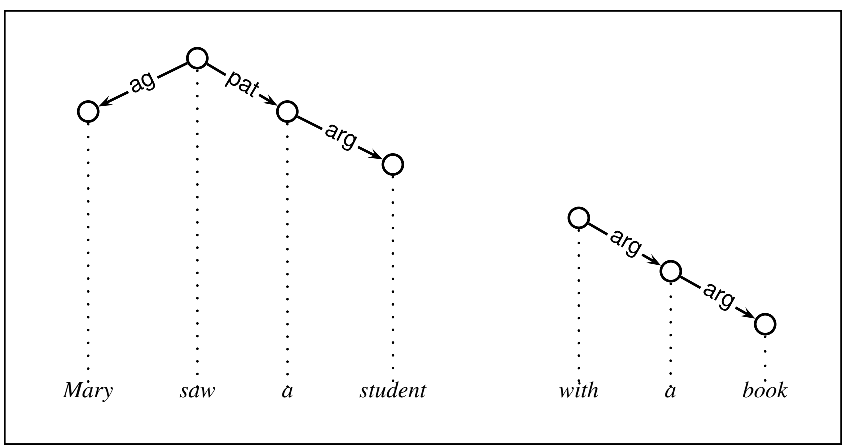
Inferences from syntax to semantics



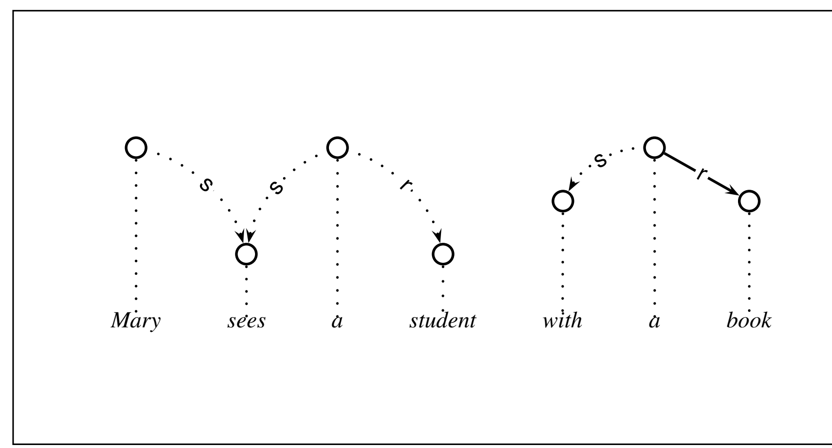
Immediate Dominance (ID)



Linear Precedence (LP)

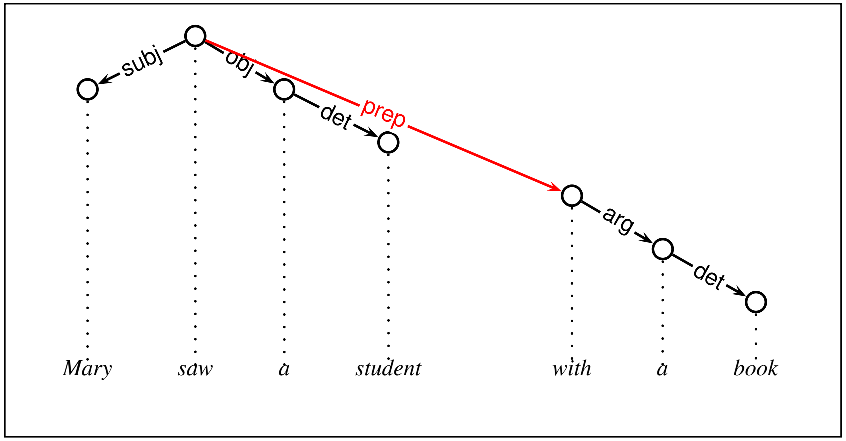


Predicate–Argument (PA)

