Flexible Composition for Optional and Derived Arguments[*]

Ash Asudeh
ash.asudeh@ling-phil.ox.ac.uk
Carleton University &
University of Oxford

Gianluca Giorgolo
Gianluca_Giorgolo@carleton.ca
Carleton University

June 29, 2012
LFG 2012 · Bali

1 Introduction

- There is broad **agreement** in linguistic theory that some distinction must be made between arguments and adjuncts.
- There is broad **disagreement** in linguistic theory as to how the distinction is to be represented and how borderline cases should be captured.
  - ⊙ Representational options:
    - ○ Structural position (P&P, LFG)
    - ○ Distinguished lists (HPSG)
    - ○ Grammatical functions (LFG)
  - ⊙ Borderline cases:
    - ○ Optional arguments

      (1) Any child of Kim's is unfortunately likely to drink __.

      (2) Kim ate __ at 10 o'clock.
    - ○ Derived arguments (Needham and Toivonen 2011)

(3)	The hole was plugged <u>by Kim</u>.	Passive *by*-phrase
(4)	Kim plugged the hole <u>with a cork</u>.	Instrumental
(5)	<u>Kim's</u> obstruction of the hole	Possessive phrase in event nominal
(6)	Kim plugged the hole <u>for them</u>.	Benefactive
(7)	The hole crawled <u>with bugs</u>.	Displaced theme
(8)	It seemed <u>to Kim</u> like the hole could be plugged.	Experiencer
(9)	The bugs crawled <u>from the hole</u>.	Directional

- The aim of this paper is to develop an analysis of the borderline cases, concentrating on three cases, using standard tools from LFG and Glue Semantics.

[*]This research is supported by an Early Researcher Award from the Ontario Ministry of Research and Innovation (Asudeh), NSERC Discovery Grant #371969 (Asudeh), and SSHRC Standard Research Grant #410-2010-1841 (Toivonen).

2 Claims and Goals

- **Main claim:** A simple but insightful analysis of optional and derived arguments at the syntax–semantics interface can be provided based on established features of Lexical-Functional Grammar with Glue Semantics.
    1. **Optionality**, offered by the regular language of LFG's functional descriptions in lexical entries (Kaplan and Bresnan 1982, Dalrymple 2001).
    2. **Flexible semantic composition**, offered by the commutative glue logic of Glue Semantics (Dalrymple 1999, 2001, Asudeh 2012).
    3. **Resource-sensitive semantic composition**, again offered by the glue logic.
    4. **Generalizations over descriptions**, offered by templates (Dalrymple et al. 2004, Asudeh et al. 2008, Asudeh 2012).
- **An assumption/talking point:** Subcategorization of grammatical functions other than expletives is not represented at f-structure, but is rather captured by resource-sensitive semantic composition (Kuhn 2001, Asudeh 2012).
- **Main questions:**
    1. What are the implications of optional and derived arguments for the mapping from syntax to semantics?
    2. How can lexical generalizations about optional and derived arguments be best captured?
- **Goal of this talk:** To attempt initial answers to these questions by looking at three cases.
    1. Optional objects of semantically transitive verbs (e.g., *drink*, *eat*)
    2. Passive *by*-phrases
    3. Instrumental *with*-phrases

3 Overview

4 Background

4.1 The Correspondence Architecture

- The grammatical architecture of LFG posits that different kinds of linguistic information are modelled by distinct data structures, all of which are present simultaneously.

- Structures are related by functions, called correspondence or projection functions, which map elements of one structure to elements of another.

- This architecture is a generalization of the architecture of Kaplan and Bresnan (1982) and is called the *Parallel Projection Architecture* or *Correspondence Architecture* (Kaplan 1987, 1989, Halvorsen and Kaplan 1988, Asudeh 2006, 2012, Asudeh and Toivonen 2009).

- The architecture that we assume is shown in Figure 1.

  ⊙ We follow Butt et al. (1997) in treating argument-structure as a level interpolated between constituent-structure and functional-structure, such that the correspondence function ϕ can be understood as the composition of the correspondence functions α and λ.

  ⊙ However, for reasons that will become apparent shortly, we also postulate a direct correspondence function θ from argument-structure to semantic-structure. The function θ is not the composition of λ and σ.

  ⊙ We will revisit questions surrounding the necessity of θ at the end of the talk.

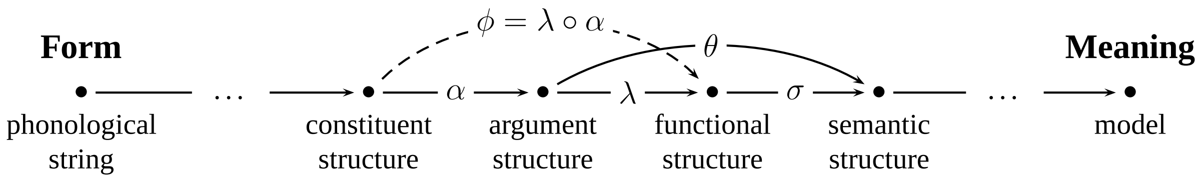


Figure 1: Detail of the Correspondence Architecture (Asudeh 2012)

4.2 Glue Semantics

- Glue Semantics (Dalrymple 1999, 2001, Asudeh 2004, 2005a,b, 2012, Lev 2007, Kokkonidis 2008) is a theory of semantic composition and the syntax–semantics interface.

- Glue *meaning constructors* are obtained from lexical items instantiated in particular syntactic structures.

  (10) $\mathcal{M} : G$

  $\mathcal{M}$ is a term from some representation of meaning, a *meaning language*, and G is a term of the Glue logic that sticks meanings together, i.e. performs composition. The colon is an uninterpreted pairing symbol.

- Linear logic (Girard 1987) serves as the Glue logic (Dalrymple et al. 1993, 1999a,b).

- The meaning constructors are used as premises in a (linear logic) proof that consumes the lexical premises to produce a sentential meaning.
- A successful Glue proof for a sentence terminates in a meaning constructor of type t:

(11) $\Gamma \vdash \mathcal{M} : G_t$

- Alternative (normalized) derivations from the same set of premises $\rightarrow$ semantic ambiguity (e.g., scope)
- Linear logic is a *resource logic*: each premise in a valid proof must be used exactly once.
  - $\odot$ Composition in Glue Semantics is therefore *resource sensitive* (Dalrymple et al. 1993, Asudeh 2005a,b, 2012).
- As discussed in detail by Dalrymple et al. (1999a), Glue Semantics is essentially a type-logical theory and is thus related to type-logical approaches to Categorial Grammar (Morrill 1994, 2011, Moortgat 1997, Carpenter 1997, Jäger 2005).
- The key difference between Glue and Categorial Grammar concerns grammatical architecture, particularly the conception of the syntax–semantics interface (Asudeh 2004, 2005b, 2006). Glue Semantics posits a strict separation between syntax and semantics, such that there is a syntax that is separate from the syntax of semantic composition. Categorial Grmamar rejects the separation of syntax from semantic composition.
- We assume a small, rather weak fragment of linear logic, multiplicative intuitionistic linear logic (MILL; Asudeh 2004, 2005b).
- Three proof rules of this fragment are of particular interest here: elimination for $\otimes$ (multiplicative conjunction) and introduction and elimination for linear implication $\multimap$.

Application : Impl. Elim.

$$\frac{\begin{array}{c}\vdots\\ a : A\end{array} \qquad \begin{array}{c}\vdots\\ f : A \multimap B\end{array}}{f(a) : B}\multimap_{\mathcal{E}}$$

Abstraction : Impl. Intro.

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

Pairwise substitution : Conj. Elim.

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

Figure 2: Linear logic proof rules with Curry-Howard correspondence

(12) Bo chortled.

(13)
$$\frac{\mathit{bo} : b \qquad \mathit{chortle} : b \multimap c}{\mathit{chortle}(\mathit{bo}) : c}\multimap_{\mathcal{E}}$$

- It is straightforward to swap the order or arguments of a function directly in the glue proof, as shown in (14). We therefore just choose a version of the lexically specified function in question that is convenient for the larger proof.

