Introduction to Glue Semantics: class 2

Applications

Jamie Y. Findlay
University of Oslo
jamie.findlay@iln.uio.no

05/08/25

1 The Resource Sensitivity Hypothesis

- Glue treats semantic composition as RESOURCE SENSITIVE in the sense we discussed yesterday: the use of linear logic to guide composition means that semantic contributions cannot be re-used or discarded.
- This seems appropriate for semantics, where e.g. *Anya kissed Xander* means (1a) and not, say, (1b), where we ignore the meaning contributed by *Xander* and use the meaning contributed by *Anya* twice:

  (1) a. $\mathbf{kiss}(\mathbf{anya}, \mathbf{xander})$
  b. $\mathbf{kiss}(\mathbf{anya}, \mathbf{anya})$

- But it also applies to other areas of the grammar, e.g. phonology and syntax (Asudeh 2012: 95–123).
  - In general, we cannot repeat or delete arbitrary phonemes:

    (2) /XY/ ≢ /XXY/
    /XY/ ≢ /X/

  - Nor can we repeat or delete arbitrary words:

    (3) $\text{WORD}_1\ \text{WORD}_2 \not\equiv \text{WORD}_1\ \text{WORD}_1\ \text{WORD}_2$
    $\text{WORD}_1\ \text{WORD}_2 \not\equiv \text{WORD}_1$

- This gives rise to what Asudeh (2004, 2012: 95) calls the Resource Sensitivity Hypothesis:

> **The Resource Sensitivity Hypothesis (RSH)**
>
> Natural language is resource sensitive.

- Asudeh (2012: 97, 106–110) discusses two version of resource sensitivity:

  - LOGICAL RESOURCE SENSITIVITY just describes the properties of a resource logic like linear logic: propositions in the logic cannot be freely re-used or discarded.
  - LINGUISTIC RESOURCE SENSITIVITY is what the RSH is about. This says that elements of combination in grammars also cannot be freely re-used or discarded.

- The most obvious way of modelling linguistic resource sensitivity is via logical resource sensitivity, and this is what Glue does for semantic composition.
- Similar logical approaches to syntax have a long pedigree, going back even further than the generative programme (e.g. Ajdukiewicz 1935; Bar-Hillel 1953; Lambek 1958).[1]
- Work in phonology is less common, though see Wheeler (1988) for one example.
- What is interesting is that (implicit) appeals to the RSH surface in many well-known principles in other frameworks.
- For example, the θ-CRITERION of Government-Binding theory is essentially just the RSH:

(4) **The θ-criterion:**
Each argument bears one and only one θ-role, and each θ-role is assigned to one and only one argument. (Chomsky 1981: 36)

- This rules out sentences like (5) or (6):

(5) **Buffy died the school.*

(6) **Spike devoured.*

- In (5), *the school* is not assigned a θ-role, and in (6), *devoured* only assigns one of its two θ-roles.
- But this is also a resource accounting problem: in (5), we have the resource for *the school* left over once we have proven a type t meaning for the `root` node – this is called a situation of RESOURCE SURPLUS.

$$\dfrac{\dfrac{\begin{array}{c}\textbf{[}\textbf{\textit{Buffy}}\textbf{]} \\ \mathbf{buffy} : E(b)\end{array} \quad \begin{array}{c}\textbf{[}\textbf{\textit{died}}\textbf{]} \\ \lambda x.\mathbf{died}(x) : E(b) \multimap T(d)\end{array}}{\mathbf{died}(\mathbf{buffy}) : T(d)} \quad \begin{array}{c}\textbf{[}\textbf{\textit{the school}}\textbf{]} \\ \vdots \\ \iota x.\mathbf{school}(x) : E(s)\end{array}}{\text{FAIL}}$$

Figure 1: A case of resource surplus

- And in (6), we will be unable to prove a type t meaning for the clause because *devoured* still needs to consume its other argument – this is a situation of RESOURCE DEFICIT.

[1] For a more readable modern introduction to this family of approaches, now known as Combinatory Categorial Grammar (CCG), see Steedman & Baldridge (2011).

$$\dfrac{\dfrac{\textbf{[}\textit{Spike}\textbf{]} \quad \textbf{[}\textit{devoured}\textbf{]}}{\mathbf{spike} : E(s) \quad \lambda x.\lambda y.\mathbf{devoured}(x, y) : E(s) \multimap [E(obj) \multimap T(d)]}}{\dfrac{\lambda y.\mathbf{devoured}(\mathbf{spike}, y) : E(obj) \multimap T(d)}{\text{FAIL}}}$$

Figure 2: A case of resource deficit

- Other similar principles which have the same goal can also be reduced to the RSH, such as the principle of Full Interpretation (Chomsky 1986: 98), the principles of Completeness and Coherence in LFG (Kaplan & Bresnan 1982), the property of bounded closure in HPSG (Klein & Sag 1985: 172), and others (see Asudeh 2012: 110–123 for further discussion).

Exercise 1: Resource sensitivity

For each of the sentences below, say whether the ungrammaticality is the result of a resource surplus or a resource deficit.

1. **Buffy was dead Willow last week.*
2. **Willow finished the book and Tara put on the shelf.*
3. **Xander attempted repeatedly.*
4. **Giles has a grimoire which he has read it several times.*

2 Modification

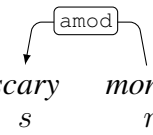
- Modifiers, like adjectives, adverbs, and more complex adjunct phrases, are prime evidence of the resource sensitivity of linguistic meaning.
- In type-theoretic terms they leave their arguments unchanged, which means they could be applied zero or multiple times without affecting the final type of the sentence; but in fact they must be applied *exactly once* each.
- Let us see how some simple modification structures are dealt with in Glue.

2.1 Nominal modifiers

- Let's think first about pre-nominal adjectives, as in *scary monster*.
- The classic problem you will encounter in intro to (formal) semantics courses is that we have a type clash: adjectives and nouns can both be predicative, and so would seem to have the type $\langle e, t\rangle$. But how can two type $\langle e, t\rangle$ meanings combine by function application when neither provides the correct input to the other?

