Lexical-Functional Grammar
&
Flexible Composition

Ash Asudeh
Oxford University & Carleton University

Goals

- Provide an overview of Lexical-Functional Grammar
- Provide an overview of Glue Semantics
- Provide an introduction to an approach to argument structure that builds on these two theories, part of what I call Flexible Composition

Lexical-Functional Grammar

History

- LFG was developed by Joan Bresnan, a syntactician, and Ron Kaplan, a social psychologist by training but then a computational linguist, as a constraint-based/declarative alternative to transformational/procedural theories of the time.
- Desiderata:
  - Formal precision
  - Psychological plausibility
  - Computational tractability

Overview

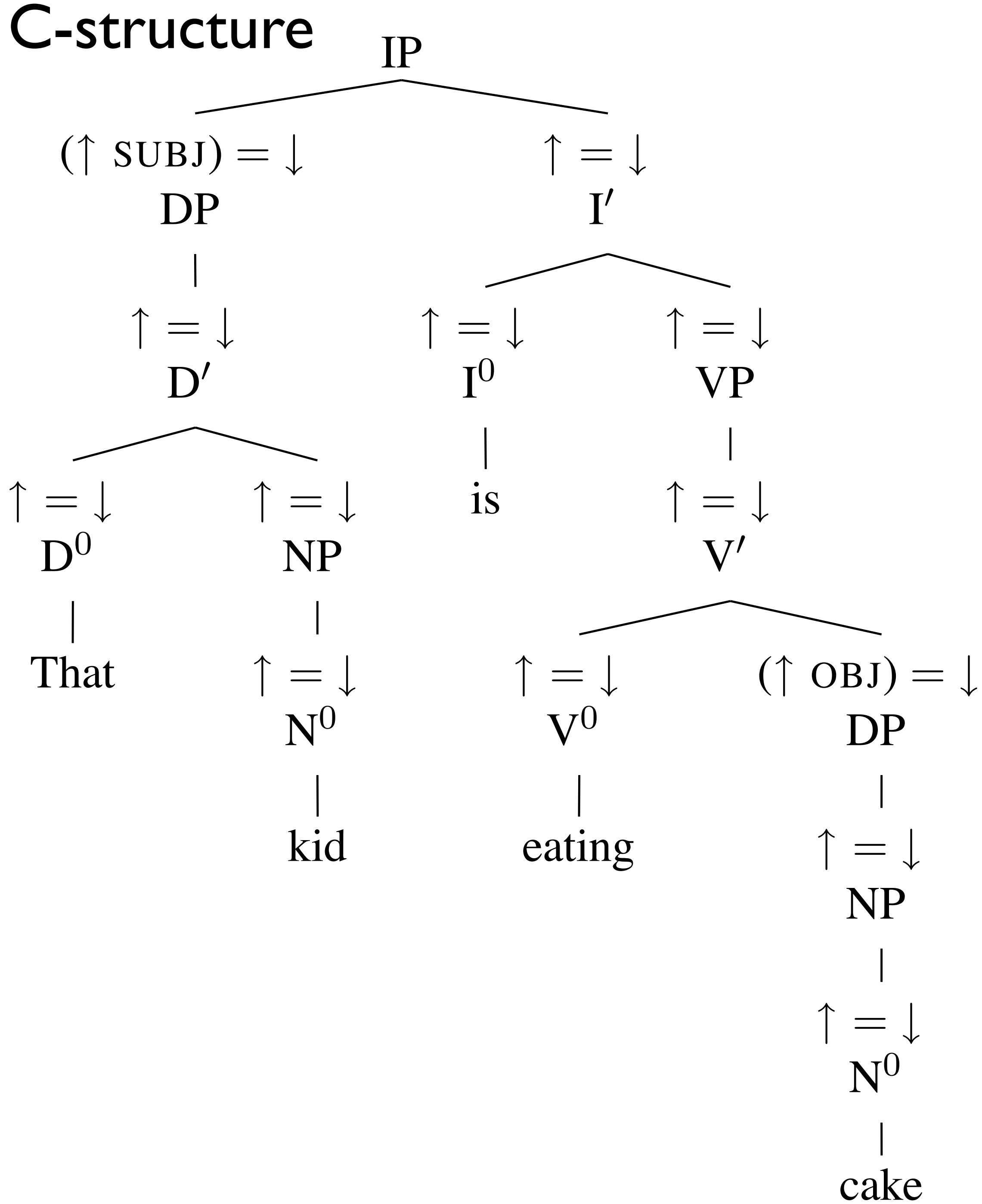
- At the heart of LFG remain its two syntactic structures:
  - C(onstituent)-structure
    - ‘Concrete syntax’: Precedence, dominance, constituency
  - F(unctional)-structure
    - ‘Abstract syntax’: Morphosyntactic features, grammatical functions, predication, subcategorization, local dependencies (agreement, control, raising), unbounded dependencies, anaphoric syntax (binding)

The ϕ correspondence function

- Elements of the c-structure are mapped to (put into *correspondence* with) elements of the f-structure by the ϕ *correspondence function* (sometimes called a *projection function*).
- This is accomplished by adding *functional descriptions* to the nodes in the c-structure tree.
- These equations use the ↑ ("up arrow") and ↓ ("down arrow") *metavariables*.
- A ↑ on a c-structure node *n* refers to the f-structure of the (c-structure) mother of *n*.
- A ↓ on a c-structure node n refers to the f-structure of node *n*.
- Examples:
  - ↑=↓ on *n* means that *n* and *n*'s mother map to the same f-structure.
  - (↑ SUBJECT) =↓ on *n* means that the f-structure of *n* is the value of the SUBJECT attribute in the f-structure of *n*'s mother.