(14)
$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\lambda y \lambda x.f(x,y) : a \multimap b \multimap c \quad [v : a]^{1}}{\lambda x.f(x,v) : b \multimap c} \qquad [u : b]^{2}}{f(u,v) : c}}{\lambda v.f(u,v) : a \multimap c}\ {\multimap_{\mathcal{I},1}}}{\lambda u \lambda v.f(u,v) : b \multimap a \multimap c}\ {\multimap_{\mathcal{I},2}}}{\lambda x \lambda y.f(x,y) : b \multimap a \multimap c}\ {\Rightarrow_{\alpha}}$$

5 The Problem

- The basic intuition behind the argument/adjunct distinction is that arguments are 'semantically necessary' in some way that adjuncts are not.
- As pointed out by Needham and Toivonen (2011), despite the intuitive appeal of this claim, it actually fails spectacularly because many clear adjuncts, such as those involving time and place, are also clearly 'semantically necessary': every event that we refer to linguistically happens at some time, in some place.
- This points to a different understanding of the intuition, which Needham and Toivonen (2011) call 'verb specificity': arguments are 'semantically distinctive' in that they are associated with particular verb classes, such that these are distinguished from other classes. Thus, time and place are generally poor arguments, *because* they are ubiquitous and fail to distinguish between verb classes.
- The semantic function that arguments play is typically tied to their obligatory realization in syntax, with optionality often being taken to be a hallmark of adjuncts.
- However, there are cases of clear arguments, according to any plausible semantic criterion, which are nevertheless syntactically optional, such as the objects of *drink* and *eat* in English.
- Similarly, there are argument-like functions (Needham and Toivonen 2011: 'derived arguments'), such as instrumentals, that distinguish verb classes according to verb specificity, but which seem to always be optional.
- Most solutions to this problem can be characterized as some version of the solution of Bresnan (1978), which proposes two distinct versions of, e.g., the verb *eat*.

(15)

eat:	V,	[___ NP],	NP$_1$ 'eat' NP$_2$
		[___],	($\exists$ y) NP$_1$ 'eat' y

- *The challenge* is to capture the core argument structure of verb classes that display optional or derived arguments in a way that:
  1. Doesn't simply treat the distinct valencies as homonymous, i.e. just accidentally related.
  2. Supports a systematic, formal semantic treatment of optional and derived arguments.
  3. Enables semantic restrictions on optional arguments to be stated.
  4. Captures commonalities between derived arguments and adjuncts

6 Analysis

6.1 Optional Transitives

6.1.1 *Drink*

- The verb *drink* has two 'core participants' (Needham and Toivonen 2011: 404): the drinker and the drink.
- It is thus underlyingly transitive, but can leave its object unexpressed (unlike, e.g., *quaff*).
- As noted by, among others, Fillmore (1986), when the object is unexpressed there are additional inferences associated with the patient; in this case, that the drink is alcoholic.
- Analysis:
    - ⊙ *Drink* is 'semantically transitive', even when it is syntactically intransitive. Its argument structure is therefore dyadic, so we represent an agent and a patient at a-structure.
    - ⊙ There is no OBJECT at f-structure, unless we posit a null pronoun. But there are a number of problems with this. First, English does not in general freely allow topic drop. Second, standard syntactic tests such as pronominalization, ellipsis, and secondary predication do not support an OBJECT.

      (16) a. Kim drank a beer, but it turned out to be Sandy's.
      b. *Kim drank, but it turned out to be Sandy's.

      (17) a. Kim is drinking a beer, and so is Sandy. (strict or sloppy)
      b. Kim is drinking, and so is Sandy. (sloppy only)

      (18) a. Kim drank the whiskey neat.
      b. *Kim drank neat.
    - ⊙ We do not postulate an OBJECT in f-structure, but instead represent at a-structure the argument that would correspond to the object at a-structure, as per the linking theory of Butt et al. (1997) (see also Asudeh et al. 2008: 79–80).
    - ⊙ We map the argument directly from a-structure to s-structure.
    - ⊙ In semantic composition, our analysis simultaneously existentially closes the object argument—capturing the fact that even though the argument is unexpressed, it is still an understood argument—and appropriately restricts the object argument to be alcoholic.
- Example:

(19) Kim drank last night.

(20) *drank* V $(\uparrow \text{PRED}) = \text{'drink'}$
$(\uparrow \text{TENSE}) = \text{PAST}$

$$\lambda y \lambda x \lambda e.drink(e) \ \wedge \ agent(e) = x \ \wedge \ patient(e) = y : \\ (\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$$

$$\left(\begin{array}{l} \lambda P \exists x.[P(x) \ \wedge \ alcoholic(x)] : \\ [(\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap \uparrow_\sigma] \multimap \uparrow_\sigma \end{array} \right)$$

(21) $drink' := \lambda e \lambda x \lambda y.drink(e) \wedge agent(e) = x \wedge patient(e) = y$

