Lexical-Functional Grammar

Ash Asudeh
Carleton University

University of Iceland
July 3, 2009

Architecture and Structures

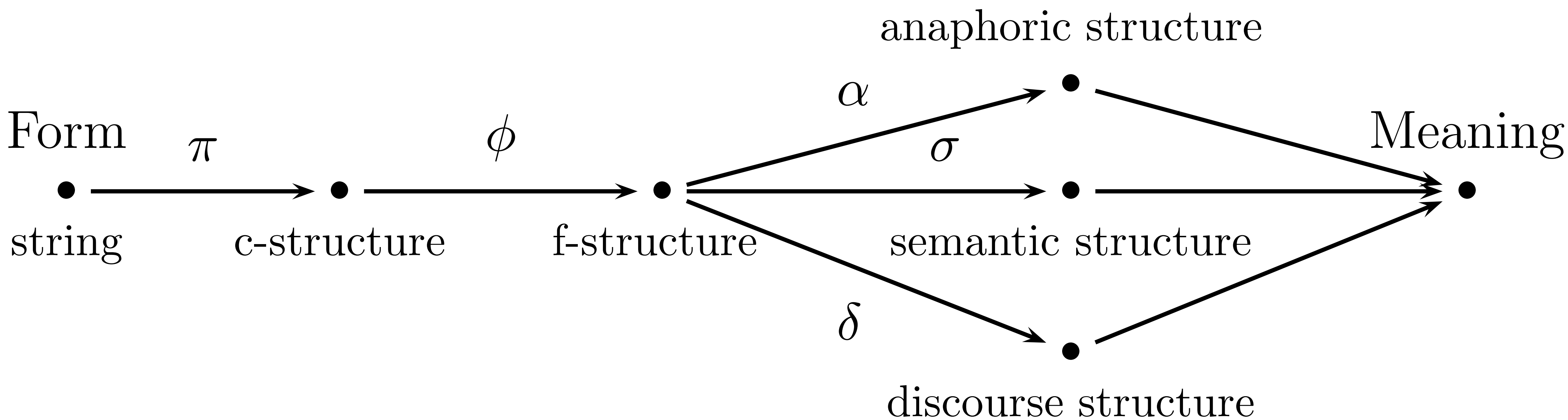
Basic Syntactic Architecture of LFG

- Two basic, **simultaneous** representations of syntax:
  - **C(onstituent)-structure**: constituency, dominance, word order, phrase structure
    Annotated trees
  - **F(unctional)-structure**: abstract grammatical relations/functions (subject, object, etc.), tense, case, agreement, predication, local and non-local dependencies
    Feature structures/attribute-value matrices
- Kaplan & Bresnan (1982):

$$\text{constituent structure} \xrightarrow{\phi} \text{functional structure}$$

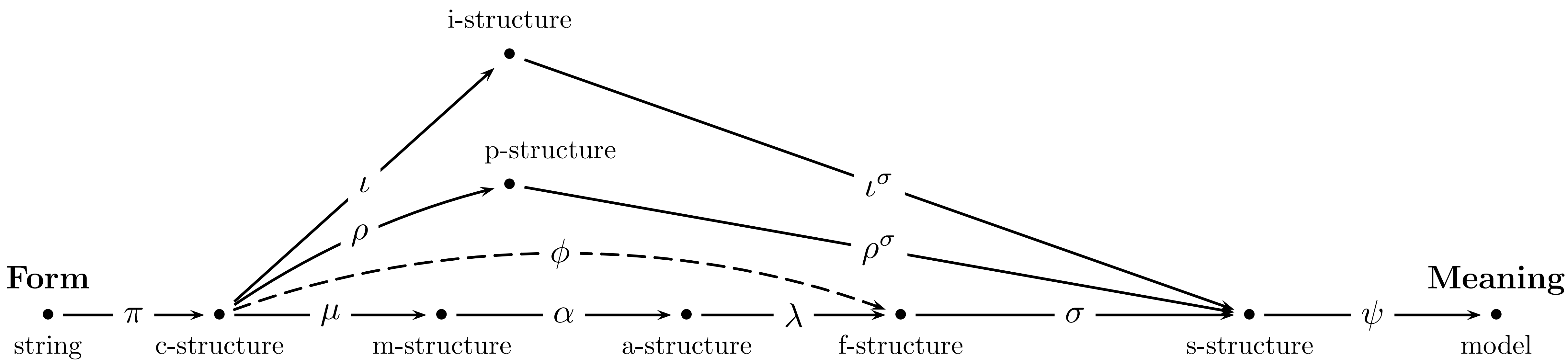
LFG's Parallel Projection Architecture

- Kaplan (1987,1989):



LFG's Parallel Projection Architecture

- Asudeh (2006):



Design Principles

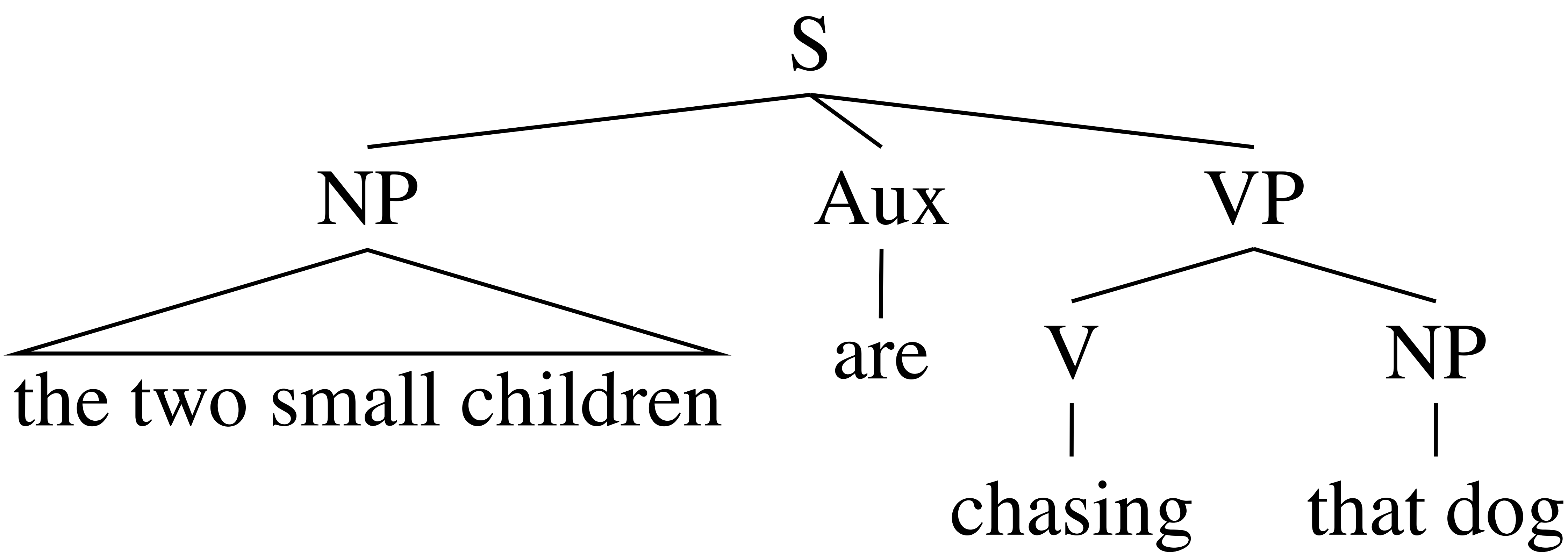
- **Principle I: Variability**
  External structures (modelled by LFG **c-structures**) vary across languages.
- **Principle II: Universality**
  Internal structures (modelled by LFG **f-structures**) are largely invariant across languages.
- **Principle III: Monotonicity**
  The mapping from c-structure to f-structure is not one-to-one, but it is monotonic (information-preserving).

Nonconfigurationality

- Two fundamental ways for language to realize underlying concepts:
  - Phrase structure (**groups**)
  - Morphology (**shapes**)
- Bresnan (1998, 2001): **‘Morphology competes with syntax**’
  - English: phrase structure strategy (**configurational**)
  - Warlpiri: morphological strategy (**nonconfigurational**)

English

- Underlying meaning:
  That of 'the two small children are chasing that dog'



English and Warlpiri

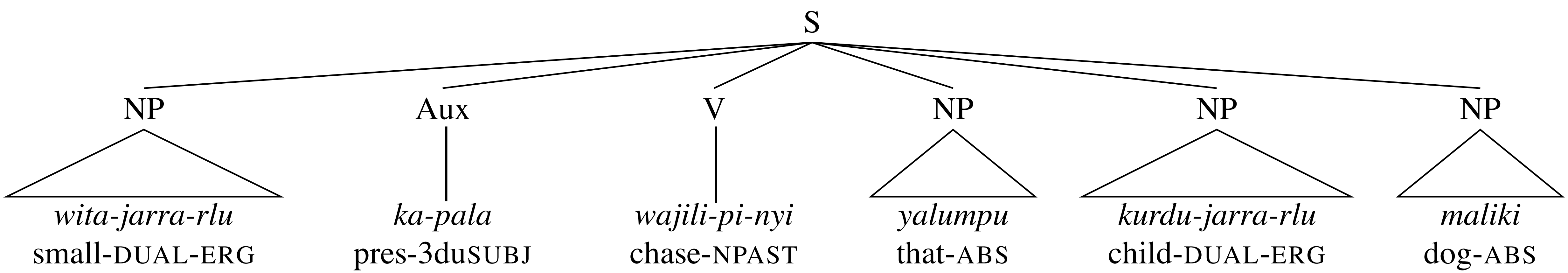
- English:

(1) a. The two small children are chasing that dog.
b. * The two small are chasing that children dog.
c. * The two small are dog chasing children that.
d. * Chasing are the two small that dog children.
e. * That are children chasing the two small dog.

- Warlpiri:
  - All of the permutations in (1) are grammatical ways to express the same underlying concept of ‘the two small children are chasing that dog’
  - Even more permutations than this are possible
  - Only restriction: Aux must be in second position
    (Note: this is a slight simplification)

Warlpiri

- Underlying meaning:
  That of ‘the two small children are chasing that dog’



Abstract Syntax

- Despite the striking structural differences between English and Warlpiri, there are nevertheless common syntactic constraints on the two languages.
  - Example: a subject can bind an object reflexive, but not vice versa

(1) a. Lucy is hitting herself.
b. * Herself is hitting Lucy.

(2) a. Napaljarri-rli ka-nyanu paka-rni
Napaljarri-ERG PRES-REFL hit-NONPAST
'Napaljarri is hitting herself.'

b. * Napaljarri ka-nyanu paka-rni
Napaljarri.ABS PRES-REFL hit-NONPAST
'Herself is hitting Napaljarri.'

➡ How should abstract grammatical relations be captured?
Transformational Grammar: configurationally, using a uniform syntactic representation

LFG: non-configurationally, using a separate syntactic representation

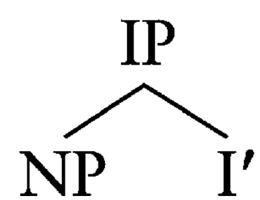
C-structure

- Language variation in phrasal expression:
  - Basic word order:
    - SVO (English), SOV (Japanese), VSO (Irish), VOS (Malagasy)
  - Constituency:
    - Grouping of verb and complements,
    - Grouping of noun and modifiers
  - Strict vs. free word order:
    - configurational languages vs. case-marking languages

Constraints on C-structures: Phrase Structure Rules

- LFG distinguishes between the objects in the model and descriptions of those objects (i.e. **constraints** on the objects).
- C-structure trees are **constrained** by phrase structure rules.

$$\text{IP} \rightarrow \text{NP I}'$$



- Right-hand side of LFG phrase structure rules are **regular expressions**:

  ➡disjunction, optionality, arbitrary repetition (Kleene plus [+] and star [*])

$$\text{V}' \rightarrow (\text{V})\ (\text{NP})\ \text{PP}^*$$

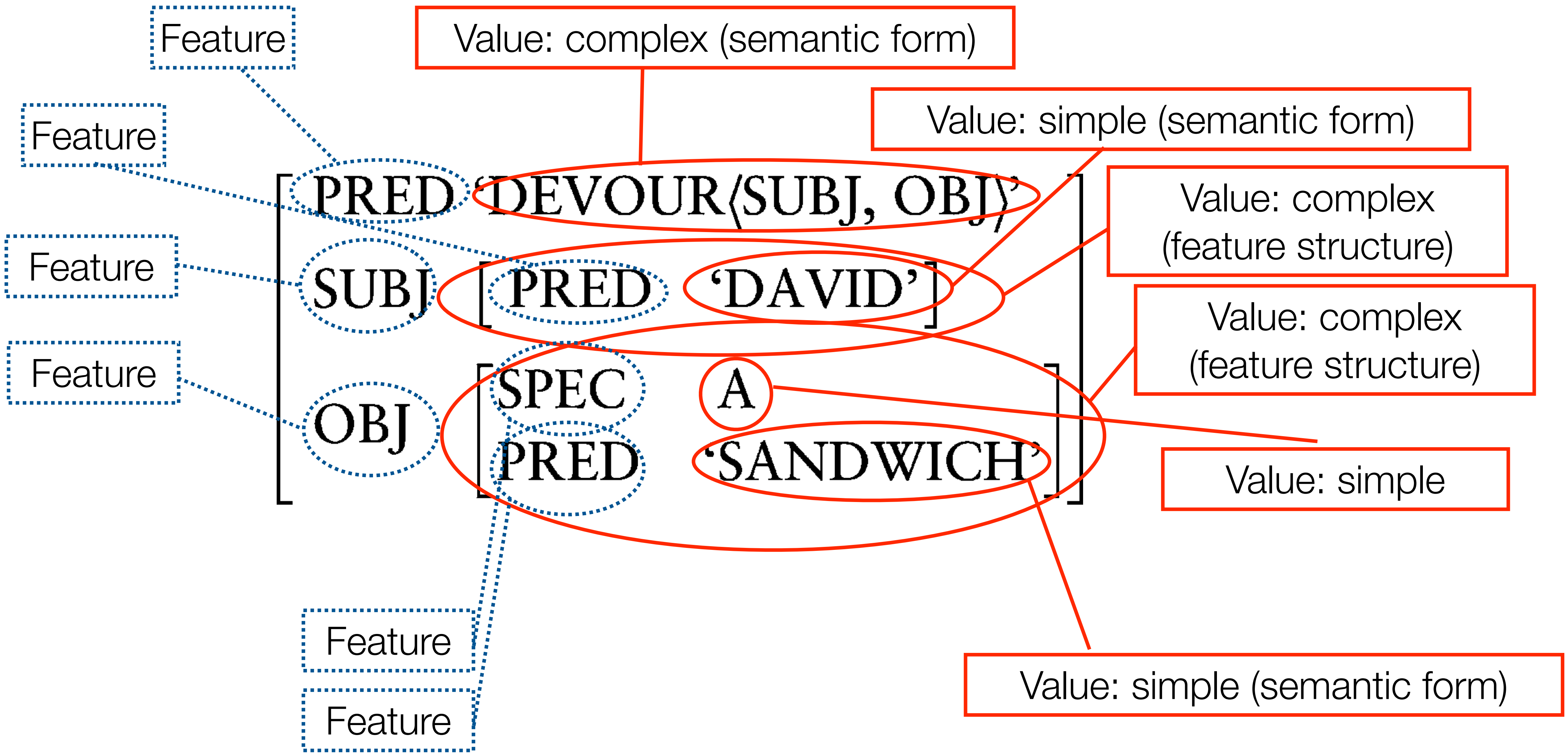
F-structures

- F-structures represent abstract **grammatical functions** (subject, object, etc.), **grammatical features** (tense, case, person, number, etc.), and **grammatical dependencies** (raising, control, unbounded dependencies)

(1) David devoured a sandwich.

$$\begin{bmatrix} \text{PRED} & \text{'DEVOUR}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'DAVID'} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{PRED} & \text{'SANDWICH'} \end{bmatrix} \end{bmatrix}$$

Anatomy of an F-structure



General Constraints on F-structures: Completeness, Coherence, Uniqueness

- **Completeness**:
  All the grammatical functions subcategorized by a predicate must be present in the f-structure.