- There have been two main solutions to this issue:
  1. We can sidestep function application and introduce a new rule of Predicate Modification that directly combines two type $\langle e, t\rangle$ predicates (the approach famously taken by Heim & Kratzer 1998).
  2. We can use a rule of type raising to turn one of the predicates into a type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ function which is then applied to the other.
- Glue takes the second approach, but this time the type raising doesn't come for free; we need to introduce a meaning constructor to achieve it.
- We do this by allowing for CONSTRUCTIONAL meanings, whereby the presence of a particular syntactic structure can contribute a meaning constructor.
- In our dependency grammar setup, we assume that particular dependency relations can introduce meaning constructors, which are then associated with the target of that dependency.
- This allows us to give adjectives the lower, predicative type in their lexical entries, and associate the attributive position – in UD this has the label `amod` – with a meaning constructor that lifts them into the modifier type:

(7)



(8)

$$\begin{aligned}
\textit{scary} &\rightsquigarrow \lambda x.\mathbf{scary}(x) : \\
&\quad E(\bullet) \multimap T(\bullet) \\
\texttt{amod} &\rightsquigarrow \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x) : \\
&\quad [E(\bullet) \multimap T(\bullet)] \multimap [[E(\hat{\bullet}) \multimap T(\hat{\bullet})] \multimap E(\hat{\bullet}) \multimap T(\hat{\bullet})] \\
\textit{monster} &\rightsquigarrow \lambda x.\mathbf{monster}(x) : \\
&\quad E(\bullet) \multimap T(\bullet)
\end{aligned}$$

- Note that alongside the meta-variable $\bullet$, which refers to the node that bears the meaning constructor, we also use the symbol '$\hat{\bullet}$' as a meta-variable which refers to the *mother* of the node bearing the meaning constructor, i.e. the node you reach by following the incoming dependency to its origin.[2]
- Figure 3 shows the proof for the meaning of *scary monster*, using the node labels from the tree in (7).

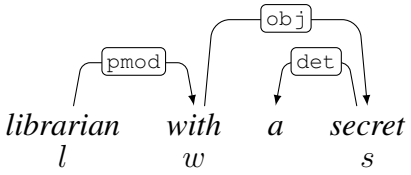
[2]Note that this assumes that multi-dominance, where two dependencies have the same target, is not allowed. In many implementations of dependency grammar, this is indeed the case. However, this often leads to rather inadequate treatments of certain constructions, and we will see in the discussion of control and raising below that I don't follow this restriction there. That is something of a technical abuse, but as long as we never refer to $\hat{\bullet}$ in a context where a node might have multiple mothers, it is a harmless one.

$$\dfrac{\dfrac{\textbf{[}\texttt{amod}\textbf{]}\quad \lambda P.\lambda Q.\lambda x.P(x)\wedge Q(x) : [E(s)\multimap T(s)]\multimap[[E(m)\multimap T(m)]\multimap E(m)\multimap T(m)] \qquad \textbf{[}\textit{scary}\textbf{]}\quad \lambda x.\mathbf{scary}(x) : E(s)\multimap T(s)}{\lambda Q.\lambda x.\mathbf{scary}(x)\wedge Q(x) : [E(m)\multimap T(m)]\multimap E(m)\multimap T(m)} \qquad \textbf{[}\textit{monster}\textbf{]}\quad \lambda x.\mathbf{monster}(x) : E(m)\multimap T(m)}{\lambda x.\mathbf{scary}(x)\wedge\mathbf{monster}(x) : E(m)\multimap T(m)}$$

Figure 3: Glue proof for *scary monster*

- Exactly the same process can be applied to other kinds of modifiers like PPs.
- For example, we will say an NP like *librarian with a secret* has the following parse:[3]

(9)



- The relevant meaning constructors are given below. We use exactly the same meaning constructor for `pmod` as we did for `amod`.

(10)
$$\begin{aligned}
\textit{librarian} &\rightsquigarrow \lambda x.\mathbf{librarian}(x) : E(\bullet)\multimap T(\bullet)\\
\texttt{pmod} &\rightsquigarrow \lambda P.\lambda Q.\lambda x.P(x)\wedge Q(x) : [E(\bullet)\multimap T(\bullet)]\multimap[[E(\hat{\bullet})\multimap T(\hat{\bullet})]\multimap E(\hat{\bullet})\multimap T(\hat{\bullet})]\\
\textit{with} &\rightsquigarrow \lambda y.\lambda x.\mathbf{with}(x,y) : E(\bullet\ \texttt{obj})\multimap[E(\bullet)\multimap T(\bullet)]\\
\textit{a} &\rightsquigarrow \lambda P.\lambda Q.\exists x.P(x)\wedge Q(x) : [E(\hat{\bullet})\multimap T(\hat{\bullet})]\multimap\forall\alpha.[[E(\hat{\bullet})\multimap T(\alpha)]\multimap T(\alpha)]\\
\textit{secret} &\rightsquigarrow \lambda x.\mathbf{secret}(x) : E(\bullet)\multimap T(\bullet)
\end{aligned}$$

- To save space in the main proof, we pre-combine *a* with *secret*:

$$\dfrac{\textbf{[}\textit{a}\textbf{]}\quad \lambda P.\lambda Q.\exists x.P(x)\wedge Q(x) : [E(s)\multimap T(s)]\multimap\forall\alpha.[[E(s)\multimap T(\alpha)]\multimap T(\alpha)] \qquad \textbf{[}\textit{secret}\textbf{]}\quad \lambda x.\mathbf{secret}(x) : E(s)\multimap T(s)}{\lambda Q.\exists x.\mathbf{secret}(x)\wedge Q(x) : \forall\alpha.[E(s)\multimap T(\alpha)]\multimap T(\alpha)}$$

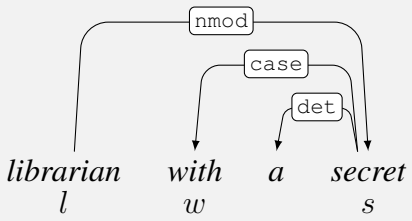
Figure 4: Combining *a* with *secret*

[3] We depart from the UD convention here, but stick with more traditional dependency grammar approaches, by treating the preposition *with* as the head of the modifier rather than *secret*.