Example: *That kid is eating cake*

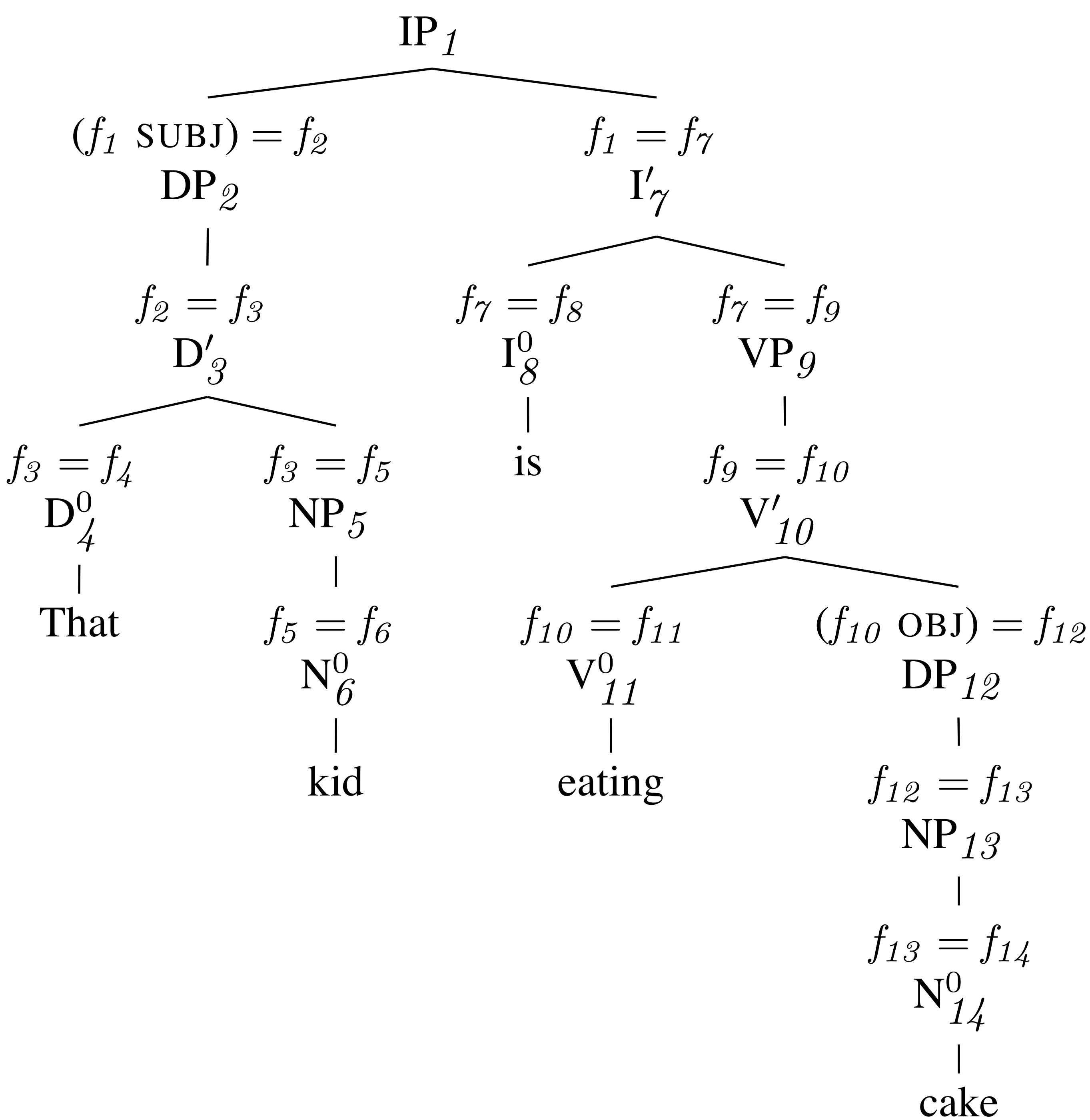
C-structure



Lexical entries

that, D^0	$(\uparrow \text{DEFINITE}) = +$ $(\uparrow \text{DEIXIS}) = \text{DISTAL}$ $(\uparrow \text{NUMBER}) = \text{SG}$ $(\uparrow \text{PERSON}) = 3$
kid, N^0	$(\uparrow \text{PRED}) = \text{'kid'}$ $(\uparrow \text{NUMBER}) = \text{SG}$ $(\uparrow \text{PERSON}) = 3$
is, I^0	$(\uparrow \text{SUBJ NUMBER}) = \text{SG}$ $(\uparrow \text{SUBJ PERSON}) = 3$ $(\uparrow \text{TENSE}) = \text{PRESENT}$ $(\uparrow \text{PARTICIPLE}) =_c \text{PRESENT}$
eating, V^0	$(\uparrow \text{PRED}) = \text{'eat}\langle\text{SUBJ,OBJ}\rangle\text{'}$ $(\uparrow \text{ASPECT}) = \text{PROGRESSIVE}$ $(\uparrow \text{PARTICIPLE}) = \text{PRESENT}$
cake, N^0	$(\uparrow \text{PRED}) = \text{'cake'}$ $(\uparrow \text{NUMBER}) = \text{SG}$ $(\uparrow \text{PERSON}) = 3$

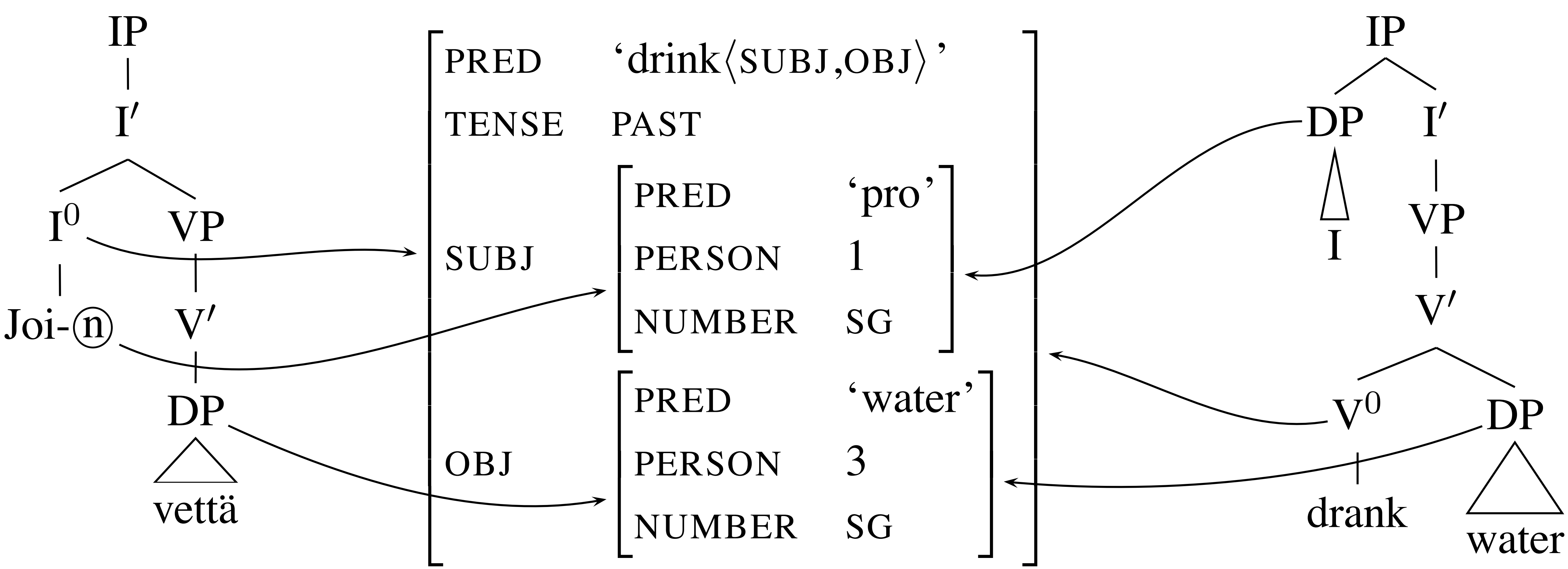
Example:
That kid is eating cake



$$
\begin{array}{l}
f_1\\ f_7\\ f_8\\ f_9\\ f_{10}\\ f_{11}
\end{array}
\left[\begin{array}{lll}
\text{PRED} & & \text{'eat}\langle\text{SUBJ,OBJ}\rangle\text{'} \\
\text{SUBJ} & \begin{array}{l} f_2\\ f_3\\ f_4\\ f_5\\ f_6 \end{array} & \left[\begin{array}{ll} \text{PRED} & \text{'kid'} \\ \text{DEIXIS} & \text{DISTAL} \\ \text{DEFINITE} & + \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{array}\right] \\
\text{OBJ} & \begin{array}{l} f_{12}\\ f_{13}\\ f_{14} \end{array} & \left[\begin{array}{ll} \text{PRED} & \text{'cake'} \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{array}\right] \\
\text{TENSE} & & \text{PRESENT} \\
\text{ASPECT} & & \text{PROGRESSIVE} \\
\text{PARTICIPLE} & & \text{PRESENT}
\end{array}\right]
$$

Flexibility in mapping

Finnish | Common (subsumptive) f-structure | English



Example:
the book that she likes

C-structure rules

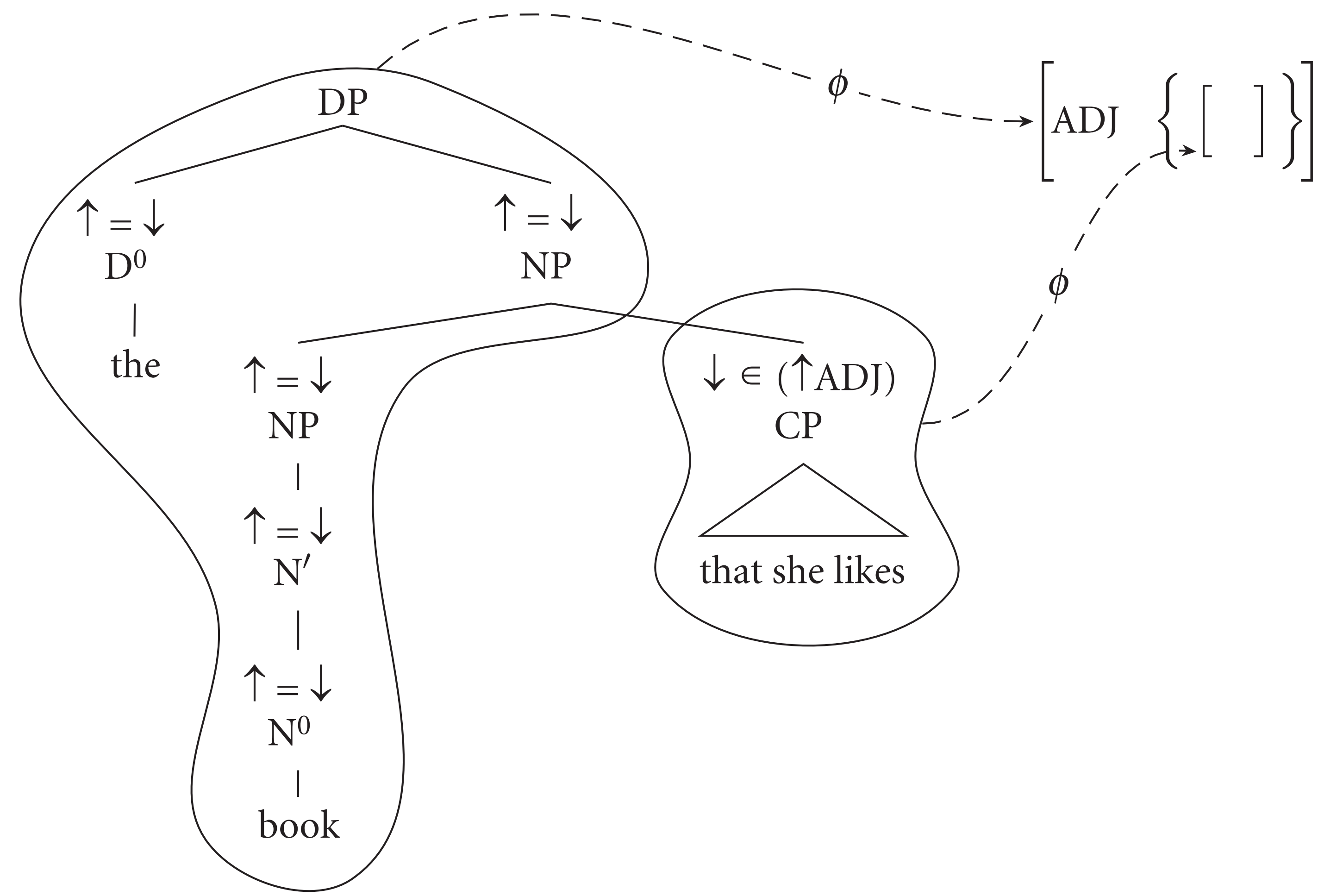
$$\mathrm{DP} \longrightarrow \underset{\uparrow = \downarrow}{\mathrm{D}^0} \quad \underset{\uparrow = \downarrow}{\mathrm{NP}}$$

