Transitivity and Composition*

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

November 29, 2012
University of Essex

1 Introduction

- There is broad agreement in linguistic theory that arguments and adjuncts must be distinguished, but there is substantial disagreement as to how the distinction is to be represented and how borderline cases should be captured.

- There are a number of representational options:
  - ⊙ In Principles and Parameters Theory (Chomsky 1981, 1995), an argument is either the complement or specifier of a head, whereas an adjunct is adjoined at the XP level.
  - ⊙ In some versions of Head-Driven Phrase Structure Grammar, an adjunct is distinguished by being a member of the DEPS list but not a member of the VALENCE lists or of the ARG-ST list (Bouma et al. 2001).
  - ⊙ In LFG, there is a hybrid approach. Adjuncts are distinguished at f-structure by being a member of a predicate's ADJUNCT set, whereas arguments fill specific grammatical functions, such as SUBJ, OBJ, etc. However, given the structure-function mapping principles proposed by Bresnan (2001) and developed further by Toivonen (2001, 2003) (see also Bresnan et al. 2013), adjuncts normally appear in distinguished c-structural positions.

- In this talk, I will focus on the related question of 'optional transitivity'. On the one hand, there are verbs that are 'semantically transitive' (perhaps 'relational' is a better characterization), but which can leave their second argument unexpressed; e.g. *eat*. On the other hand, there are verbs that are 'semantically intransitive', but which can express a 'cognate object'; e.g. *laugh*.

---

*This talk is based on joint work with Gianluca Giorgolo (King's College, London). This research was supported by an Early Researcher Award from the Ontario Ministry of Research and Innovation (Asudeh), NSERC Discovery Grant #371969 (Asudeh), SSHRC Standard Research Grant #410-2010-1841 (Toivonen) and by a grant from the John Fell Oxford University Press Research Fund (Asudeh).

1.1 Main Claims and Questions

- I will present some initial developments in a theory of adjuncts and arguments, building on recent work by Needham and Toivonen (2011), that uses LFG and Glue Semantics (Dalrymple 1999, 2001, Asudeh 2012) to treat the argument/adjunct distinction not narrowly as an issue of syntactic representation, but rather as a distinction that primarily concerns semantic composition.[1]

- The main questions the following:

    1. What are the implications of optional arguments, such as unexpressed PATIENTs, and 'derived arguments', such as cognate objects, for the mapping from syntax to semantics?
    2. How can lexical generalizations about optional and derived arguments best be captured?

- I will look at two cases:

    1. Optional objects of semantically relational verbs (e.g., *drink*, *eat*)
    2. Cognate objects (e.g., *laugh a cruel laugh*, *die a peaceful death*)

- 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. **Mismatches** are allowed between distinct structures in the grammatical architecture. For example, an argument is not necessarily realized as a grammatical function in f-structure. As another example, a null pronominal realizes a grammatical function at f-structure, but is not represented at c-structure.
    2. **Optionality**, offered by the regular language of LFG's functional descriptions in lexical entries (Kaplan and Bresnan 1982, Dalrymple 2001).
    3. **Flexible semantic composition**, offered by the commutative glue logic of Glue Semantics (Dalrymple 1999, 2001, Asudeh 2012).
    4. **Resource-sensitive semantic composition**, again offered by the glue logic.
    5. **Generalizations over descriptions**, offered by templates (Dalrymple et al. 2004, Asudeh et al. 2008, Asudeh 2012). .

2 Overview of the Rest of the Talk

3. Some key data
4. The problem
5. An informal sketch of the approach
6. Background (LFG and Glue Semantics)
7. Analysis (optional arguments, cognate objects)
8. Conclusion

---

[1] Also see (Giorgolo and Asudeh 2012), which takes a distinct formal approach that uses monads, building on Giorgolo and Asudeh (2011). However, that paper maintains the key insight that the argument/adjunct distinction is an issue of semantic composition.

3 Some Key Data

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

(2) Kim ate __ at noon.

(3) a. Isak quaffed his milk at lunch.
b. *Isak quaffed __ at lunch.

(4) a. Thora devoured her cake after dinner.
b. *Thora devoured __ after dinner.

- The distinctions between *drink*/*quaff* and *eat*/*devour* need to be captured lexically somehow — in other words, it is part of what we know as language speakers that *drink* can drop its object argument but that *quaff* does not.[2]

- I'll refer to these sorts of cases as 'optional arguments'.

- These sorts of optional arguments constitute semantically underlying arguments that are not syntactically expressed.

- It has been noted (e.g., Fillmore 1986) that there may be restrictions on implicit arguments that are absent for their explicit counterparts:

(5) a. Fido ate this morning.
$\Rightarrow$ Whatever Fido ate counts as food for Fido
b. Fido ate my homework.
$\not\Rightarrow$ My homework counts as food for Fido

(6) a. Kim drank last night.
$\Rightarrow$ Whatever Kim drank last night is alcoholic/intoxicating
b. Kim drank milk last night.
$\not\Rightarrow$ Milk is alcoholic/intoxicating

- In many respects, 'cognate objects' constitute the opposite scenario, in which a verb that is not normally transitive can appear with a restricted kind of object (among others, Jones 1988, Massam 1990).

(7) a. Kim laughed a cruel laugh.
b. His uncle laughed a loud guffaw. (Google)
c. Kim laughed the most startling laugh we had ever heard.
d. *Kim laughed Sandy.
e. *Kim laughed a pizza.
f. *Kim laughed every laugh last night.