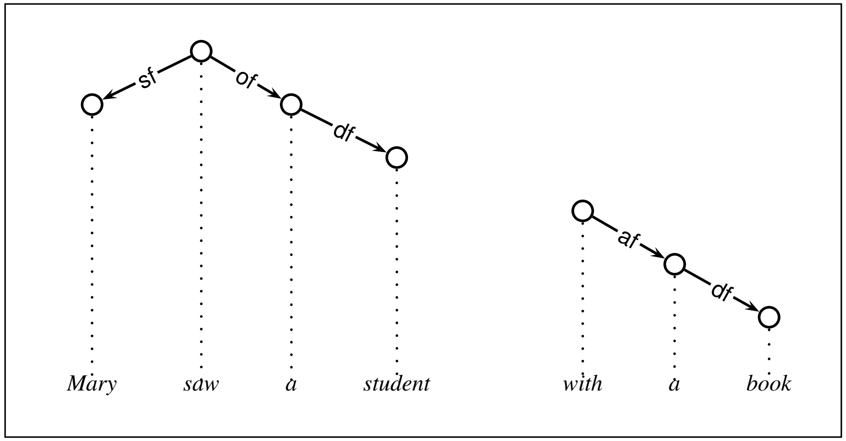


Scope (SC)

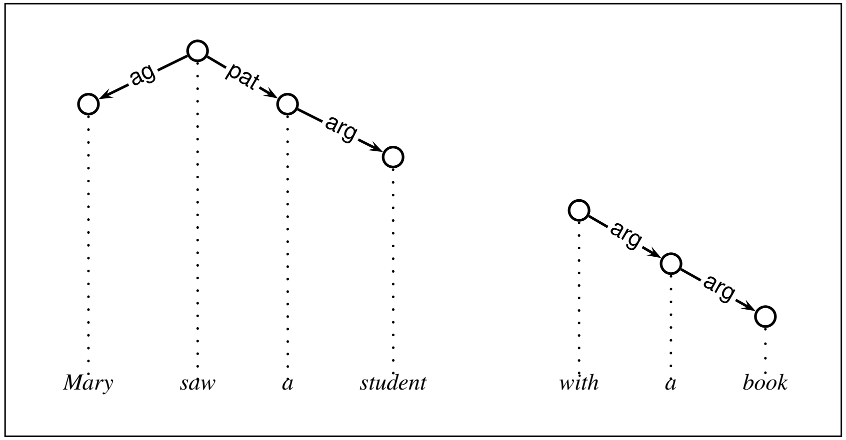
Inferences from syntax to semantics



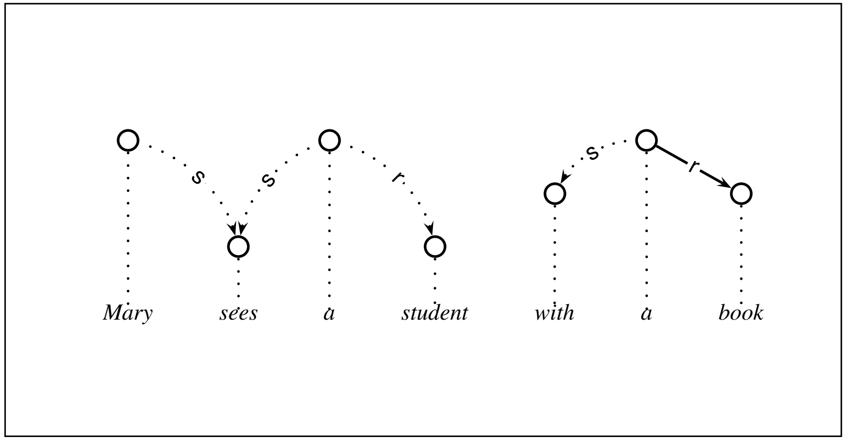
Immediate Dominance (ID)



Linear Precedence (LP)