- Now the proof proceeds as shown in Figure 5, on the following page.
- Relative clauses are also modifiers of nouns, and will therefore have the same $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ type, but we will explore those in more detail in Section 4.

Exercise 2: Modifying meaning constructors

In the Universal Dependencies framework, the parse for *librarian with a secret* would look different – *with* would be the dependent of *secret* instead of the other way around:



Which of the meaning constructors in (10) would we need to change, and how would we need to change them, in order to accommodate this alternative syntactic analysis?

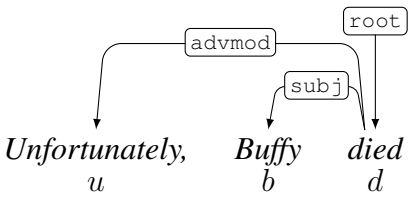
2.2 Sentential modifiers

- Sentential modifiers can be straightforwardly treated as providing type $\langle t,t\rangle$ meanings.
- Of course, there are also intensional adverbs like *probably*, *possibly*, *necessarily*, etc., which would require the introduction of meanings relativised to possible worlds. We won't touch on intensional semantics here, but Glue can easily be extended to handle such a move.
- Sticking to the extensional domain, an adverb like *unfortunately* has the following meaning constructor:

(11) $\lambda p.\mathbf{unfortunately}(p) : T(\hat{\bullet}) \multimap T(\hat{\bullet})$

- In a dependency grammar, sentential adverbs will be dependents of the head of the clause:

(12)



$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\lambda x.\lambda y.\mathbf{with}(x,y) : E(w) \multimap [E(s) \multimap T(w)]}\textbf{[with]}
\qquad
\begin{bmatrix} x : \\ E(w) \end{bmatrix}^1
}{\lambda y.\mathbf{with}(x,y) : E(s) \multimap T(w)}
\qquad
\dfrac{}{\lambda Q.\exists y.\mathbf{secret}(y) \wedge Q(y) : [E(s) \multimap T(w)] \multimap T(w)}\textbf{[a secret]}
}{\exists y.\mathbf{secret}(y) \wedge \mathbf{with}(x,y) : T(w)}
}{\lambda x.\exists y.\mathbf{secret}(y) \wedge \mathbf{with}(x,y) : E(w) \multimap T(w)}\multimap_{\mathcal{I},1}
\qquad
\dfrac{}{\lambda x.\mathbf{librarian}(x) : E(l) \multimap T(l)}\textbf{[librarian]}
}{\lambda Q.\lambda x.Q(x) \wedge \mathbf{with}(x,y) : [E(l) \multimap T(l)] \multimap E(l) \multimap T(l)}
}{\lambda x.\mathbf{librarian}(x) \wedge \exists y.\mathbf{secret}(y) \wedge \mathbf{with}(x,y) : E(l) \multimap T(l)}
$$

$$
\dfrac{
\dfrac{}{\lambda P.\lambda Q.\lambda x.Q(x) \wedge P(x) : [E(w) \multimap T(w)] \multimap [[E(l) \multimap T(l)] \multimap E(l) \multimap T(l)]}\textbf{[pmod]}
}{\lambda Q.\lambda x.Q(x) \wedge \exists y.\mathbf{secret}(y) \wedge \mathbf{with}(x,y) : [E(l) \multimap T(l)] \multimap E(l) \multimap T(l)}
$$

Figure 5: Glue proof for *librarian with a secret*

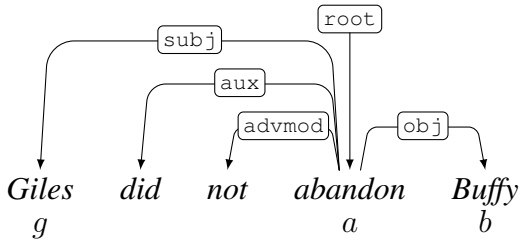
Exercise 3: Sentential adverbs

Give a Glue proof for the sentence in (12).

- Basing the connection to the syntax on more abstract relations also helps with things like negation: while *not* might appear clause-internally, we want it to apply to the sentence as a whole (at least on one straightforward reading). This isn't a problem in Glue:

(13) $\textit{not} \quad \rightsquigarrow \quad \lambda p.\neg p : T(\hat{\bullet}) \multimap T(\hat{\bullet})$

(14)



Exercise 4: Negation

Give a Glue proof for the sentence in (14).

(Assume *did* makes no semantic contribution, since we are not worrying about tense or aspect.)

2.3 VP modifiers

- We delay discussion of VP modifiers (manner adverbials, etc.) until tomorrow, when we introduce event semantics.

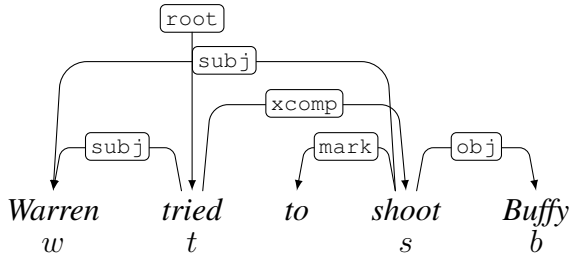
3 Control and raising

- A potential challenge to resource sensitivity comes from CONTROL verbs.
- These are verbs like English *try* or *promise*, where a single phrase is a semantic argument of two verbs at the same time:[4,5]

[4]The label `xcomp` is used for an 'open' complement clause, i.e. one which has an argument that must be identified with some other element in the clause.