- Cognate objects are not underlying arguments, because they don't passivize (Jones 1988).

(8) *A cruel laugh was laughed by Kim.

---

[2] It has been noted (e.g., Jackendoff 2002) that this may be predictable based on semantic factors, since devouring/quaffing is a particular manner of eating/drinking, etc., but this would just seem to mean that the lexical generalization may be stated in a more general fashion, perhaps in a hierarchically organized lexicon, not that it is not part of lexical knowledge.

(9) *A gruesome death was died by John. (Jones 1988: 91)

(10) *An uneventful life was lived by Harry. (Jones 1988: 91)

(11) *A weary sigh was sighed by Bill. (Jones 1988: 91)

- Nevertheless, unbounded dependencies can target the cognate object (Jones 1988).

(12) a. What sort of a death did John die?
b. What a (gruesome) death John died!

- In LFG-theoretic terms, the passive and extraction data together indicates that the cognate object does not receive a thematic role (it is not a semantic argument), but does fill the OBJ grammatical function in f-structure.

- In contrast, standard syntactic tests such as pronominalization, ellipsis, and secondary predication do not support an OBJECT at f-structure when the second argument is unexpressed:

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

(14) a. Kim is eating a cake, and so is Sandy. (strict or sloppy)
b. Kim is eating, and so is Sandy. (sloppy only)

(15) a. Kim drank the whiskey neat.
b. *Kim drank neat.

- Optional arguments are thus underlying arguments that are not syntactically expressed, whereas cognate objects are syntactically expressed but are not underlying arguments.

- Cognate objects thus count as 'derived arguments', in the sense of Needham and Toivonen (2011), who review a number of other cases in which a syntactic phrase seems to be an adjunct in some ways, but an argument in others. For example, passive *by*-phrases and *with*-instrumentals also fall into this class. I'll leave these other cases aside here, but there are semantic derivations at the end of the paper for: a short passive, a *by*-passive, and a *with*-instrumental.

4 The Problem

- The semantic function that arguments play is typically tied to their obligatory realization in syntax, with optionality often taken to be a hallmark of adjuncts.

- Optional arguments would seem to uncontroversially be arguments according to any plausible semantic criterion, but are nevertheless syntactically optional.

- Cognate objects may be analysed as underlying arguments (Massam 1990, Sailer 2010) or not (Jones 1988) — both moves seem plausible and both sorts of arguments have been made in the literature.

- Most solutions to this valency problem can be characterized as some version of the solution of Bresnan (1978), which proposes two distinct versions of, e.g., the verb *eat*.

(16) *eat:* V, [___ NP], NP_1 'eat' NP_2
[___], $(\exists\ y)\ NP_1$ 'eat' y

- However, this kind of approach is clearly unappealing, because it basically posits an ambiguity for each relevant verb and misses the generalization that, e.g., the 'eating' is the same sort of thing in both cases.

- **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 distinct valencies as accidentally related (homonymous).
  2. Supports a systematic 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

5 An Informal Sketch of The Approach

- The basic strategy will be to break apart lexical information in such a way that, for example, a transitive verb with an optional object can supply semantic information about the implicit object just in case the object is unexpressed. However, a single lexical entry for the verb handles both the intransitive and transitive instantiation of the verb.

- We can exemplify the general approach with the following schematized lexical entry for *eat*:

(17) Kim ate at noon.

(18) *ate* V $(\uparrow \textsc{pred}) = $ 'eat'

F-structure constraints

Obligatory Glue meaning constructor;
encodes general semantic information that is
common to transitive and intransitive uses

(**Optional Glue meaning constructor;**
encodes semantic information that is
specific to the intransitive use)

- The PRED feature of this lexical entry does not encode whether it is transitive or instransitive. We assume that 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). If this were not the case, the formal f-structure description language would force a disjunctive lexical entry, but for theoretically uninteresting reasons (also see Giorgolo and Asudeh 2012 for further discussion).

- The lexical entry in (18) is different from the disjunctive lexical entries suggested by Bresnan (1978), in an important respect. The two Glue meaning constructors in (18) do not stand in a purely disjunctive relationship, whereas the two options in (16) do.

- The logic of the relevant part of the entry in (18) can be represented as $A \veebar (A \wedge B)$, where A is the obligatory meaning constructor, B is the optional meaning constructor, and $\veebar$ is exclusive disjunction. In contrast, the logic of the lexical entry in (16) is purely disjunctive: $A \vee B$, where A is the transitive option and B is the intransitive option. Observe that $(A \veebar (A \wedge B)) \not\equiv (A \veebar B)$.

6 Background

6.1 Lexical-Functional Grammar

- LFG is a declarative, constraint-based linguistic theory (Kaplan and Bresnan 1982).
- The motivation behind LFG is to have a theory that contributes in three ways to our understanding of language:
  1. Theory, including language universals and typology
  2. Psycholinguistics, including language acquisition
  3. Computational linguistics, including automatic parsing and generation, machine translation, and language modelling
- 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).
- Syntax: constituent structure (c-structure) and functional structure (f-structure).
- C-structure is represented by phrase structure trees:
  1. Word order
  2. Dominance
  3. Constituency
  4. Syntactic categories
- F-structure is represented by feature structures (also known as attribute value matrices):
  1. Grammatical functions, such as SUBJECT and OBJECT
  2. Case
  3. Agreement
  4. Tense and aspect
  5. Local dependencies (e.g., control and raising)
  6. Unbounded dependencies (e.g., question formation, relative clause formation)

