POTTSIAN LFG

Doug Arnold and Louisa Sadler
University of Essex

Proceedings of the LFG10 Conference

Miriam Butt and Tracy Holloway King (Editors)

2010

CSLI Publications

http://csli-publications.stanford.edu/

1 Introduction

Constructions that one might broadly call ‘parenthetical’ (including various kinds of appositive construction) have received very little attention in the LFG literature. This is surprising because, taken broadly, parentheticals include a wide range of rather common constructions including supplemental/appositional constructions, expressives, and ’pure’ parentheticals of various sorts. Apart from some brief coverage in Butt et al. (1999), the most extensive discussion of such constructions probably occurs in a series of papers in which Fortmann looks at some parenthetical constructions in German (Fortmann, 2005, 2006). Fortmann’s concern is largely syntactic: in Fortmann (2005) he argues that parentheticals such as (1) are ‘non-regular’ constituents in that they are integrated into the c-structure, but not the f-structure, of the host clause.[1]

(1)

Theo	hat	—	ein	Klempner	war	nicht	zu	erreichen	—	den	Rohrbruch
Theo	has	—	a	plumber	was	not	to	reach	—	the	pipe burst
selbst	repariert.										
self	fixed										

However, Fortmann (2006) shows that the structural integration facts are different for different kinds of parenthetical, specifically that at least some *verba dicendi* parentheticals are fully integrated at both c- and f-structure.

Outside LFG there has been a considerable amount of work on the syntax of parentheticals and supplemental/appositive constructions, a major focus being whether or not parenthetical elements should be analysed as syntactically integrated or syntactically orphaned (see Arnold (2007) for references). Beyond purely syntactic accounts, an important contribution is made by Potts (2005). Potts provides a very appealing account of the semantics of a range of parenthetical expressions, including supplemental/appositive (non-restrictive) relative clauses, and expressives (e.g. *damn Republican*).

In the current paper we explore whether an approach to the analysis of supplementals and appositions inspired by the work of Potts (2005) can be expressed in the framework of LFG. We concentrate on ARCs, and focus on how Potts’ approach can be implemented using a resource sensitive glue approach to semantic composition (see e.g. Dalrymple, 1999; Asudeh, 2004). An LFG account of the syntax and semantics of English ARCs is a side effect of our discussion (no account seems to have been previously proposed).

The paper is structured as follows. Section 2 sets out the basic properties of English ARCs. Section 3 outlines Potts’ approach. In Section 4 we describe

[†]We are grateful to the participants at LFG10 in Ottawa, Canada, for insightful and stimulating discussion, and in particular to Ash Asudeh and Mary Dalrymple. We should also thank Tracy Holloway-King for detailed comments on a earlier version of the paper.

[1]Fortmann does not provide an English translation, but it would presumably be along the line of ‘No plumber being available, Theo has fixed the burst pipe himself’.

what seems to be the most direct implementation of Potts' ideas, building on the standard LFG analysis of restrictive relative clauses from Dalrymple (2001). In Section 5 we consider an alternative analysis that makes more use of the LFG projection architecture. Section 6 provides a conclusion and notes some open questions.

2 ARCs: The Facts

As is well known, supplemental or appositive relative clauses (ARCs), such as in (2), are distinguishable from restrictive relative clauses (RRCs), such as in (3), on a number of grounds. Phonologically, an ARC as in (2) is often associated with special 'comma' intonation, which is not typical of an RRC such as in (3). Semantically, in (3) the relative clause restricts the denotation of the noun *pets*, in (2) it does not, instead it simply provides additional or supplementary information about the pets (hence the terms 'non-restrictive' or 'supplementary' relative clause). As a consequence a restrictive can give rise to a 'contrast set' of entities that are in the denotation of the head noun, but do not satisfy the relative clause. Thus there is a potential antecedent for *the others* in (3), while there is no antecedent for *the others* in (2).