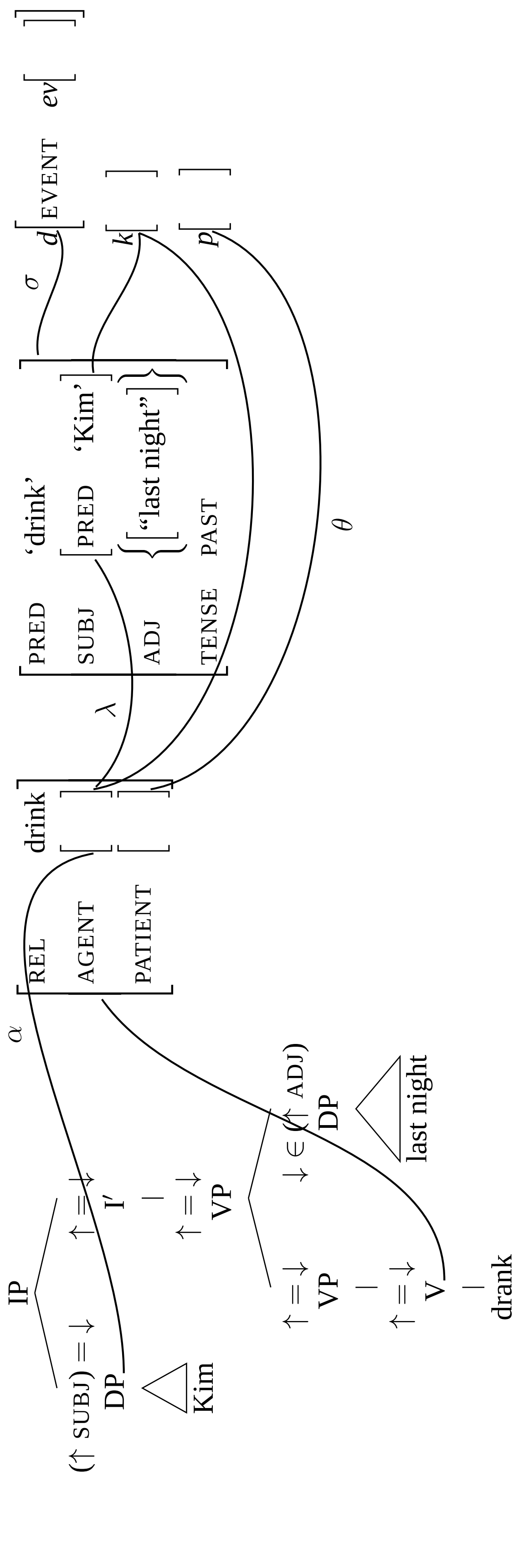


Figure 3: Relevant structures and correspondences for *Kim drank last night*

$$
\dfrac{
\begin{array}{l}\textbf{PAST}\\ \lambda P \exists e.[P(e) \wedge past(e)] :\\ (ev \multimap d) \multimap d\end{array}
\quad
\dfrac{
\begin{array}{l}\textbf{last night}\\ \lambda P \lambda e''.[P(e'') \wedge last.night(e'')] :\\ (ev \multimap d) \multimap (ev \multimap d)\end{array}
\quad
\dfrac{
\dfrac{
\begin{array}{l}\textbf{drank (opt.)}\\ \lambda P \exists x.[P(x) \;\wedge\; alcoholic(x)] :\\ (p \multimap d) \multimap d\end{array}
\quad
\dfrac{
\dfrac{
\begin{array}{l}\textbf{drank}\\ drink' :\\ ev \multimap k \multimap p \multimap d\end{array}
\quad [e' : ev]^1
}{drink'(e') : k \multimap p \multimap d}
\quad
\begin{array}{l}\textbf{Kim}\\ kim :\\ k\end{array}
}{drink'(e')(kim) : p \multimap d}
}{\exists x.[drink'(e')(kim)(x) \wedge alcoholic(x)] : d}
}{\lambda e' \exists x.[drink'(e')(kim)(x) \wedge alcoholic(x)] : ev \multimap d}\multimap_{\mathcal{I},1}
}{\lambda e'' \exists x.[drink'(e'')(kim)(x) \wedge alcoholic(x) \wedge last.night(e'')] : ev \multimap d}
}{
\dfrac{\exists e \exists x.[drink'(e)(kim)(x) \wedge alcoholic(x) \wedge last.night(e) \wedge past(e)] : d}{\exists e \exists x.[drink(e) \wedge agent(e) = kim \wedge patient(e) = x \wedge alcoholic(x) \wedge last.night(e) \wedge past(e)] : d}\Rightarrow_{\beta}
}
$$

Figure 4: Proof for *Kim drank last night.*

6.1.2 *Eat*

- Our second example of a semantically transitive, syntactically intransitive verb is intransitive *eat*.
- *Eat* raises similar issues to *drink*, but the function in the optional premise must take the subject as an argument, since when the object is unexpressed the understanding is that what was eaten was whatever the eater normally eats. That is, it is not enough for the unexpressed argument to be edible, it must be edible *for* the subject. Contrast the following:

(22) My cousin Kim ate with gusto last night.

(23) My cow Kim ate with gusto last night.

- Example:

(24) Kim ate at noon.

(25) *ate* V $(\uparrow \text{PRED}) = \text{‘eat’}$
$(\uparrow \text{TENSE}) = \text{PAST}$

$$\lambda y \lambda x \lambda e.eat(e) \ \wedge \ agent(e) = x \ \wedge \ patient(e) = y : \\ (\widehat{*}_{\alpha}\ \text{PATIENT})_{\theta} \multimap (\uparrow \text{SUBJ})_{\sigma} \multimap (\uparrow_{\sigma} \text{EVENT}) \multimap \uparrow_{\sigma}$$

$$\left(\begin{array}{l} \lambda P \lambda y \exists x.[P(x)(y) \ \wedge \ \mathit{food.for}(x,y)] : \\ [(\widehat{*}_{\alpha}\ \text{PATIENT})_{\theta} \multimap (\uparrow \text{SUBJ})_{\sigma} \multimap \uparrow_{\sigma}] \multimap (\uparrow \text{SUBJ})_{\sigma} \multimap \uparrow_{\sigma} \end{array} \right)$$

(26) $eat' := \lambda e \lambda y \lambda x.eat(e) \wedge agent(e) = x \wedge patient(e) = y$

$$
\dfrac{
\dfrac{
\begin{array}{l}\textbf{PAST}\\ \lambda P\exists e.[P(e) \wedge \mathit{past}(e)] :\\ (ev \multimap e) \multimap e\end{array}
\quad
\dfrac{
\begin{array}{l}\textbf{at noon}\\ \lambda P\lambda e''.[P(e'') \wedge \mathit{at.noon}(e'')] :\\ (ev \multimap e) \multimap (ev \multimap e)\end{array}
\quad
\dfrac{
\dfrac{
\dfrac{
\begin{array}{l}\textbf{ate (opt.)}\\ \lambda P\lambda y\exists x.[P(x)(y) \;\wedge\; \mathit{food.for}(x,y)] :\\ [p \multimap k \multimap e] \multimap k \multimap e\end{array}
\quad
\dfrac{\begin{array}{l}\textbf{ate}\\ \mathit{eat}' :\\ ev \multimap p \multimap k \multimap e\end{array} \quad [e' : ev]^1}{\mathit{eat}'(e) : p \multimap k \multimap e}
}{\lambda y\exists x.[\mathit{eat}'(e')(x)(y) \wedge \mathit{food.for}(x,y)] : k \multimap e}
\quad
\begin{array}{l}\textbf{Kim}\\ \mathit{kim} :\\ k\end{array}
}{\exists x.[\mathit{eat}'(e')(x)(\mathit{kim}) \wedge \mathit{food.for}(x,\mathit{kim})] : e}
}{\lambda e'\exists x.[\mathit{eat}'(e')(x)(\mathit{kim}) \wedge \mathit{food.for}(x,\mathit{kim})] : ev \multimap e} \multimap_{\mathcal{I},1}
}{\lambda e''\exists x.[\mathit{eat}'(e'')(x)(\mathit{kim}) \wedge \mathit{food.for}(x,\mathit{kim}) \wedge \mathit{at.noon}(e'')] : ev \multimap e}
}{\exists e\exists x.[\mathit{eat}'(e)(x)(\mathit{kim}) \wedge \mathit{food.for}(x,\mathit{kim}) \wedge \mathit{at.noon}(e) \wedge \mathit{past}(e)] : e}
}{\exists e\exists x.[\mathit{eat}(e) \wedge \mathit{agent}(e) = \mathit{kim} \wedge \mathit{patient}(e) = x \wedge \mathit{food.for}(x,\mathit{kim}) \wedge \mathit{at.noon}(e) \wedge \mathit{past}(e)] : e} \Rightarrow_{\beta}
$$

Figure 5: Proof for *Kim ate at noon.*

6.2 Passives

- In the absence of a *by*-phrase, the suppressed argument of a passive is not represented at f-structure, but is represented at a-structure.
- A short passive is thus semantically transitive, but syntactically intransitive, much like our previous cases.
- Analysis:
  - ⊙ We again propose a direct mapping from the suppressed argument's a-structure role to s-structure.
  - ⊙ The suppressed argument is optionally existentially closed.
    - ○ Given Glue's resource-sensitive semantic composition, this option only leads to a well-formed proof in the absence of a *by*-phrase, as discussed further below.
    - ○ In the absence of a *by*-phrase, resource sensitivity similarly guarantees that the optional premise contributed by the passive verb *must* be realized, because otherwise the dependency on the highest role (e.g., the AGENT in the example below) is not discharged.
- Example:

(27) Kim was eaten last night.

(28) *eaten* V $(\uparrow \textsc{pred}) = \text{`eat'}$
$(\uparrow \textsc{voice}) = \textsc{passive}$

$\lambda x \lambda y \lambda e.eat(e) \;\wedge\; agent(e) = x \;\wedge\; patient(e) = y :$
$(\widehat{*}_\alpha \; \text{AGENT})_\theta \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

$\big(\; \lambda P \exists x.[P(x)] : [(\widehat{*}_\alpha \; \text{AGENT})_\theta \multimap \uparrow_\sigma] \multimap \uparrow_\sigma \; \big)$

(29) $eat' := \lambda e \lambda y \lambda x.eat(e) \wedge agent(e) = x \wedge patient(e) = y$