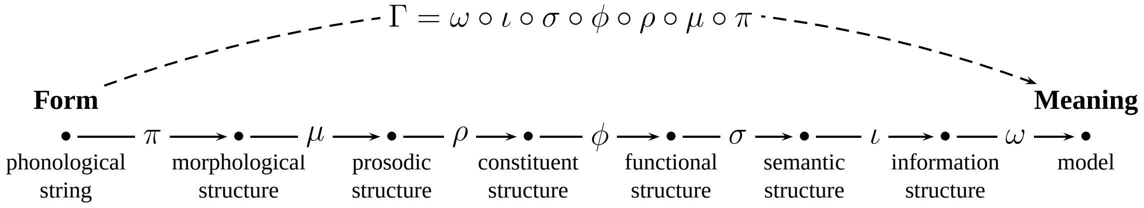
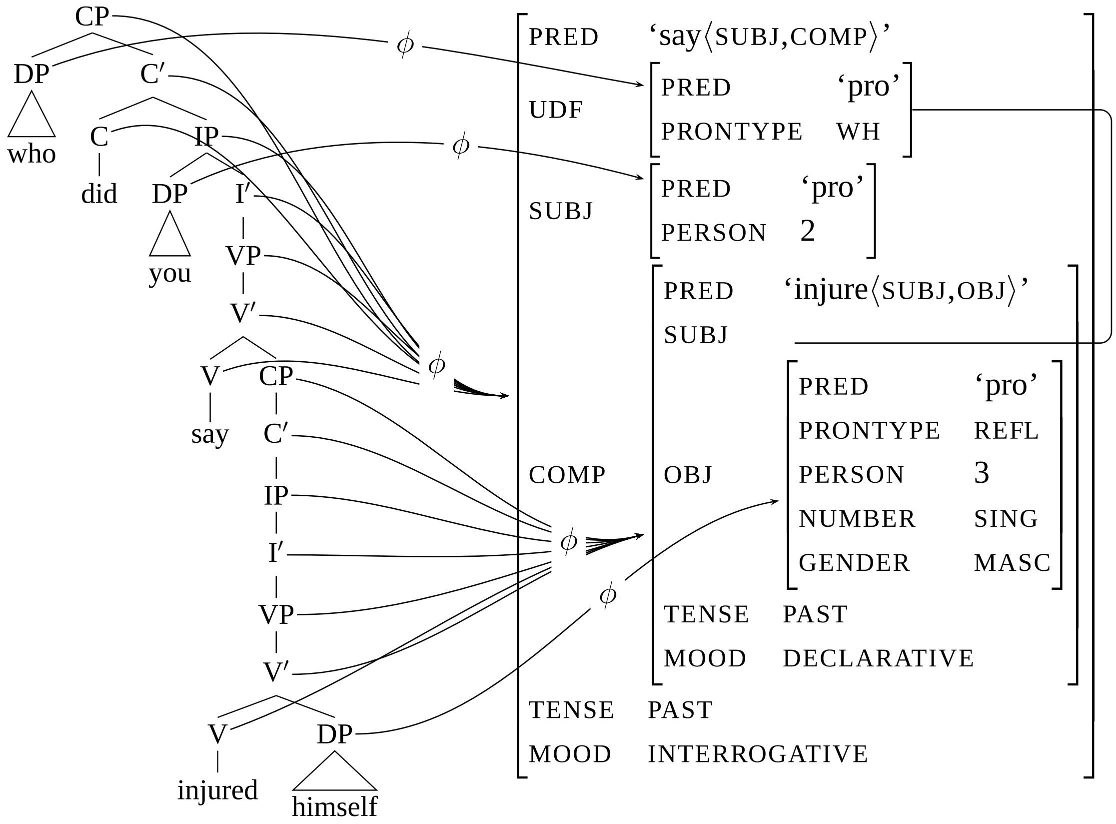


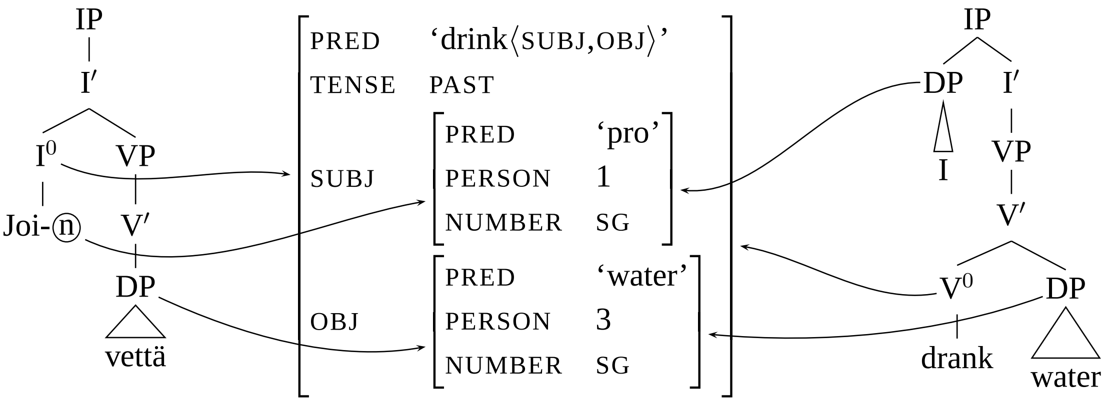
Figure 1: The Correspondence Architecture, pipeline version (based on Asudeh 2012)

6.1.1 Examples

(19)



(20)



6.2 Glue Semantics

- Glue Semantics (Dalrymple 1999, 2001, Asudeh 2004, 2005, 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.