(2) Kim has three pets, which a neighbour looks after.
#The others fend for themselves. [ARC]
(3) Kim has three pets which a neighbour looks after.
The others fend for themselves. [RRC]

English ARCs are subject to a number of surface morphosyntactic restrictions, in particular, they must be finite and +WH, as illustrated by the contrast between (4) and (5)-(6). The corresponding RRCs are acceptable, witness (7)-(8).

(4) Kim, who Sandy spoke to, will arrive later.
(5) *Kim, for Sandy to speak to, will arrive later.
(6) *Kim, that Sandy spoke to, will arrive later.

(7) A person for Sandy to speak to will arrive later.
(8) A person that Sandy invited will arrive later.

In the remainder of this section, we will establish two key properties of ARCs in English. These are (i) that they show semantic 'wide scope' effects; and (ii) that they are syntactically integrated, at both c- and f-structure.[2]

The existence of 'wide scope' effects with ARCs is relatively well known. Consider the pair (9a)-(9b). A reading of (9a) is available in which Sandy's aunt (also) lives in Sweden, i.e. where the content of the relative clause (roughly 'she lives in Sweden') is part of the antecedent of *so*; but (9b) provides no information about where Sandy's mother lives – the ARC is

[2]See Arnold (2007) and references there for more discussion of the empirical issues.

not part of the antecedent of *so*. Similarly, in (10b) *who use the IPA* scopes outside Kim's belief set (Kim's belief is about linguists in general), while in (10a) Kim's beliefs concern those linguists who use the IPA. In (11b) *who use the IPA* similarly falls outside the scope of the question.

(9) a. Kim is visiting her aunt who lives in Sweden, and so is Sandy. [RRC]
b. Kim is visiting her mother, who lives in Sweden, and so is Sandy. [ARC]

(10) a. Kim believes that linguists who use the IPA are clever. [RRC]
b. Kim believes that linguists, who use the IPA, are clever. [ARC]

(11) a. Are linguists who use the IPA invariably clever people? [RRC]
b. Are linguists, who use the IPA, invariably clever people? [ARC]

The existence of wide scope effects with ARCs is not controversial.[3] The syntactic status of ARCs is more problematic, though we think the emerging consensus favours analyses where they are fully integrated with few, if any, structural distinctions between ARCs and RRCs.

As regards c-structure, we believe the evidence that ARCs are integrated is overwhelming. First, a range of standard constituency tests show that the ARC and the NP antecedent form a constituent. For example, it is in general impossible to separate an ARC from its antecedent in a raising context (other 'movement' phenomena show exactly the same restriction):

(12) [Kim, who Sandy invited], seemed Δ to leave early.
(13) *Kim seemed, [Δ who Sandy invited], to leave early.

The same conclusion follows from the possibilities of placement for the possessive *'s* when the possessor is modified by an ARC. If we assume that possessive *'s* attaches to the right edge of NP, the following provide convincing evidence that an ARC and its antecedent together form a NP (i.e. they are integrated at c-structure). Notice also that the phonological form of the possessive varies according to the final segment of the ARC, as one would expect if the ARC is integrated.

(14) Kim – who you hit –'s mother (wants to talk). (/s/)
(15) Kim – who you hid –'s mother (wants to talk). (/z/)
(16) Kim – who you miss –'s mother (wants to talk). (/ɪz/)
(17) *Kim's – who you miss – mother (wants to talk).

The issue of f-structure is more subtle, but what evidence there is suggests that the ARC is also integrated at f-structure.

[3]But not entirely unproblematic. It is widely assumed that ARCs invariably take wide scope, but this is not the case. Arnold (2007) discusses circumstances where ARCs appear to take narrow scope, and Harris and Potts (2010) give convincing examples, bolstered with some persuasive experimental results, which indicate that ARCs can sometimes take narrow scope.

One argument in favour of this conclusion may be provided by data concerning conditions on VP ellipsis (VPE) in English. Potsdam (1997), following Warner (1993), observes that in ellipsis of a VP with an auxiliary verb, the antecedent VP must not be finite, so (18) is grammatical but (19) is ungrammatical. This shows that the resolution process that finds the antecedent of an ellided VP needs to check some morphosyntactic properties (f-structure attributes) of potential antecedents.