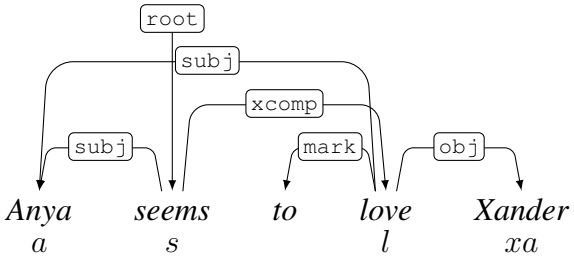
[5]Having two dependencies pointing to the same word means that the dependency parse is no longer merely a tree, but rather a more general graph structure. To retain the attractive computational/mathematical properties of trees, dependency grammars often omit certain kinds of annotations, such as the controlled subjects of open complement clauses. In the Universal Dependencies project, such richer annotations are optionally encoded in an alternative 'Enhanced' representation (Schuster & Manning 2016), although comparatively few UD treebanks actually make use of this (Findlay & Haug 2021).

(15)



- *Warren* is a semantic argument of both *tried* (he is the one trying) and *shoot* (his goal is to make *Warren shot Buffy* true) at the same time.
- Control verbs differ from the syntactically similar RAISING verbs, where the shared phrase is only a semantic argument of the downstairs predicate:

(16)



- *Anya* is a syntactic argument of both *seems* and *love*, but is only a semantic argument of the latter.
- One way to bring out this difference is in the behaviour of these verbs with respect to expletives or idiom chunks (Bresnan 1982: 76f.):

(17) a. *It seemed to rain.*
b. **It tried to rain.*

(18) a. *Strings seemed to be pulled for her.*
b. **Strings tried to be pulled for her.*

- Another is in the possibility of using an expletive subject and a finite clause with raising verbs, since their subject is not a semantic argument, vs. the impossibility of this with control verbs, since their subject *is* a semantic argument:

(19) *It seems that Anya loves Xander.*

(20) **It tried that Warren shot Buffy.*

- We can represent the different status of the subject in terms of the translations of such sentences into predicate logic:

(21) $\mathbf{seems}(\mathbf{love}(\mathbf{anya}, \mathbf{xander}))$

(22) $\mathbf{tried}(\mathbf{warren}, \mathbf{shoot}(\mathbf{warren}, \mathbf{buffy}))$

- That is, raising verbs take a single, propositional, argument, corresponding to the embedded clause, while control verbs take two arguments, one corresponding to the embedded clause and one corresponding to their subject.
- For raising verbs, there is no problem of resource sensitivity, since only the downstairs verb will consume the resource corresponding to the subject; the raising verb itself simply consumes that proposition when it is saturated.
- A verb like *seems* can thus have the following meaning constructor:

(23) $\lambda p.\mathbf{seems}(p) : T(\bullet\ \texttt{xcomp}) \multimap T(\bullet)$

- The control verbs, though, pose a *prima facie* challenge, since it would seem that both the downstairs *and* upstairs verbs want to consume one and the same resource corresponding to the subject, which would result in resource deficit.
- In fact, Asudeh (2005) shows there is a quite straightforward solution to this challenge, owing once again to Glue's separation of the syntax of composition from syntax proper.
- The meaning constructor for *tried* is the one which consumes the subject's resource, but it also consumes the embedded predicate's dependency on that subject, and on the meaning side combines them:

(24) $\lambda x.\lambda P.\mathbf{tried}(x, P(x)) :$
$E(\bullet\ \texttt{subj}) \multimap [[E(\bullet\ \texttt{xcomp subj}) \multimap T(\bullet\ \texttt{xcomp})] \multimap T(\bullet)]$

- In this way, the subject/controller can be a semantic argument of both predicates, even though compositionally speaking it is only the control verb which consumes it.

Exercise 5: Control and raising

Give a Glue proof for each of the sentences in (15) and (16).

- These meaning constructors also give the right results when it comes to scope interactions.
- Observe that raising and control verbs also differ with respect to whether they allow wide or narrow scope readings of quantifiers in subject position:

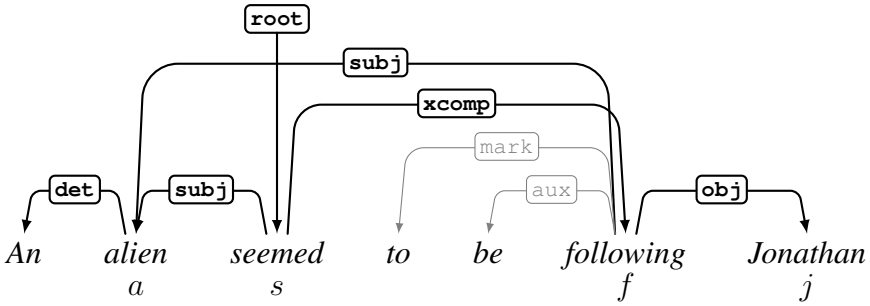
(25) *An alien seemed to be following Jonathan.*
This can mean (a) or (b):[6]
a. $\exists x.\mathbf{alien}(x) \wedge \mathbf{seemed}(\mathbf{follow}(x, \mathbf{jonathan}))$
b. $\mathbf{seemed}(\exists x.\mathbf{alien}(x) \wedge \mathbf{follow}(x, \mathbf{jonathan}))$

[6]For the sake of simplicity, we ignore the continuous aspect of *be following*.