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

  $\mathcal{M}$ 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:

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

- Alternative derivations from the same set of premises → semantic ambiguity (e.g., scope)

- Linear logic is a *resource logic*: each premise in valid linear logic proof must be used exactly once.

- 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, 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, 2005, 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.

- I assume a small, rather weak fragment of linear logic, multiplicative intuitionistic linear logic (MILL; Asudeh 2004, 2005).

- 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.

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

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

(23) Bo chortled.

(24)
$$\dfrac{\mathit{bo} : b \qquad \mathit{chortle} : b \multimap c}{\mathit{chortle}(\mathit{bo}) : c}\multimap_{\mathcal{E}}$$

- Anaphora in Glue Semantics are typically treated as functions on their antecedents (Dalrymple et al. 1999c, Dalrymple 2001). This is a kind of a variable-free treatment of anaphora, which has also been adopted in certain Categorial Grammar analyses (Jacobson 1999, Jäger 2005, among others), although the two variable-free traditions developed separately.
- A variable-free treatment of anaphora is quite natural in Glue, because the commutative linear logic allows anaphora to combine directly with their antecedents, in opposition to the kind of intervening operations that are necessary for variable-free anaphoric resolution in non-commutative Categorial Grammar.
- The meaning constructor for a pronominal has the following general form, where $\uparrow$ is the f-structure of the pronoun and $\uparrow_\sigma$ is its σ-projection in sem-structure:

(25)
$$\lambda z.z \times z : (\uparrow_\sigma \text{ANTECEDENT}) \multimap [(\uparrow_\sigma \text{ANTECEDENT}) \otimes \uparrow_\sigma]$$

- The pronoun's type is therefore $\langle \sigma, \langle \sigma, \tau \rangle \rangle$, where σ is the type of the antecedent and τ is the type of the pronoun. I here assume that both σ and τ are type e (individuals).

(26) Bo fooled himself.

(27)
$$\dfrac{\dfrac{\dfrac{\overset{\textbf{Bo}}{\mathit{bo} : b} \qquad \overset{\textbf{himself}}{\lambda z.z \times z : b \multimap (b \otimes p)}}{\mathit{bo} \times \mathit{bo} : b \otimes p}\multimap_{\mathcal{E}} \qquad \dfrac{\dfrac{[x : b]^1 \qquad \overset{\textbf{fooled}}{\lambda u \lambda v.\mathit{fool}(u,v) : b \multimap p \multimap f}}{\lambda v.\mathit{fool}(x,v) : p \multimap f}\multimap_{\mathcal{E}} \qquad [y : p]^2}{\mathit{fool}(x,y) : f}\multimap_{\mathcal{E}}}{\mathit{let}\ \mathit{bo} \times \mathit{bo}\ \mathit{be}\ x \times y\ \mathit{in}\ \mathit{fool}(x,y) : f}\otimes_{\mathcal{E},1,2}}{\mathit{fool}(\mathit{bo},\mathit{bo}) : f}\Rightarrow_\beta$$

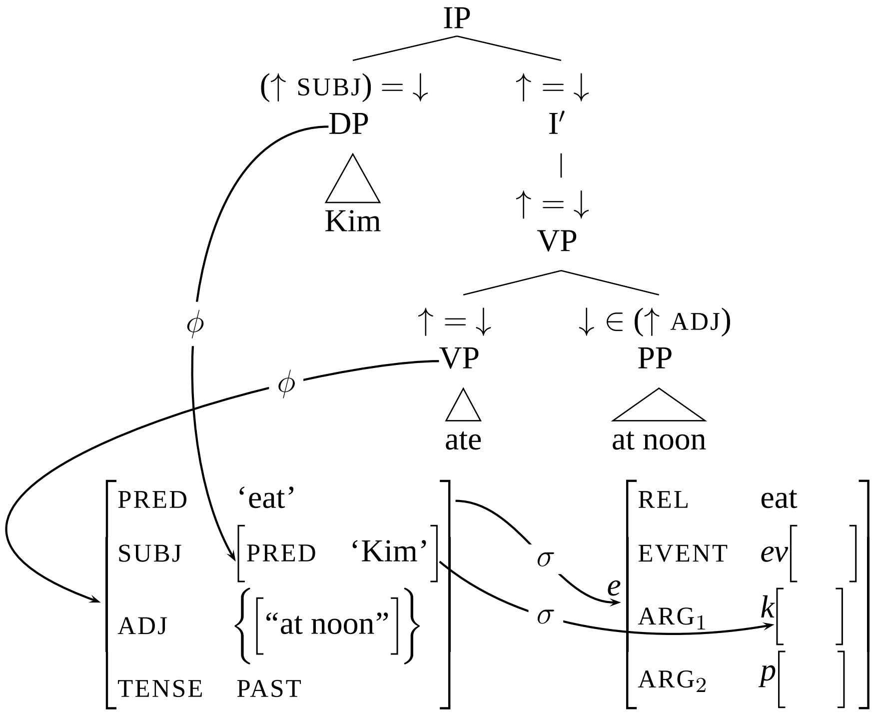


Figure 3: C-structure, f-structure, and semantic structure for *Kim ate at noon.*

7 Analysis

- Butt et al. (1997) treat argument structure (a-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 (from c-structure to a-structure) and λ (from a-structure to f-structure).