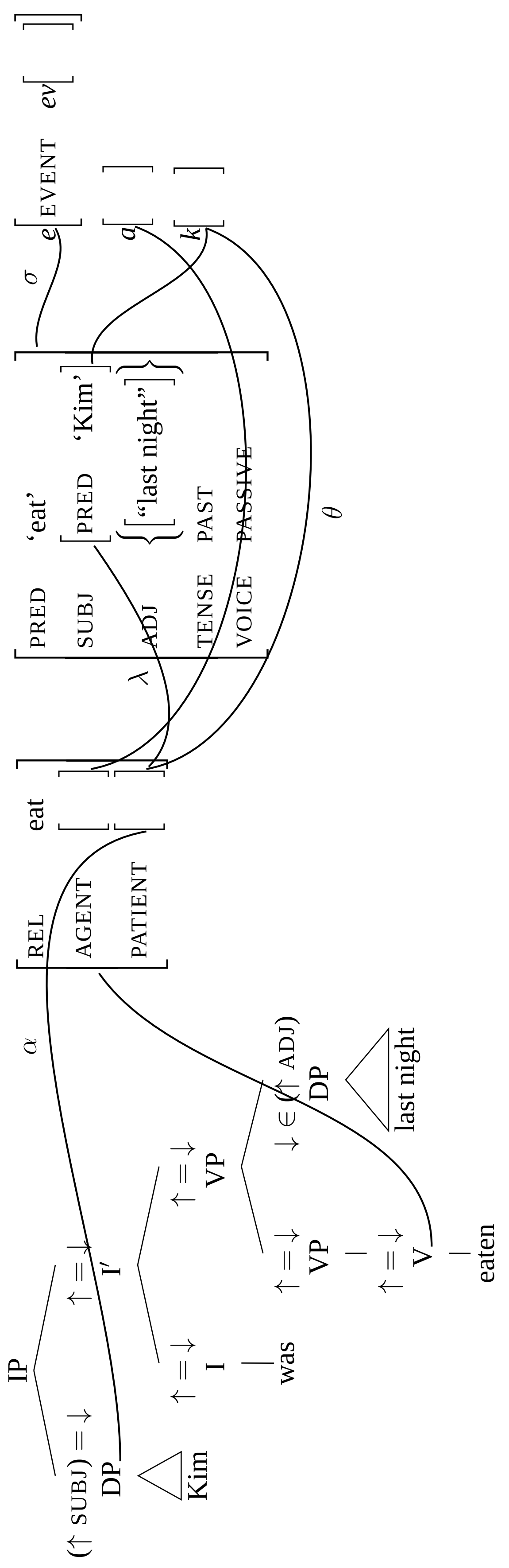


Figure 6: Relevant structures and correspondences for *Kim was eaten last night.*

$$
\cfrac{\cfrac{\begin{array}{l}\textbf{was}\\ \lambda P\exists e.[P(e)\wedge past(e)]:\\ (ev\multimap e)\multimap e\end{array}\quad \cfrac{\begin{array}{l}\textbf{last night}\\ \lambda P\lambda e''.[P(e'')\wedge last.night(e'')]:\\ (ev\multimap e)\multimap(ev\multimap e)\end{array}\quad \cfrac{\cfrac{\begin{array}{l}\textbf{eaten (opt.)}\\ \lambda P\exists x.[P(x)]:\\ (a\multimap e)\multimap e\end{array}\quad \cfrac{\cfrac{\begin{array}{l}\textbf{eaten}\\ eat':\\ ev\multimap k\multimap a\multimap e\end{array}\quad [e':ev]^1}{eat'(e): k\multimap a\multimap e}\quad \begin{array}{l}\textbf{Kim}\\ kim:\\ k\end{array}}{eat'(e')(kim): a\multimap e}}{\exists x.[eat'(e')(kim)(x)]: e}}{\lambda e'\exists x.[eat'(e')(kim)(x)]: ev\multimap e}\multimap_{\mathcal{I},1}}{\lambda e''\exists x.[eat'(e')(kim)(x)\wedge last.night(e'')]: ev\multimap e}}{\exists e\exists x.[eat'(e)(kim)(x)\wedge last.night(e)\wedge past(e)]: e}}{\exists e\exists x.[eat(e)\wedge agent(e)=x\wedge patient(e)=kim\wedge last.night(e)\wedge past(e)]: e}\Rightarrow_{\beta}
$$

Figure 7: Proof for *Kim was eaten last night.*

- The lexical entry for the optional *by*-phrase is shown in (30).

(30) *by* P $(\uparrow \textsc{pred}) = \text{`by'}$
$((\textsc{obl}\ \uparrow)\ \textsc{voice}) =_c \textsc{passive}$
$\lambda x \lambda P.[P(x)]$:
$(\uparrow \textsc{obj})_\sigma \multimap [\uparrow_\sigma \multimap (\textsc{obl}\ \uparrow)_\sigma] \multimap (\textsc{obl}\ \uparrow)_\sigma$

- Rather than existentially closing the suppressed argument of its passive verb, the *by*-phrase saturates the corresponding argument of the passive with the OBJ in the *by*-phrase (e.g., *Sandy* in *by Sandy*).
- There has been some inconsistency in the LFG literature regarding the realization of the *by*-phrase at f-structure: Is it an ADJ or an OBL? (See Needham and Toivonen 2011 for discussion and references.)
- This choice does not substantively affect our analysis, but we assume the *by*-phrase is an OBL here; otherwise, replace (OBL ↑) with (ADJ ∈ ↑).
- Needham and Toivonen (2011) also note that the *by*-phrase must fill the role of whatever was the highest/suppressed argument of the verb that it modifies.
- We assume that this is taken care of in the mapping from a-structure to f-structure, such that the highest role is mapped to the f-structure OBL.
- The resource-sensitivity of the glue logic is important in controlling the compositional possibilities of the passive.
  - ⊙ If the *by*-phrase is present, then the optional meaning constructor contributed by the passive verb cannot also be present.
  - ⊙ If the optional meaning constructor were present, there would be two dependencies on the resource corresponding to the highest role (e.g., the AGENT of *eaten*, which would be mapped to the *by*-phrase, if one is present).
  - ⊙ We thus correctly predict that there is existential closure of the suppressed argument if and only if there is no *by*-phrase.
- Example:

(31) Kim was eaten by Godzilla last night.

(32) $eat' := \lambda e \lambda y \lambda x.eat(e) \wedge agent(e) = x \wedge patient(e) = y$

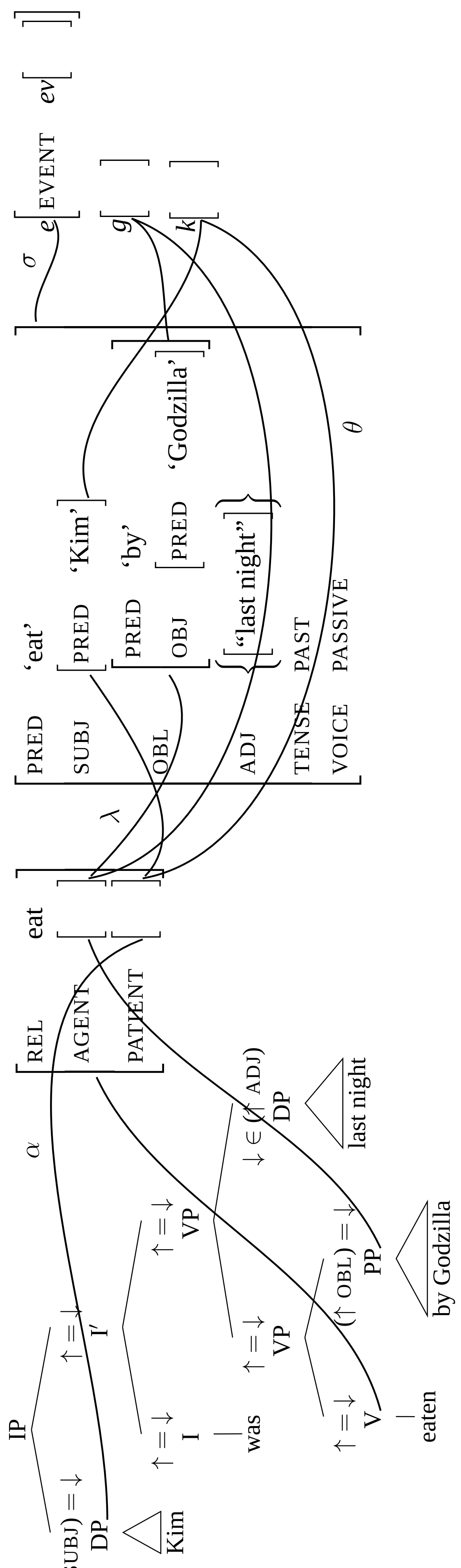


Figure 8: Relevant structures and correspondences for *Kim was eaten by Godzilla last night.*