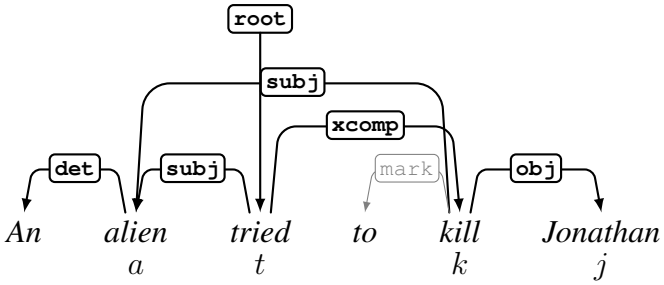
(26) *An alien tried to kill Jonathan.*
This can only mean (a):
a. $\exists x.\mathbf{alien}(x) \wedge \mathbf{tried}(x, \mathbf{kill}(x, \mathbf{jonathan}))$

- In both (25a) and (26a), the existence of an alien is asserted.
- In (25b), no aliens need exist: this reading is equivalent to that expressed by *It seemed that an alien was following Jonathan*.
- Put differently, raising verbs allow both a DE RE and a DE DICTO reading of their subject, while control verbs only allow a DE RE reading.
- This follows straightforwardly from the meaning constructors for the two verbs. The basic intuition is this:
  - In the raising but not the control context, the embedded verb actually needs to consume its subject. This means that the quantifier can scope at that point, *or* we can use hypothetical reasoning to postpone the dependency until the higher clause.
  - In the control context, though, it is the upstairs verb that consumes the subject, and so if the quantifier were to scope below it, there would be no way to discharge the matrix verb's dependency on the subject.
- The proofs are given in Figures 7–10; to save space, we once again pre-combine the determiner and the noun, the proof for which is shown in Figure 6.
- The relevant parses are shown in (27)–(28); the common components are highlighted, to show again that the syntactic structures of both raising and control verbs are parallel.
- The instantiated meaning constructors are shown in (29).

(27)



(28)



(29)

$$
\begin{aligned}
\textit{an} &\rightsquigarrow \lambda P.\lambda Q.\exists x.P(x) \wedge Q(x) : \\
&\quad [E(a) \multimap T(a)] \multimap \forall \beta.[[E(a) \multimap T(\beta)] \multimap T(\beta)] \\
\textit{alien} &\rightsquigarrow \lambda x.\mathbf{alien}(x) : \\
&\quad E(a) \multimap T(a) \\
\textit{seemed} &\rightsquigarrow \lambda p.\mathbf{seemed}(p) : \\
&\quad T(f) \multimap T(s) \\
\textit{tried} &\rightsquigarrow \lambda P.\lambda x.\mathbf{tried}(x, P(x)) : \\
&\quad [E(a) \multimap T(k)] \multimap [E(a) \multimap T(t)] \\
\textit{following} &\rightsquigarrow \lambda y.\lambda x.\mathbf{follow}(x, y) : \\
&\quad E(j) \multimap [E(a) \multimap T(f)] \\
\textit{kill} &\rightsquigarrow \lambda y.\lambda x.\mathbf{kill}(x, y) : \\
&\quad E(j) \multimap [E(a) \multimap T(k)] \\
\textit{Jonathan} &\rightsquigarrow \mathbf{jonathan} : \\
&\quad E(j)
\end{aligned}
$$

[*an*]	**[*alien*]**
$\lambda P.\lambda Q.\exists x.P(x) \wedge Q(x) :$	$\lambda x.\mathbf{alien}(x) :$
$[E(a) \multimap T(a)] \multimap \forall \beta.[[E(a) \multimap T(\beta)] \multimap T(\beta)]$	$E(a) \multimap T(a)$

$$\lambda Q.\exists x.\mathbf{alien}(x) \wedge Q(x) : \forall \beta.[E(a) \multimap T(\beta)] \multimap T(\beta)$$

Figure 6: Combining *an* with *alien*

$$\dfrac{\dfrac{\dfrac{\dfrac{\textbf{[}\textit{following}\textbf{]}\quad \lambda y.\lambda x.\mathbf{follow}(x,y) : E(j) \multimap [E(a) \multimap T(f)] \qquad \textbf{[}\textit{Jonathan}\textbf{]}\quad \mathbf{jonathan} : E(j)}{\lambda x.\mathbf{follow}(x,\mathbf{jonathan}) : E(a) \multimap T(f)} \qquad \left[\begin{array}{l} x : \\ E(a) \end{array}\right]^1}{\mathbf{follow}(x,\mathbf{jonathan}) : T(f)} \qquad \textbf{[}\textit{seemed}\textbf{]}\quad \lambda p.\mathbf{seemed}(p) : T(f) \multimap T(s)}{\mathbf{seemed}(\mathbf{follow}(x,\mathbf{jonathan})) : T(s)}}{\lambda x.\mathbf{seemed}(\mathbf{follow}(x,\mathbf{jonathan})) : E(a) \multimap T(s)}\ \multimap_{\mathcal{I},1}$$

$$\dfrac{\textbf{[}\textit{an alien}\textbf{]}\quad \lambda Q.\exists x.\mathbf{alien}(x) \wedge Q(x) : [E(a) \multimap T(s)] \multimap T(s) \qquad \lambda x.\mathbf{seemed}(\mathbf{follow}(x,\mathbf{jonathan})) : E(a) \multimap T(s)}{\exists x.\mathbf{alien}(x) \wedge \mathbf{seemed}(\mathbf{follow}(x,\mathbf{jonathan})) : T(s)}$$

Figure 7: Glue proof for the wide scope (*de re*) reading of *An alien seemed to be following Jonathan*

$$\dfrac{\dfrac{\dfrac{\textbf{[}\textit{following}\textbf{]}\quad \lambda y.\lambda x.\mathbf{follow}(x,y) : E(j) \multimap [E(a) \multimap T(f)] \qquad \textbf{[}\textit{Jonathan}\textbf{]}\quad \mathbf{jonathan} : E(j)}{\lambda x.\mathbf{follow}(x,\mathbf{jonathan}) : E(a) \multimap T(f)} \qquad \textbf{[}\textit{an alien}\textbf{]}\quad \lambda Q.\exists x.\mathbf{alien}(x) \wedge Q(x) : [E(a) \multimap T(f)] \multimap T(f)}{\exists x.\mathbf{alien}(x) \wedge \mathbf{follow}(x,\mathbf{jonathan}) : T(f)} \qquad \textbf{[}\textit{seemed}\textbf{]}\quad \lambda p.\mathbf{seemed}(p) : T(f) \multimap T(s)}{\mathbf{seemed}(\exists x.\mathbf{alien}(x) \wedge \mathbf{follow}(x,\mathbf{jonathan})) : T(s)}$$