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

  1. We achieve a simplified architecture, which eliminates a separate a-structure projection, without losing information.
  2. We do not lose linking relations and they are still post-constituent structure.
  3. Many of the meaning constructors for semantic composition are more elegant and simplified.
  4. We regain the simple, traditional ϕ mapping from c-structure to f-structure.
  5. We gain a connected semantic structure.

- Figure 3 shows relevant structures and correspondences for the following example:

  (28) Kim ate at noon.

- Since we will be assuming an event semantics for our meaning language, such that thematic roles are functions from events to individuals (Parsons 1990), we avoid redundancy in the argument structure by using attributes like ARG_1 instead of AGENT, etc.

- If an alternative meaning language that does not encode thematic roles is used, it may be preferable to represent the nature of the arguments directly in a-structure, using the more specific attributes.

7.1 Optional Arguments

- Optional arguments are semantic arguments that can be syntactically unexpressed, as exemplified by the optional transitivity of *eat* and *drink* versus *devour* and *quaff*.
- In semantic composition, our analysis simultaneously existentially closes the argument that is alternatively expressed by the object — capturing the fact that even though the argument is unexpressed, it is still an understood argument — and appropriately restricts the existentially closed argument (Fillmore 1986).
- For example, the existentially closed argument of *drink* is an intoxicating beverage and that of *eat* is food.[3]
- Moreover, the predicate that expresses this in the semantics must be a relation that also takes the subject as an argument. That is, it is not enough, e.g., for the unexpressed argument to be edible, it must be edible *for* the subject. Contrast the following:

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

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

- My cousin Kim and my cow Kim eat different sorts of things, and our understanding of these sentences reflects that.[4]
- The lexical entry for *ate* is shown in (32) and the Glue proof for example (31) is shown in Figure 5, assuming other standard premises as appropriate and with premises instantiated as per Figure 3.

(31) Kim ate at noon.

(32) *ate* V

$(\uparrow \textsc{pred}) = \text{‘eat’}$
$(\uparrow \textsc{tense}) = \textsc{past}$

$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{arg}_1)$
$(\uparrow_\sigma \textsc{arg}_2)$

$$\lambda y \lambda x \lambda e.eat(e) \wedge agent(e) = x \wedge patient(e) = y : \\ (\uparrow_\sigma \textsc{arg}_2) \multimap (\uparrow_\sigma \textsc{arg}_1) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$$

$$\left(\begin{array}{l} \lambda P \lambda y \exists x.[P(x)(y) \wedge \mathit{food.for}(x,y)] : \\ [(\uparrow_\sigma \textsc{arg}_2) \multimap (\uparrow_\sigma \textsc{arg}_1) \multimap \uparrow_\sigma] \multimap [(\uparrow_\sigma \textsc{arg}_1) \multimap \uparrow_\sigma] \end{array} \right)$$

- The predicate $\mathit{food.for}(x,y)$ is interpreted such that x is food for y.
- It is straightforward to swap the order of arguments of a function directly in the Glue proof, as shown in (33).

---

[3] This information is perhaps better treated as a presupposition or conventional implicature than a straight entailment, but we leave this aside here, since it would be straightforward to augment the analysis in standard ways to capture this aspect.

[4] This is not obvious for *drink*, but it seems to be equally the case. For example if Dr. McCoy from *Star Trek* utters “Every subject drank”, referring to a group of alien beings in his lab, we expect that each subject drank something compatible with its biology (see also Giorgolo and Asudeh 2012).

(33)

$$\begin{prooftree}
\AxiomC{$\lambda y \lambda x.f(x,y) : a \multimap b \multimap c$}
\AxiomC{$[v : a]^{1}$}
\BinaryInfC{$\lambda x.f(x,v) : b \multimap c$}
\AxiomC{$[u : b]^{2}$}
\BinaryInfC{$f(u,v) : c$}
\RightLabel{$\multimap_{\mathcal{I},1}$}
\UnaryInfC{$\lambda v.f(u,v) : a \multimap c$}
\RightLabel{$\multimap_{\mathcal{I},2}$}
\UnaryInfC{$\lambda u \lambda v.f(u,v) : b \multimap a \multimap c$}
\RightLabel{$\Rightarrow_{\alpha}$}
\UnaryInfC{$\lambda x \lambda y.f(x,y) : b \multimap a \multimap c$}
\end{prooftree}$$

- I therefore adopt the convention of choosing a version of the lexically specified function in question that is convenient for the larger proof, abbreviating the function as eat', etc., until the final line of proofs, when the abbreviation is unpacked.

- The same lexical entry in (32) is used for the analysis of an example like this:

  (34) Kim ate the cake at noon.

- In this case the resource sensitivity of Glue Semantics (Asudeh 2004, 2012) ensures that the optional premise cannot be selected.

- The obligatory premise is the only consumer of the object resource in the relevant resource pool.

- If the optional premise is also in the resource pool, then the optional premise acts as a modifier of the obligatory premise, as shown below, such that there is no longer a consumer for the object premise.

- Therefore, selection of the optional premise leads to a successful Glue proof if and only if there is no object resource. If the object is expressed and therefore contributes a resource, the optional premise is not selected and the obligatory premise consumes its object as per usual. The proof for (34) is shown in Figure 6.