(18) You may not be confused, but you really should. [be confused]
(19) *You are not confused, but you really should. [be confused]

The key point for present purposes is how this fact about VPE interacts with ARCs. VPE is possible both into and out of ARCs and RRCs. The antecedent of VPE in an ARC can be inside a restrictive as in (20a), and the antecedent of VPE in a restrictive relative clause can be inside an ARC, as in (20b). As (20c) additionally shows, both antecedent and VPE can be within ARCs. Given that VPE is subject to morphosyntactic constraints, these data provide an important argument that ARCs are contained within (accessible within) whatever representation is relevant to the statement of morphosyntactic conditions such as those noted by Potsdam: in LFG this would be f-structure. The data in (21a)-(21f) support this conclusion by showing that the nonfiniteness requirement (on VPE) determines the grammaticality. Notice in particular that the pattern of judgements is the same regardless of whether the examples involve ARCs or RRCs.

(20) a. Someone that [supports the war]$_i$ insulted Kim, who doesn't Δ_i
b. Sandy, who [supports the war]$_i$, insulted someone that doesn't Δ_i
c. Sandy, who [supports the war]$_i$, insulted Kim, who doesn't Δ_i

(21) a. Someone that [may be worried]$_i$ told Kim, who really should Δ_i
b. *Someone that [is worried]$_i$ told Kim, who really should Δ_i
c. Sandy, who [may be worried]$_i$, told someone that really should Δ_i
d. *Sandy, who [is worried]$_i$, told someone that really should Δ_i
e. Sandy, who [may be worried]$_i$, insulted Kim, who really should Δ_i
f. *Sandy, who [is worried]$_i$, insulted Kim, who really should Δ_i

At very least, these data show that the facts about VPE will be easier to describe if ARCs are part of f-structure in the same way as RRCs.

A second argument for f-structure integration comes from the fact that Right-Node Raising operates out of both ARCs and restrictives alike, as shown in the following examples:

(22) Sam met someone [that buys Δ], and Kim met someone [that sells Δ], the pretentious garbage that some call Modern Art. RRC

(23) Sam met Sandy, [who buys Δ], and Kim met Leslie, [who sells Δ], the pretentious garbage that some call Modern Art. ARC

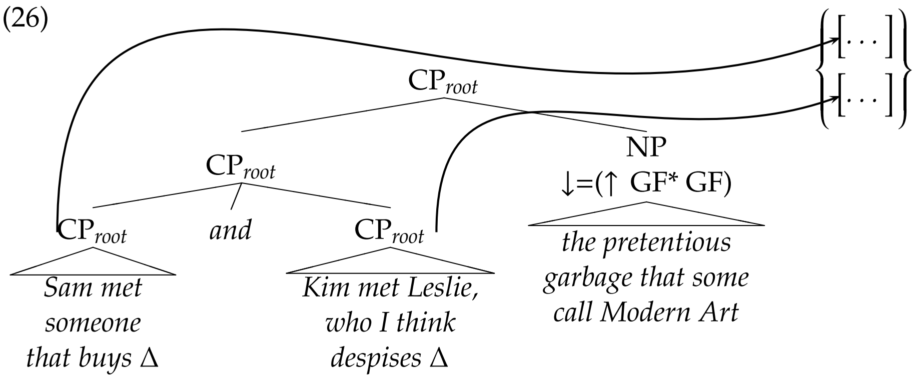
(24) a. Sam met someone [that buys Δ], and Kim met Leslie, [who sells Δ],

the pretentious garbage that some call Modern Art. RRC,ARC

b. Sam met Sandy, [who buys Δ], and Kim met someone [that sells Δ], the pretentious garbage that some call Modern Art. ARC,RRC

Simplifying considerably, one might have a structure for (25) such as (26). A functional uncertainty statement is associated with the RNR-ed NP (given here in maximally general form as ↓= (↑GF* GF)). Since ↑ is a set, the functional uncertainty may be solved differently in each member of the set (in each conjunct), but in each it will have to be a continuous path through the f-structure, for example, OBJ RELMOD OBJ in the first conjunct and OBJ RELMOD COMP OBJ in the second conjunct. Since it appears that these paths can reach indiscriminately into either ARCs or RRCs, the implication is that ARCs are integrated into the f-structure just as RRCs are — or at least, a description of RNR will be easier to formulate if ARCs are part of f-structure in the same way as RRCs.