$$\mathrm{NP} \longrightarrow \underset{\uparrow = \downarrow}{\mathrm{NP}} \quad \underset{\downarrow \in (\uparrow \text{ ADJ})}{\mathrm{CP}}$$

$$\mathrm{NP} \longrightarrow \underset{\uparrow = \downarrow}{\mathrm{N}'}$$

$$\mathrm{N}' \longrightarrow \underset{\uparrow = \downarrow}{\mathrm{N}^0}$$

C-structure and corresponding f-structure



Example:
the book that she likes

Lexical entries

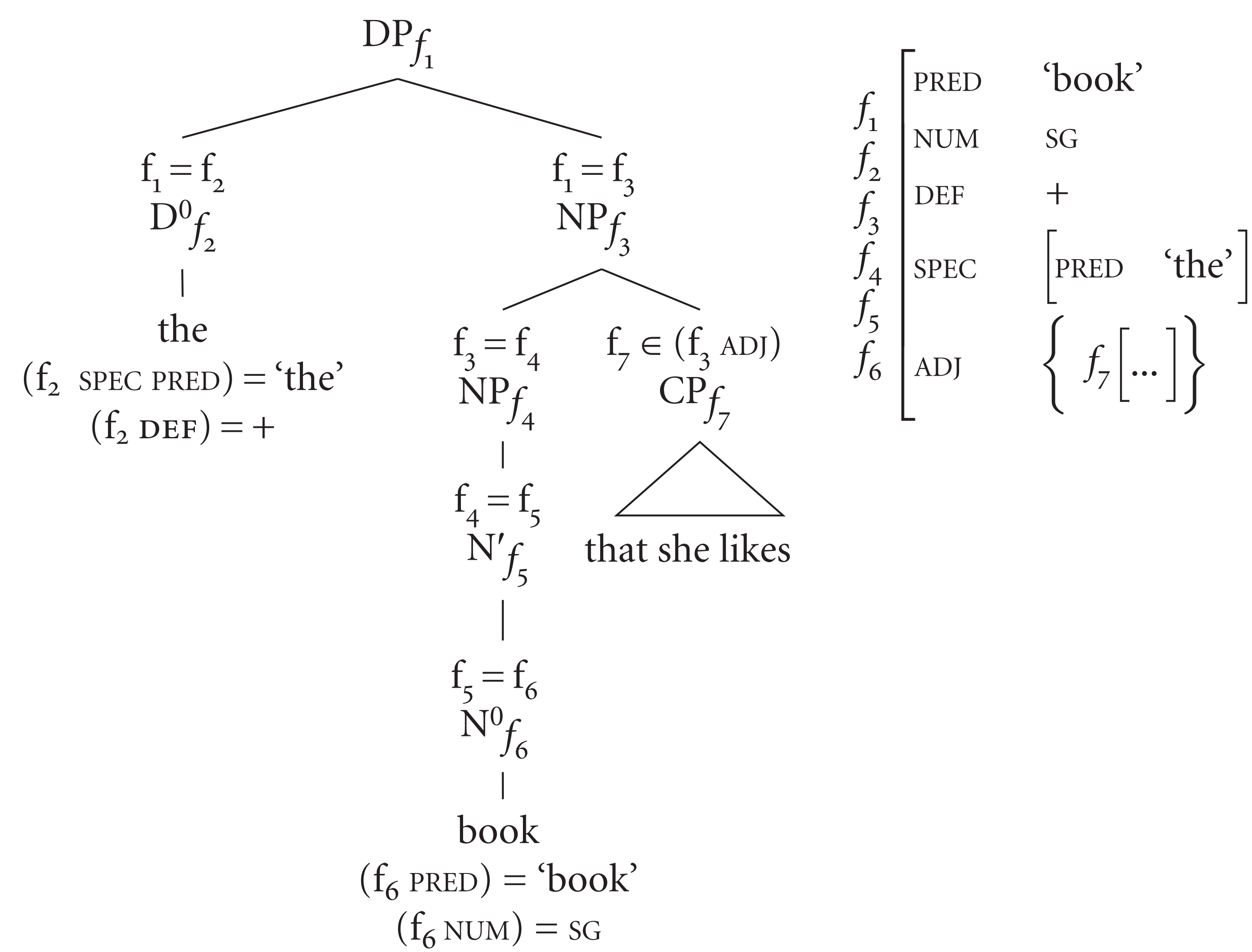
she, D^0 — (↑ PRED) = 'pro'
(↑ PERSON) = 3
(↑ NUMBER) = SINGULAR
(↑ GENDER) = FEMININE

likes, V^0 — (↑ PRED) = 'like⟨SUBJ,OBJ⟩'
(↑ TENSE) = PRESENT
(↑ SUBJECT PERSON) = 3
(↑ SUBJECT NUMBER) = SINGULAR

the, D^0 — (↑ SPEC PRED) = 'the'
(↑ DEFINITE) = +

book, N^0 — (↑ PRED) = 'book'
(↑ NUMBER) = SINGULAR

C-structure with lexical information and instantiated f-structure



Example:
the book that she likes

Penultimate f-structure

$$
b\begin{bmatrix}
\textsc{pred} & \text{‘book’} \\
\textsc{num} & \textsc{sg} \\
\textsc{def} & + \\
\textsc{spec} & \begin{bmatrix} \textsc{pred} & \text{‘the’} \end{bmatrix} \\
\textsc{adj} & \left\{ l\begin{bmatrix}
\textsc{pred} & \text{‘like’} \\
\textsc{subj} & p\begin{bmatrix}
\textsc{pred} & \text{‘pro’} \\
\textsc{pers} & 3 \\
\textsc{num} & \textsc{sg} \\
\textsc{gend} & \textsc{fem}
\end{bmatrix} \\
\textsc{tense} & \textsc{pres}
\end{bmatrix} \right\}
\end{bmatrix}
$$

Notes:

1. I often adopt the practice of labelling f-structures mnemonically with the first letter of the PRED value.
2. I often leave the subcategorization out of the PRED. (There’s a principled reason for this; we can discuss it in question time.)

General wellformedness constraints on f-structures

- *Completeness*
  All subcategorized grammatical functions in a PRED feature must be present in the f-structure.
- *Coherence*
  All grammatical functions that are present in the f-structure must be subcategorized by a PRED feature.
- *Consistency* (a.k.a. *Uniqueness*)
  Each f-structure attribute has one value.

Example: Violations of *Completeness*, *Coherence*, *Consistency*

Completeness

$$\begin{bmatrix} \text{PRED} & \text{`like}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \quad \end{bmatrix} \end{bmatrix}$$

Coherence

$$\begin{bmatrix} \text{PRED} & \text{`like}\langle\text{SUBJ, OBJ}\rangle\text{'} \\ \text{SUBJ} & \begin{bmatrix} \quad \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \quad \end{bmatrix} \\ \text{OBL} & \begin{bmatrix} \quad \end{bmatrix} \end{bmatrix}$$