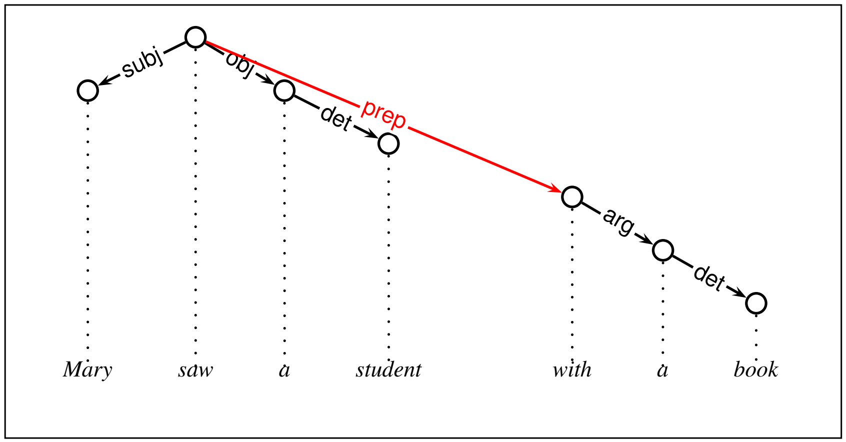


Predicate–Argument (PA)

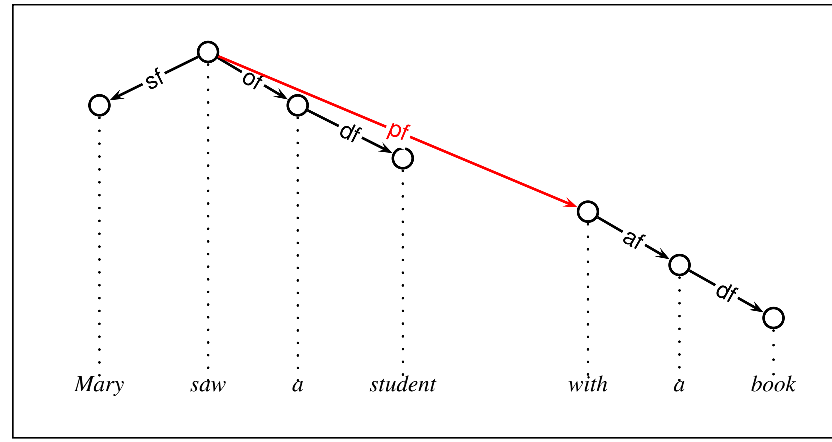


Scope (SC)

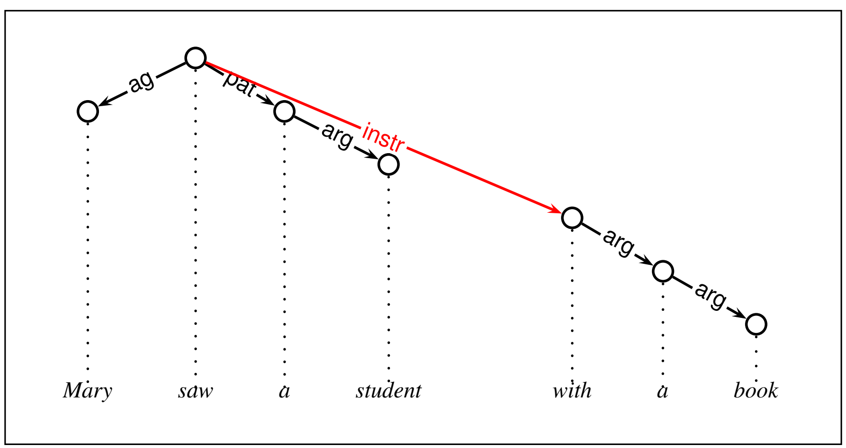
Inferences from syntax to semantics



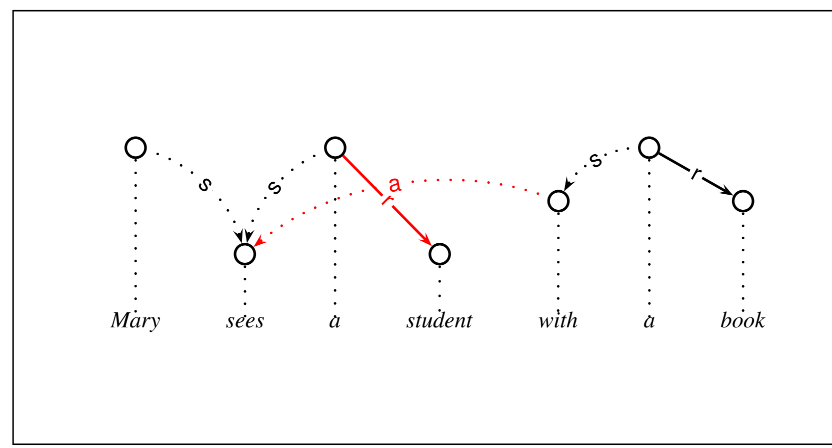
Immediate Dominance (ID)