(25) Sam met someone that buys Δ and Kim met Leslie, who I think despises Δ, the pretentious garbage that some call Modern Art.

(26)


CP_root
CP_root
NP
↓=(↑ GF* GF)
CP_root
and
CP_root
Sam met someone that buys Δ
Kim met Leslie, who I think despises Δ
the pretentious garbage that some call Modern Art
[...]
[...]

3 Potts' Approach

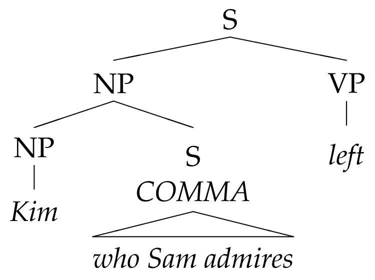
Potts' central idea is that the interpretation of an expression involves (at least) two dimensions: an *at-issue* dimension of normal truth-conditional content, and another dimension, which Potts argues to consist of Gricean conventional implicatures — the *ci* dimension. It is to this dimension that supplemental expressions, including ARCs, make their contribution. Since we assume that one could accept the idea of a special 'supplemental' dimension of meaning without accepting Potts' Gricean characterization of it, we will not pursue the latter topic, interesting though it is. We also focus only on ARCs, leaving aside other parentheticals and supplementals, but note that Potts' own proposal has much wider scope.

Potts' framework is type-theoretic, so the idea is that every expression is associated with a pair of meanings, and a pair of types $\langle \alpha^a, \gamma^c \rangle$, where α^a is an *at-issue* type, and γ^c is a *ci* type (note the superscripts). In what follows we will normally suppress the superscript on *at-issue* types, to avoid notational clutter. Semantic composition involves constructing a semantic parsetree, whose nodes are subject to a number of node-admissibility conditions. These are most easily understood via an example, such as (27).

(27) Kim, who Sam admires, left.

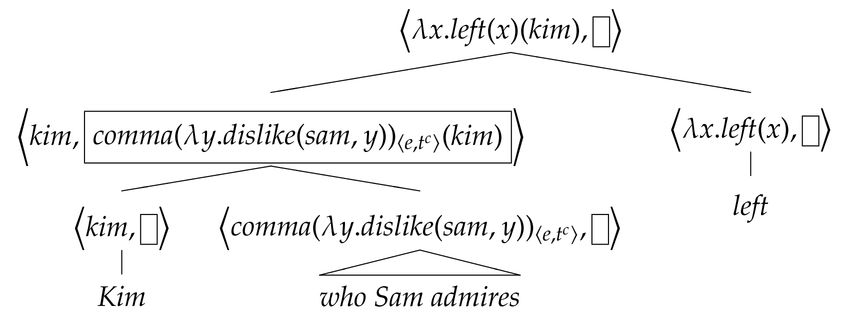
The basic syntactic structure Potts assumes is along the lines of (28) (modulo the node labels, e.g. Potts has DP where we have used NP). From the point of view of the discussion above, the only important points to note are (a) that the ARC is syntactically integrated, and (b) the presence of a COMMA feature on the ARC. This syntactic feature provides an interface between the phonology (comma intonation) and the semantics (*ci* type content).

(28)



The corresponding semantic tree is along the lines of (29), in which each node is associated with a pair denoting the *at-issue* and *ci* content, and where we have highlighted *ci* content by putting it in a box. Notice that most nodes have no *ci* content (the box is empty). We have only indicated the types of expressions where they are non-standard.