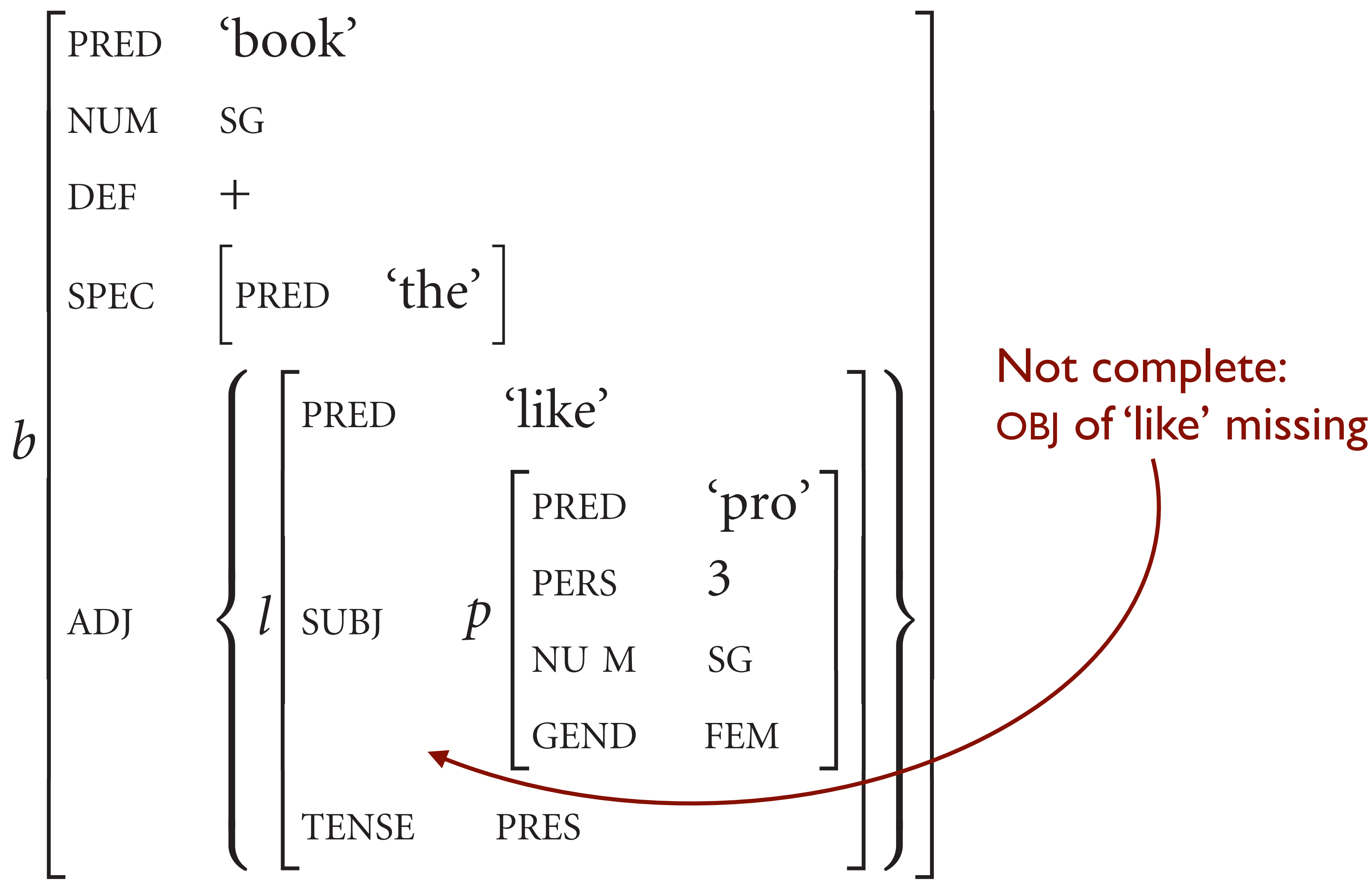
Consistency

$$\begin{bmatrix} \text{NUM} & \text{SG} \\ \text{NUM} & \text{PL} \end{bmatrix}$$

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`hello'} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{`world'} \end{bmatrix} \end{bmatrix}$$

Example:
the book that she likes

Penultimate f-structure



Unbounded dependencies

- *Extended Coherence Condition*
  An UNBOUNDED DEPENDENCY FUNCTION (UDF) must be linked to the semantic predicate argument structure of the sentence in which it occurs, either by functionally or by anaphorically binding an argument.

$$(\uparrow \text{UDFPATH}) = (\uparrow \text{COMP}^* \ \text{GF})$$

(1) Who did you see?

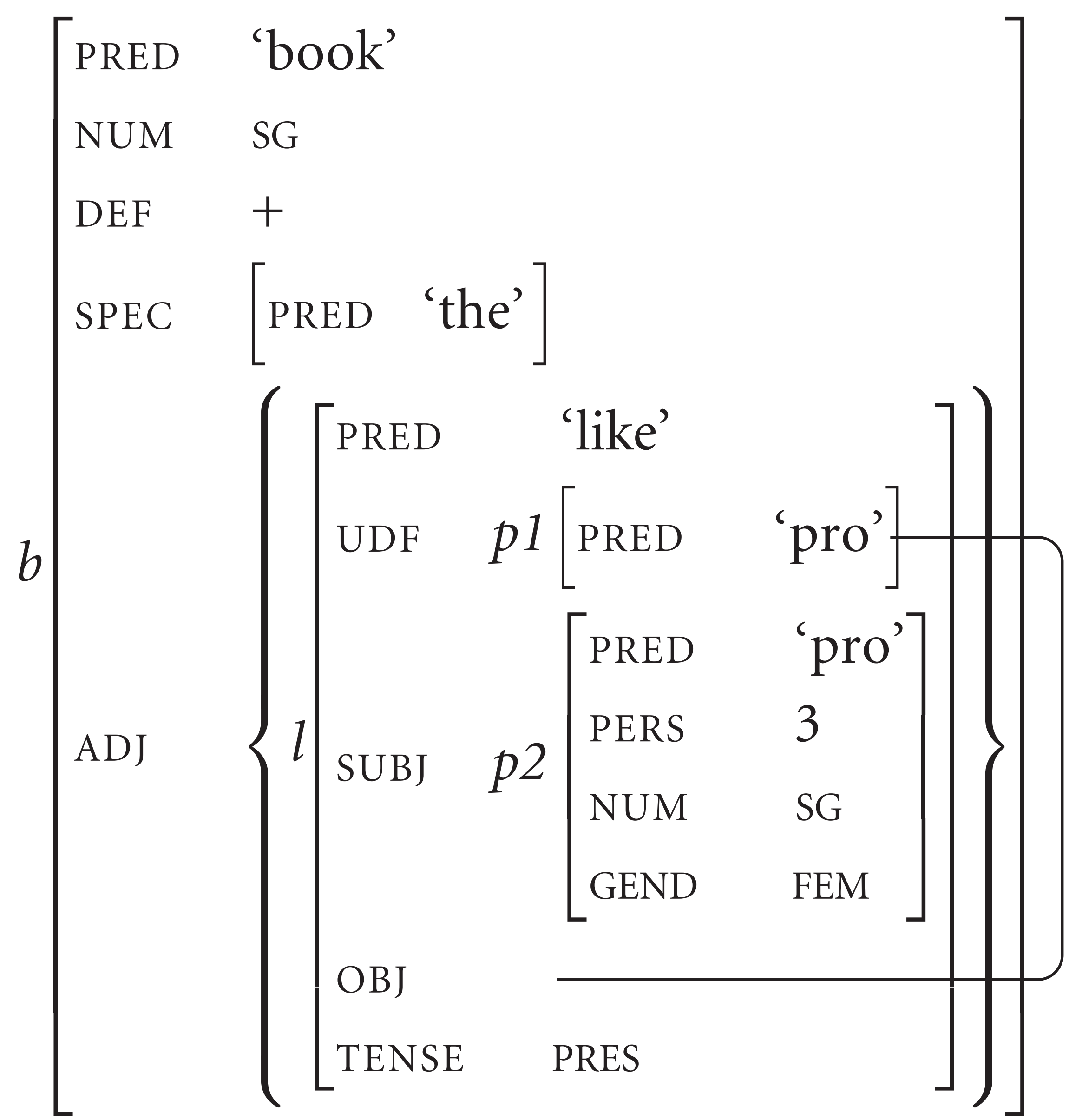
(2) Who did Kim say that you saw?

(3) Who did Kim claim that Sandy alleged that you saw?