  (1)* David devoured.

  Devour <SUBJ, OBJ>

- **Coherence**:
  Only the grammatical functions subcategorized by a predicate may be present in the f-structure.

  (2)* David devoured a sandwich that it was raining.

- **Uniqueness**:
  No attribute may have more than one value.

Uniqueness and Semantic Forms

- Semantic forms (values of PRED features) are **unique**.

$$
\begin{bmatrix}
\text{PRED} & \text{‘DEVOUR}_{37}\langle\text{SUBJ,OBJ}\rangle\text{’} \\
\text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘DAVID}_{42}\text{’} \end{bmatrix} \\
\text{OBJ} & \begin{bmatrix} \text{SPEC} & \text{A} \\ \text{PRED} & \text{‘SANDWICH}_{14}\text{’} \end{bmatrix}
\end{bmatrix}
$$

➡Multiple instances of semantic forms **cannot unify**, even if the semantic forms are otherwise compatible.

(1) * David devoured a sandwich a sandwich.

Features and the Lexicon in LFG

Lexical Entries in LFG

yawns V

$(\uparrow$ PRED$)=$'yawn$\langle$SUBJ$\rangle$'
$(\uparrow$ VFORM$)=$FINITE
$(\uparrow$ TENSE$)=$PRES
$(\uparrow$ SUBJ PERS$)=3$
$(\uparrow$ SUBJ NUM$)=$SG

F(unctional)-description, made up of functional schemata

Two Main Kinds of F-structure Constraints: Defining Equations and Constraining Equations

- Functional schemata and functional descriptions are often referred to as equations. This is a little inaccurate, because equality is not always the relevant relation, but it is certainly the most common way of specifying constraints on f-structures in LFG. So the term has stuck.

- There are two main classes of f-structure constraints in LFG:

  1. Defining Equations
     These equations define the f-structure by specifying which features have which values. They 'make it so'. Defining equations are stated with a simple equality (or other relation symbol).

$$(f \text{ SUBJ NUM}) = \text{SG}$$

Two Main Kinds of F-structure Constraints: Defining Equations and Constraining Equations

- There are two main classes of f-structure constraints in LFG:

  2. Constraining Equations
  These equations further constrain the f-structure once it has been constructed.

  In other words:
  1. Satisfy defining equations, setting aside constraining equations, to get minimal model.
  2. Satisfy constraining equations.

  There are a number of different kinds of constraining equations, but the ones that check feature-value pairs are written with a subscript c on the equality like this:

$$(f \text{ SUBJ NUM}) =_c \text{SG}$$

Other Kinds of Constraining Equations

Negative equation: $(f \text{ TENSE}) \neq \text{PRESENT}$

Existential constraint: $(f \text{ TENSE})$

Negative existential constraint: $\neg(f \text{ TENSE})$

Optionality, Disjunction, Conjunction, Negation

sneeze $(f\ \text{PRED}) = \text{‘SNEEZE}\langle\text{SUBJ}\rangle\text{’}$
$\{(f\ \text{VFORM}) = \text{BASE}\ |$
$(f\ \text{TENSE}) = \text{PRES}$
$\neg\{(f\ \text{SUBJ PERS}) = 3$
$(f\ \text{SUBJ NUM}) = \text{SG}\}\}$

Conjunction (implicit)

Negation ¬ A or ¬{ ... }

Disjunction { A | B }

- The lexical entry for ‘sneeze’ (from Dalrymple 2001:87) says the following: The PRED of ‘sneeze’ is ‘SNEEZE<SUBJ>’. Also (conjunction): **Either** (disjunction) the VFORM is BASE (i.e. it’s a non-finite form) **or** it has present tense and **it is not the case that** (negation) its subject has third person singular agreement features (cf. *She sneeze*.)

$((f\ \text{SUBJ PRED}) = \text{‘PRO’})$

Optionality (A)

Hint: ‘pro-drop’ in LFG!

Outside-In and Inside-Out equations

- Outside-in equations with respect to an f-structure *f* make specifications about paths leading **in from** *f*:

$$(\uparrow \text{COMP TENSE}) = \text{PRESENT}$$

- Inside-out equations with respect to an f-structure *f* make specifications about paths leading **out from** *f*:

$$(\text{COMP} \uparrow)$$

- The two kinds of equation can be combined:

$$((\text{COMP} \uparrow) \text{ TENSE}) = \text{PRESENT}$$

Outside-In and Inside-Out equations

- Outside-in equations with respect to an f-structure *f* make specifications about paths leading **in from** *f*:

$$(f \text{ COMP TENSE}) = \text{PRESENT}$$

- Inside-out equations with respect to an f-structure *f* make specifications about paths leading **out from** *f*:

$$(\text{COMP } f)$$

- The two kinds of equation can be combined:

$$((\text{COMP } f) \text{ TENSE}) = \text{PRESENT}$$

Functional Uncertainty

- Simple or limited functional uncertainty can be expressed by defining abbreviatory symbols disjunctively:

$$\text{GF} = \{ \text{SUBJ} \mid \text{OBJ} \mid \text{OBJ}_\theta \mid \text{OBL} \mid \text{COMP} \mid \text{XCOMP} \mid \text{ADJ} \mid \text{XADJ} \}$$

- Unlimited functional uncertainty can be expressed with Kleene star (*) or Kleene plus (+), where X* means ‘0 or more X’ and X+ means ‘1 or more X’:

$$(\uparrow \text{FOCUS}) = (\uparrow \{\text{XCOMP} \mid \text{COMP}\}^* \text{ GF})$$

$$(\uparrow \text{INDEX}) = ((\text{GF}^+ \uparrow) \text{ SUBJ INDEX})$$

- Note that f-descriptions are therefore written in a regular language, as is also the case for the right-hand side of c-structure rules.

Functional Descriptions and Subsumption

- F-descriptions are true of not just the smallest, 'intuitively intended' f-structure, but also any larger f-structure that contains the same information.*

  * This relationship is called **subsumption**:
  In general, a structure A subsumes a structure B if and only if A and B are identical or B contains A and additional information not included in A.

$$f\begin{bmatrix} \text{PRED} & \text{`GO}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix} \qquad g\begin{bmatrix} \text{PRED} & \text{`GO}\langle\text{SUBJ}\rangle\text{'} \\ \text{TENSE} & \text{FUTURE} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`PRO'} \\ \text{CASE} & \text{NOM} \\ \text{NUM} & \text{SG} \end{bmatrix} \end{bmatrix}$$

f subsumes *g*

- An f-description is therefore true of not just the **minimal** f-structure that satisfies the description: the f-description is also true of the infinitely many other f-structures that the intended, minimal f-structure subsumes.

Minimization

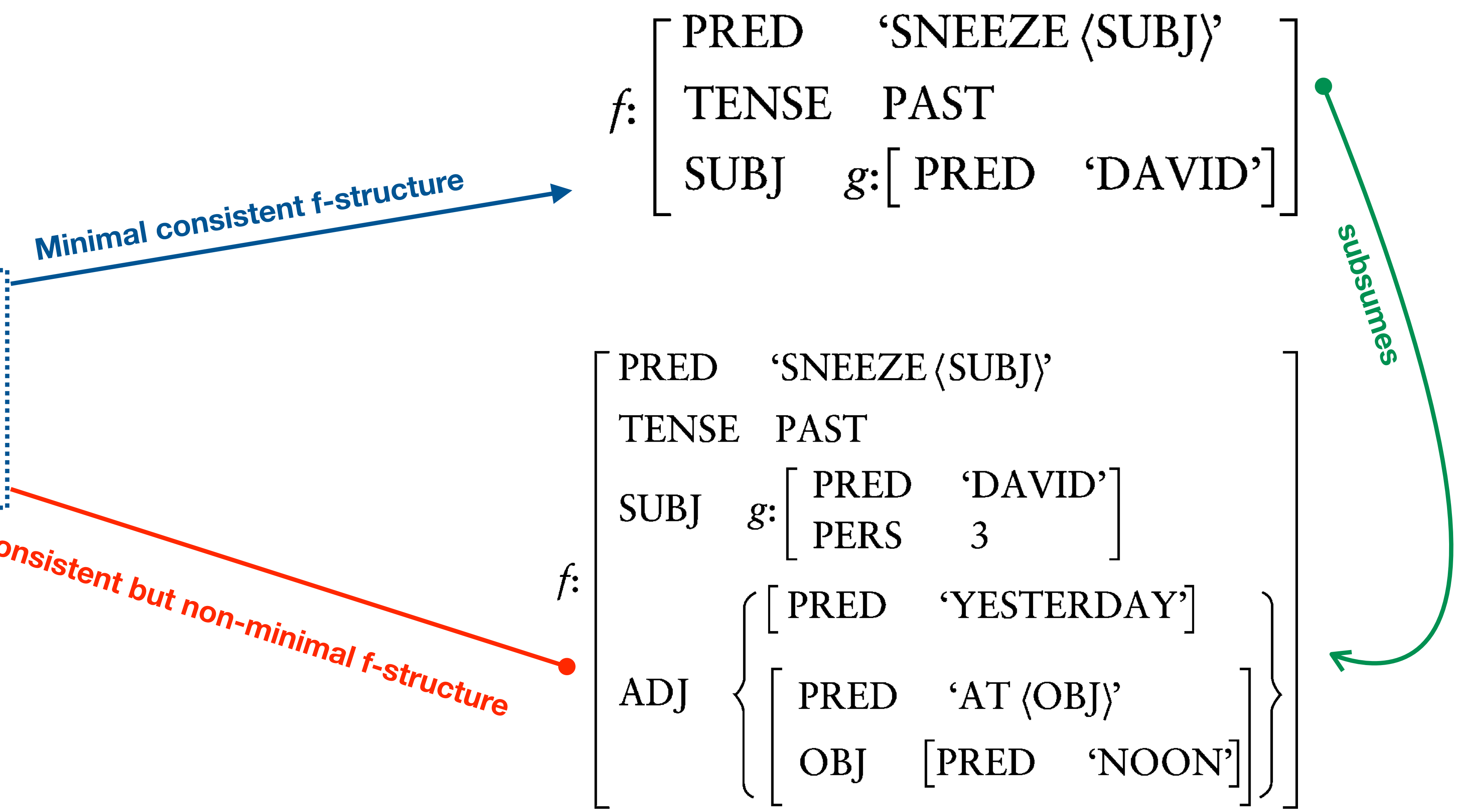
- There is a general requirement on LFG's solution algorithm that it yield the **minimal** solution: no features that are not mentioned in the f-description may be included.

- Let's look at an example from Dalrymple (2001).

  (1) David sneezed.

- F-description:

$$
\begin{aligned}
&(f\ \text{PRED}) = \text{`SNEEZE}\langle\text{SUBJ}\rangle\text{'}\\
&(f\ \text{TENSE}) = \text{PAST}\\
&(f\ \text{SUBJ}) = g\\
&(g\ \text{PRED}) = \text{`DAVID'}
\end{aligned}
$$



Lexical Generalizations in LFG

yawns V

$(\uparrow \text{PRED}) = \text{`yawn}\langle \text{SUBJ} \rangle\text{'}$

$(\uparrow \text{VFORM}) = \text{FINITE}$

$(\uparrow \text{TENSE}) = \text{PRES}$

$(\uparrow \text{SUBJ PERS}) = 3$

$(\uparrow \text{SUBJ NUM}) = \text{SG}$

A lot of this f-description is shared by other verbs.