(29)


⟨λx.left(x)(kim), □⟩
⟨kim, comma(λy.dislike(sam, y))⟨e,t^c⟩(kim)⟩
⟨λx.left(x), □⟩
left
⟨kim, □⟩
⟨comma(λy.dislike(sam, y))⟨e,t^c⟩, □⟩
Kim
who Sam admires

Notice that the *at-issue* content of the whole sentence is $\lambda x.left(x)(kim)$, that is $left(kim)$, which would also be the *at-issue* content of a simple sentence without a parenthetical (viz. *Kim left*). Notice likewise that the *at-issue* content of *Kim, who Sandy admires,* is the same as the *at-issue* content of *Kim* (the content of the ARC has disappeared from the *at-issue* dimension). The *ci* content of *Kim, who Sandy admires,* is produced from the *at-issue* content of *who Sandy admires* and the *at-issue* content of *Kim* in a way we will describe directly. It will turn out to be the *ci* proposition $dislike(sam, kim)_{t^c}$. Finally, notice that the *ci* content of *who Sandy admires* is empty – at the level of the relative clause, the content is still in the *at-issue* dimension. Intuitively, what happens is that the content of the relative clause 'starts out' as *at-issue* content, but then moves to the *ci* dimension, and then plays no further part in the construction of the *at-issue* content of the main clause. This is in contrast with the content of *Kim*, which one might think of as being 'copied' from the *at-issue* dimension into the *ci* dimension, since it plays a role in the *ci* content, and also plays a role in the *at-issue* content of the clause as a whole (i.e. it is used twice).

The mechanics here involves two admissibility conditions, and the definition of *comma*, which is the semantic correlate of the COMMA feature.[4]

Potts defines *comma* as in (30) – it is simply a function that changes the type of its argument from being of type $\langle e, t\rangle$ to being of type $\langle e, t^c\rangle$; it takes a function from entities to normal propositions, and produces a function from entities to propositions in the *ci* dimension.

(30) $\mathit{comma} = \lambda X.\lambda x.X(x)_{\langle\langle e,t\rangle,\langle e,t^c\rangle\rangle}$

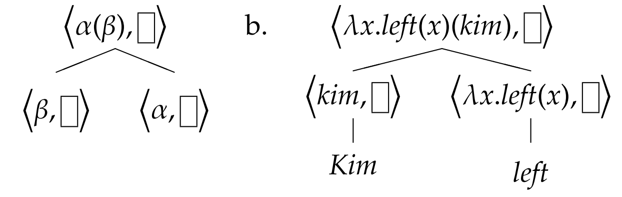
In the example above, the denotation of *who Sam admires* is of type $\langle e, t\rangle$, so *comma* will apply to it as in (31), giving (32) as the *ci* content of *Kim, who Sam admires*, as promised above.

(31) $\mathit{comma}(\lambda y.\mathit{dislike}(\mathit{sam}, y))_{\langle e,t^c\rangle} =$
$\lambda X.\lambda x.X(x)(\lambda y.\mathit{dislike}(\mathit{sam}, y))_{\langle e,t^c\rangle} =$
$\lambda x.\mathit{dislike}(\mathit{sam}, x)_{\langle e,t^c\rangle}$

(32) $\lambda x.\mathit{dislike}(\mathit{sam}, x)_{\langle e,t^c\rangle}(\mathit{kim}) =$
$\mathit{dislike}(\mathit{sam}, \mathit{kim})_{t^c}$

The first of Potts' node admissibility conditions is given schematically in (33a), and exemplified in (33b). This is used for 'normal' (*at-issue*) content. (The order of daughters in these trees is not relevant; we have used α for the functor daughter, and β for the argument; more generally, functor and argument are identified by their types, which we have not bothered to indicate here).

(33) a. b.

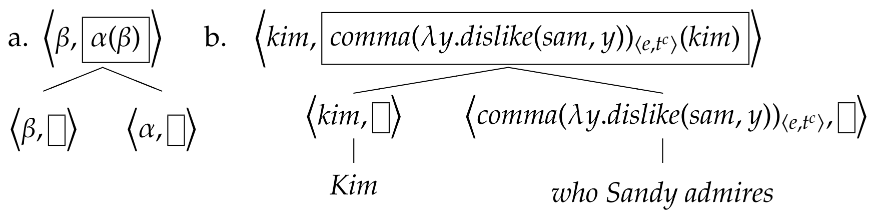


In words: the *at-issue* content of the mother is produced by applying the *at-issue* content of the functor argument to the *at-issue* content of the argument daughter. The *ci* content is not touched or affected in any way (note, in particular, that it is not percolated). Essentially, this is just normal

[4]Potts assumes a third admissibility condition, which licenses *ci* content which does not take any piece of *at-issue* content as an argument. This is used for the parenthetical part of examples like *Kim (and you won't believe this) ate fifty eggs*. It is not relevant here.

semantic composition.