- What about obligatory transitives, such as *devour* and *quaff*, which do not allow their objects to be unexpressed (see (3) and (4) above)? The lexical entries for these verbs lack the optional, modificational premise:

  (35) *devoured* V
  $(\uparrow$ PRED$) =$ ‘devour’
  $(\uparrow$ TENSE$) =$ PAST

  $(\uparrow$ SUBJ$)_\sigma = (\uparrow_\sigma$ ARG$_1)$
  $(\uparrow$ OBJ$)_\sigma = (\uparrow_\sigma$ ARG$_2)$

  $\lambda y \lambda x \lambda e.devour(e) \;\wedge\; agent(e) = x \;\wedge\; patient(e) = y :$
  $(\uparrow_\sigma \text{ ARG}_2) \multimap (\uparrow_\sigma \text{ ARG}_1) \multimap (\uparrow_\sigma \text{ EVENT}) \multimap \uparrow_\sigma$

- Resource-sensitive composition ensures that predicates like this must have an expressed object that contributes the ARG$_2$ resource; otherwise the dependency on this resource is not properly discharged and there is no valid Glue proof.

7.2 Scope

- Fodor and Fodor (1980) note that a quantifier in subject position must take wide scope over the existentially closed implicit argument of a syntactically intransitive but semantically relational verb:[5]

  (36) Every student ate.
  $\Rightarrow$ For every student x, there is some thing y such that x ate y.
  $\not\Rightarrow$ There is some thing y such that, for every student x, x ate y.

- Our analysis captures this scope generalization.
- The quantifier and the optional premise contributed by the verb *ate* both constitute dependencies on a dependency on the subject. That is, both the quantifier and the optional premise are consumers of a premise that can be schematized as $subj \multimap predicate$.
- There is only one such premise (the verb's premise, having consumed the implicit argument's resource).
- The optional premise, however, is a modifier-type premise that outputs the same dependency again.
- Therefore, the quantifier can consume the output of the optional premise.
- In contrast, the quantifier does not output a premise of this type, but rather one of a propositional type. Therefore, the optional premise cannot consume the output of the quantifier.
- This means that the quantifier must come later in the proof, which entails that it scopes wide.
- The successful proof for the wide scope reading is shown in Figure 7.[6]

---

[5] This claim has been refined by Lasersohn (1993), based on distributed readings, but he does not seem to have found the correct generalization. This is discussed further in Giorgolo and Asudeh (2012).

[6] Our approach allows the subject quantifier and existential event closure to scope freely with respect to each other, since examples like (36) are ambiguous between a single event of every student eating and separate events of each student eating. The proof in Figure 7 captures only the first of these readings.

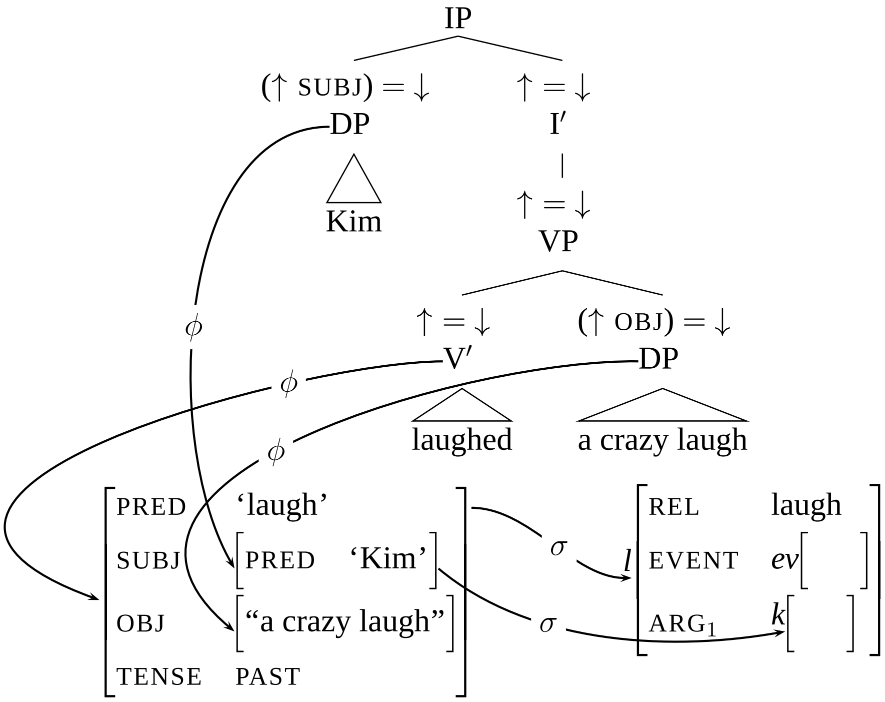


Figure 4: C-structure, f-structure, and semantic structure for *Kim laughed a crazy laugh*

7.3 Cognate Objects

- The lexical entry for *laughed* is shown in (38) and the Glue proof for example (37) is shown in Figure 8, assuming other standard premises as appropriate and with premises instantiated as per Figure 4.

(37) Kim laughed a crazy laugh.

(38) *laugh* V
$(\uparrow \textsc{pred}) = \text{‘laugh’}$
$(\uparrow \textsc{tense}) = \textsc{past}$

$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{arg}_1)$

$\lambda x \lambda e.laugh(e) \ \wedge \ agent(e) = x :$
$(\uparrow_\sigma \textsc{arg}_1) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

- The cognate object is treated compositionally like an adjunct (Sailer 2010), in that it does not map to an argument in semantic structure and in how it composes (as a modifier), but note that it is in fact an OBJ in f-structure.

- This accounts for the object-like syntactic behaviour of the cognate object, without forcing us to treat it as an underlying argument or postulating a transitive version of *laugh*.

8 Conclusion

- I have provided an analysis of optional arguments, such as unexpressed PATIENTs, and derived arguments, such as cognate objects, which treats the phenomena as essentially a problem of semantic composition.