Figure 8: Glue proof for the narrow scope (*de dicto*) reading of *An alien seemed to be following Jonathan*

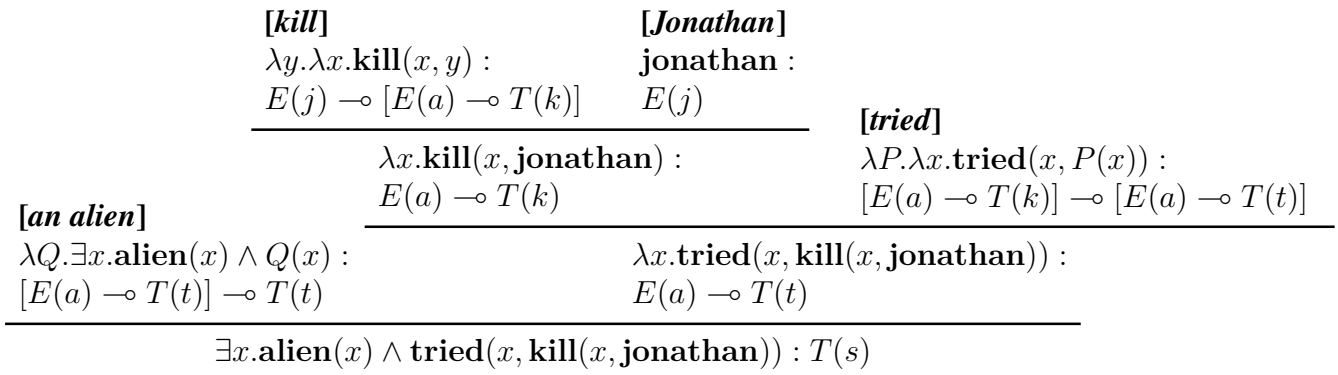


Figure 9: Glue proof for the wide scope (*de re*) reading of *An alien tried to kill Jonathan*

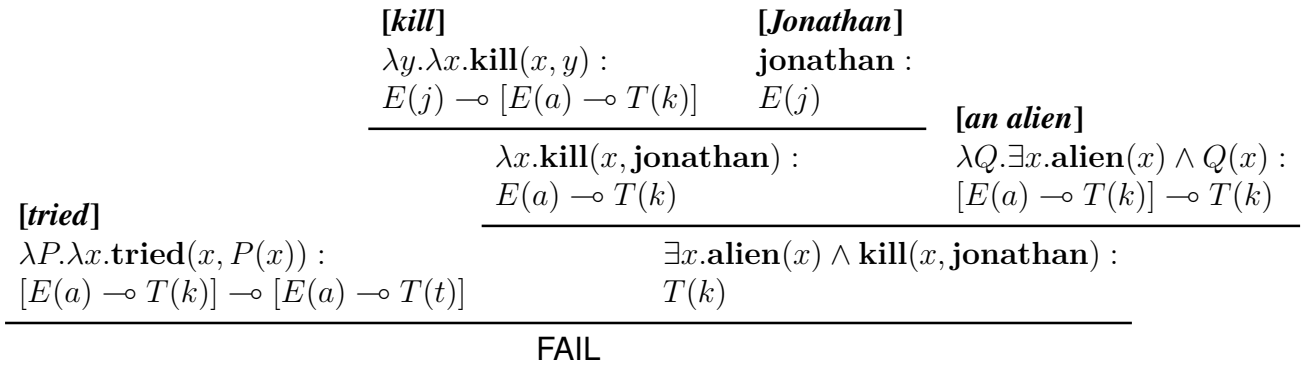
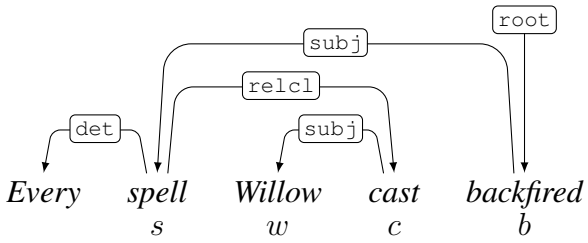


Figure 10: Attempted Glue proof for a non-existent narrow scope (*de dicto*) reading of *An alien tried to kill Jonathan*

4 Long-distance dependencies

- Glue offers an elegant analysis of LDDs which does not require the presence of empty elements in the syntax, or of movement.
- Here is the dependency parse for a sentence involving a relative clause:

(30)

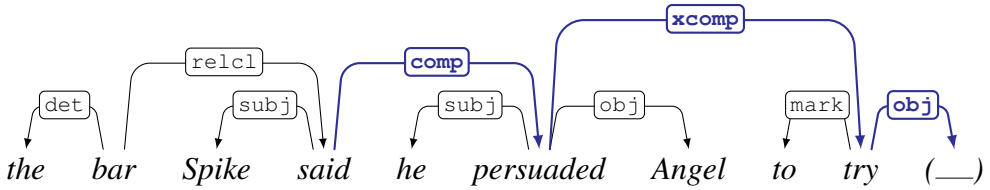


- The dependency `relcl` is a relation between a noun and the head of a relative clause that modifies it.
- Since relative clauses denote predicates and we want to combine them with the head noun they modify to create a new predicate, we once again introduce a constructional meaning along the same lines as we did for `amod`/`pmod` earlier.
- This time it is triggered by the presence of the `relcl` relation, and it is slightly different in form:

(31) $\texttt{relcl} \rightsquigarrow \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)$
$\forall\alpha.[E(\alpha) \multimap T(\bullet)] \multimap [E(\hat{\bullet}) \multimap T(\hat{\bullet})] \multimap E(\hat{\bullet}) \multimap T(\hat{\bullet})$

- The meaning side is the same as we saw above for `amod`/`pmod`, but the Glue side is more complex.
- This time we quantify over the initial type e dependency, which here corresponds to the gapped argument, because it can be potentially highly embedded.
- For example, in (32), the path from *said* to the gap is given by the expression in (33):

(32)



(33) $\bullet$ `comp xcomp obj`

- Importantly, we do not require there to actually be an empty node in the syntactic representation where the gap is. All that matters is that it matches the missing argument in the verb's meaning constructor.
- We can pick any arbitrary label to represent this hypothetical node, so long as we pick the same one in both the verb's meaning constructor and the relative clause meaning constructor.