The second is given schematically in (34a) and exemplified in (34b).

(34)

a. ⟨β, α(β)⟩
⟨β, □⟩
⟨α, □⟩
b. ⟨kim, comma(λy.dislike(sam, y))⟨e,t^c⟩(kim)⟩
⟨kim, □⟩
⟨comma(λy.dislike(sam, y))⟨e,t^c⟩, □⟩
Kim
who Sandy admires

Here the *at-issue* content of the mother is just that of the argument daughter, but as well as being 'passed up' to the mother, the content of this daughter is also used as an argument in the *ci* content of the mother. The functor in the *ci* content comes from the *at-issue* content of the functor daughter. This admissibility condition requires α to be a function from *at-issue* content to *ci* content: $\langle \sigma^a, \tau^c \rangle$ for some types σ, τ — in the example in (34b) the function is of type $\langle e, t^c \rangle$. Producing this sort of type from a 'normal' *at-issue* type is of course the semantic effect of the COMMA feature. One way of thinking of this node admissibility condition is as moving the functor into the *ci* dimension, and 'copying' the argument into the *ci* dimension (the argument also remains in the *at-issue* dimension; in resource logic terms, this means the argument is 'consumed twice').

Semantic parsetrees are interpreted according to the following principle (ignoring intensionality). Let T be a semantic parsetree with the *at-issue* term α_{σ^a} on its root node (that is, a semantic expression α of type σ^a), and distinct terms $\beta^1_{\tau^c}, \ldots, \beta^n_{\tau^c}$ on its nodes, then the interpretation of T is the tuple in (35).

(35) $\left\langle\ [\![\alpha_{\sigma^a}]\!]^{M,g},\ [\![\beta^1_{\tau^c}]\!]^{M,g}, \ldots,\ [\![\beta^n_{\tau^c}]\!]^{M,g} \right\rangle$

That is, a tuple consisting of the interpretation of the *at-issue* content, and all the *ci* content from anywhere in the tree, all interpreted relative to the same model and variable assignment.

Several features of this approach are worth noting. First, we have seen there are expressions of type $\langle \sigma^a, \tau^c \rangle$, that is, functors which take *at-issue* content to *ci* content (ARCs are of this type, and *comma* is designed to produce expressions of this type). However, Potts sets up the type theory so that there are no expressions which work the other way: there are no expressions of type $\langle \tau^c, \sigma^a \rangle$. Intuitively, this means there are no expressions which take *ci* content and move or copy it into the *at-issue* dimension. There is only one way traffic from the *at-issue* dimension into the *ci* dimension.

The fact that there are no functions from *ci* content means that *ci* content escapes the scope of all 'normal' (*at-issue*) operators, and makes *ci* content 'scopeless'. The effect of (35) is to combine *ci* content at the highest level – to give it 'wide scope'. Potts' approach thus captures what we take to be two key features of ARCs: syntactic integration and semantic wide scope.

Second, notice that the *ci* dimension is not percolated around the semantic parsetree. Instead, once an appositive has been formed, it is left where it is until the interpretation of the whole parsetree assimilates it to the interpretation of the whole main clause. This further underlines the limited possibilities for interaction across the dimension: once an appositive has been formed it is entirely inaccessible to the *at-issue* content — as though it was not part of the higher semantic tree. Potts considers and rejects introducing *ci* projection and *ci* storage mechanisms, on the grounds that the approach adopted makes the conceptual separation of the dimensions clearer.