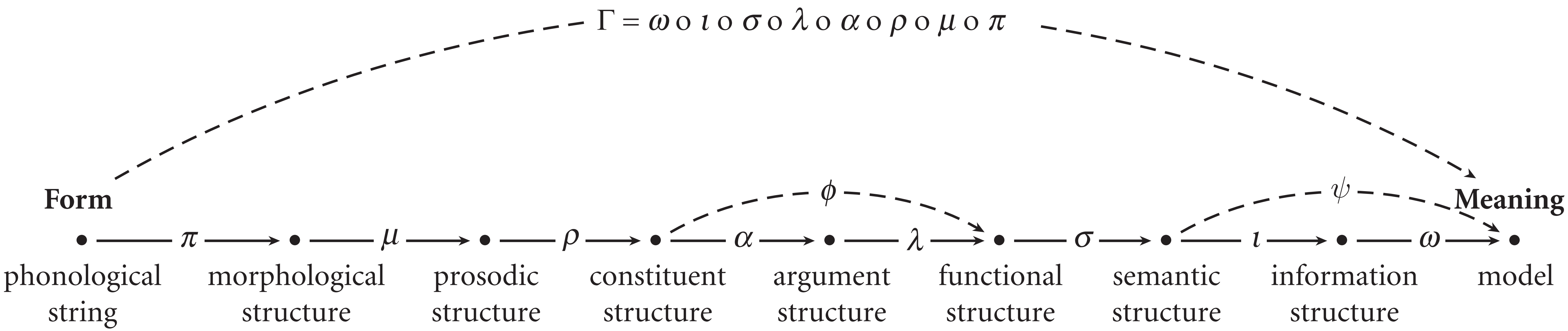
$$\text{CP} \longrightarrow \left\{ \begin{array}{c} \text{XP} \\ (\uparrow \text{UDF}) = \downarrow \\ (\uparrow \text{UDF}) = (\uparrow \text{UDFPATH}) \end{array} \middle| \begin{array}{c} \epsilon \\ (\uparrow \text{UDF PRED}) = \text{'pro'} \\ (\uparrow \text{UDF}) = (\uparrow \text{UDFPATH}) \end{array} \right\} \begin{array}{c} \text{C}' \\ \uparrow = \downarrow \\ \end{array}$$

Example:
the book that she likes

Final f-structure



Language as a form–meaning mapping: The Correspondence Architecture



Templates:
Generalizations over named descriptions

- An LFG template is nothing more than a named functional description (i.e., a set of equations that describe linguistic structures).
- For any LFG grammar defined in terms of templates, we could construct a completely equivalent grammar which does not use templates, simply by replacing each template with the description that it abbreviates.
  - The same grammatical descriptions would be associated with words and phrases in each of the two grammars, and the grammars would produce the same c-structures and f-structures for the words and phrases of the language.
  - However, the grammar without templates would lack the means of expressing generalizations across lexical entries and grammar rules which templates make available.
- In sum:
  - Templates name LFG grammatical descriptions such that the same description can be used in different parts of the grammar.
  - The semantics of template calling/invocation is just substitution: The grammatical description that the template names is substituted where the template is called.

Example: Present tense intransitive verbs

$3\text{SG} = \quad (\uparrow \text{SUBJ NUM}) = 3$
$\qquad\qquad (\uparrow \text{SUBJ PERS}) = \text{SG}$

laughs V $(\uparrow \text{PRED}) = \text{`laugh}\langle\text{SUBJ}\rangle\text{'}$
$(\uparrow \text{TENSE}) = \text{PRESENT}$
$@3\text{SG}$

laugh V $(\uparrow \text{PRED}) = \text{`laugh}\langle\text{SUBJ}\rangle\text{'}$
$\{ (\uparrow \text{TENSE}) = \text{PRESENT}$
$\neg@3\text{SG} \mid$
$\neg(\uparrow \text{TENSE}) \}$

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

$\text{BARE-V} = \quad \{ @\text{TENSE(PRESENT)}$
$\neg@3\text{SG} \mid$
$\neg(\uparrow \text{TENSE}) \}$

laughs V $@\text{INTRANSITIVE(laugh)}$
$@\text{TENSE(PRESENT)}$
$@3\text{SG}$

laugh V $@\text{INTRANSITIVE(laugh)}$
$@\text{BARE-V}$

Templates: Lexical entries and phrasal configurations

- Templates can be associated with lexical entries, but as they are just named descriptions, they can also be associated with c-structure configurations by calling the template in the c-structure rule.
- Example: English relative clauses have bare and non-bare alternatives

(1) the book Kim read

(2) the book which Kim read

- Suppose we have a template REL that captures the relativizing information.
  - $\text{REL} = \lambda Q.\lambda P.\lambda x.P(x) \wedge Q(x) : clause \multimap nominal \multimap nominal$
- This template can now be associated with a relative pronoun or with the rule for a bare/reduced relative clause

(3) $which$ D @REL

(4) $\text{CP} \longrightarrow \begin{pmatrix} \text{RelP} \\ \ldots \end{pmatrix} \quad \begin{matrix} \text{C}' \\ (\ @\text{REL}\ ) \end{matrix}$

Template hierarchies & type hierarchies

- As we've seen, template definitions may contain reference to other templates.
- This effectively creates a hierarchy of templates, similar to the type hierarchies of Head-Driven Phrase Structure Grammar.
- Differences:

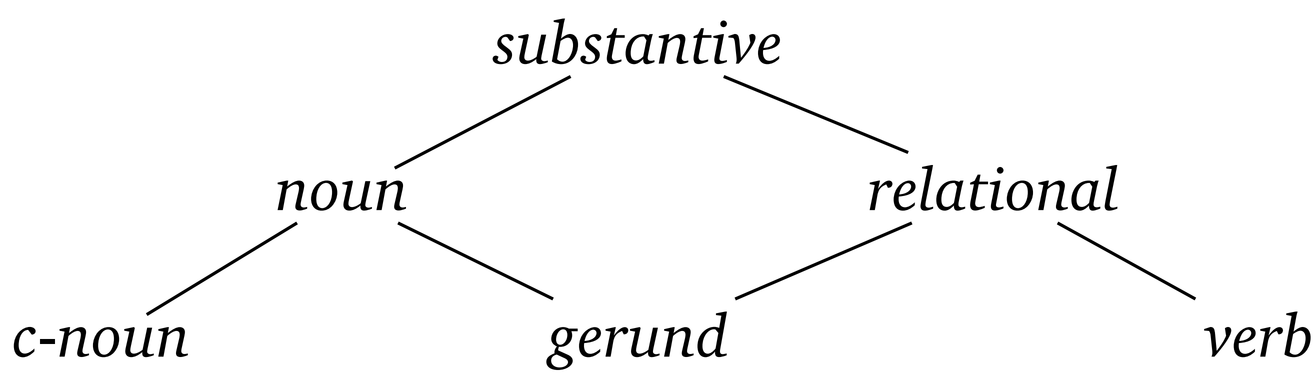
1. Type hierarchies represent relations between structures, whereas template hierarchies represent relations between descriptions of structures: Templates do not appear in the actual structures of the theory, but only in descriptions that the structures must satisfy.

2. Type hierarchies represent *inheritance* in an *and/or* semilattice.

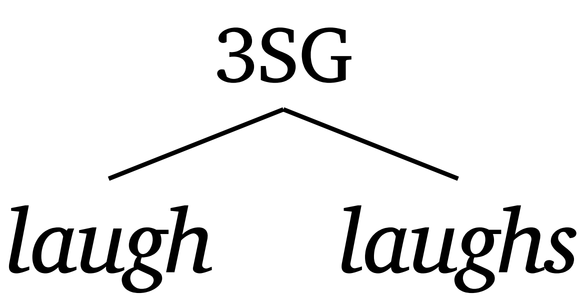
    - The daughters of a type represent disjoint subtypes (*or*).
    - Multiple mothers for a type represent conjoined super- types (and).

Template hierarchies & type hierarchies

Type hierarchy: Inheritance



Template hierarchy: Inclusion



laughs $(\uparrow \text{PRED}) = \text{`laugh}\langle\text{SUBJ}\rangle\text{'}$
@3SG

laugh $(\uparrow \text{PRED}) = \text{`laugh}\langle\text{SUBJ}\rangle\text{'}$
$\neg$@3SG

Glue Semantics

Overview

- Glue Semantics is a method of semantic composition in which a potentially representationally rich meaning language is paired with a very constrained logic of composition that 'glues' pieces of meaning together to obtain larger meanings.
    - You can think of the pieces of meaning like Lego pieces: They can only fit together in certain ways.
- The meaning language is some logic that supports the lambda calculus and has a model-theoretic interpretation.
- The glue logic is (a fragment of) Linear Logic, a logic originally developed for theoretical computer science.
- Linear Logic, and hence the glue logic, is a *resource logic*: All meanings obtained through the syntactic parse must be used exactly once.
    - It's like you have to use all your Lego pieces to build something and, obviously, no piece may be used more than once.

Meaning constructors

- Meaningful linguistic expressions, particularly but not necessarily lexical items, are associated with *meaning constructors* of the following form:

  $$\mathcal{M} : G$$

- The expression on the left is a term from the meaning language. The expression on the right is the associated term from the glue logic. The colon is an uninterpreted pairing symbol.

The logic of implication: Filling in missing pieces

- Linear implication is a cute *lollipop*.
- We can define its logical behaviour in terms of two simple proof rules.