LFG Templates: Relations between Descriptions

yawns
$(\uparrow \text{PRED}) = \text{`yawn}\langle\text{SUBJ}\rangle\text{'}$
$(\uparrow \text{VFORM}) = \text{FINITE}$
$(\uparrow \text{TENSE}) = \text{PRES}$
$(\uparrow \text{SUBJ PERS}) = 3$
$(\uparrow \text{SUBJ NUM}) = \text{SG}$

$$\text{PRESENT} = \begin{array}{l}(\uparrow \text{VFORM}) = \text{FINITE}\\ (\uparrow \text{TENSE}) = \text{PRES}\end{array}$$

$$3\text{SG} = \begin{array}{l}(\uparrow \text{SUBJ PERS}) = 3\\ (\uparrow \text{SUBJ NUM}) = \text{SG}\end{array}$$

⇩

yawns
$(\uparrow \text{PRED}) = \text{`yawn}\langle\text{SUBJ}\rangle\text{'}$
@PRESENT
@3SG

Templates: Factorization and Hierarchies

FINITE = ($\uparrow$ VFORM)=FINITE

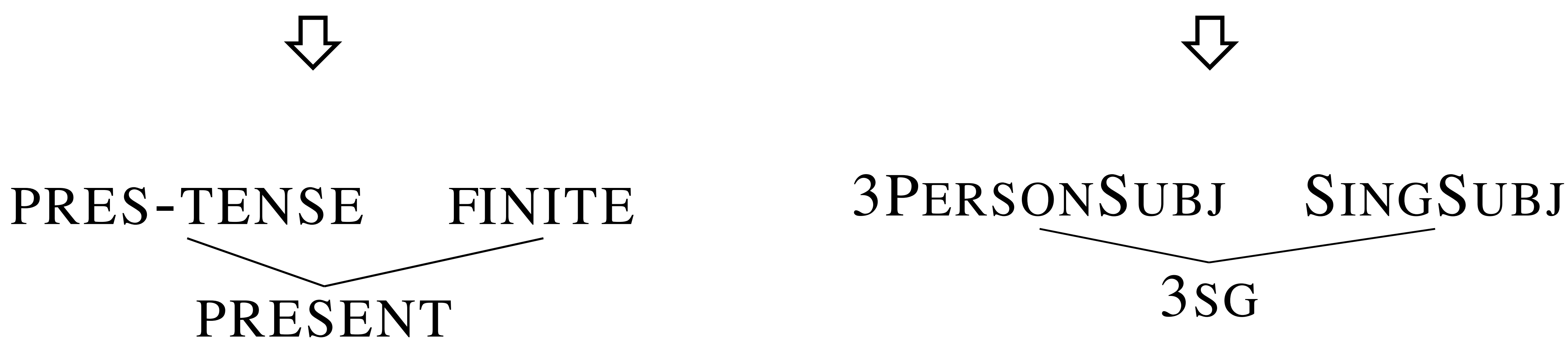
PRES-TENSE = ($\uparrow$ TENSE)=PRES

PRESENT = @FINITE
@PRES-TENSE

3PERSONSUBJ = ($\uparrow$ SUBJ PERS)=3

SINGSUBJ = ($\uparrow$ SUBJ NUM)=SG

3SG = @3PERSONSUBJ
@SINGSUBJ



Templates: Factorization and Hierarchies

yawns	(↑ PRED)='yawn⟨SUBJ⟩'	PRES3SG	=	@PRESENT
	@PRESENT			@3SG
	@3SG			

⇩

yawns	(↑ PRED)='yawn⟨SUBJ⟩'
	@PRES3SG

PRES-TENSE   FINITE   3PERSONSUBJ   SINGSUBJ

PRESENT   3SG

PRES3SG

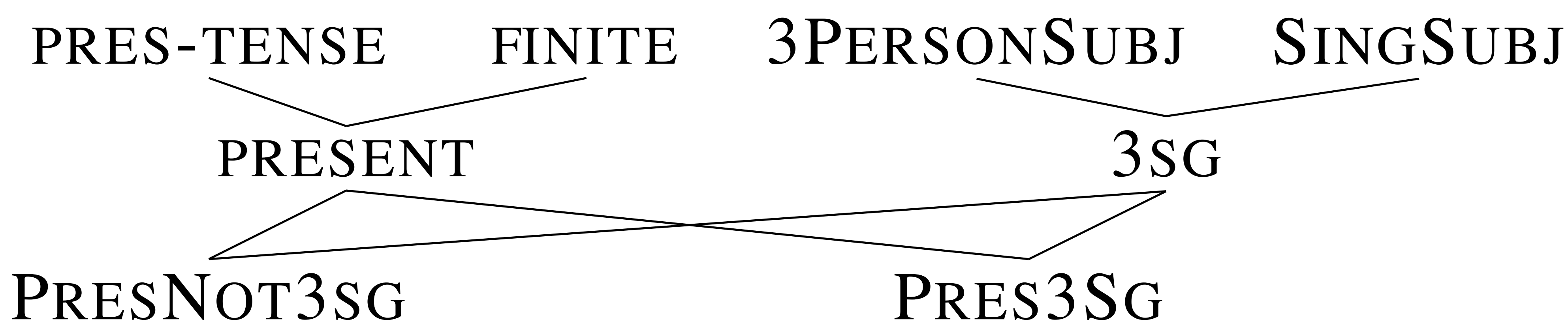
Templates: Boolean Operators

PresNot3Sg = @present
¬@3sg

⇨

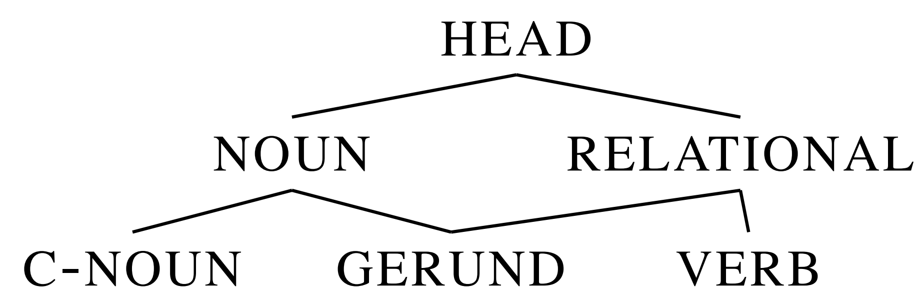
(↑ VFORM)=FINITE
(↑ TENSE)=PRES
¬{(↑ SUBJ PERS)=3
(↑ SUBJ NUM)=SG}

Negation



Hierarchies: Templates vs. Types

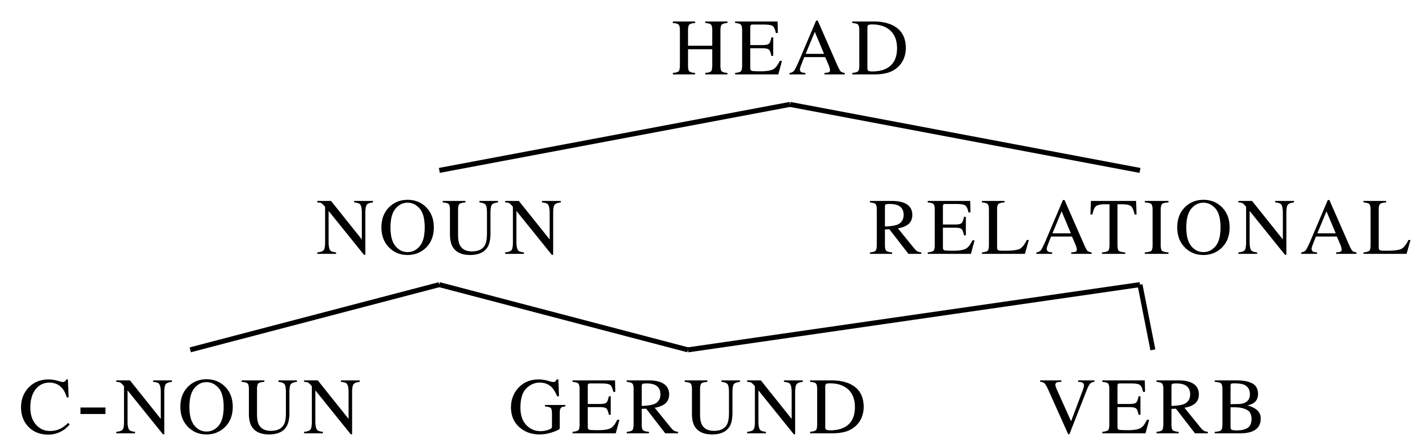
- Type hierarchies are *and/or* lattices:
  - Motherhood: ***or***
  - Multiple Dominance: ***and***



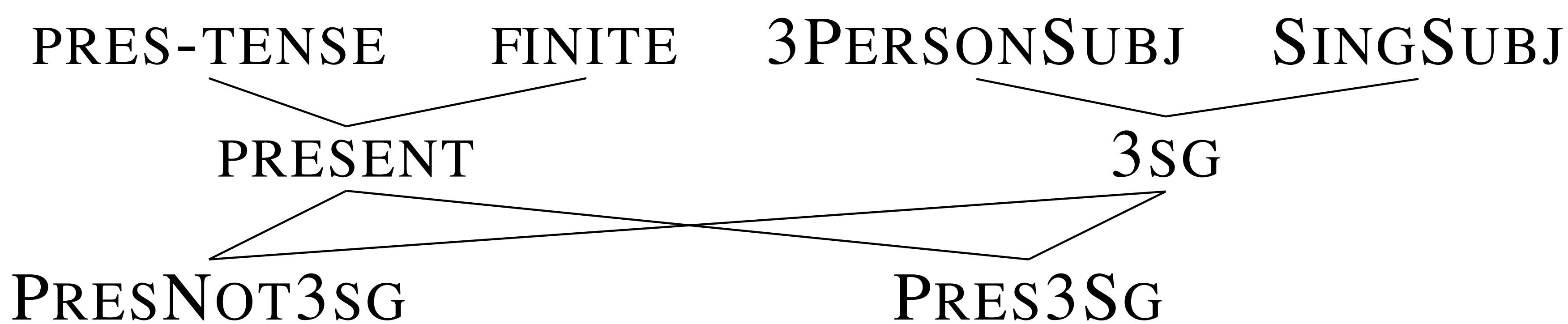
- Type hierarchies encode inclusion/inheritance and place constraints on how the inheritance is interpreted.
- LFG template hierarchies encode only inclusion: multiple dominance not interpreted as conjunction, no real status for motherhood.
- LFG hierarchies relate descriptions only: mode of combination (logical operators) is determined contextually at invocation or is built into the template.
- HPSG hierarchies relate first-class ontological objects of the theory.
- LFG hierarchies are abbreviatory only and have no real ontological status.

Hierarchies: Templates vs. Types

HPSG



LFG



Parameterized Templates

yawns $(\uparrow \text{ PRED})=\text{`yawn}\langle\text{SUBJ}\rangle\text{'}$
@PRES3SG

INTRANSITIVE(P) = $(\uparrow \text{ PRED})=\text{`P}\langle\text{SUBJ}\rangle\text{'}$

⇩

yawns @INTRANSITIVE(yawn)
@PRES3SG

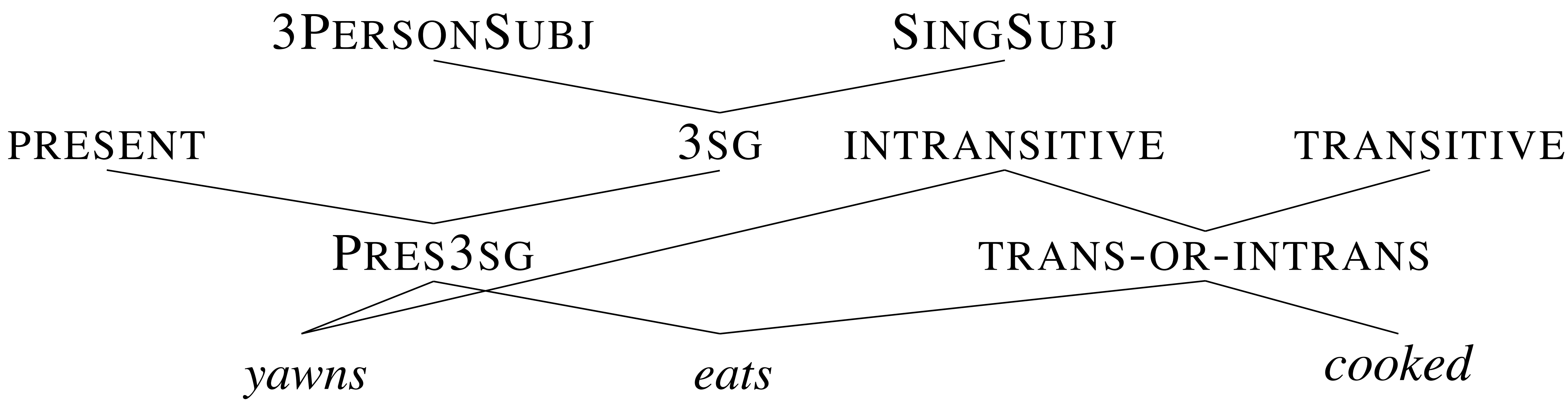
Parameterized Templates

$$\text{TRANSITIVE(P)} \quad = \quad (\uparrow \text{ PRED})=\text{`P}\langle\text{SUBJ, OBJ}\rangle\text{'}$$

$$\text{TRANS-OR-INTRANS(P)} \quad = \quad @\text{TRANSITIVE(P)} \vee @\text{INTRANSITIVE(P)}$$

$$(\uparrow \text{ PRED})=\text{`}\mathit{eat}\langle\text{SUBJ, OBJ}\rangle\text{'} \vee (\uparrow \text{ PRED})=\text{`}\mathit{eat}\langle\text{SUBJ}\rangle\text{'}$$

Temple Hierarchy with Lexical Leaves



Defaults in LFG

$(\uparrow \text{CASE}) \vee (\uparrow \text{CASE}){=}\text{NOM}$

The f-structure must have case and if nothing else provides its case, then its case is nominative.

$\text{DEFAULT}(D\ V) \quad = \quad D \vee D{=}V$

Paramerized template for defaults.