The third, and for current purposes the most interesting, feature of the approach is the potential problem it raises for resources sensitive approaches to semantic interpretation — as we have seen, some content (e.g. the content of *Kim* above) is used twice, once in the *at-issue* dimension, once in the *ci* dimension.

Potts is aware of this issue, and sketches out a possible solution (Potts, 2005, p85ff). He exemplifies it with 'expressive' adjectives. The basic idea is that the semantics of an expressive adjective like *damn* is such that the semantics of *damn Republican* is associated with an *at-issue*/*ci* pair like (36), where the *at-issue* content is just the normal content of *Republican*, which can be used in the normal way (as in, e.g. *The Republicans will win*), and the *ci* content predicates *damn* of *Republican*, conveying a negative attitude to Republicans.

(36) $\left\langle \mathit{Republican}_{\langle e,t\rangle}, \boxed{\mathit{damn}(\mathit{Republican})_{t^c}} \right\rangle$

Producing this is mainly a matter of giving the right semantics to *damn* in the lexicon — specifically, giving it a type that yields a *ci* proposition when applied to its argument. We need not pursue this here. What matters here is that issues of resource sensitivity arise with *damn* as they do with ARCs above, because the semantics of *Republican* is used in both *at-issue* and *ci* dimensions.

Potts suggests this issue can be addressed by associating *damn* with a resource like (37). He gives a derivation of *damn Republican* as in (38).[5]

(37) $f \multimap [f \otimes p^c]$

The idea is that *damn* should consume the resource associated with the content of a noun like *Republican* to produce a composite ('tensor') resource, consisting of another normal resource (f), and a *ci* resource (p^c), thus duplicating the resource associated with *Republican* and avoiding the apparent

[5]Here we use the notation Potts uses, with 'meaning' and 'glue' expressions linked by '$\leadsto$'. Below we will use a different notation, this will typically involve a name (abbreviation) for a meaning constructor, followed by a meaning expression (a piece of lambda calculus) and a glue expression separated by a comma, so we would have something like **[Republican]** $\lambda x.\mathit{Republican}(x) : f$ in place of (38a).

problem of ’double consumption’ of resources.[6]

(38) a. [**Republican**] $\rightsquigarrow f$
b. [**damn**] $\rightsquigarrow f \multimap [f \otimes p^c]$
c. [**damn-Republican**] $\rightsquigarrow f \otimes p^c$

Potts does not develop this suggestion beyond this description of a resource for expressive adjectives, in particular, he does not consider whether the approach can be generalized to deal with *ci* content that is not lexically based. In what follows, we attempt to remedy this, filling out details, and applying the approach to the analysis of ARCs.

4 An LFG-Glue Implementation

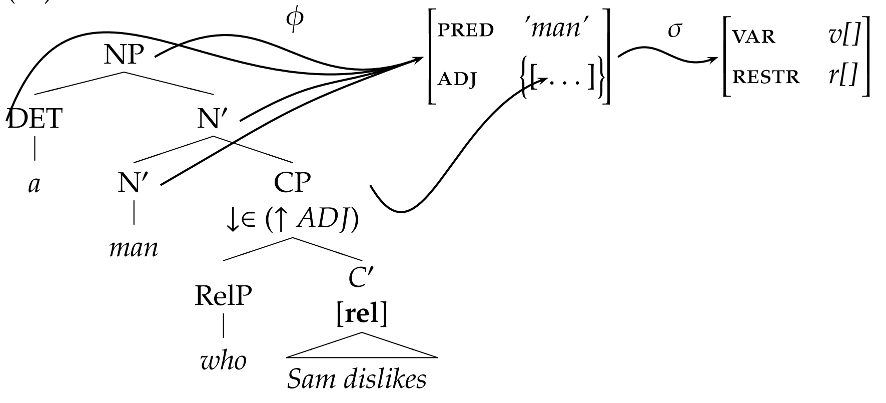
Assuming ARCs are syntactically integrated like RRCs, then a reasonable starting point for our analysis is the approach to English RRCs in Dalrymple (2001, 416ff), which provides an account of the syntax and the semantics of English RRCs. Since our account makes crucial use of the meaning constructors that she proposes, we begin by outlining her analysis.