- Figure 11 shows the derivation of the sentence in (30), using the meaning constructors in (34):

(34)
$$\begin{aligned}
\textit{every} &\rightsquigarrow \lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x) : \\
&\quad [E(s) \multimap T(s)] \multimap \forall\alpha.[[E(s) \multimap T(\alpha)] \multimap T(\alpha)] \\
\textit{spell} &\rightsquigarrow \lambda x.\mathbf{spell}(x) : \\
&\quad E(s) \multimap T(s) \\
\textit{Willow} &\rightsquigarrow \mathbf{willow} : \\
&\quad E(w) \\
\textit{cast} &\rightsquigarrow \lambda y.\lambda x.\mathbf{cast}(x, y) : \\
&\quad E(obj) \multimap [E(w) \multimap T(c)] \\
\texttt{relcl} &\rightsquigarrow \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x) : \\
&\quad \forall\alpha.[E(\alpha) \multimap T(c)] \multimap [E(s) \multimap T(s)] \multimap E(s) \multimap T(s) \\
\textit{backfired} &\rightsquigarrow \lambda x.\mathbf{backfired}(x) : \\
&\quad E(s) \multimap T(b)
\end{aligned}$$

- We assume that any relative pronoun or complementiser is semantically empty, to preserve the same analysis across relative clauses with and without an overt relative pronoun/ complementiser.[7]

5 Further reading

- We have touched on a handful of empirical topics here, but there are many more that have been given a treatment in Glue Semantics.
- Part III of Dalrymple et al. (2019) presents Glue analyses of a number of different phenomena, several of which we have discussed, but some which we have not.
- On **anaphora**, for example, see Dalrymple et al. (2019: ch. 14). In addition, Dalrymple et al. (1999) is the *locus classicus* of work on anaphora in Glue; see Asudeh (2012) for a modern implementation of some of the same ideas. However, contemporary work in LFG tends to treat anaphora as involving mostly syntax and pragmatics, rather than semantics – see Dalrymple et al. (2018) for a recent example.
- When it comes to **coordination**, Dalrymple et al. (2019: ch. 16) is mostly a presentation of Asudeh & Crouch (2002), the most thorough and detailed work on coordination in Glue.

[7] Although the relative pronoun *who* does contribute the information that the gap is a person – see Dalrymple (2001: 416–422) for an analysis in an LFG setting which includes such an interpretation for *who*.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\textbf{willow} : E(w)}\,\textbf{[Willow]}
\qquad
\dfrac{}{\begin{array}{l}\lambda y.\lambda x.\mathbf{cast}(x,y) : \\ E(obj) \multimap [E(w) \multimap T(c)]\end{array}}\,\textbf{[cast]}
}{\begin{array}{l}\mathbf{cast}(\mathbf{willow},y) : \\ T(c)\end{array}}
\qquad
\dfrac{}{\begin{array}{l}y : \\ E(obj)\end{array}}\,{}^{1}
}{\begin{array}{l}\lambda y.\mathbf{cast}(\mathbf{willow},y) : \\ E(obj) \multimap T(c)\end{array}}\,{}^{\multimap_{\mathcal{I},1}}
\qquad
\dfrac{}{\begin{array}{l}\lambda x.\mathbf{spell}(x) : \\ E(s) \multimap T(s)\end{array}}\,\textbf{[spell]}
}{\begin{array}{l}\lambda Q.\lambda x.Q(x) \wedge \mathbf{cast}(\mathbf{willow},x) : \\ [E(s) \multimap T(s)] \multimap E(s) \multimap T(s)\end{array}}
\qquad
\dfrac{}{\begin{array}{l}\lambda P.\lambda Q.\lambda x.Q(x) \wedge P(x) : \\ [E(obj) \multimap T(c)] \multimap [E(s) \multimap T(s)] \multimap E(s) \multimap T(s)\end{array}}\,\textbf{[relcl]}
}{\begin{array}{l}\lambda P.\lambda Q.\lambda x.Q(x) \wedge P(x) : \\ \forall \alpha.[E(\alpha) \multimap T(c)] \multimap [E(s) \multimap T(s)] \multimap E(s) \multimap T(s)\end{array}}\,\forall_{\mathcal{E}}
}{\begin{array}{l}\lambda x.\mathbf{backfired}(x) : \\ E(s) \multimap T(c)\end{array}}\,\textbf{[backfired]}
}{\begin{array}{l}\lambda x.\mathbf{spell}(x) \wedge \mathbf{cast}(\mathbf{willow},x) : \\ E(s) \multimap T(s)\end{array}}
}{\begin{array}{l}\lambda Q.\forall x.[\mathbf{spell}(x) \wedge \mathbf{cast}(\mathbf{willow},x)] \to Q(x) : \\ [E(s) \multimap T(c)] \multimap T(c)\end{array}}\,\textbf{[every]}
}{\begin{array}{l}\forall x.[\mathbf{spell}(x) \wedge \mathbf{cast}(\mathbf{willow},x)] \to \mathbf{backfired}(x) : \\ T(c)\end{array}}
$$

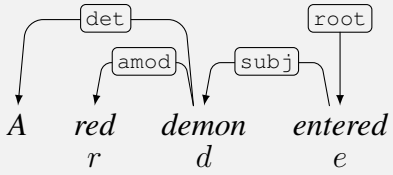
Figure 11: Derivation for *Every spell Willow cast backfired*

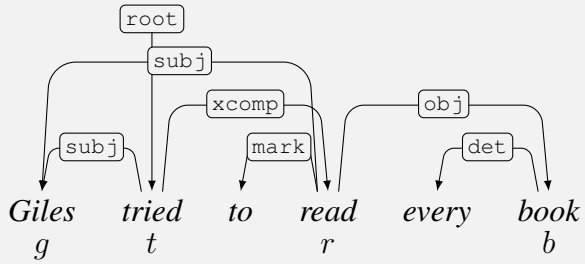
Exercise 6: Putting it together

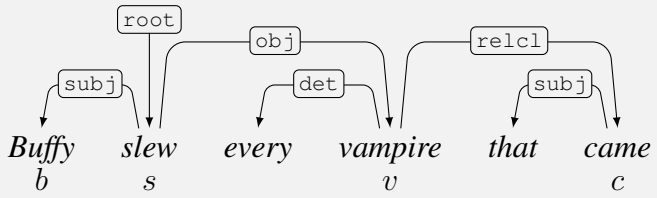
Give a Glue proof for each of the following sentences. In each case, you are given a dependency parse, but must provide meaning constructors yourself.