$$
\dfrac{
\begin{array}{c}\textbf{was}\\ \lambda P\exists e.[P(e)\wedge past(e)] :\\ (ev\multimap e)\multimap e\end{array}
\qquad
\dfrac{
\begin{array}{c}\textbf{last night}\\ \lambda P\lambda e''.[P(e'')\wedge last.night(e'')] :\\ (ev\multimap e)\multimap(ev\multimap e)\end{array}
\qquad
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}\textbf{by}\\ \lambda x\lambda P.[P(x)] :\\ g\multimap(a\multimap e)\multimap e\end{array}
\quad
\begin{array}{c}\textbf{Godzilla}\\ godzilla :\\ g\end{array}
}{\lambda P.[P(godzilla)] : (a\multimap e)\multimap e}
\quad
\dfrac{
\dfrac{
\begin{array}{c}\textbf{eaten}\\ eat' :\\ ev\multimap k\multimap a\multimap e\end{array}
\quad
[e' : ev]^{1}
}{eat'(e) : k\multimap a\multimap e}
\quad
\begin{array}{c}\textbf{Kim}\\ kim :\\ k\end{array}
}{eat'(e')(kim) : a\multimap e}
}{eat'(e')(kim)(godzilla) : e}
}{\lambda e'.[eat'(e')(kim)(godzilla)] : ev\multimap e}\multimap_{\mathcal{I},1}
}{\lambda e''.[eat'(e')(kim)(godzilla)\wedge last.night(e'')] : ev\multimap e}
}{
\dfrac{\exists e.[eat'(e)(kim)(godzilla)\wedge last.night(e)\wedge past(e)] : e}{\exists e.[eat(e)\wedge agent(e) = godzilla\wedge patient(e) = kim\wedge last.night(e)\wedge past(e)] : e}\Rightarrow_{\beta}
}
$$

Figure 9: Proof for *Kim was eaten by Godzilla last night.*

6.3 Instrumentals

- The last case we look at is instrumental *with*-phrases, including the contrast in (33) and (34):

(33) a. Robin killed Sandy.
b. Robin killed Sandy with dynamite.

(34) a. An explosion killed Sandy.
b. #An explosion killed Sandy with dynamite.

- Instrumental *with*-phrases, like passive *by*-phrases, are instances of Needham and Toivonen's 'derived arguments'.
- Following Reinhart (2002), Needham and Toivonen (2011: 415) note that the instrumental *with*-phrase are only well-formed with "agent verbs".
- Stated as such, a generalization seems to be missed, because we seem to require two verbs *kill*: agentive *kill*$_1$ and non-agentive *kill*$_2$.
- Analysis:
  - ⊙ Our analysis instead captures the contrast through the same kind of standard restrictive semantics used for *eat* above, by imposing a requirement of animacy on the subject argument while simultaneously adding the information that the object of the *with*-phrase is an instrument in the event.
  - ⊙ We assume that, unlike passive *by*-phrases, instrumental *with*-phrases add an argument that is not otherwise linguistically represented. However, this is accomplished through lexical information associated with instrumental *with*, rather than by directly modifying the lexical entry of the verb.
  - ⊙ Once again, for consistency with Needham and Toivonen (2011), we treat the *with*-phrase as an OBL, but once again this does not substantively affect our analysis: if the *with*-phrase is better analyzed as an ADJ, replace (OBL ↑) with (ADJ ∈ ↑).
  - ⊙ Here is the lexical entry for instrumental *with*:

(35) *with* P $(\uparrow \text{PRED}) = \text{`with'}$

$$\lambda y \lambda P \lambda x \lambda e.[P(x)(e) \;\wedge\; \mathit{animate}(x) \;\wedge\; \mathit{instrument}(e) = y] :$$
$$(\uparrow \text{OBJ})_\sigma \multimap$$
$$[((\text{OBL} \uparrow)\ \text{SUBJ})_\sigma \multimap ((\text{OBL} \uparrow)_\sigma\ \text{EVENT}) \multimap (\text{OBL} \uparrow)_\sigma] \multimap$$
$$((\text{OBL} \uparrow)\ \text{SUBJ})_\sigma \multimap ((\text{OBL} \uparrow)_\sigma\ \text{EVENT}) \multimap (\text{OBL} \uparrow)_\sigma$$

  - ⊙ In our analysis, no mention is made of the underlying role of the subject argument, allowing it to be the same role whether the instrumental is present or not.

- Example:

(36) Kim tapped Sandy with Excalibur.

(37) *tapped* V $(\uparrow \text{PRED}) = \text{`tap'}$
$(\uparrow \text{TENSE}) = \text{PAST}$

$$\lambda y \lambda x \lambda e.\mathit{tap}(e) \;\wedge\; \mathit{agent}(e) = x \;\wedge\; \mathit{patient}(e) = y :$$
$$(\uparrow \text{OBJ})_\sigma \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$$

(38) $\mathit{tap}' := \lambda y \lambda x \lambda e.\mathit{tap}(e) \wedge \mathit{agent}(e) = x \wedge \mathit{patient}(e) = y$

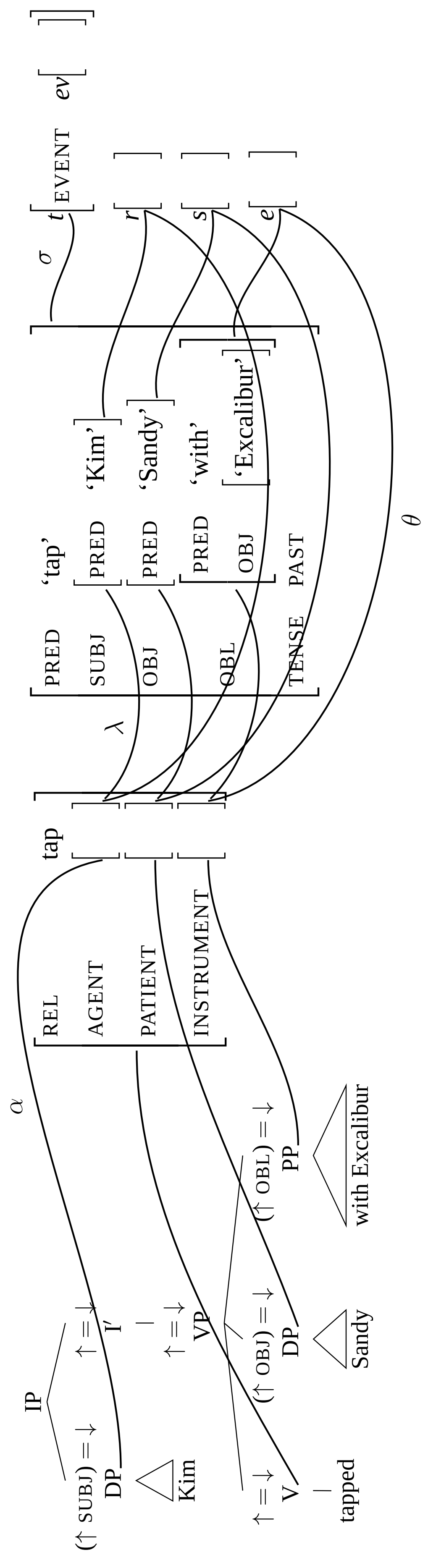


Figure 10: Relevant structures and correspondences for *Kim tapped Sandy with Excalibur.*