Linear Precedence (LP)

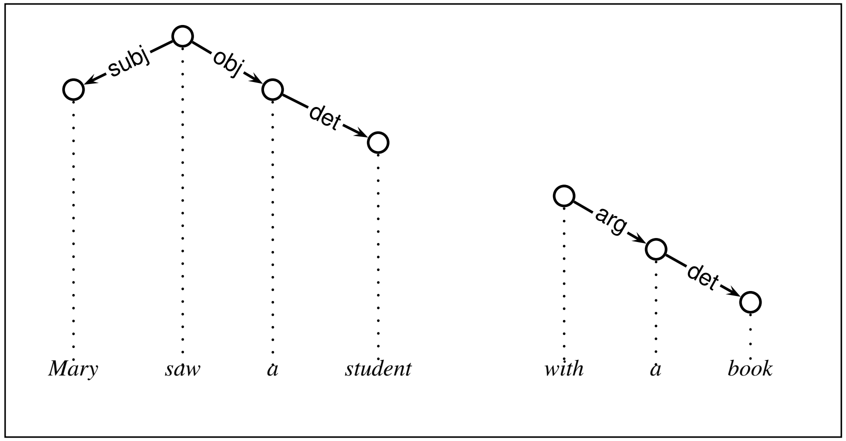


Predicate–Argument (PA)

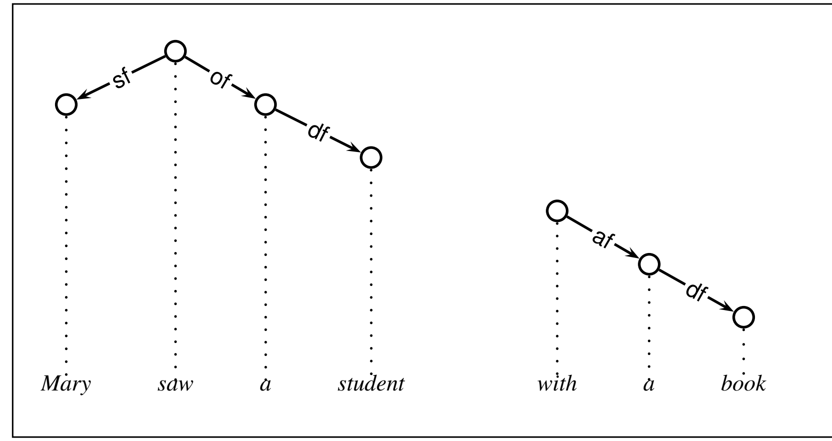


Scope (SC)