Also illustrates that parameterized templates can have multiple arguments

⇩

$@\text{DEFAULT}((\uparrow \text{CASE})\ \text{NOM})$

C-structure Annotation of Templates

$$\begin{array}{lll} VP \longrightarrow & V & ADVP^* \\ & \uparrow=\downarrow & \downarrow \in (\uparrow \text{ADJUNCT}) \\ & & (\downarrow \text{ADJUNCT-TYPE})=\text{VP-ADJ} \end{array}$$

$$\begin{array}{lll} \text{ADJUNCT}(\text{P}) & = & \downarrow \in (\uparrow \text{ADJUNCT}) \\ & & @\text{ADJUNCT-TYPE}(\text{P}) \end{array}$$

$$\text{ADJUNCT-TYPE}(\text{P}) \quad = \quad (\downarrow \text{ADJUNCT-TYPE})=\text{P}$$

⇩

$$\begin{array}{lll} VP \longrightarrow & V & ADVP^* \\ & \uparrow=\downarrow & @\text{ADJUNCT}(\text{VP-ADJ}) \end{array}$$

Features in the Minimalist Program

Features and Explanation

- The sorts of features that are associated with functional heads in the Minimalist Program are well-motivated morphosyntactically, although other theories may not draw the conclusion that this merits phrase structural representation (cf. Blevins 2008).
- Care must be taken to avoid circular reasoning in feature theory:
  - The ‘strong’ meta-feature: “This thing has whatever property makes things displace, as evidenced by its displacement.”
  - The ‘weak’ meta-feature: “This thing lacks whatever property makes things displace, as evidenced by its lack of displacement.”
  - The EPP feature: “This thing has whatever property makes things move to subject position, as evidenced by its occupying subject position.”

Features and Simplicity

- Adger (2003, 2008) considers three kinds of basic features:
  - Privative, e.g. [singular]
  - Binary, e.g. [singular +]
  - Valued, e.g. [number   singular]
- Adger considers the privative kind the simplest in its own right.
- This may be true, but only if it does not introduce complexity elsewhere in the system (Culicover & Jackendoff 2005: ‘honest accounting’).
- Notice that only the final type of feature treats number features as any kind of natural class within the theory (as opposed to meta-theoretically).

Kinds of Feature-Value Combinations

- Adger (2003):
  - Privative
    - [singular], [V], ...
  - Binary
    - [singular: +] (?)
  - Attribute-value
    - [Tense: past]

Interpreted vs. Uninterpreted Features

- Interpreted features:
  - [F]
- Uninterpreted features:
  - [*u*F]
- All uninterpreted features must be eliminated (‘checked’).
- Interpreted features are interpreted by the semantics.
  - Presupposes an interpretive (non-combinatorial) semantics.

[Notation from Adger 2003]

Feature Strength

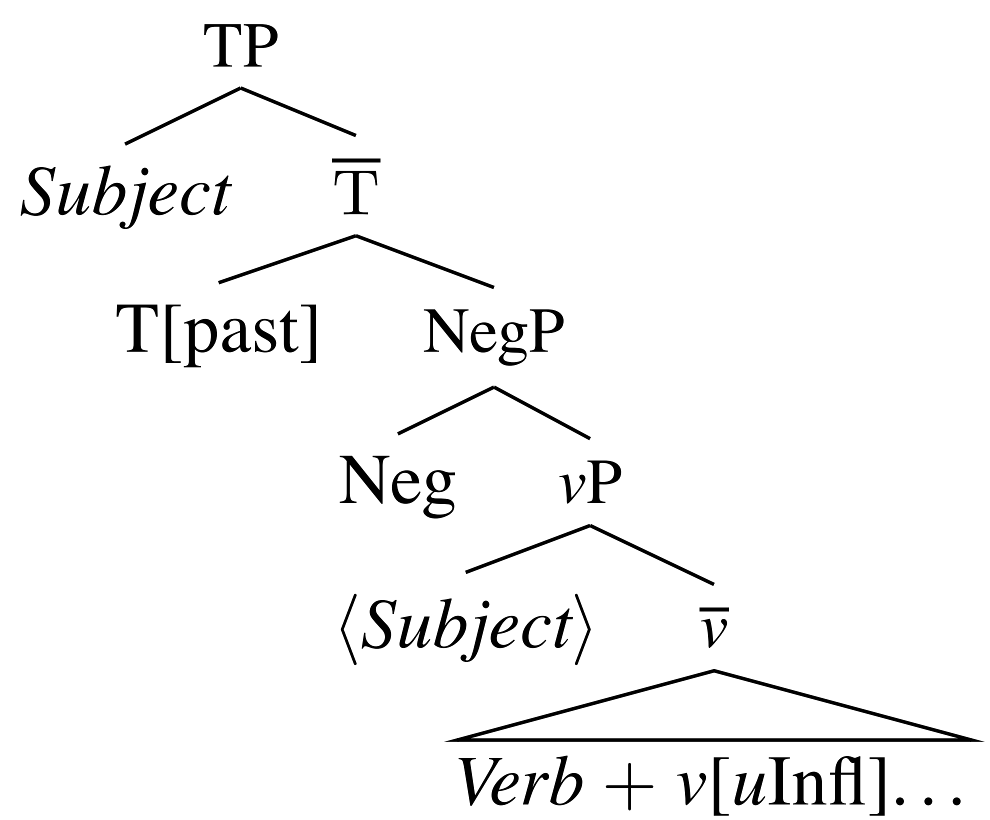
- Strong features must be checked locally:
  Trigger Move/Internal Merge/Remerge
  - [F*]
- Weak features do not have to be checked locally:
  Do not trigger Move
  - [F]

[Notation from Adger 2003]

An Example: Auxiliaries

- Adger (2003:181)

  "When [*u*Infl:] on Aux is valued by T, the value is strong; when [*u*Infl:] on *v* is valued by T, the value is weak."



Locality of Feature Matching

- Adger (2003:218)

  Locality of Matching
  Agree holds between a feature F on X and a matching feature F on Y if and only if there is no intervening Z[F].

  Intervention
  In a structure [X ... Z ... Y], Z intervenes between X and Y iff X c-commands Z and Z c-commands Y.

Feature-Value Unrestrictiveness & Free Valuation

- Asudeh & Toivonen (2006) argue that the Minimalist feature system of Adger (2003) has two undesirable properties.

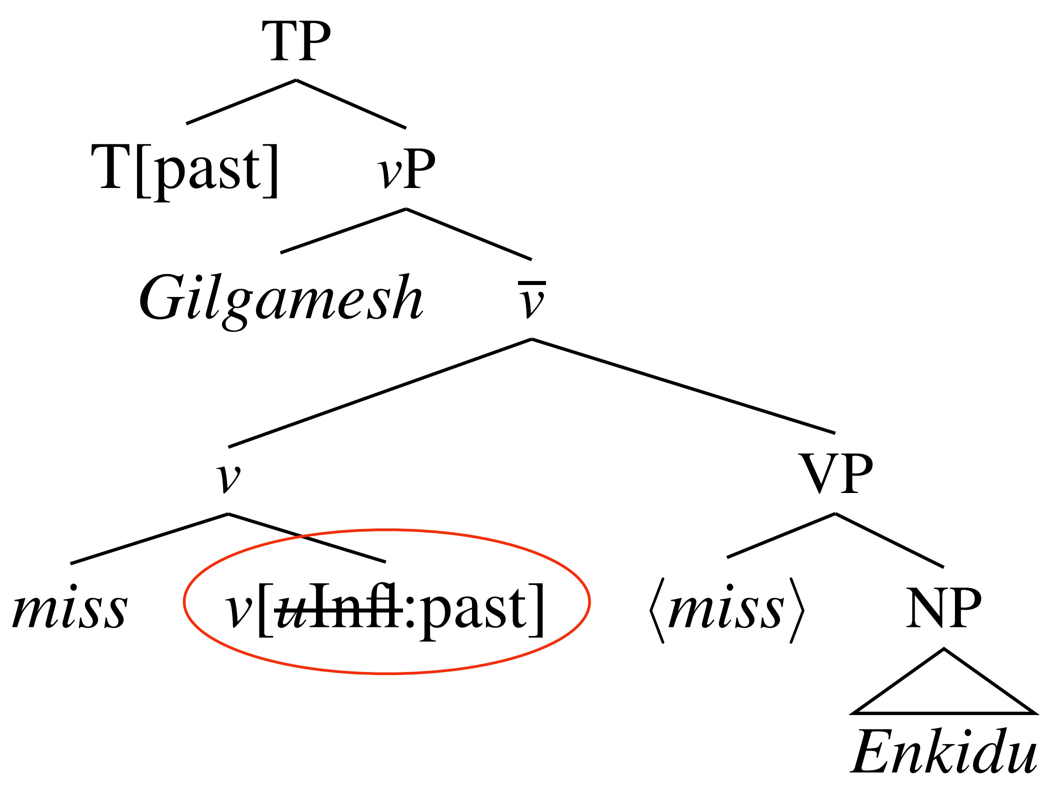
  Feature-value unrestrictiveness
  Feature valuation is unrestricted with respect to what values a valued feature may receive.

  Free valuation
  Feature valuation appears freely, subject to locality conditions.

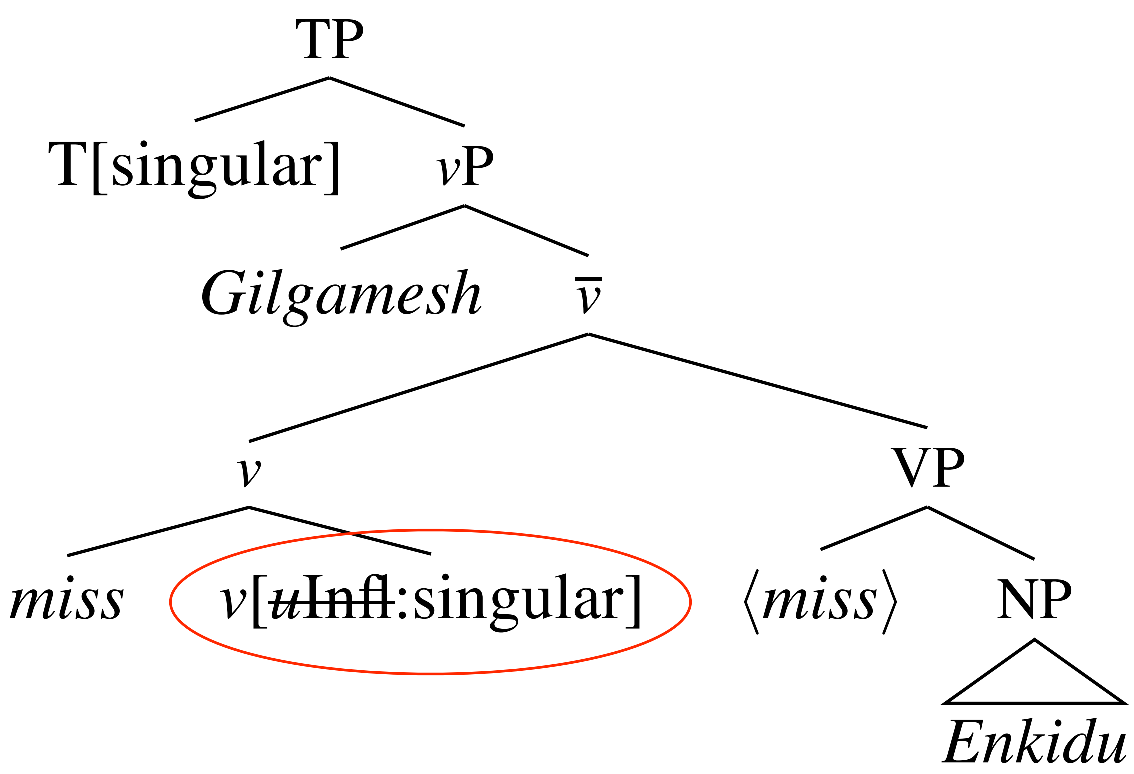
- This results in a very unconstrained theory of features.
- This may sound good, because it's less stipulative and hence more Minimal, but from a theory perspective it is bad: unconstrained theories are less predictive.

Example: English Subject Agreement

(1) Gilgamesh missed Enkidu



(2) Gilgamesh misses Enkidu



- Contrast with HPSG: MP has no typing of values (feature value unrestrictiveness)
- Contrast with LFG: MP has valuation without specification (free valuation)

Two Contrasting Feature Theories

- HPSG (Pollard & Sag 1994): features are not just valued, the values are also *typed*
  - If two values can unify, they must be in a typing relation (one must be a subtype of the other).
  - Feature values in HPSG are thus tightly restricted by types.
- LFG (Kaplan & Bresnan 1982, Bresnan 2001): features are not restricted, but there is no free valuation
  - A feature cannot end up with a given value unless there is an explicit equation in the system.

Feature Simplicity and Constraint Types

- LFG offers the opportunity to consider Adger's three feature types in light of a single feature type, with varying constraint types.
  - LFG features are valued (*f* is an LFG f(unctional)-structure):

$$f\begin{bmatrix}\text{NUMBER} & \text{singular}\end{bmatrix}$$

  - Types of LFG feature constraints.
    - Defining equation: $(f\ \text{NUMBER}) = \text{singular}$
    - Existential constraint: $(f\ \text{NUMBER})$
    - Negative existential constraint: $\neg(f\ \text{NUMBER})$
    - Constraining equation: $(f\ \text{NUMBER}) =_c \text{singular}$
    - Negative constraining equation: $(f\ \text{NUMBER}) \neq \text{singular}$

Feature Simplicity and Constraint Types

- All features treated as valued features: no restriction on constraint types
- All features treated as binary features: only positive and negative constraining equations allowed
- All features treated as privative: only negative and existential constraints allowed
  - This understanding of privative features actually does treat number as a natural class.
- This treats the notion of feature simplicity as a kind of meta-theoretical statement in an explicit, non-ad-hoc feature theory.