Functional application : Implication elimination (modus ponens)

$$\frac{f : A \multimap B \qquad a : A}{f(a) : B} \multimap_{\mathcal{E}}$$

Functional abstraction : Implication introduction (hypothetical reasoning)

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

Example: *Kim hugged Robin*

- Meaning constructors from lexical entries

Kim	e	$kim : \uparrow_\sigma$
Robin	e	$robin : \uparrow_\sigma$
hugged	$e \rightarrow (e \rightarrow t)$	$\lambda y.\lambda x.hug(x, y) : (\uparrow \text{OBJ})_\sigma \multimap (\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$

- F-structure

$$h\begin{bmatrix} \text{PRED} & \text{`hug}\langle\text{SUBJ}, \text{OBJ}\rangle\text{'} \\ \text{SUBJ} & k\begin{bmatrix}\text{PRED} & \text{`Kim'}\end{bmatrix} \\ \text{OBJ} & r\begin{bmatrix}\text{PRED} & \text{`Robin'}\end{bmatrix} \end{bmatrix}$$

Example:
Kim hugged Robin

- Instantiated meaning constructors

Kim	e	$kim : k_\sigma$
Robin	e	$robin : r_\sigma$
hugged	$e \rightarrow (e \rightarrow t)$	$\lambda y.\lambda x.hug(x,y) : r_\sigma \multimap k_\sigma \multimap h_\sigma$

- Proof

$$\dfrac{\dfrac{\dfrac{\lambda y.\lambda x.hug(x,y) : r_\sigma \multimap k_\sigma \multimap h_\sigma \quad robin : r_\sigma}{(\lambda y.\lambda x.hug(x,y))(robin) : k_\sigma \multimap h_\sigma}\multimap_{\mathcal{E}}}{\lambda x.hug(x,robin) : k_\sigma \multimap h_\sigma}\Rightarrow_\beta \quad kim : k_\sigma}{\dfrac{(\lambda x.hug(x,robin))(kim) : h_\sigma}{hug(kim,robin) : h_\sigma}\Rightarrow_\beta}\multimap_{\mathcal{E}}$$

Flexible Composition

Overview

- *Flexible Composition* is the name I've given to a theory of semantic composition, or more specifically how compositional meanings are packaged, that my collaborators (Mary Dalrymple, Gianluca Giorgolo, and Ida Toivonen) and I have been developing for a number of years.
- Basic intuition: Templates can be used to factor out common meanings across lexical items, e.g. argument structure regularities, and across phrasal configurations, e.g. so-called "constructional meanings" (but without actual constructions).

Some features of Flexible Composition

1. The representation of core semantic information, such that the same lexical entry can be involved in a number of valency realizations
   - (1) The hamster ate a sheet of newspaper this morning.
   - (2) The hamster ate this morning.
   - (3) The hamster ate its way through a sheet of newspaper this morning.
2. The representation of missing/understood arguments
3. The representation of additional/derived arguments
   - (4) *The performer laughed the children.
   - (5) The performer laughed a funny laugh.
4. The possibility of associating meanings with syntactic configurations
   - (6) The performer sang the children a song.
5. Templates as generalizations over lexically encoded meaning
6. Templates as the locus of specification of meanings which can be associated with lexical entries or c-structure rules

Some templates for Flexible Composition

AGENT =
@ARG1
$\lambda P \lambda x \lambda e. P(e) \wedge agent(e) = x :$
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{ARG}_1) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

PATIENT =
@ARG2
$\lambda P \lambda x \lambda e. P(e) \wedge patient(e) = x :$
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{ARG}_2) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

AGENT-PATIENT =
@AGENT
@PATIENT

PASSIVE =
$(\uparrow \text{VOICE}) = \text{PASSIVE}$
@ADDMAP(PLUSR,ARG_1)
$(\ \lambda P \exists x.[P(x)] : [(\uparrow_\sigma \text{ARG}_1) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma\ )$

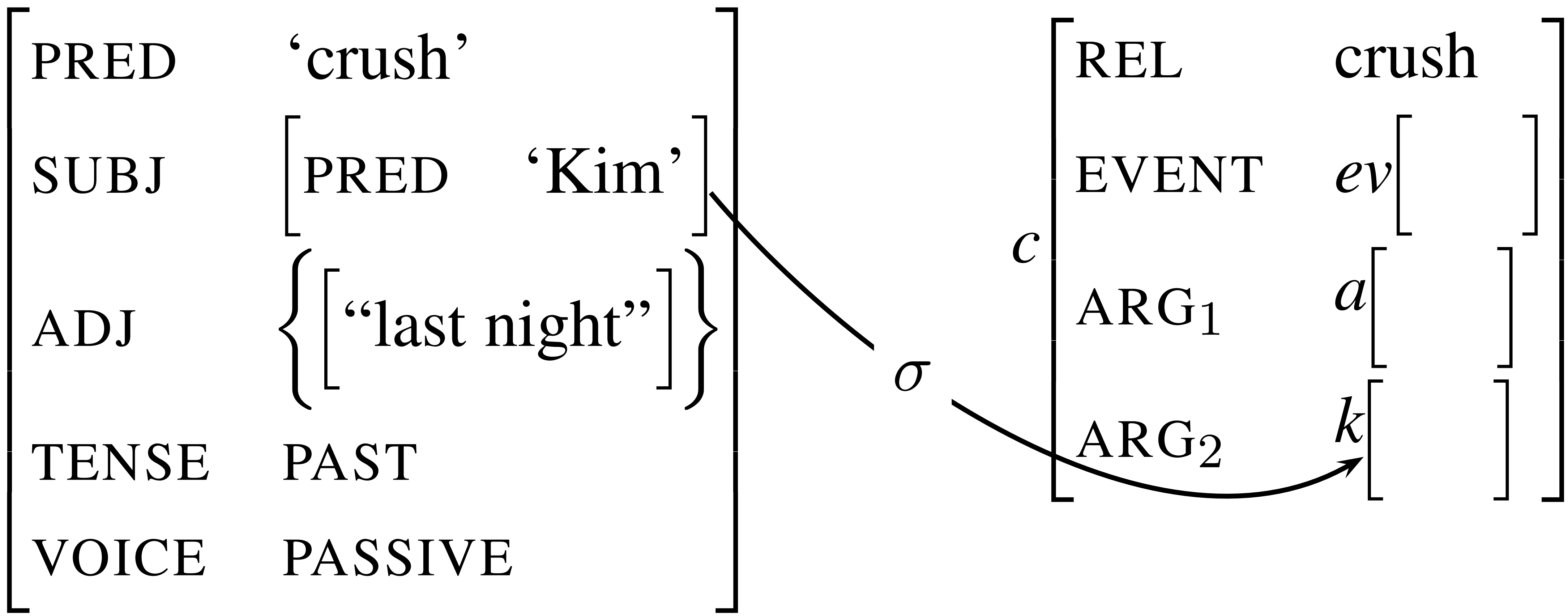
COGNATEOBJECT
$\lambda x \lambda P \lambda e. P(e) \wedge x = \varepsilon(e) :$
$(\uparrow \text{OBJ})_\sigma \multimap [(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

BENEFACTIVE =
@ARG3
$\lambda x \lambda y \lambda P \lambda e. P(y)(e) \wedge beneficiary(e) = x :$
$(\uparrow_\sigma \text{ARG}_2) \multimap (\uparrow_\sigma \text{ARG}_3) \multimap [(\uparrow_\sigma \text{ARG}_2) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

UNDERSTOODOBJECT =
$\lambda P \exists x.[P(x)] : [(\uparrow_\sigma \text{ARG}_2) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

PAST =
$(\uparrow \text{TENSE}) = \text{PAST}$
$\lambda P \exists e.[P(e) \wedge past(e)] :$
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

Example:
Kim was crushed last night



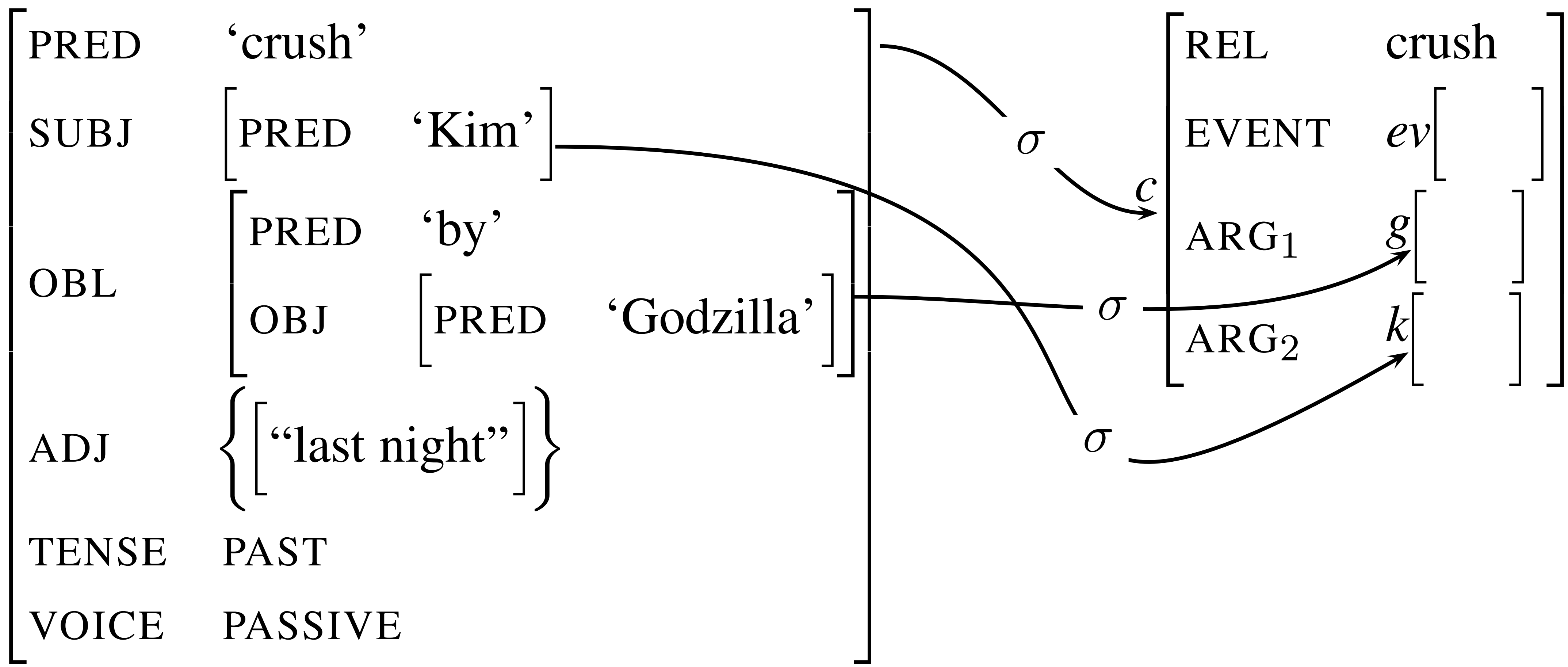
crushed V $(\uparrow \text{PRED}) = \text{`crush'}$