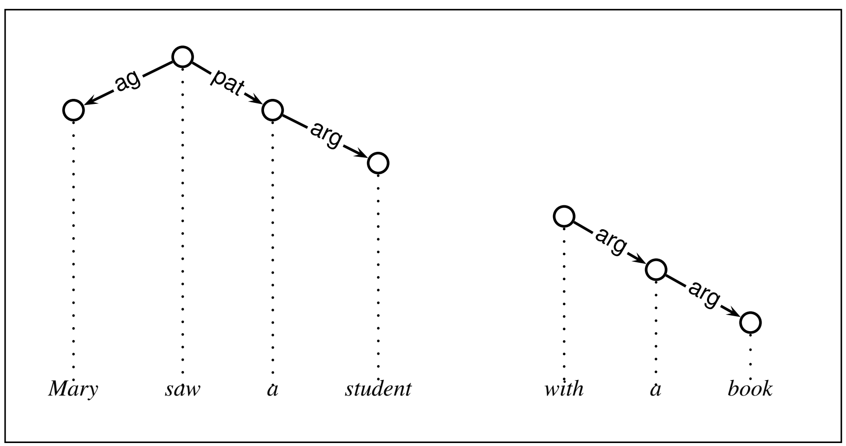
Inferences from semantics to syntax



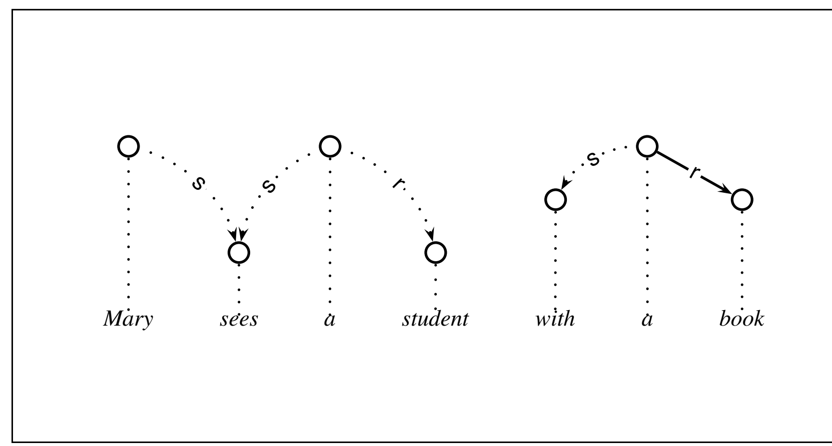
Immediate Dominance (ID)



Linear Precedence (LP)

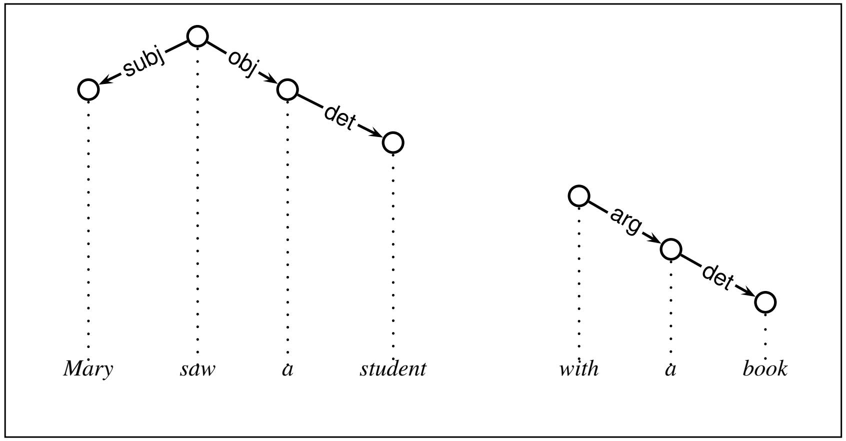


Predicate−Argument (PA)

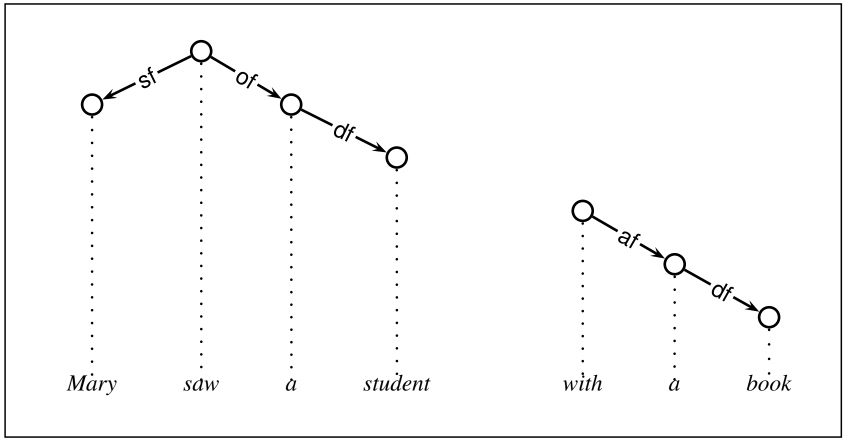


Scope (SC)