- The analysis is based on established features of Lexical-Functional Grammar with Glue Semantics:

    1. **Mismatches** are allowed between distinct structures in the grammatical architecture. For example, an argument is not necessarily realized as a grammatical function in f-structure. As another example, a null pronominal realizes a grammatical function at f-structure, but is not represented at c-structure.
    2. **Optionality**, offered by the regular language of LFG's functional descriptions in lexical entries (Kaplan and Bresnan 1982, Dalrymple 2001).
    3. **Flexible semantic composition**, offered by the commutative glue logic of Glue Semantics (Dalrymple 1999, 2001, Asudeh 2012).
    4. **Resource-sensitive semantic composition**, again offered by the glue logic.

References

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

—. 2005. 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.

Bouma, Gosse, Robert Malouf, and Ivan A. Sag. 2001. Satisfying Constraints on Extraction and Adjunction. *Natural Language and Linguistic Theory* 19: 1–65.

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.

—. 2001. *Lexical-Functional Syntax*. Oxford: Blackwell.

Bresnan, Joan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2013. *Lexical-Functional Syntax*. Oxford: Wiley-Blackwell, 2nd edn. Forthcoming.

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.

Butt, Miriam, and Tracy Holloway King, eds. 2011. *Proceedings of the LFG11 Conference*. Stanford, CA: CSLI Publications.

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

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

—. 1995. *The Minimalist Program*. 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.

—. 1999c. Quantification, Anaphora, and Intensionality. In Dalrymple 1999, 39–89.

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.

Fodor, Jerry A., and Janet Dean Fodor. 1980. Functional Structure, Quantifiers, and Meaning Postulates. *Linguistic Inquiry* 11(4): 759–770.

Giorgolo, Gianluca, and Ash Asudeh. 2011. Multidimensional Semantics with Unidimensional Glue Logic. In Butt and King 2011, 236–256.

—. 2012. Missing Resources in a Resource-Sensitive Semantics. In Miriam Butt and Tracy Holloway King, eds., *Proceedings of the LFG12 Conference*, This volume. Stanford, CA: CSLI Publications.

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.

Jackendoff, Ray. 2002. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford: Oxford University Press.

Jacobson, Pauline. 1999. Towards a Variable-Free Semantics. *Linguistics and Philosophy* 22(2): 117–184.

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

Jones, Michael Allan. 1988. Cognate Objects and the Case-Filter. *Journal of Linguistics* 24(1): 89–110.

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.

Lasersohn, Peter N. 1993. Lexical Distributivity and Implicit Arguments. In *Proceedings from Semantics and Linguistic Theory III*, 145–161.

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

Massam, Diane. 1990. Cognate Objects as Thematic Objects. *Canadian Journal of Linguistics* 35: 161–190.

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.

Needham, Stephanie, and Ida Toivonen. 2011. Derived Arguments. In Butt and King 2011, 401–421.

Parsons, Terence. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*. Cambridge, MA: MIT Press.

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.

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

Sailer, Manfred. 2010. The Family of English Cognate Object Constructions. In Stefan Müller, ed., *Proceedings of the HPSG10 Conference*, 191–211. Stanford, CA: CSLI Publications.

Toivonen, Ida. 2001. The Phrase Structure of Non-Projecting Words. Ph.D. thesis, Stanford University.

—. 2003. *Non-Projecting Words: A Case Study of Swedish Verbal Particles*. Dordrecht: Kluwer.

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

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

$$\dfrac{\dfrac{\begin{array}{l}\textbf{PAST}\\ \lambda P \exists e.[P(e) \wedge past(e)] :\\ (ev \multimap e) \multimap e\end{array} \quad \dfrac{\begin{array}{l}\textbf{at noon}\\ \lambda P \lambda e'.[P(e') \wedge at.noon(e')] :\\ (ev \multimap e) \multimap (ev \multimap e)\end{array} \quad \dfrac{\dfrac{\begin{array}{l}\textbf{ate}\\ eat' :\\ c \multimap k \multimap ev \multimap e\end{array} \quad \begin{array}{l}\textbf{the cake}\\ \iota x.[cake(x)] :\\ c\end{array}}{eat'(\iota x.[cake(x)]) : k \multimap ev \multimap e} \quad \begin{array}{l}\textbf{Kim}\\ kim :\\ k\end{array}}{eat'(\iota x.[cake(x)])(kim) : ev \multimap e}}{\lambda e'.[eat'(\iota x.[cake(x)])(kim)(e') \wedge at.noon(e')] : ev \multimap e}}{\exists e.[eat'(\iota x.[cake(x)])(kim)(e) \wedge at.noon(e) \wedge past(e)] : e}}{\exists e.[eat(e) \wedge agent(e) = kim \wedge patient(e) = \iota x.[cake(x)] \wedge at.noon(e) \wedge past(e)] : e} \Rightarrow_\beta$$