@AGENT-PATIENT

{ @PAST | @PASSIVE }

$\lambda e.crush(e) : (\uparrow_\sigma \text{ EVENT}) \multimap \uparrow_\sigma$

Example:
Kim was crushed by Godzilla last night



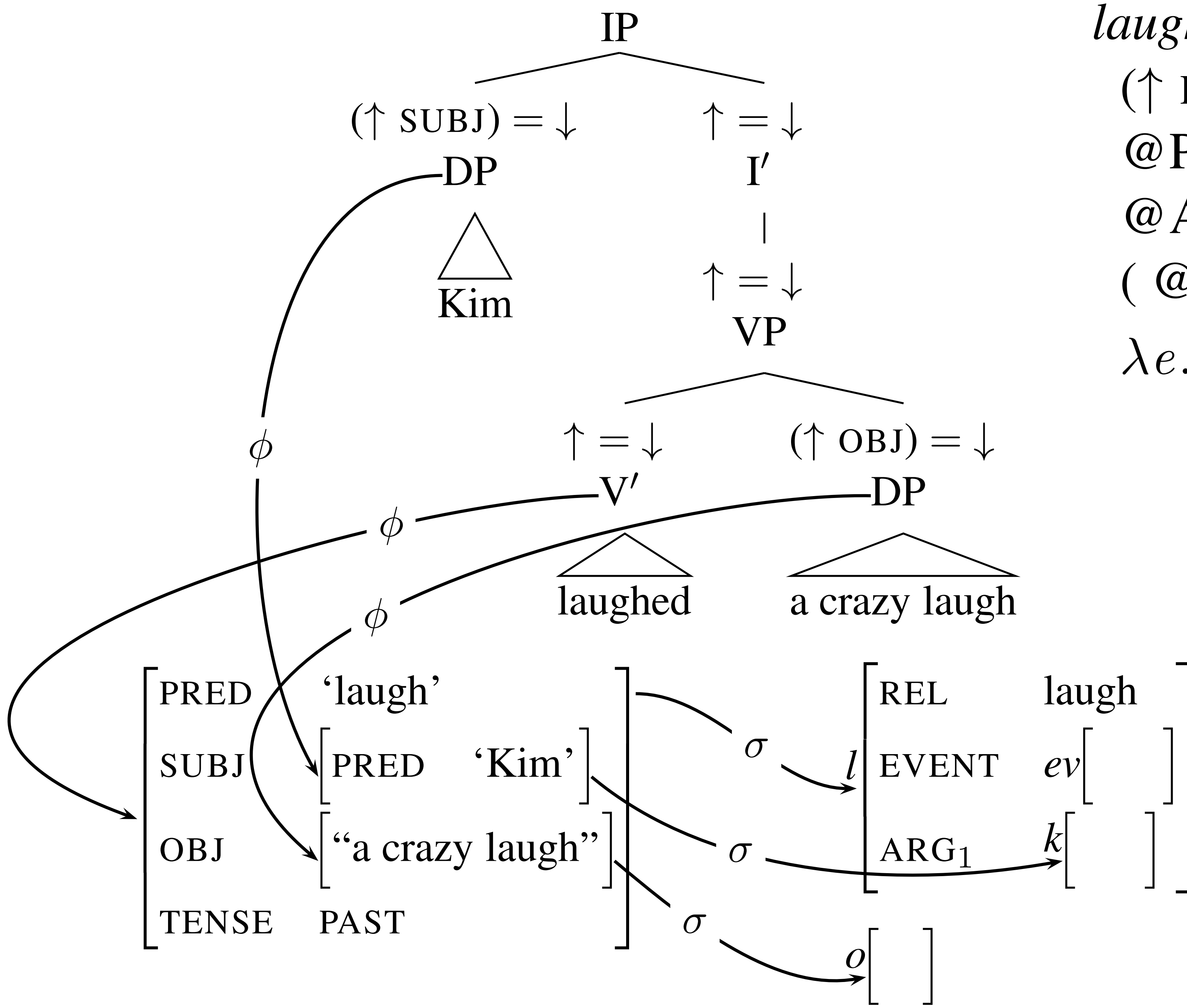
by P $(\uparrow \text{PRED}) = \text{'by'}$

$((\text{OBL} \uparrow)\ \text{VOICE}) =_c \text{PASSIVE}$

$(\uparrow \text{OBJ})_\sigma = ((\text{OBL} \uparrow)_\sigma\ \text{ARG}_1)$

$\lambda x \lambda P.[P(x)] : (\uparrow_\sigma \text{ARG}_1) \multimap [\uparrow_\sigma \multimap (\text{OBL} \uparrow)_\sigma] \multimap (\text{OBL} \uparrow)_\sigma$

Example:
Kim laughed a crazy laugh



laughed V

(↑ PRED) = ‘laugh’

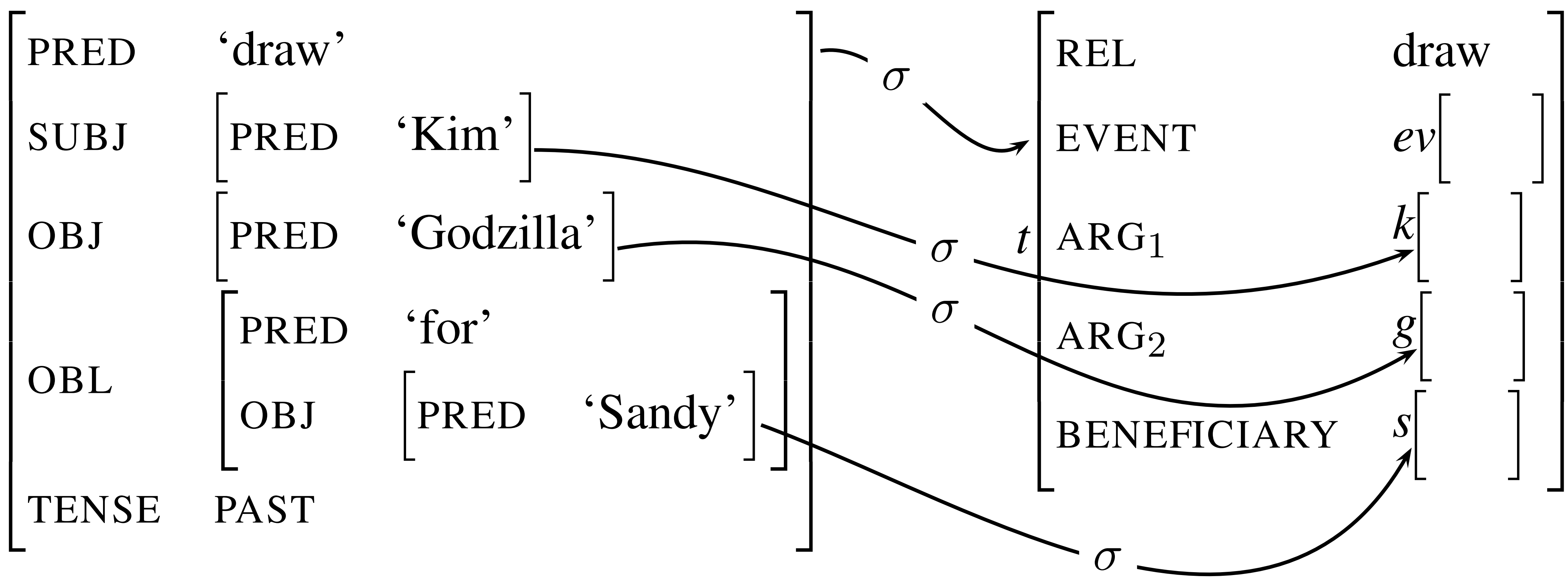
@PAST

@AGENT

(@COGNATEOBJECT)