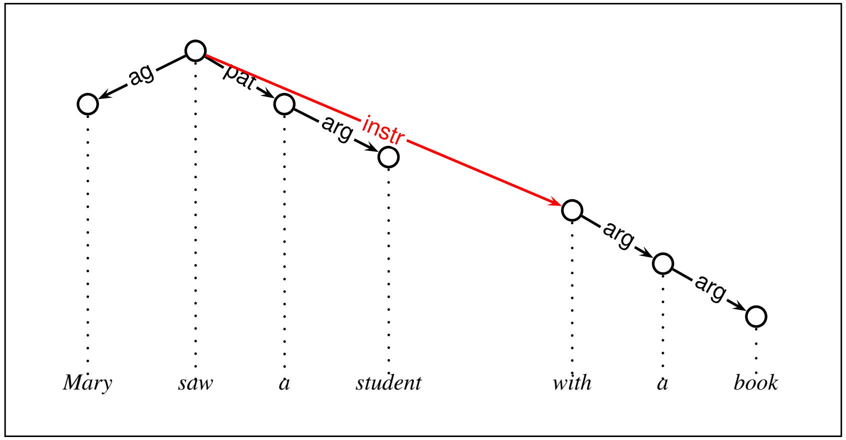
Inferences from semantics to syntax



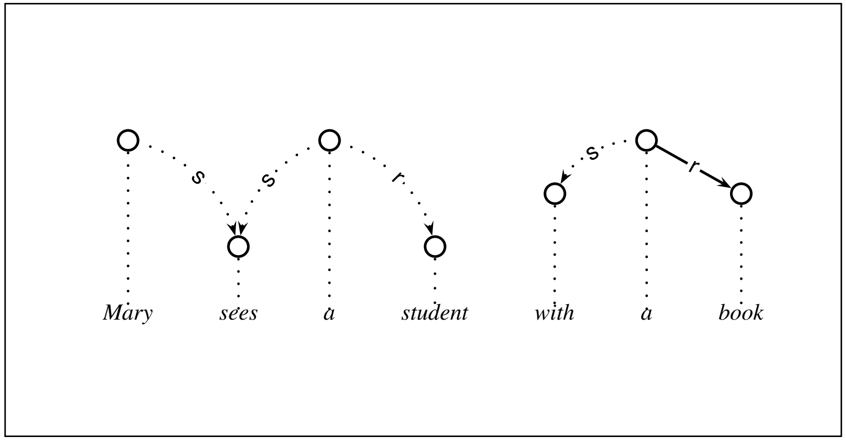
Immediate Dominance (ID)



Linear Precedence (LP)

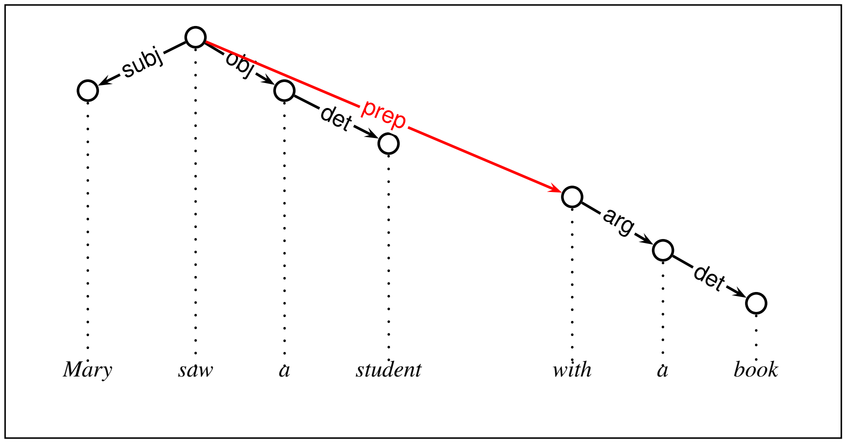


Predicate–Argument (PA)