Control and Raising

Lexical Entries

tried V $(\uparrow \text{PRED}) = \text{‘try}\langle\text{SUBJ,XCOMP}\rangle\text{’}$

$(\uparrow \text{SUBJ}) = (\uparrow \text{XCOMP SUBJ})$

seemed V $(\uparrow \text{PRED}) = \text{‘seem}\langle\text{CF}\rangle\text{SUBJ’}$

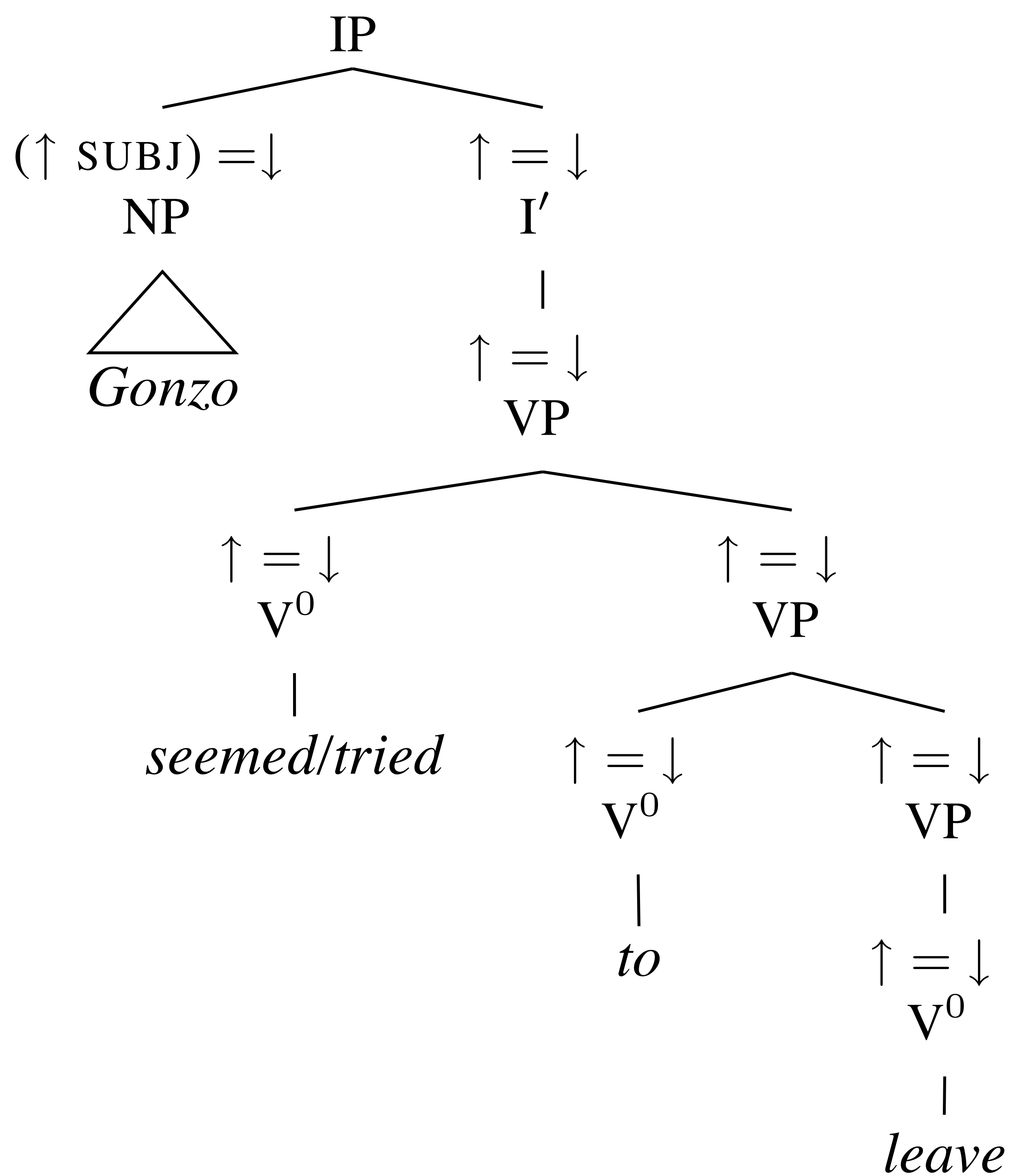
$\{ (\uparrow \text{SUBJ}) = (\uparrow \text{XCOMP SUBJ}) \mid$

$(\uparrow \text{SUBJ PRONTYPE}) = \text{EXPLETIVE}$

$(\uparrow \text{SUBJ FORM}) = \text{IT}$

$(\uparrow \text{COMP}) \}$

Raising to Subject/Subject Control C-structure



F-structures

$$\begin{bmatrix} \text{PRED} & \text{‘seem}\langle(\text{XCOMP})\rangle(\text{SUBJ})\text{’} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘Gonzo’} \end{bmatrix} \\ \text{XCOMP} & \begin{bmatrix} \text{PRED} & \text{‘leave}\langle(\text{SUBJ})\rangle\text{’} \\ \text{SUBJ} & \_\_\_\_ \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{PRED} & \text{‘try}\langle(\text{SUBJ}),(\text{XCOMP})\rangle\text{’} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{‘Gonzo’} \end{bmatrix} \\ \text{XCOMP} & \begin{bmatrix} \text{PRED} & \text{‘leave}\langle(\text{SUBJ})\rangle\text{’} \\ \text{SUBJ} & \_\_\_\_ \end{bmatrix} \end{bmatrix}$$

Copy Raising

Data

(1) Thora seems like she enjoys hot chocolate.

(2) Thora seems like Isak pinched her again.

(3) Thora seems like Isak ruined her book.

(4) * Thora seems like Isak enjoys hot chocolate.

(5) * Thora seems like Isak pinched Justin again.

(6) * Thora seems like Isak ruined Justin’s book.

Data

(7) It seems like there is a problem here.

(8) It seems like Thora is upset.

(9) It seems like it rained last night.

(10) There seems like there's a problem here.

(11) * There seems like it rained last night.

Lexical Entries

$like_1$ $\quad P^0 \quad (\uparrow \text{PRED}) = \text{`like}\langle\text{SUBJ,COMP}\rangle\text{'}$

$like_2$ $\quad P^0 \quad (\uparrow \text{PRED}) = \text{`like}\langle\text{CF}\rangle\text{SUBJ'}$

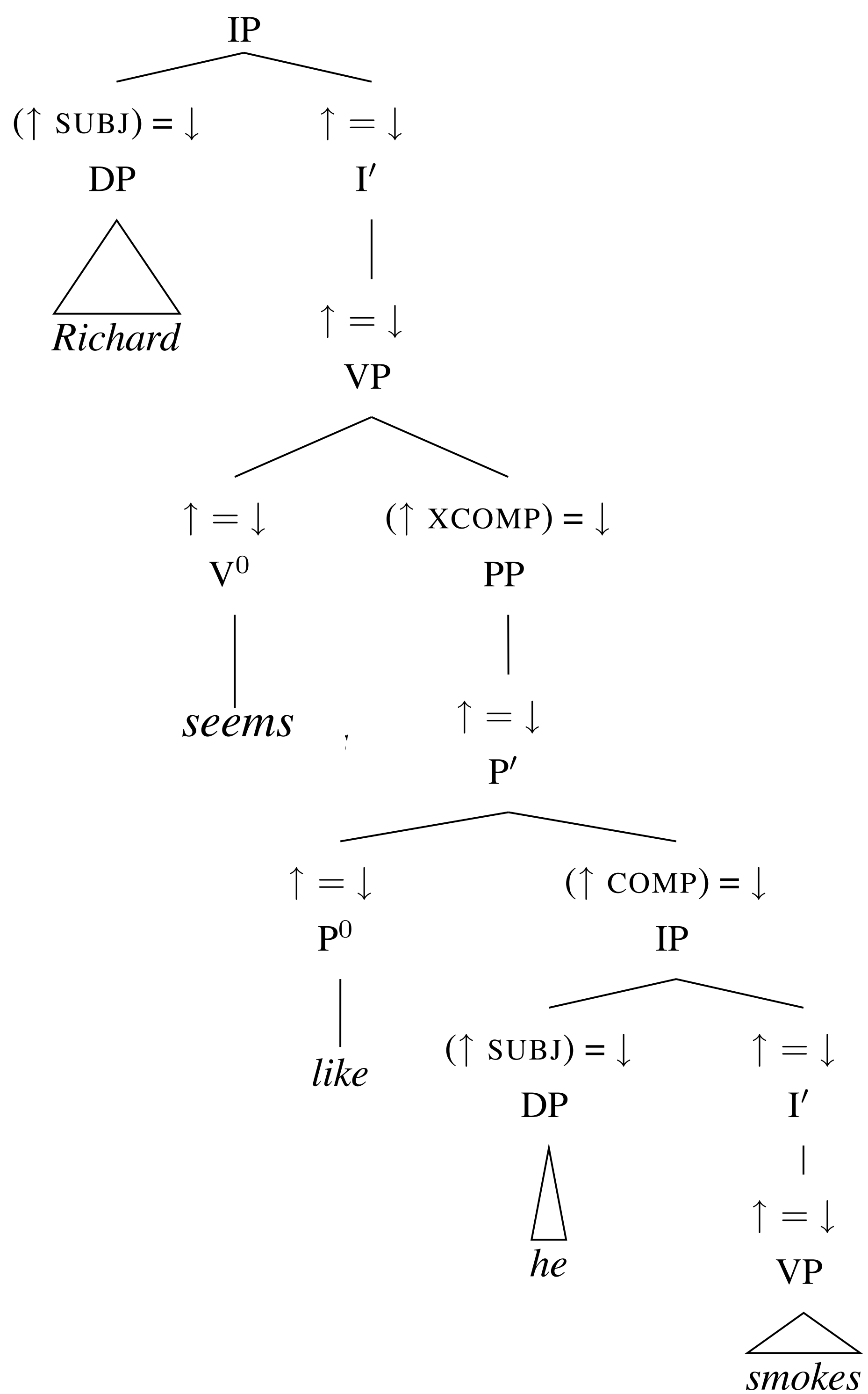
$$\{\ (\uparrow \text{SUBJ}) = (\uparrow \text{XCOMP SUBJ})\ |$$