$\lambda e.laugh(e) : (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

Example:
Kim drew Godzilla for Sandy



drew V

$(\uparrow \text{PRED}) = \text{`draw'}$

@PAST

@AGENT-PATIENT

$\lambda e.draw(e) : (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

for P

$(\uparrow \text{PRED}) = \text{`for'}$

$(\uparrow \text{OBJ})_\sigma = ((\text{OBL} \uparrow)_\sigma \text{ BENEFICIARY})$

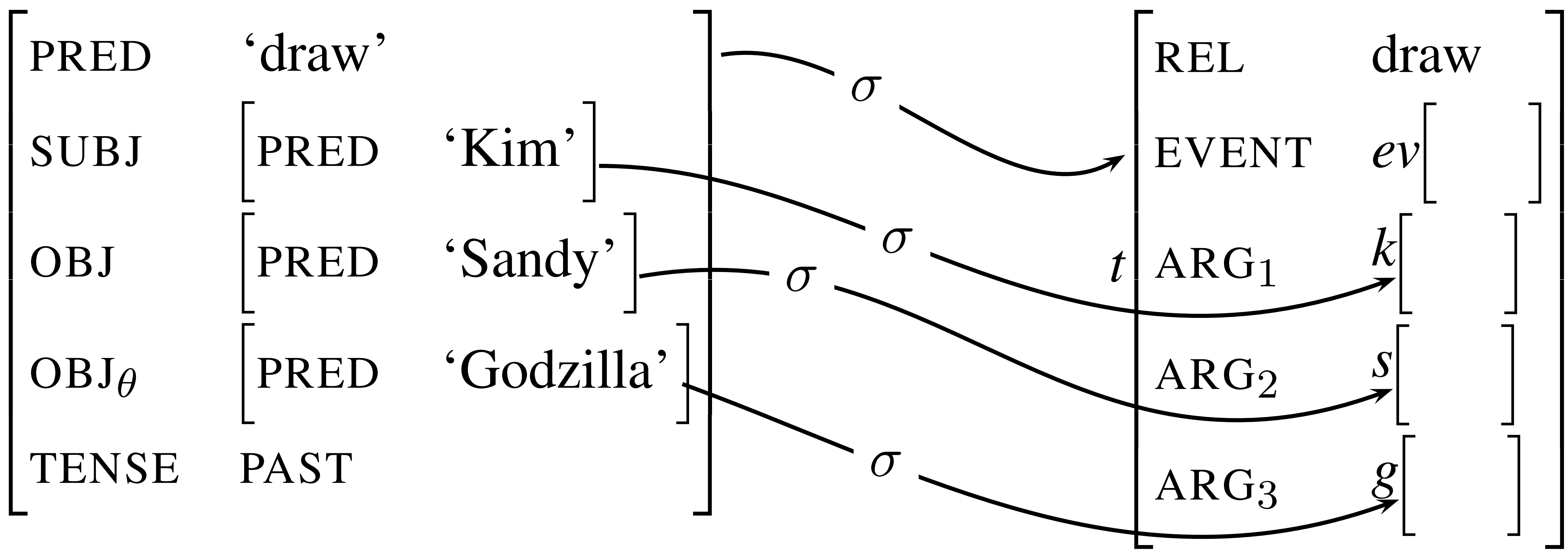
$\lambda y \lambda P \lambda e.[P(e) \wedge \mathit{beneficiary}(e) = y] :$

$(\uparrow_\sigma \text{BENEFICIARY}) \multimap$

$[((\text{OBL} \uparrow)_\sigma \text{ EVENT}) \multimap (\text{OBL} \uparrow)_\sigma] \multimap$

$((\text{OBL} \uparrow)_\sigma \text{ EVENT}) \multimap (\text{OBL} \uparrow)_\sigma$

Example:
Kim drew Sandy Godzilla



$V' \rightarrow$	V	DP	DP
	$\uparrow = \downarrow$	$(\uparrow \text{OBJ}) = \downarrow$	$(\uparrow \text{OBJ}_\theta) = \downarrow$
	(@BENEFACTIVE)		

Note: It is a normal assumption in LFG that elements of c-structure rules are optional. This rule therefore allows the passive verb to be inserted under V. (It would also have to have a PP node added for the oblique, but this is a trivial modification.)

Proof:
Kim was crushed last night

$crush' =$

@Patient
$\lambda P \lambda x \lambda e. P(e) \wedge patient(e) = x :$
$(ev \multimap c) \multimap k \multimap ev \multimap c$

crushed
$\lambda e. crush(e) :$
$ev \multimap c$

$\lambda x \lambda e. crush(e) \wedge patient(e) = x : k \multimap ev \multimap c \qquad [x : k]^1$

$\lambda e. crush(e) \wedge patient(e) = x : ev \multimap c$

@Agent
$\lambda P \lambda y \lambda e. P(e) \wedge agent(e) = y :$
$(ev \multimap c) \multimap a \multimap ev \multimap c$

$\lambda y \lambda e. crush(e) \wedge patient(e) = x \wedge agent(e) = y : a \multimap ev \multimap c$

$\lambda x \lambda y \lambda e. crush(e) \wedge patient(e) = x \wedge agent(e) = y : k \multimap a \multimap ev \multimap c \quad \multimap_{\mathcal{I},1}$

$crush'$

Kim
$kim :$
k

$crush'(kim) : a \multimap ev \multimap c \qquad [y : a]^2$

$crush'(kim)(y) : ev \multimap c \qquad [e' : ev]^3$

$crush'(kim)(y)(e') : c$

$\lambda y. crush'(kim)(y)(e') : a \multimap c \quad \multimap_{\mathcal{I},2}$

@Passive
$\lambda P \exists x.[P(x)] :$
$(a \multimap c) \multimap c$

$\exists x.[crush'(kim)(x)(e')] : c$

$\lambda e' \exists x.[crush'(kim)(x)(e')] : ev \multimap c \quad \multimap_{\mathcal{I},3}$

last night
$\lambda P \lambda e''.[P(e'') \wedge last.night(e'')] :$
$(ev \multimap c) \multimap (ev \multimap c)$

$\lambda e'' \exists x.[crush'(kim)(x)(e'') \wedge last.night(e'')] : ev \multimap c$

was
$\lambda P \exists e.[P(e) \wedge past(e)] :$
$(ev \multimap c) \multimap c$

$\exists e \exists x.[crush'(kim)(x)(e) \wedge last.night(e) \wedge past(e)] : c$

$\exists e \exists x.[crush(e) \wedge patient(e) = kim \wedge agent(e) = x \wedge last.night(e) \wedge past(e)] : c \quad \Rightarrow_{\beta}$

What just happened?

- Overview of Lexical-Functional Grammar
- Overview of Glue Semantics
- Introduction to Flexible Composition as an approach to argument structure

Thank you

Some key references

Asudeh, Ash. 2012. *The Logic of Pronominal Resumption*. Oxford: Oxford University Press.

Asudeh, Ash, Mary Dalrymple, and Ida Toivonen. 2008. Constructions with Lexical Integrity: Templates as the Lexicon-Syntax Interface. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG08 Conference*, 68–88. Stanford, CA: CSLI Publications.

Asudeh, Ash, Mary Dalrymple, and Ida Toivonen. 2013. Constructions with Lexical Integrity. *Journal of Language Modelling* 1(1): 1–54.

Asudeh, Ash, and Gianluca Giorgolo. 2012. Flexible Composition for Optional and Derived Arguments. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG12 Conference*, 64–84. Stanford, CA: CSLI Publications.

Asudeh, Ash, Gianluca Giorgolo, and Ida Toivonen. 2014. Meaning and Valency. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG14 Conference*. Stanford, CA: CSLI Publications.

Asudeh, Ash, and Ida Toivonen. 2014. *With* Lexical Integrity. *Theoretical Linguistics* 40(1–2): 175–186.

Asudeh, Ash, and Ida Toivonen. 2015. Lexical-Functional Grammar. In Bernd Heine and Heiko Narrog, eds., The Oxford Handbook of Linguistic Analysis, 373–406. Oxford: Oxford Uni- versity Press, 2nd edn.

Bresnan, Joan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2016. *Lexical-Functional Syntax*. Malden, MA: Wiley-Blackwell, 2nd edn.

Dalrymple, Mary, ed. 1999. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. Cambridge, MA: MIT Press.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*. San Diego, CA: Academic Press.

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, eds. 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications.

Kaplan, Ronald M., and Joan Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Joan Bresnan, ed., *The Mental Representation of Grammatical Relations*, 173–281. Cambridge, MA: MIT Press. Reprinted in Dalrymple et al. (1995: 29–135).