$$\dfrac{\dfrac{\begin{array}{l}\textbf{PAST}\\ \lambda P \exists e.[P(e) \wedge past(e)] :\\ (ev \multimap t) \multimap t\end{array} \qquad \dfrac{\dfrac{\dfrac{\begin{array}{l}\textbf{with}\\ \lambda y \lambda P \lambda x \lambda e.[P(x)(e) \wedge animate(x) \wedge instrument(e) = y] :\\ e \multimap (k \multimap ev \multimap t) \multimap k \multimap ev \multimap t\end{array} \quad \begin{array}{l}\textbf{Excalibur}\\ excalibur :\\ e\end{array}}{\begin{array}{l}\lambda P \lambda x \lambda e.[P(x)(e) \wedge animate(x) \wedge instrument(e) = excalibur] :\\ (k \multimap ev \multimap t) \multimap k \multimap ev \multimap t\end{array}} \quad \dfrac{\begin{array}{l}\textbf{tapped}\\ tap' :\\ s \multimap k \multimap ev \multimap t\end{array} \quad \begin{array}{l}\textbf{Sandy}\\ sandy :\\ s\end{array}}{tap'(sandy) : k \multimap ev \multimap t}}{\lambda x \lambda e.[tap'(sandy)(x)(e) \wedge animate(x) \wedge instrument(e) = excalibur] : k \multimap ev \multimap t} \quad \begin{array}{l}\textbf{Kim}\\ kim :\\ k\end{array}}{\lambda e.[tap'(sandy)(kim)(e) \wedge animate(kim) \wedge instrument(e) = excalibur] : ev \multimap t}}{\exists e.[tap'(sandy)(kim)(e) \wedge animate(kim) \wedge instrument(e) = excalibur] : t}}{\exists e.[tap(e) \wedge agent(e) = kim \wedge patient(e) = sandy \wedge animate(kim) \wedge instrument(e) = excalibur \wedge past(e)] : t} \Rightarrow_\beta$$

Figure 11: Proof for *Kim tapped Sandy with Excalibur.*

7 An Alternative Analysis: Argument Structure at S-Structure

- We now present an alternative analysis that does away with the λ-projection, the λ correspondence function, and the θ correspondence function. Argument structure is effectively captured in semantic structure instead. Some of the benefits of this approach are as follows:

  1. We achieve a simplified architecture, without losing information.
  2. We remove the non-determinacy introduced by the λ and θ correspondence functions.
  3. Many of the meaning constructors are more elegant and simplified.
  4. We regain the simple, traditional ϕ mapping.
  5. We gain a connected s-structure.
  6. We do not lose linking relations and they are still post-c-structure.

(39) *drank* V $(\uparrow \textsc{pred}) = \text{`drink'}$

$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{agent})$
$(\uparrow_\sigma \textsc{patient})$

$\lambda y \lambda x \lambda e.drink(e) \wedge agent(e) = x \wedge patient(e) = y :$
$(\uparrow_\sigma \textsc{patient}) \multimap (\uparrow_\sigma \textsc{agent}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

$$\left(\begin{array}{l} \lambda P \exists x.[P(x) \wedge alcoholic(x)] : \\ [(\uparrow_\sigma \textsc{patient}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma \end{array} \right)$$

(40) *quaffed* V $(\uparrow \textsc{pred}) = \text{`quaff'}$

$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{agent})$
$(\uparrow \textsc{obj})_\sigma = (\uparrow_\sigma \textsc{patient})$

$\lambda y \lambda x \lambda e.quaff(e) \wedge agent(e) = x \wedge patient(e) = y :$
$(\uparrow_\sigma \textsc{patient}) \multimap (\uparrow_\sigma \textsc{agent}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

(41) *eaten* V $(\uparrow \textsc{pred}) = \text{`eat'}$
$(\uparrow \textsc{voice}) = \textsc{passive}$

$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{patient})$
$(\uparrow_\sigma \textsc{agent})$

$\lambda x \lambda y \lambda e.eat(e) \wedge agent(e) = x \wedge patient(e) = y :$
$(\uparrow_\sigma \textsc{agent}) \multimap (\uparrow_\sigma \textsc{patient}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

$\left(\lambda P \exists x.[P(x)] : [(\uparrow_\sigma \textsc{agent}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma \right)$

(42) *by* P $(\uparrow \textsc{pred}) = \text{`by'}$
$((\textsc{obl} \uparrow) \textsc{voice}) =_c \textsc{passive}$

$\lambda x \lambda P.[P(x)] : (\uparrow \textsc{obj})_\sigma \multimap [\uparrow_\sigma \multimap (\textsc{obl} \uparrow)_\sigma] \multimap (\textsc{obl} \uparrow)_\sigma$

(43) *with* P $(\uparrow \textsc{pred}) = \text{`with'}$

$\uparrow_\sigma = ((\textsc{obl} \uparrow)_\sigma \textsc{instrument})$

$\lambda y \lambda P \lambda x \lambda e.[P(x)(e) \wedge animate(x) \wedge instrument(e) = y] :$
$(\uparrow \textsc{obj})_\sigma \multimap$
$[((\textsc{obl} \uparrow) \textsc{subj})_\sigma \multimap ((\textsc{obl} \uparrow)_\sigma \textsc{event}) \multimap (\textsc{obl} \uparrow)_\sigma] \multimap$
$((\textsc{obl} \uparrow) \textsc{subj})_\sigma \multimap ((\textsc{obl} \uparrow)_\sigma \textsc{event}) \multimap (\textsc{obl} \uparrow)_\sigma$

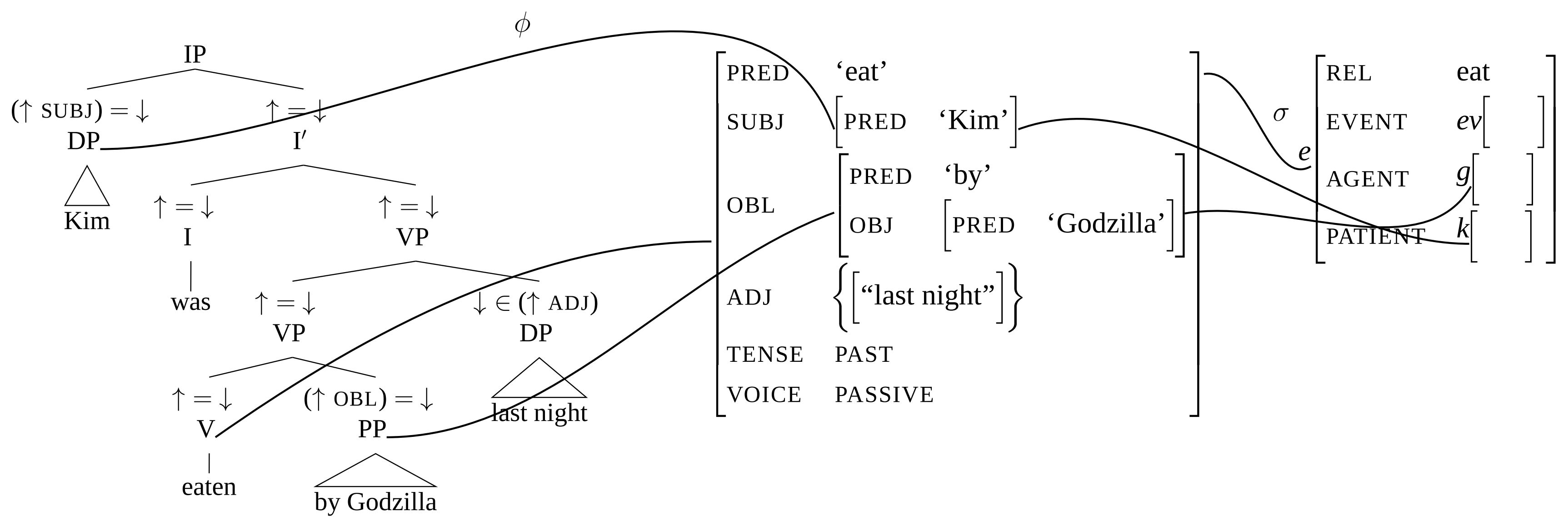


Figure 12: Relevant structures and correspondences for alternative analysis of *Kim was eaten by Godzilla last night*.