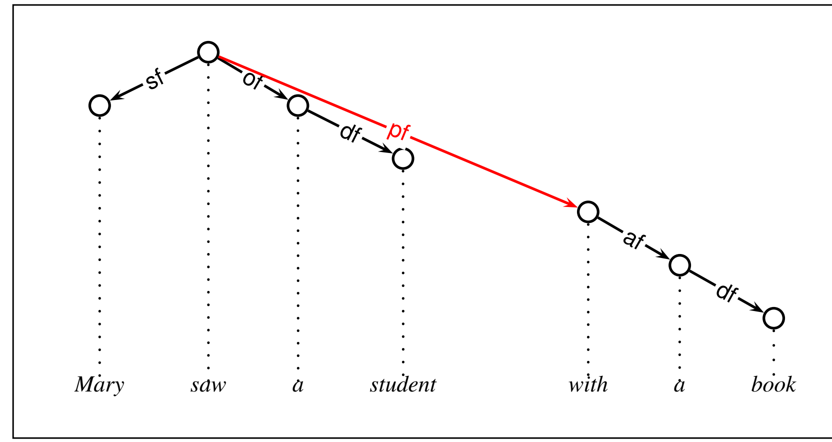


Scope (SC)

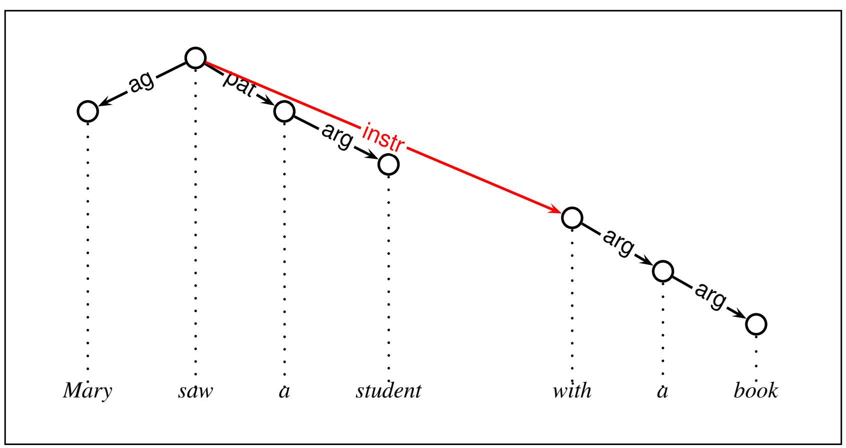
Inferences from semantics to syntax



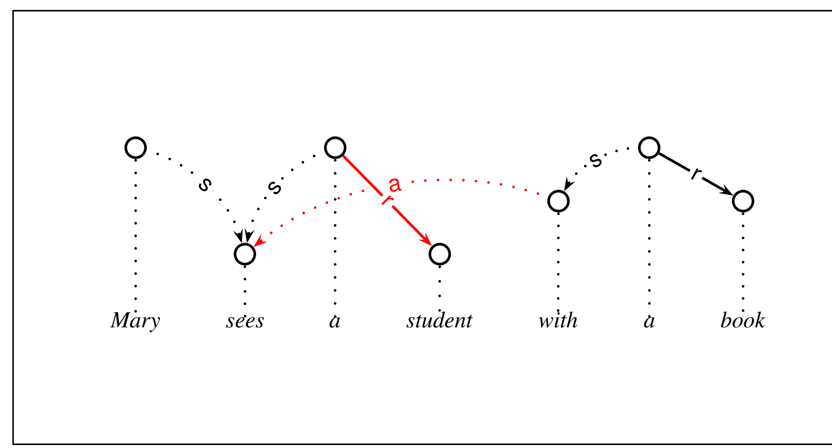
Immediate Dominance (ID)



Linear Precedence (LP)



Predicate–Argument (PA)



Scope (SC)

Summary

- XDG can be used to implement a *relational syntax-semantics interface* that supports the *concurrent flow of information*
- supports *underspecification*
- the dimensions can be linked by *multi-dimensional principles* and *mutually constrain each other*
- no dimension is more “basic” than another, each leads a life on its own

Future Work

- find a *uniform representation formalism for principles*
- *generalization of XDG and CLLS* into a single formalism, working title *Graph Configuration Meta Language* (GCML)
- make *XDG efficient on large grammars*
- *import of large grammars* (e.g. XTAG, ERG)
- *induction of large grammars* (e.g. from Penn TB, PDT)