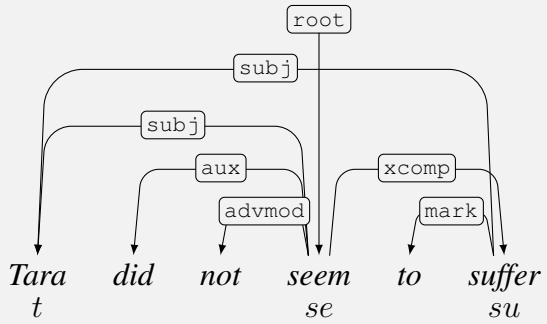
Do not necessarily assume that every word should contribute a meaning constructor. Some dependency relations may also contribute a meaning constructor, but not all of them need to.

Treat *to* and *that* as semantically vacuous.

1. 

2. 

3. 

4. 

References

Ajdukiewicz, Kazimierz. 1935. Die syntaktische Konnexität. *Studia Philosophica (Warszawa)* 1. 1–28. English translation by H. Weber available in McCall, Storrs (ed.). 1967. *Polish logic 1920–1939*. Oxford: Oxford University Press.

Asudeh, Ash. 2004. *Resumption as resource management*. Ph.D. thesis, Stanford University.

Asudeh, Ash. 2005. Control and semantic resource sensitivity. *Journal of Linguistics* 41(3). 465–511. https://doi.org/10.1017/S0022226705003427.

Asudeh, Ash. 2012. *The logic of pronominal resumption*. Oxford: Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199206421.001.0001.

Asudeh, Ash & Richard Crouch. 2002. Coordination and parallelism in Glue Semantics: integrating discourse cohesion and the Element Constraint. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG02 Conference*, 19–39. Stanford, CA: CSLI Publications. https://typo.uni-konstanz.de/lfg-proceedings/LFGprocCSLI/LFG2002/pdfs/lfg02asudehcrouch-num.pdf.

Bar-Hillel, Yehoshua. 1953. A quasi-mathematical notation for syntactic description. *Language* 29(1). 47–58. https://www.jstor.org/stable/410452.

Bresnan, Joan. 1982. The passive in lexical theory. In Joan Bresnan (ed.), *The mental representation of grammatical relations*, 3–86. Cambridge, MA: MIT Press.

Chomsky, Noam. 1981. *Lectures on government and binding*. Dordrecht: Foris.

Chomsky, Noam. 1986. *Knowledge of language: its nature, origin, and use.* New York, NY: Praeger.

Dalrymple, Mary. 2001. *Lexical Functional Grammar* (Syntax and Semantics 34). San Diego, CA: Academic Press.

Dalrymple, Mary, Dag T. T. Haug & John J. Lowe. 2018. Integrating LFG's binding theory with PCDRT. *Journal of Language Modelling* 6(1). 87–129. https://doi.org/10.15398/jlm.v6i1.204.

Dalrymple, Mary, John Lamping & Fernando Pereira Vijay Saraswat. 1999. Quantification, anaphora, and intensionality. In Mary Dalrymple (ed.), *Semantics and syntax in Lexical Functional Grammar: the resource logic approach*, 39–90. Cambridge, MA: MIT Press.

Dalrymple, Mary, John J. Lowe & Louise Mycock. 2019. *The Oxford reference guide to Lexical Functional Grammar*. Oxford: Oxford University Press. https://doi.org/10.1093/oso/9780198733300.001.0001.

Findlay, Jamie Y. & Dag T. T. Haug. 2021. How useful are Enhanced Universal Dependencies for semantic interpretation? In *Proceedings of the Sixth International Conference on Dependency Linguistics (Depling, SyntaxFest 2021)*, 22–34. Sofia: Association for Computational Linguistics. https://aclanthology.org/2021.depling-1.3.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar* (Blackwell Textbooks in Linguistics 13). Oxford: Blackwell.

Kaplan, Ronald M. & 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.

Klein, Ewan & Ivan A. Sag. 1985. Type-driven translation. *Linguistics and Philosophy* 8(2). 163–201. https://doi.org/10.1007/BF00632365.

Lambek, Joachim. 1958. The mathematics of sentence structure. *American Mathematical Monthly* 65(3). 154–170. https://doi.org/10.1080/00029890.1958.11989160.

Schuster, Sebastian & Christopher D. Manning. 2016. Enhanced English Universal Dependencies: an improved representation for natural language understanding tasks. In *Proceedings of the tenth international conference on language resources and evaluation (lrec'16)*, https://aclanthology.org/L16-1376/.

Steedman, Mark & Jason Baldridge. 2011. Combinatory categorial grammar. In Robert D. Borsley & Kersti Börjars (eds.), *Non-transformational syntax: formal and explicit models of grammar*, 181–224. Oxford: Wiley-Blackwell. https://doi.org/10.1002/9781444395037.ch5.

Wheeler, Deirdre. 1988. Consequences of some categorially-motivated phonological assumptions. In Richard T. Oehrle, Emmon Bach & Deirdre Wheeler (eds.), *Categorial grammars and natural language structures* (Studies in Linguistics and Philosophy 32), 467–488. Dordrecht: Springer. https://doi.org/10.1007/978-94-015-6878-4_17.