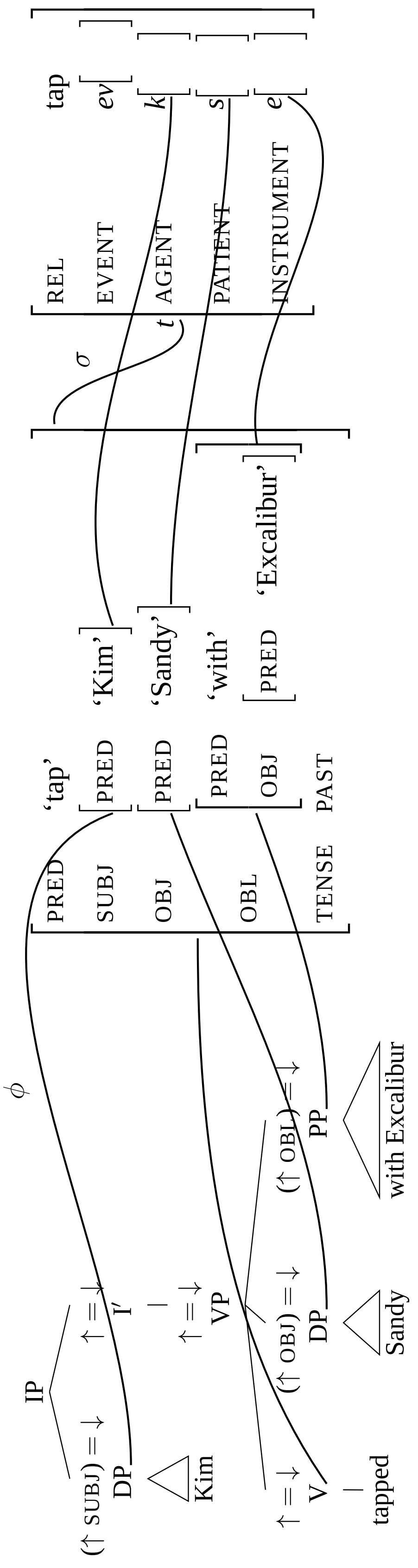


Figure 13: Relevant structures and correspondences for alternative analysis of *Kim tapped Sandy with Excalibur.*

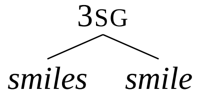
8 Capturing Lexical Generalizations

- An LFG template is a label for a functional description — a set of equations and constraints that describes linguistic structures, such as the functional descriptions that describe f-structures (Dalrymple et al. 2004, Asudeh et al. 2008, Crouch et al. 2011, Asudeh 2012).
- Template invocation is denoted by the prefix @ in a functional description.
- The semantics of template invocation is substitution. Any occurrence of a template in a lexical entry or rule can be equivalently replaced by the grammatical description that the template is associated with.
- Templates are therefore purely abbreviatory devices, but can nevertheless capture linguistic generalizations, since associating a grammatical description with a template treats the description as a natural class.
- Thus, a grammar with templates is extensionally equivalent to the same grammar with all templates replaced by their associated grammatical descriptions, but the first grammar might express generalizations that the second grammar does not.
- Templates can also encode information hierarchically, since template definitions may refer to other templates. This is reminiscent of the type hierarchies of HPSG (Pollard and Sag 1987, 1994) and Sign-Based Construction Grammar (SBCG; Michaelis 2010, Sag 2010).
- However, template hierarchies represent inclusion, rather than inheritance. If template B is a sub-template of template A, then the description that A labels is included in the description that B labels.
- Illustration: English agreement

(44) 3$\textsc{sg}$ = $(\uparrow \textsc{subj pers}) = 3$
$(\uparrow \textsc{subj num}) = \textsc{sg}$

(45) *smiles* $(\uparrow \textsc{pred}) =$ ‘smile$\langle\textsc{subj}\rangle$’
@3$\textsc{sg}$

(46) *smile* $(\uparrow \textsc{pred}) =$ ‘laugh$\langle\textsc{subj}\rangle$’
¬@3$\textsc{sg}$

(47)


- Templates can also take arguments.

(48) $\textsc{intrans}$(P) = $(\uparrow \textsc{pred}) =$ ‘P$\langle\textsc{subj}\rangle$’

- Given the substitutional semantics of templates, the following two lexical entries are equivalent:

(49) *smiles* @$\textsc{intrans}$(smile)
@3$\textsc{sg}$

(50) *smiles* $(\uparrow \textsc{pred}) =$ ‘smile$\langle\textsc{subj}\rangle$’
$(\uparrow \textsc{subj pers}) = 3$
$(\uparrow \textsc{subj num}) = \textsc{sg}$

8.1 Lexical Entries with Templates

8.1.1 λ-Projection Analysis

(51) PAST = $(\uparrow \text{TENSE}) = \text{PAST}$

(52) AGENT-PATIENT-VERB =
$\lambda P \lambda y \lambda x \lambda e. P(e) \wedge \mathit{agent}(e) = x \wedge \mathit{patient}(e) = y :$
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

(53) UNDERSTOOD-OBJECT = $\lambda P \exists x.[P(x)] : [(\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

(54) *drank* V $(\uparrow \text{PRED}) =$ ‘drink’
@PAST
@AGENT-PATIENT-VERB

$$\left(\begin{array}{l} \text{@UNDERSTOOD-OBJECT} \\ \\ \lambda P \lambda x \lambda e. P(x)(e) \wedge \mathit{alcoholic}(x) : \\ {[(\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap} \\ \quad (\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma \end{array} \right)$$

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

(55) *ate* V $(\uparrow \text{PRED}) =$ ‘ate’
@PAST
@AGENT-PATIENT-VERB

$$\left(\begin{array}{l} \text{@UNDERSTOOD-OBJECT} \\ \\ \lambda P \lambda x \lambda y \lambda e. P(x)(y)(e) \wedge \mathit{food.for}(x, y) : \\ {[(\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow_\sigma \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap} \\ \quad (\widehat{*}_\alpha \text{ PATIENT})_\theta \multimap (\uparrow_\sigma \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma \end{array} \right)$$

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

(56) PASSIVE(P) = $(\uparrow \text{VOICE}) = \text{PASSIVE}$
$(\ \lambda P \exists x.[P(x)] : [(\widehat{*}_\alpha \text{ P})_\theta \multimap \uparrow_\sigma] \multimap \uparrow_\sigma\ )$

(57) *eaten* V $(\uparrow \text{PRED}) =$ ‘eat’
@PASSIVE(AGENT)

$\lambda x \lambda y \lambda e.\mathit{eat}(e) \wedge \mathit{agent}(e) = x \wedge \mathit{patient}(e) = y :$
$(\widehat{*}_\alpha \text{ AGENT})_\theta \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

8.1.2 Non-λ Analysis

(58) AGENT-PATIENT-VERB =
$(\uparrow \text{SUBJ})_\sigma = (\uparrow_\sigma \text{AGENT})$
$\lambda P \lambda y \lambda x \lambda e.P(e) \ \wedge \ agent(e) = x \ \wedge \ patient(e) = y$:
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{PATIENT}) \multimap (\uparrow_\sigma \text{AGENT}) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

(59) UNDERSTOOD-OBJECT = $(\uparrow_\sigma \text{PATIENT})$
$\lambda P \exists x.[P(x)] : [(\uparrow_\sigma \text{PATIENT}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

(60) *drank* V $(\uparrow \text{PRED}) =$ ‘drink’
@PAST
@AGENT-PATIENT-VERB

$$\left(\begin{array}{l} @\text{UNDERSTOOD-OBJECT} \\ \lambda P \lambda x \lambda e.P(x)(e) \wedge alcoholic(x) : \\ {[(\uparrow_\sigma \text{PATIENT}) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap} \\ \quad (\uparrow_\sigma \text{PATIENT}) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma \end{array} \right)$$

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

(61) *quaffed* V $(\uparrow \text{PRED}) =$ ‘quaff’
@PAST
@AGENT-PATIENT-VERB
$(\uparrow \text{OBJ})_\sigma = (\uparrow_\sigma \text{PATIENT})$
$\lambda e.quaff(e) : (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