$$(\uparrow \text{SUBJ PRONTYPE}) = \text{EXPLETIVE}$$

$$(\uparrow \text{SUBJ FORM}) = \text{IT}$$

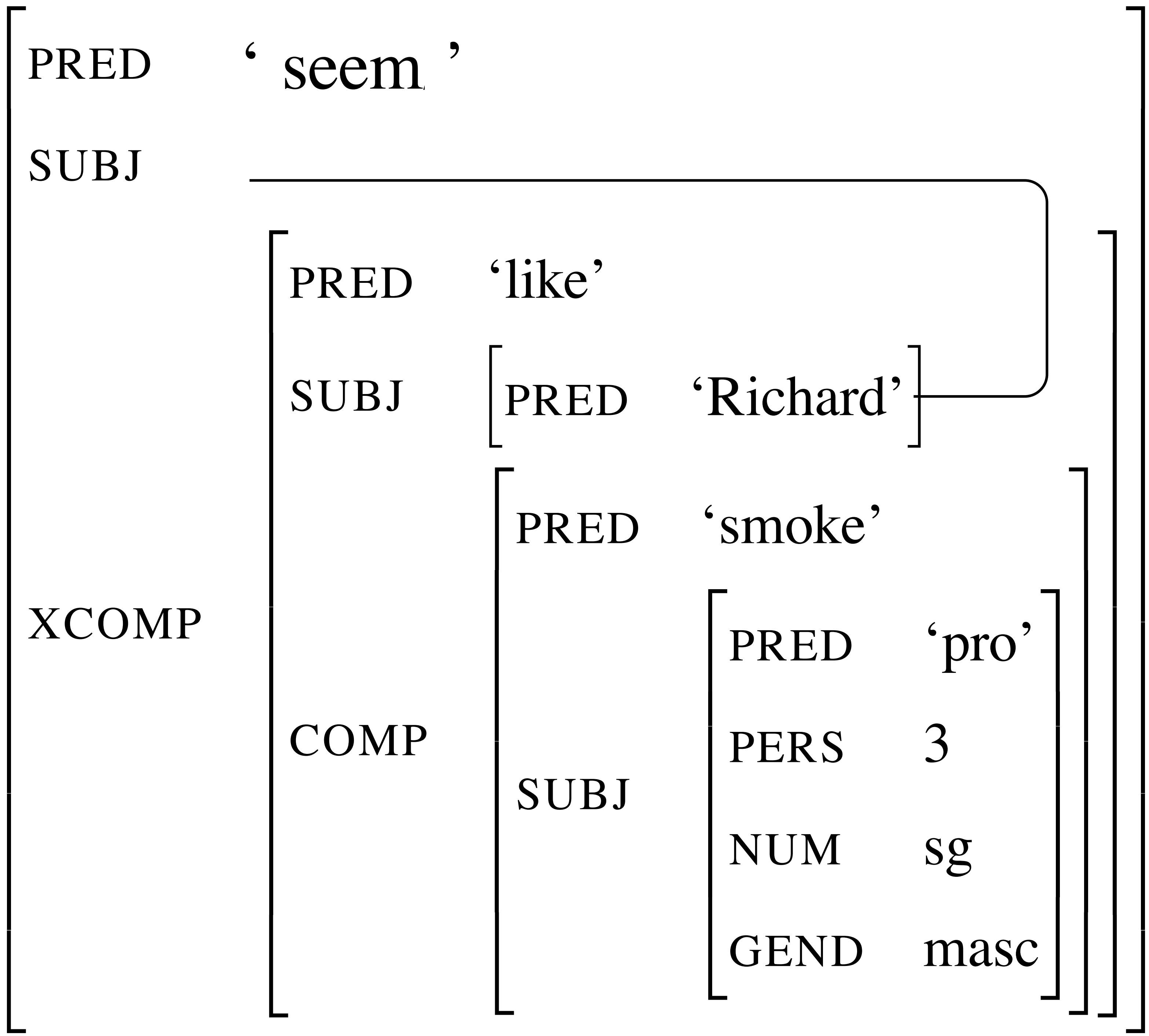
$$(\uparrow \text{COMP})\ \}$$

C-structure

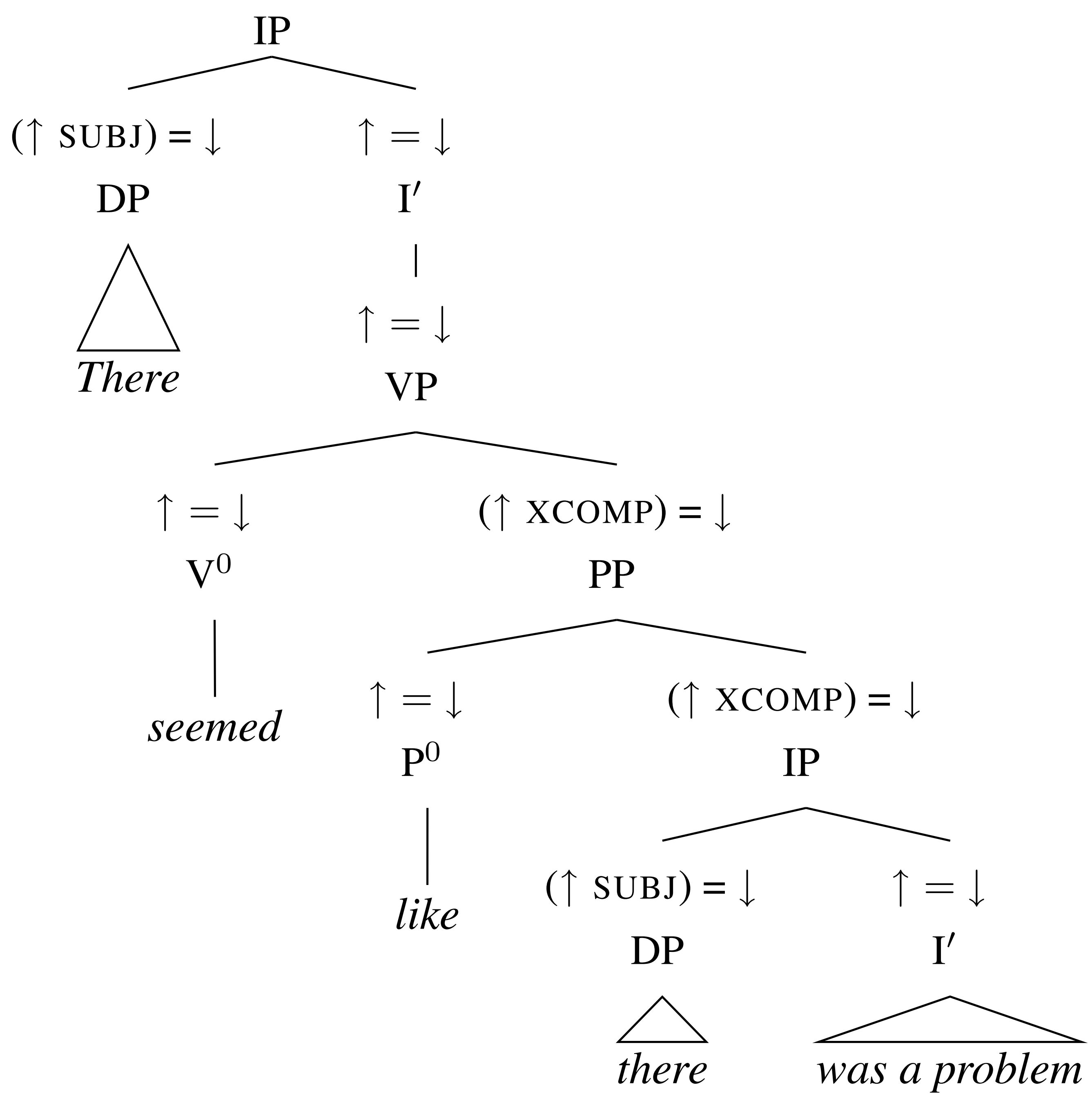


IP
(↑ SUBJ) = ↓
DP
Richard
↑ = ↓
I′
↑ = ↓
VP
↑ = ↓
V0
seems
(↑ XCOMP) = ↓
PP
↑ = ↓
P′
↑ = ↓
P0
like
(↑ COMP) = ↓
IP
(↑ SUBJ) = ↓
DP
he
↑ = ↓
I′
↑ = ↓
VP
smokes

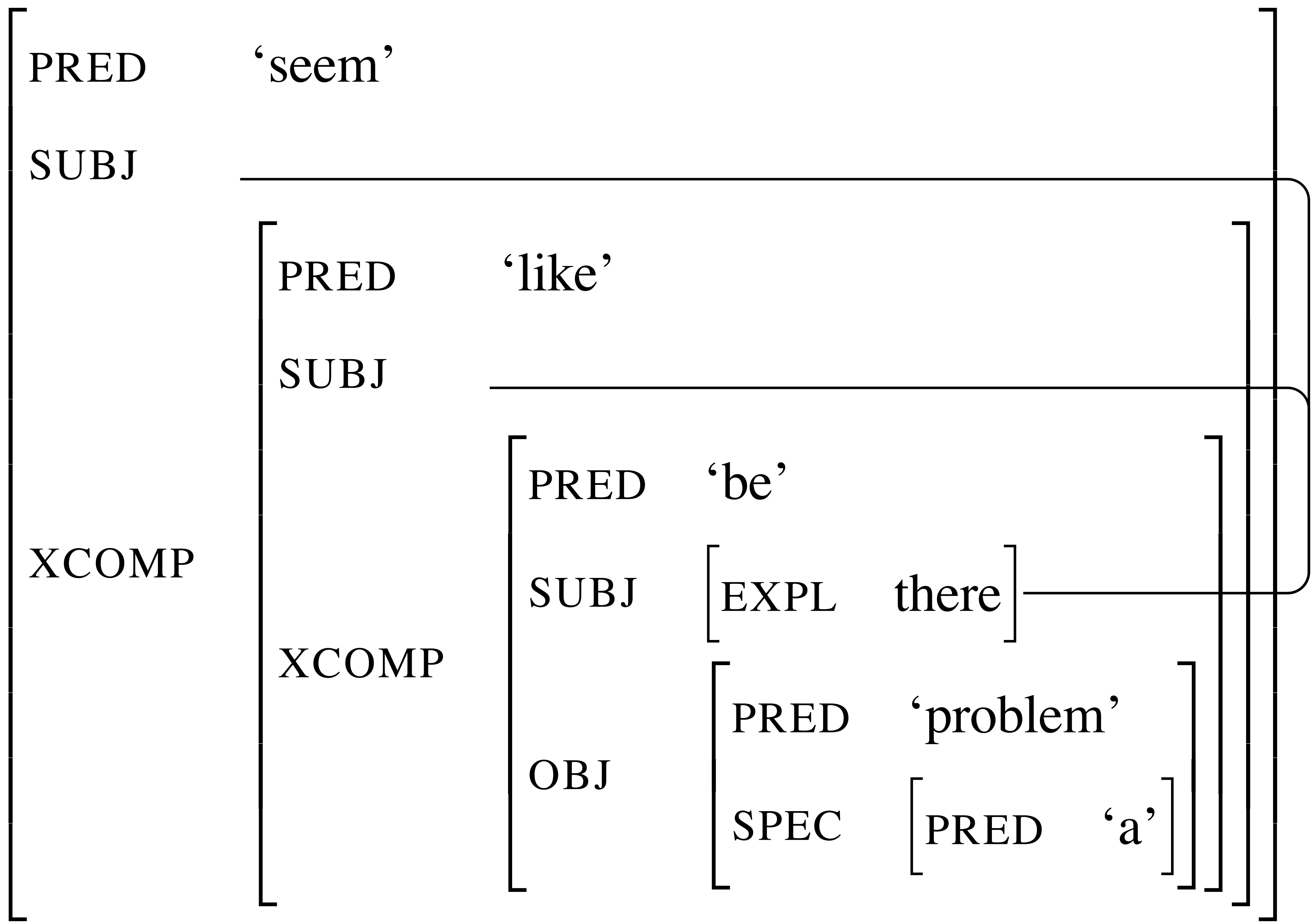
F-structure


PRED ‘ seem ’
SUBJ
XCOMP
PRED ‘like’
SUBJ
PRED ‘Richard’
COMP
PRED ‘smoke’
SUBJ
PRED ‘pro’
PERS 3
NUM sg
GEND masc

C-structure


IP
(↑ SUBJ) = ↓
DP
There
↑ = ↓
I′
↑ = ↓
VP
↑ = ↓
V0
seemed
(↑ XCOMP) = ↓
PP
↑ = ↓
P0
like
(↑ XCOMP) = ↓
IP
(↑ SUBJ) = ↓
DP
there
↑ = ↓
I′
was a problem

F-structure


PRED
'seem'
SUBJ
XCOMP
PRED
'like'
SUBJ
XCOMP
PRED
'be'
SUBJ
EXPL
there
OBJ
PRED
'problem'
SPEC
PRED
'a'

Unbounded Dependencies

Filler-Gap Dependencies

Functional Uncertainty

- The syntactic relationship between the top and bottom of an unbounded dependency is represented with a functional uncertainty:
  - Top = MiddlePath-Func-Uncertainty   Bottom-Func-Uncertainty

(1) [What] [did Kim claim that Sandy suspected that Robin knew]   [ ]

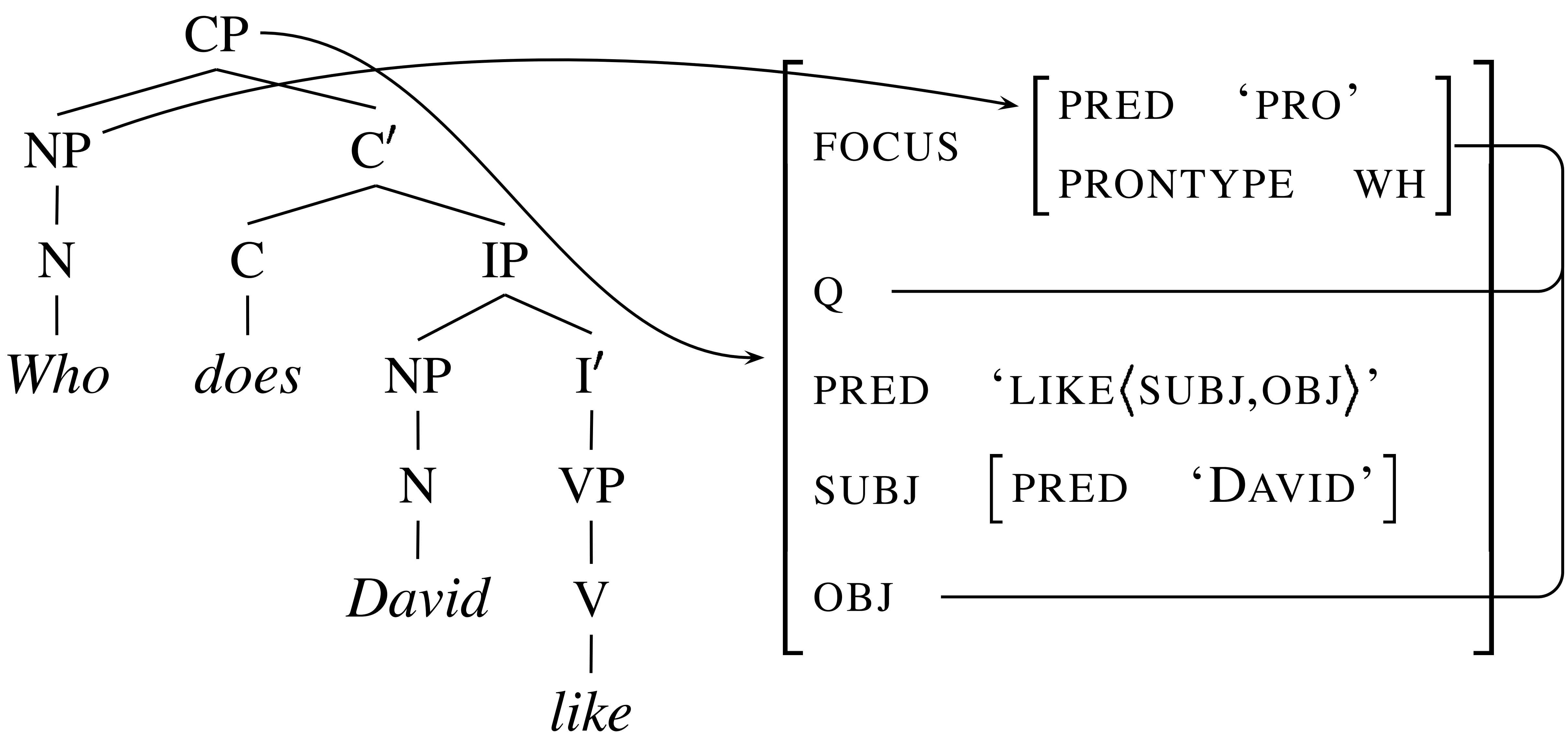
top   middle   bottom

$$(\uparrow \text{FOCUS}) = (\uparrow \text{COMP}^* \ \{\text{OBJ} \mid \text{OBJ}_\theta\})$$

top   middle   bottom

(2) [What] [did Kim claim that Sandy suspected that Robin gave Bo]   [ ]

Wh-Questions: Example



Wh-Questions: Annotated PS Rule

$$\text{CP} \longrightarrow \begin{pmatrix} \text{QuesP} \\ (\uparrow \textsc{focus}) = \downarrow \\ (\uparrow \textsc{focus}) = (\uparrow \textsc{QFocusPath}) \\ (\uparrow \textsc{q}) = (\uparrow \textsc{focus WhPath}) \\ (\uparrow \textsc{q prontype}) =_c \textsc{wh} \end{pmatrix} \begin{pmatrix} \text{C}' \\ \uparrow = \downarrow \end{pmatrix}$$

Wh-Questions: QuesP Metacategory

$$\text{QuesP} \equiv \{\text{NP} \mid \text{PP} \mid \text{AdvP} \mid \text{AP}\}$$

(1) NP: Who do you like?

(2) PP: To whom did you give a book?

(3) AdvP: When did you yawn?

(4) AP: How tall is Chris?