Figure 6: Proof for *Kim ate the cake at noon.*

$$
\dfrac{
\dfrac{
\begin{array}{l}\textbf{PAST}\\ \lambda P\exists e.[P(e)\wedge past(e)] :\\ (ev\multimap e)\multimap e\end{array}
\quad
\dfrac{
\dfrac{
\begin{array}{l}\textbf{every student}\\ \lambda P\forall z.[student(z)\to\ P(z)] :\\ \forall X.[(s\multimap X)\multimap X]\end{array}
\quad
\dfrac{
\begin{array}{l}\textbf{ate (opt.)}\\ \lambda P\lambda y\exists x.[P(x)(y)\ \wedge\ food.for(x,y)] :\\ [p\multimap s\multimap e]\multimap s\multimap e\end{array}
\quad
\dfrac{\begin{array}{l}\textbf{ate}\\ eat' :\\ ev\multimap p\multimap s\multimap e\end{array}\qquad [e' : ev]^1}{eat'(e') : p\multimap s\multimap e}
}{\lambda y\exists x.[eat'(e')(x)(y)\wedge food.for(x,y)] : s\multimap e}
}{\forall z.[student(z)\to\ \exists x.[eat'(e')(x)(z)\wedge food.for(x,z)]] : e}\ \forall_{\mathcal{E}}[e/X]
}{\lambda e'\forall z.[student(z)\to\ \exists x.[eat'(e')(x)(z)\wedge food.for(x,z)]] : ev\multimap e}\ \multimap_{\mathcal{I},1}
}{\exists e.[\forall z.[student(z)\to\ \exists x.[eat'(e'')(x)(z)\wedge food.for(x,z)]]\wedge at.noon(e'')\wedge past(e)] : e}
}{\exists e.[\forall z.[student(z)\to\ \exists x.[eat(e)\wedge agent(e)=z\wedge patient(e)=x\wedge food.for(x,z)]]\wedge at.noon(e)\wedge past(e)] : e}\ \Rightarrow_\beta
$$

Figure 7: Proof for subject wide scope reading of *Every student ate*.

$$
\dfrac{
\dfrac{
\begin{array}{l}\textbf{PAST}\\ \lambda P\exists e.[P(e)\wedge past(e)] :\\ (ev\multimap l)\multimap l\end{array}
\quad
\dfrac{
\begin{array}{c}\vdots\\ \begin{array}{l}\textbf{a crazy laugh}\\ \lambda P\lambda e'.[P(e')\wedge manner(e')=crazy] :\\ (ev\multimap l)\multimap(ev\multimap l)\end{array}\end{array}
\quad
\dfrac{\begin{array}{l}\textbf{laughed}\\ laugh' :\\ k\multimap ev\multimap l\end{array}\qquad \begin{array}{l}\textbf{Kim}\\ kim :\\ k\end{array}}{laugh'(kim) : ev\multimap l}
}{\lambda e'.[laugh'(kim)(e')\wedge manner(e')=crazy] : ev\multimap l}
}{\exists e.[laugh'(kim)(e)\wedge manner(e)=crazy\wedge past(e)] : l}
}{\exists e.[laugh(e)\wedge agent(e)=kim\wedge manner(e)=crazy\wedge past(e)] : l}\ \Rightarrow_\beta
$$

Figure 8: Proof for *Kim laughed a crazy laugh*.

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

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

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

Figure 10: Proof for *Kim was eaten by Godzilla last night.*

$$\frac{\begin{array}{l}\textbf{with}\\ \lambda y\lambda P\lambda x\lambda e.[P(x)(e)\wedge animate(x)\wedge instrument(e)=y] :\\ e\multimap(k\multimap ev\multimap t)\multimap k\multimap ev\multimap t\end{array}\qquad \begin{array}{l}\textbf{Excalibur}\\ excalibur :\\ e\end{array}}{\begin{array}{l}\lambda P\lambda x\lambda e.[P(x)(e)\wedge animate(x)\wedge instrument(e)=excalibur] :\\ (k\multimap ev\multimap t)\multimap k\multimap ev\multimap t\end{array}}$$

$$\frac{\begin{array}{l}\textbf{tapped}\\ tap' :\\ s\multimap k\multimap ev\multimap t\end{array}\qquad \begin{array}{l}\textbf{Sandy}\\ sandy :\\ s\end{array}}{tap'(sandy) : k\multimap ev\multimap t}$$

$$\frac{\begin{array}{l}\lambda P\lambda x\lambda e.[P(x)(e)\wedge animate(x)\wedge instrument(e)=excalibur] :\\ (k\multimap ev\multimap t)\multimap k\multimap ev\multimap t\end{array}\qquad tap'(sandy) : k\multimap ev\multimap t}{\lambda x\lambda e.[tap'(sandy)(x)(e)\wedge animate(x)\wedge instrument(e)=excalibur] : k\multimap ev\multimap t}$$

$$\frac{\lambda x\lambda e.[tap'(sandy)(x)(e)\wedge animate(x)\wedge instrument(e)=excalibur] : k\multimap ev\multimap t\qquad \begin{array}{l}\textbf{Kim}\\ kim :\\ k\end{array}}{\lambda e.[tap'(sandy)(kim)(e)\wedge animate(kim)\wedge instrument(e)=excalibur] : ev\multimap t}$$

$$\frac{\begin{array}{l}\textbf{PAST}\\ \lambda P\exists e.[P(e)\wedge past(e)] :\\ (ev\multimap t)\multimap t\end{array}\qquad \lambda e.[tap'(sandy)(kim)(e)\wedge animate(kim)\wedge instrument(e)=excalibur] : ev\multimap t}{\exists e.[tap'(sandy)(kim)(e)\wedge animate(kim)\wedge instrument(e)=excalibur] : t}$$

$$\frac{\exists e.[tap'(sandy)(kim)(e)\wedge animate(kim)\wedge instrument(e)=excalibur] : t}{\exists e.[tap(e)\wedge agent(e)=kim\wedge patient(e)=sandy\wedge animate(kim)\wedge instrument(e)=excalibur\wedge past(e)] : t}\Rightarrow_{\beta}$$

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