(62) PASSIVE(P,Q) = $(\uparrow \text{VOICE}) = \text{PASSIVE}$
$(\uparrow \text{SUBJ})_\sigma = (\uparrow_\sigma \text{Q})$
$(\uparrow_\sigma \text{P})$
$(\ \lambda P \exists x.[P(x)] : [(\uparrow_\sigma \text{P}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma \ )$

(63) *eaten* V $(\uparrow \text{PRED}) =$ ‘eat’
@PASSIVE(AGENT,PATIENT)
$\lambda x \lambda y \lambda e.eat(e) \ \wedge \ agent(e) = x \ \wedge \ patient(e) = y$:
$(\uparrow_\sigma \text{AGENT}) \multimap (\uparrow \text{SUBJ})_\sigma \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$

9 Conclusion

- We noted above that the challenge posed by these phenomena is to capture the core argument structure of verb classes that display optional or derived arguments in a way that:

  1. Doesn't simply treat the distinct valencies as homonymous, i.e. just accidentally related.
  2. Supports a systematic, formal semantic treatment of optional and derived arguments.
  3. Enables semantic restrictions on optional arguments to be stated.
  4. Captures commonalities between derived arguments and adjuncts

- These challenges have been met based on the following three features of LFG+Glue:

  1. **Optionality** in functional descriptions in lexical entries allows a single lexical entry for verbs like *eat*, capturing the theoretical intuition that this is an instance of polysemy, not homonymy.
  2. The formal treatment uses **flexible composition**, which allows optional and derived arguments to be specified in additional meaning constructors, whether optionally instantiated in the verb's lexical entry or in lexical entries for prepositional heads of derived argument. Semantic restrictions on optional arguments are simultaneously captured lexically. Lastly, derived arguments have a modifier type in the semantic composition, which is a step towards explaining their puzzling adjunct-like behaviour.
  3. **Resource-sensitive composition** ensures proper interaction of optional information in the verb's entry and information contributed by derived arguments, e.g. the optional existential closure of the suppressed argument in the passive and the realization of the suppressed argument in the *by*-phrase.

- In addition to meeting the challenges we identified, we demonstrated how **generalizations over descriptions**, as offered by templates, can capture lexical generalizations concerning optional and derived arguments.

- We also considered two different architectural arrangements in light of these phenomena, with the second, s-structure approach having several advantages.

  1. We achieve a simplified architecture that does away with the λ-projection, the λ correspondence function, the θ correspondence function. Nevertheless, the relevant information is not lost.
  2. We remove the non-determinacy introduced by the λ and θ correspondence functions.
  3. Many of the meaning constructors are more elegant and simplified.
  4. We regain the simple, traditional ϕ mapping.
  5. We gain a connected s-structure.
  6. We do not lose linking relations and they are still post-c-structure.

References

Asudeh, Ash. 2004. Resumption as Resource Management. Ph.D. thesis, Stanford University.

—. 2005a. Control and Semantic Resource Sensitivity. *Journal of Linguistics* 41(3): 465–511.

—. 2005b. Relational Nouns, Pronouns, and Resumption. *Linguistics and Philosophy* 28(4): 375–446.

—. 2006. Direct Compositionality and the Architecture of LFG. In Miriam Butt, Mary Dalrymple, and Tracy Holloway King, eds., *Intelligent Linguistic Architectures: Variations on Themes by Ronald M. Kaplan*, 363–387. Stanford, CA: CSLI Publications.

—. 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, and Ida Toivonen. 2009. Lexical-Functional Grammar. In Heine and Narrog 2009, 425–458.

Bresnan, Joan. 1978. A Realistic Transformational Grammar. In Morris Halle, Joan Bresnan, and George A. Miller, eds., *Linguistic Theory and Psychological Reality*, 1–59. Cambridge, MA: MIT Press.

Butt, Miriam, Mary Dalrymple, and Anette Frank. 1997. An Architecture for Linking Theory in LFG. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG97 Conference*. Stanford, CA: CSLI Publications.

Carpenter, Bob. 1997. *Type-Logical Semantics*. Cambridge, MA: MIT Press.

Crouch, Richard, Mary Dalrymple, Ronald M. Kaplan, Tracy King, John T. Maxwell, III, and Paula Newman. 2011. *XLE Documentation*. Palo Alto Research Center (PARC), Palo Alto, CA.

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, Vaneet Gupta, John Lamping, and Vijay Saraswat. 1999a. Relating Resource-Based Semantics to Categorial Semantics. In Dalrymple 1999, 261–280.

Dalrymple, Mary, Ronald M. Kaplan, and Tracy Holloway King. 2004. Linguistic Generalizations over Descriptions. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG04 Conference*, 199–208. Stanford, CA: CSLI Publications.

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

Dalrymple, Mary, John Lamping, Fernando Pereira, and Vijay Saraswat. 1999b. Overview and Introduction. In Dalrymple 1999, 1–38.

Dalrymple, Mary, John Lamping, and Vijay Saraswat. 1993. LFG Semantics via Constraints. In *Proceedings of the Sixth Meeting of the European ACL*, 97–105. European Chapter of the Association for Computational Linguistics, University of Utrecht.

Fillmore, Charles J. 1986. Pragmatically Controlled Zero Anaphora. In *Proceedings of the Twelfth Annual Meeting of the Berkeley Linguistics Society*, 95–107. Berkeley: Berkeley Linguistics Society.

Girard, Jean-Yves. 1987. Linear Logic. *Theoretical Computer Science* 50(1): 1–102.

Halvorsen, Per-Kristian, and Ronald M. Kaplan. 1988. Projections and Semantic Description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, 1116–1122. Institute for New Generation Systems, Tokyo. Reprinted in Dalrymple et al. (1995: 279–292).

Heine, Bernd, and Heiko Narrog, eds. 2009. *The Oxford Handbook of Linguistic Analysis*. Oxford: Oxford University Press.

Jäger, Gerhard. 2005. *Anaphora and Type Logical Grammar*. Dordrecht: Springer.

Kaplan, Ronald M. 1987. Three Seductions of Computational Psycholinguistics. In Peter Whitelock, Mary McGee Wood, Harold L. Somers, Rod Johnson, and Paul Bennett, eds., *Linguistic Theory and Computer Applications*, 149–181. London: Academic Press. Reprinted in Dalrymple et al. (1995: 339–367).

—. 1989. The Formal Architecture of Lexical-Functional Grammar. In Chu-Ren Huang and Keh-Jiann Chen, eds., *Proceedings of ROCLING II*, 3–18. Reprinted in Dalrymple et al. (1995: 7–27).

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).

Kokkonidis, Miltiadis. 2008. First-Order Glue. *Journal of Logic, Language and Information* 17(1): 43–68.

Kuhn, Jonas. 2001. Resource Sensitivity in the Syntax-Semantics Interface and the German Split NP Construction. In W. Detmar Meurers and Tibor Kiss, eds., *Constraint-Based Approaches to Germanic Syntax*. Stanford, CA: CSLI Publications.

Lev, Iddo. 2007. Packed Computation of Exact Meaning Representations. Ph.D. thesis, Stanford University.

Michaelis, Laura. 2010. Sign-Based Construction Grammar. In Heine and Narrog 2009, 139–158.

Moortgat, Michael. 1997. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen, eds., *Handbook of Logic and Language*, 93–177. Cambridge, MA: MIT Press. Co-published with Elsevier Science B.V., Amsterdam.

Morrill, Glyn V. 1994. *Type Logical Grammar*. Dordrecht: Kluwer.

—. 2011. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford: Oxford University Press.

Needham, Stephanie, and Ida Toivonen. 2011. Derived Arguments. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG11 Conference*, 401–421. Stanford, CA: CSLI Publications.

Pollard, Carl, and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics*. CSLI Publications.

—. 1994. *Head-driven Phrase Structure Grammar*. Chicago, IL and Stanford, CA: The University of Chicago Press and CSLI Publications.

Reinhart, Tanya. 2002. The Theta System — An Overview. *Theoretical Linguistics* 28: 229–290.

Sag, Ivan A. 2010. English Filler-Gap Constructions. *Language* 86(3): 486–545.