Wh-Questions: Unbounded Dependency Equation

English QFOCUSPATH:

$$\Big\{ \text{XCOMP} \mid \underset{(\rightarrow \text{LDD}) \neq -}{\text{COMP}} \mid \underset{(\rightarrow \text{TENSE})}{\text{OBJ}} \Big\}^{*} \quad \Big\{ \big(\text{ADJ} \underset{\neg(\rightarrow \text{TENSE})}{\in}\big) \quad (\text{GF}) \mid \text{GF} \Big\}$$

Wh-Questions: Pied Piping

English WHPATH:

$$\left\{ \text{SPEC}^* \mid \text{OBJ} \right\}$$

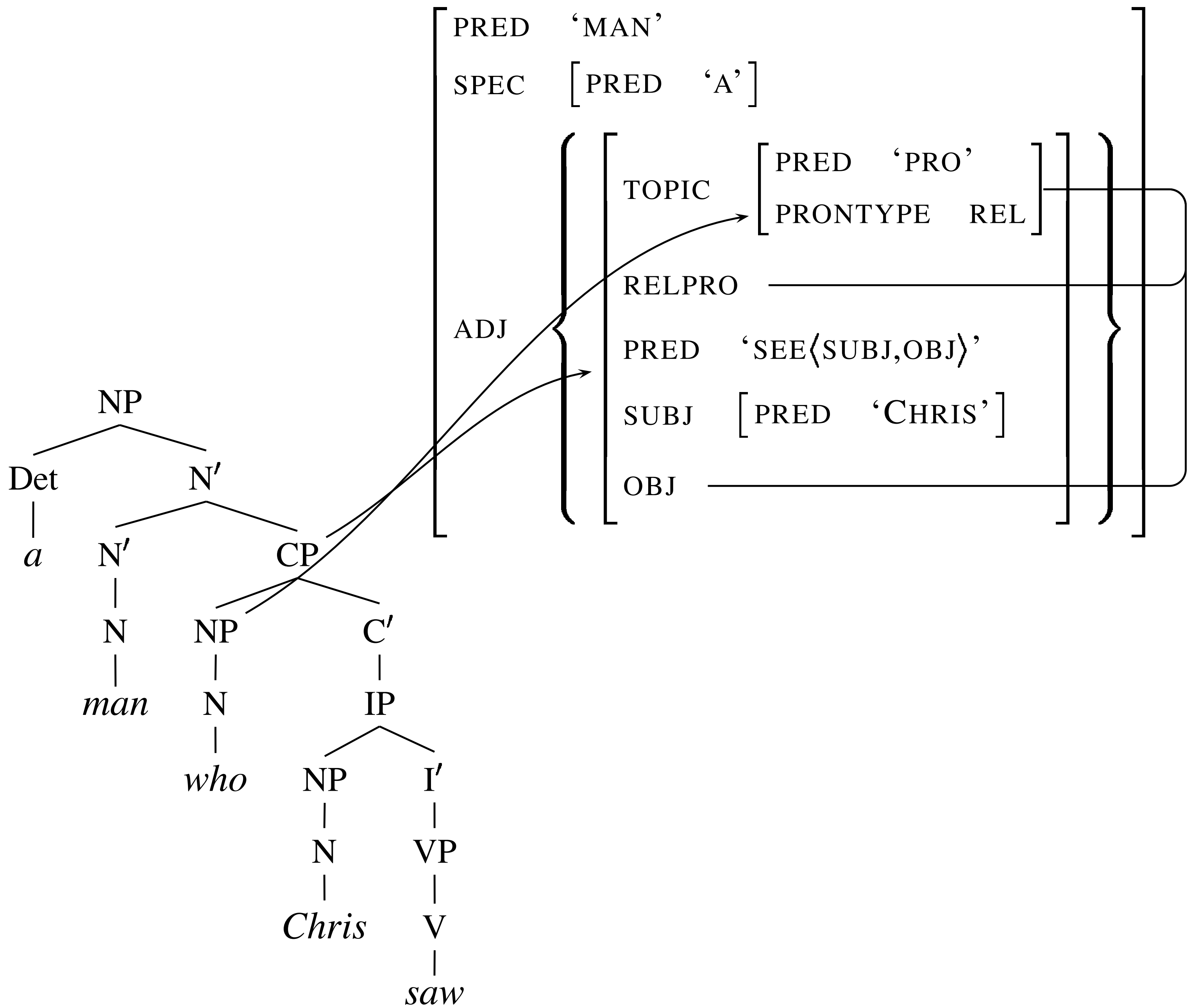
(1) [Whose book] did you read?

(2) [Whose brother's book] did you read?

(3) [In which room] do you teach?

Relative Clauses: Example

a man who Chris saw



Relative Clauses: Annotated PS Rule

$$\text{CP} \longrightarrow \left(\begin{array}{c} \text{RelP} \\ (\uparrow \text{ TOPIC}) = \downarrow \\ (\uparrow \text{ TOPIC}) = (\uparrow \text{ RTOPICPATH}) \\ (\uparrow \text{ RELPRO}) = (\uparrow \text{ TOPIC RELPATH}) \\ (\uparrow \text{ RELPRO PRONTYPE}) =_c \text{REL} \end{array}\right) \left(\begin{array}{c} \text{C}' \\ \uparrow = \downarrow \end{array}\right)$$

Relative Clauses: RelP Metacategory

$$\text{RelP} \equiv \{\text{NP} \mid \text{PP} \mid \text{AP} \mid \text{AdvP}\}$$

(1) NP: a man <u>who</u> I selected

(2) PP: a man <u>to whom</u> I gave a book

(3) AP: the kind of person <u>proud of whom</u> I could never be

(4) AdvP: the city <u>where</u> I live

Relative Clauses: Unbounded Dependency Equation

English RTOPICPATH:

$$\left\{ \text{XCOMP} \;\middle|\; \begin{array}{c}\text{COMP}\\ (\rightarrow \text{LDD}) \neq -\end{array} \;\middle|\; \begin{array}{c}\text{OBJ}\\ (\rightarrow \text{TENSE})\end{array} \right\}^{*} \quad \left\{ \left(\text{ADJ} \; \begin{array}{c}\in\\ \neg(\rightarrow \text{TENSE})\end{array} \right) \; (\text{GF}) \;\middle|\; \text{GF} \right\}$$

Relative Clauses: Pied Piping

English RELPATH:

$$\left\{\text{SPEC}^{*} \;\middle|\; \left[(\text{OBL}_{\theta})\,\text{OBJ}\right]^{*}\right\}$$

(1) the man [who] I met

(2) the man [whose book] I read

(3) the man [whose brother's book] I read

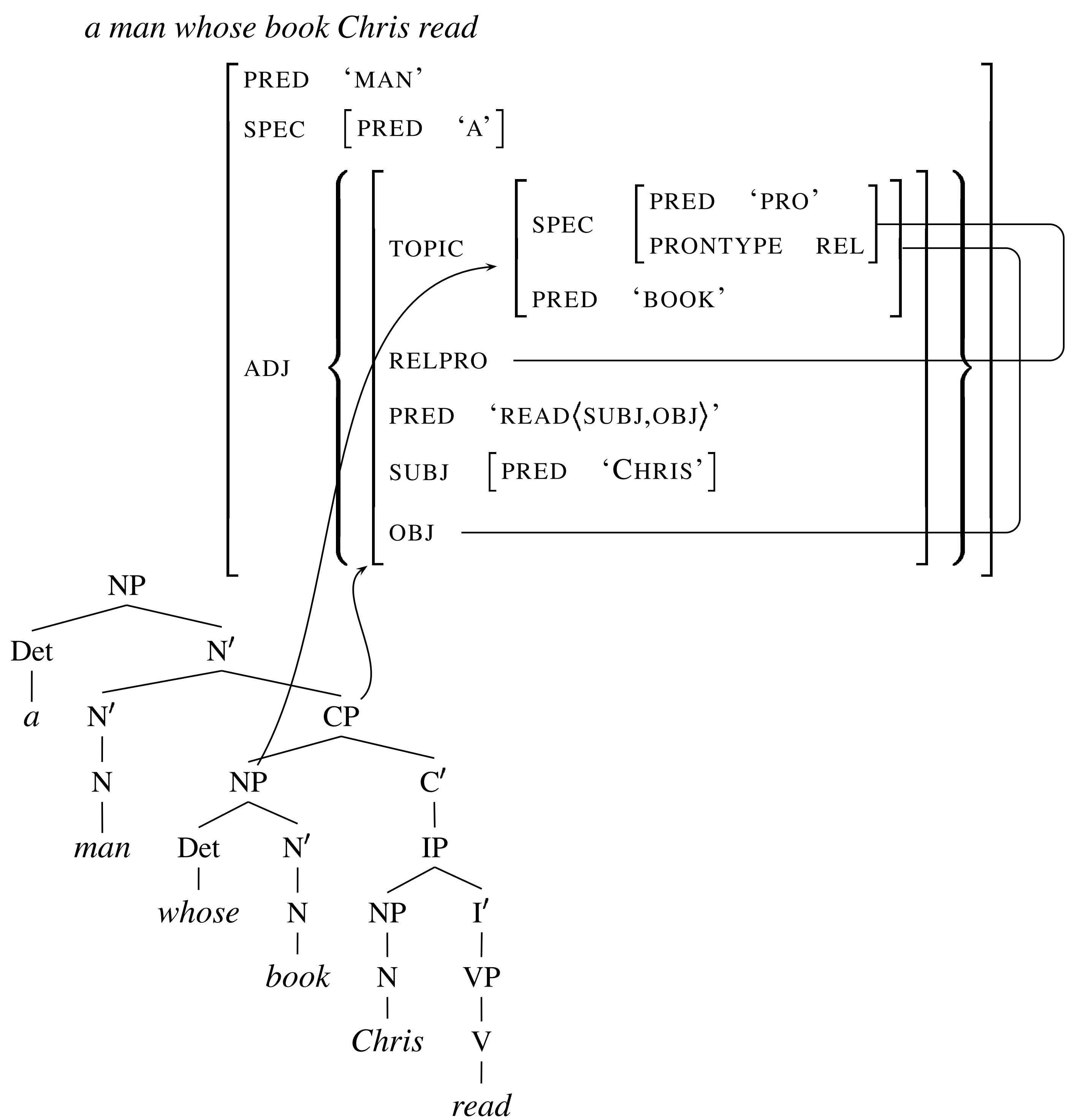
(4) the report [the cover of which] I designed

(5) the man [faster than whom] I can run

(6) the kind of person [proud of whom] I could never be

(7) the report [the height of the lettering on the cover of which] the government prescribes

Relative Clauses: Pied Piping Example



Constraints on Extraction

Empty Category Principle/*That*-Trace

(1) Who do you think [__ left]?

(2) * Who do you think [that __ left]?

(3) * What do you wonder [if __ smells bad]?

(4) Who do you think [__ should be trusted]?

(5) * Who do you think [that __ should be trusted]?

(6) Who do you think [that, under no circumstances, __ should be trusted]?

(7) Who do you wonder [if, under certain circumstances, __ could be trusted]?

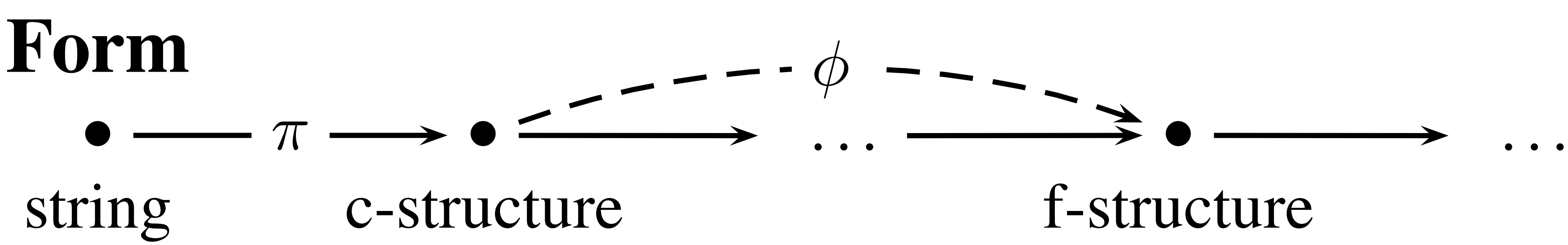
That-Trace in LFG

- LFG has a relation called f-precedence that uses the native precedence of c-structure to talk about precedence between bits of f-structure.
- F-precedence relies on LFG's projection architecture and the inverse of the c-structure–f-structure mapping function ϕ.
- The inverse is written ϕ^{-1} and returns the set of c-structure nodes that map to its argument f-structure node.

F-precedence
An f-structure *f* f-precedes an f-structure *g* ($f <_f g$) if and only if for all $n_1 \in \phi^{-1}(f)$ and for all $n_2 \in \phi^{-1}(g)$, n_1 c-precedes n_2.

That-Trace in LFG

- We can leverage LFG's projection architecture to capture the fact that *That*-Trace is a 'surfacy' phenomenon (cf. ECP as a PF constraint in recent Minimalism).



That-Trace in LFG

- Assume a native precedence relation on strings, yielding a notion of element that is string-adjacent to the right (‘next string element’), which we define as $\text{Right}_{\text{string}}(\pi^{-1}(*))$, where * designates the current c-structure node in a phrase structure rule element or lexical entry.
    - Let’s abbreviate the right string-adjacent element to * as $>$.
    - The semantics of $>$ is ‘the string element that is right string-adjacent to me’.
- Note that π^{-1} returns string elements, not sets of string elements, because π is bijective, since c-structures are trees.

That-Trace in LFG

- We can use f-precedence and > to capture the surfacy nature of *That*-Trace.
- Basically, English has a (somewhat arbitrary) constraint that the right-adjacent string element to the complementizer must be locally realized.
- This can be stated by requiring that any unbounded dependency function in the f-structure corresponding to the element that occurs in the string immediately after the complementizer should **not** f-precede the complementizer's f-structure.

Left Branch Constraint

(1) Whose car did you drive __?

(2) * Whose did you drive [__ car]?

Left Branch Constraint in LFG

- Do not include SPEC/POSS in GFs of possible extraction sites.
- Note that the equation we looked at previously already disallows the extraction from passing through a SPEC in the first part.
- We modify the equation as follows

$$\left\{ \text{XCOMP} \;\middle|\; \underset{(\rightarrow\ \text{LDD}) \neq -}{\text{COMP}} \;\middle|\; \underset{(\rightarrow\ \text{TENSE})}{\text{OBJ}} \right\}^{*} \quad \left\{ \left(\underset{\neg(\rightarrow\ \text{TENSE})}{\text{ADJ} \quad \in} \right) \ (\text{GF}) \;\middle|\; \text{GF} - \text{SPEC} \right\}$$

Wh-Islands in LFG: Off-Path Constraints

- The off-path metavariable ← refers to the f-structure that contains the attribute that the constraint is attached to.
- The off-path metavariable → refers to the f-structure that is the value of the attribute that the constraint is attached to.

$$\Big\{ \text{XCOMP} \;\Big|\; \underset{(\to\ \text{LDD}) \neq -}{\text{COMP}} \;\Big|\; \underset{(\to\ \text{TENSE})}{\text{OBJ}} \Big\}^* \quad \Big\{ \Big(\text{ADJ} \underset{\neg(\to\ \text{TENSE})}{\in} \Big) \; (\text{GF}) \;\Big|\; \text{GF} - \text{SPEC} \Big\}$$

- Use ← to state the bottom cannot be in an f-structure that has an unbounded dependency function UDF, where UDF = {TOPIC | FOCUS}.

$$\Big\{ \text{XCOMP} \;\Big|\; \underset{(\to\ \text{LDD}) \neq -}{\text{COMP}} \;\Big|\; \underset{(\to\ \text{TENSE})}{\text{OBJ}} \Big\}^* \quad \Big\{ \Big(\text{ADJ} \underset{\neg(\to\ \text{TENSE})}{\in} \Big) \; (\text{GF}) \;\Big|\; \underset{\neg(\leftarrow\ \text{UDF})}{\text{GF} - \text{SPEC}} \Big\}$$

Successive Cyclic Effects

Successive Cyclicity

- Data from languages such as Irish and Chamorro, which show successive marking along the extraction path, have motivated the claim that extraction/movement is 'cyclic' (not all at once). Cf. Phases in Minimalism.
- Of course, this data does not argue for movement per se, as some have wrongly assumed, but rather that unbounded dependencies should

  1. Be made up of a series of local relations; or
  2. Have a way to refer to their environments as the dependency is constructed.

- HPSG has adopted the first approach, LFG the second.

Data: Irish

- Note: Date from McCloskey via Bouma et al. (2001).

a. Shíl mé *goN* mbeadh sé ann
thought I PRT *would-be he there*
I thought that he would be there.

b. Dúirt mé *gurL* shíl mé *goN* mbeadh sé ann
*said I goN+*PAST *thought I* PRT *would-be he there*
I said that I thought that he would be there.

c. an fear *aL* shíl mé *aL* bheadh __ ann
[the man]$_j$ PRT *thought I* PRT *would-be* $\_\__j$ *there*
the man that I thought would be there

d. an fear *aL* dúirt mé *aL* shíl mé *aL* bheadh __ ann
[the man]$_j$ PRT *said I* PRT *thought I* PRT *would-be* $\_\__j$ *there*
The man that I said I thought would be there

e. an fear *aL* shíl __ *goN* mbeadh sé ann
[the man]$_j$ PRT *thought* $\_\__j$ PRT *would-be he there*
the man that thought he would be there

Irish Successive Cyclicity in LFG

$aL \quad \hat{\mathrm{C}} \qquad (\uparrow \text{UDF}) = (\uparrow \quad \text{CF}^* \quad \text{GF})$

$(\rightarrow \text{UDF}) = (\uparrow \text{UDF})$

Note: UDF = {TOPIC | FOCUS}, CF = {XCOMP | COMP}

$goN \quad \hat{\mathrm{C}} \qquad (\uparrow \text{TENSE})$

$\neg(\uparrow \text{UDF})$

Glue Semantics

Glue Semantics

- Glue Semantics is a type-logical semantics that can be tied to any syntactic formalism that supports a notion of headedness.
- Glue Semantics can be thought of as *categorial semantics without categorial syntax.*
- The independent syntax assumed in Glue Semantics means that the logic of composition is *commutative*, unlike in Categorial Grammar.
- Selected works:
  Dalrymple (1999, 2001), Crouch & van Genabith (2000),
  Asudeh (2004, 2005a,b, in prep.), Lev 2007, Kokkonidis (in press)

Glue Semantics

- Lexically-contributed *meaning constructors* :=

Meaning language term $\mathcal{M} : G$ Composition language term

- Meaning language := some lambda calculus
  - Model-theoretic
- Composition language := linear logic
  - Proof-theoretic
- Curry Howard Isomorphism between formulas (meanings) and types (proof terms)
- Successful Glue Semantics proof:

$$\Gamma \vdash \mathcal{M} : G_t$$

Key Glue Proof Rules with Curry-Howard Terms

Application : Implication Elimination

$$\dfrac{\begin{matrix}\vdots \\ a : A\end{matrix} \qquad \begin{matrix}\vdots \\ f : A \multimap B\end{matrix}}{f(a) : B} \multimap_{\mathcal{E}}$$

Abstraction : Implication Introduction

$$\dfrac{\begin{matrix}[x : A]^1 \\ \vdots \\ f : B\end{matrix}}{\lambda x.f : A \multimap B} \multimap_{\mathcal{I},1}$$

Pairwise Substitution : Conjunction Elimination

$$\dfrac{\begin{matrix}\vdots \\ a : A \otimes B\end{matrix} \qquad \begin{matrix}[x : A]^1 \;\; [y : B]^2 \\ \vdots \\ f : C\end{matrix}}{\mathsf{let}\ a\ \mathsf{be}\ x \times y\ \mathsf{in}\ f : C} \otimes_{\mathcal{E},1,2}$$

Beta reduction for let:

$$\mathsf{let}\ a \times b\ \mathsf{be}\ x \times y\ \mathsf{in}\ f \;\; \Rightarrow_{\beta} \;\; f[a/x, b/y]$$

Example: *Mary laughed*

1. $mary : \uparrow_{\sigma_e}$
2. $laugh : (\uparrow \text{SUBJ})_{\sigma_e} \multimap \uparrow_{\sigma_t}$

$$f\begin{bmatrix} \text{PRED} & \text{`laugh}\langle\text{SUBJ}\rangle\text{'} \\ \text{SUBJ} & g\begin{bmatrix}\text{PRED} & \text{`Mary'}\end{bmatrix} \end{bmatrix}$$

$1'.$ $mary : g_{\sigma_e}$

$2'.$ $laugh : g_{\sigma_e} \multimap f_{\sigma_t}$

$1''.$ $mary : m$

$2''.$ $laugh : m \multimap l$

Proof

1.	$mary : m$	Lex. **Mary**
2.	$laugh : m \multimap l$	Lex. **laughed**
3.	$laugh(mary) : l$	E $\multimap$, 1, 2

$\equiv$

Proof

$$\frac{mary : m \qquad laugh : m \multimap l}{laugh(mary) : l} \multimap_{\mathcal{E}}$$

Example: *Most presidents speak*

1. $\lambda R \lambda S.most(R, S) : (v \multimap r) \multimap \forall X.[(p \multimap X) \multimap X]$ Lex. **most**
2. $president^* : v \multimap r$ Lex. **presidents**
3. $speak : p \multimap s$ Lex. **speak**

$$\dfrac{\dfrac{\begin{array}{l}\lambda R \lambda S.most(R, S) : \\ (v \multimap r) \multimap \forall X.[(p \multimap X) \multimap X]\end{array} \quad \begin{array}{l}president^* : \\ v \multimap r\end{array}}{\begin{array}{l}\lambda S.most(president^*, S) : \\ \forall X.[(p \multimap X) \multimap X]\end{array}} \quad \begin{array}{l}speak : \\ p \multimap s\end{array}}{most(president^*, speak) : s} \multimap_{\mathcal{E}}, [s/X]$$

Example:
Most presidents speak at least one language

$$
\begin{bmatrix}
\text{PRED} & \text{`speak}\langle\text{SUBJ, OBJ}\rangle\text{'} \\
\text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`president'} \\ \text{SPEC} & \begin{bmatrix} \text{PRED} & \text{`most'} \end{bmatrix} \end{bmatrix} \\
\text{OBJ} & \begin{bmatrix} \text{PRED} & \text{`language'} \\ \text{SPEC} & \begin{bmatrix} \text{PRED} & \text{`at-least-one'} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Single parse

➡

Multiple scope possibilities (Underspecification through quantification)

1. $\lambda R \lambda S.most(R, S) :$ $(v1 \multimap r1) \multimap \forall X.[(p \multimap X) \multimap X]$ — Lex. **most**
2. $president^{*} : v1 \multimap r1$ — Lex. **presidents**
3. $speak : p \multimap l \multimap s$ — Lex. **speak**
4. $\lambda P \lambda Q.at\text{-}least\text{-}one(P, Q) :$ $(v2 \multimap r2) \multimap \forall Y.[(l \multimap Y) \multimap Y]$ — Lex. **at least one**
5. $language : v2 \multimap r2$ — Lex. **language**

Most presidents speak at least one language

Subject wide scope

$$\dfrac{\lambda P \lambda Q.a\text{-}l\text{-}o(P,Q) : (v2 \multimap r2) \multimap \forall Y.[(l \multimap Y) \multimap Y] \qquad lang : v2 \multimap r2}{\lambda Q.a\text{-}l\text{-}o(lang, Q) : \forall Y.[(l \multimap Y) \multimap Y]}$$

$$\dfrac{\lambda x \lambda y.speak(x,y) : p \multimap l \multimap s \qquad [z : p]^1}{\lambda y.speak(z,y) : l \multimap s}$$

$$\dfrac{\lambda Q.a\text{-}l\text{-}o(lang, Q) : \forall Y.[(l \multimap Y) \multimap Y] \qquad \lambda y.speak(z,y) : l \multimap s}{a\text{-}l\text{-}o(lang, \lambda y.speak(z,y)) : s}\ [s/Y]$$

$$\dfrac{a\text{-}l\text{-}o(lang, \lambda y.speak(z,y)) : s}{\lambda z.a\text{-}l\text{-}o(lang, \lambda y.speak(z,y)) : p \multimap s}\ \multimap_{\mathcal{I},1}$$

$$\dfrac{\lambda R \lambda S.most(R,S) : (v1 \multimap r1) \multimap \forall X.[(p \multimap X) \multimap X] \qquad president^* : v1 \multimap r1}{\lambda S.most(president^*, S) : \forall X.[(p \multimap X) \multimap X]}$$

$$\dfrac{\lambda S.most(president^*, S) : \forall X.[(p \multimap X) \multimap X] \qquad \lambda z.a\text{-}l\text{-}o(lang, \lambda y.speak(z,y)) : p \multimap s}{most(president^*, \lambda z.a\text{-}l\text{-}o(lang, \lambda y.speak(z,y))) : s}\ [s/X]$$

Most presidents speak at least one language
Object wide scope

$$\dfrac{\lambda R\lambda S.most(R,S) : (v1 \multimap r1) \multimap \forall X.[(p \multimap X) \multimap X] \qquad president^{*} : v1 \multimap r1}{\lambda S.most(president^{*}, S) : \forall X.[(p \multimap X) \multimap X]}$$

$$\dfrac{\lambda y\lambda x.speak(x,y) : l \multimap p \multimap s \qquad [z : l]^{1}}{\lambda x.speak(x,z) : p \multimap s}$$

$$\dfrac{\lambda S.most(president^{*}, S) : \forall X.[(p \multimap X) \multimap X] \qquad \lambda x.speak(x,z) : p \multimap s}{most(president^{*}, \lambda x.speak(x,z)) : s}\ [s/X]$$

$$\dfrac{most(president^{*}, \lambda x.speak(x,z)) : s}{\lambda z.most(president^{*}, \lambda x.speak(x,z)) : l \multimap s}\ \multimap_{\mathcal{I},1}$$

$$\dfrac{\lambda P\lambda Q.a\text{-}l\text{-}o(P,Q) : (v2 \multimap r2) \multimap \forall Y.[(l \multimap Y) \multimap Y] \qquad lang : v2 \multimap r2}{\lambda Q.a\text{-}l\text{-}o(lang, Q) : \forall Y.[(l \multimap Y) \multimap Y]}$$

$$\dfrac{\lambda Q.a\text{-}l\text{-}o(lang, Q) : \forall Y.[(l \multimap Y) \multimap Y] \qquad \lambda z.most(president^{*}, \lambda x.speak(x,z)) : l \multimap s}{a\text{-}l\text{-}o(lang, \lambda z.most(president^{*}, \lambda x.speak(x,z))) : s}\ [s/Y]$$

Anaphora in Glue Semantics

- Variable-free: pronouns are functions on their antecedents (Jacobson 1999, among others)
- Commutative logic of composition allows pronouns to compose **directly** with their antecedents.
  - No need for otherwise unmotivated additional type shifting (e.g. Jacobson's *z*-shift)

Anaphora in Glue Semantics

1. Joe said he bowls.

- Pronominal meaning constructor:

$$\lambda z.z \times z : A \multimap (A \otimes P)$$

$$\dfrac{\dfrac{\dfrac{joe : j \qquad \lambda z.z \times z : j \multimap (j \otimes p)}{joe \times joe : j \otimes p} \qquad \dfrac{\dfrac{[x : j]^1 \qquad \lambda u \lambda q.say(u, q) : j \multimap b \multimap s}{\lambda q.say(x, q) : b \multimap s} \qquad \dfrac{[y : p]^2 \qquad \lambda v.bowl(v) : p \multimap b}{bowl(y) : b}}{say(x,\ bowl(y)) : s}}{\textsf{let}\ joe \times joe\ \textsf{be}\ x \times y\ \textsf{in}\ say(x, bowl(y)) : s} \otimes_{\mathcal{E},1,2}}{say(joe, bowl(joe)) : s} \Rightarrow_{\beta}$$

Further Points of Interest

- Glue Semantics can be understood as a *representationalist* theory, picking up on a theme from Wednesday's semantics workshop.
    - Proofs can be reasoned about as representations (Asudeh & Crouch 2002a,b).
    - Proofs have strong identity criteria: normalization, comparison
- Glue Semantics allows recovery of a non-representationalist notion of *direct compositionality* (Asudeh 2005, 2006).
    - ➡ Flexible framework with lots of scope for exploration of questions of compositionality and semantic representation