4.1 Restrictive Relatives (Dalrymple, 2001)

Dalrymple’s c-structure is as in (40), where the RRC is adjoined to N′, and the [**rel**] meaning constructor is associated with C′.

(39) A man who Sam dislikes left.

(40)



The core of Dalrymple (2001)’s glue approach to RRCs is the meaning constructor [**rel**] which is associated with the C′ node in RRCs. The meaning constructor [**rel**] is defined as in (41), where we use $v_{\langle e \rangle}$ and $r_{\langle t \rangle}$ as abbrevi-

[6]Strictly speaking, the issue is not ’double consumption’, but ’multiple consumption’. In an example like *The damn Republicans, who I despise, will win*, the resource associated with *Republicans* is involved in three ways: in the main clause, as an argument of *damn*, and in the ARC. More complicated examples involving still more resource consumption can easily be imagined.

ations for ((ADJ $\in \uparrow)_\sigma$ VAR) and ((ADJ $\in \uparrow)_\sigma$ RESTR) , and $h_{<e>}$ and $g_{<t>}$ are ($\uparrow$RELPRO)$_\sigma$ and $\uparrow_\sigma$, respectively.[7] Intuitively, $[h_{<e>} \multimap g_{<t>}]$ corresponds to the resource contributed by the relative clause, which is a function from an individual to a proposition. Since $[v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]$ is the sort of thing one associates with a nominal modifier, [**rel**] is a meaning constructor which combines with a particular kind of (semantically incomplete) clause to produce a nominal modifier. In (42) we spell out the glue side of the definition of [**rel**] with abbreviations expanded.

(41) [**rel**] $\lambda P.\lambda Q.\lambda X.P(X) \wedge Q(X) : [h_{<e>} \multimap g_{<t>}] \multimap [[v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]]$

(42) [($\uparrow$ RELPRO)$_\sigma \multimap \uparrow_\sigma$] $\multimap$
[[((ADJ $\in \uparrow)_\sigma$ VAR) $\multimap$ ((ADJ $\in \uparrow)_\sigma$ RESTR)] $\multimap$
[((ADJ $\in \uparrow)_\sigma$ VAR) $\multimap$ ((ADJ $\in \uparrow)_\sigma$ RESTR)]]

Inside the relative clause itself the relevant meaning constructors are as shown in (43). The resource (43a) is basically a one-place predicate (i.e. it will consume an entity to provide a proposition), and the resource associated with *who*, (43b), will consume (43a) to produce (43c): *who* is taken to add the restriction that the argument in question is human.

(43) a. [**Sam-dislikes**] $\lambda Y.dislike(Sam, Y) : h_{<e>} \multimap g_{<t>}$
b. [**who**] $\lambda Q.\lambda X.person(X) \wedge Q(X) : [h_{<e>} \multimap g_{<t>}] \multimap [h_{<e>} \multimap g_{<t>}]$
c. [**who-Sam-dislikes**]
$\lambda Y.person(Y) \wedge dislike(Sam, Y) : h_{<e>} \multimap g_{<t>}$

[**rel**] will combine with [**who-Sam-dislikes**] to produce a relative clause meaning, i.e. a nominal modifier, as noted above.

(44) a. [**who-Sam-dislikes**] $\lambda Y.person(Y) \wedge dislike(Sam, Y) : h_{<e>} \multimap g_{<t>}$
b. [**rel**] $\lambda P.\lambda Q.\lambda X.P(X) \wedge Q(X) : [h_{<e>} \multimap g_{<t>}] \multimap [[v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]]$
c. [**rel-who-Sam-dislikes**]
$\lambda P.\lambda Q.\lambda X.P(X) \wedge Q(X)(\lambda Y.person(Y) \wedge dislike(Sam, Y))$
$\lambda Q.\lambda X.[\lambda Y.person(Y) \wedge dislike(Sam, Y)](X) \wedge Q(X)$
$\lambda Q.\lambda X.person(X) \wedge dislike(Sam, X) \wedge Q(X) : [v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]$

Finally, the relative clause as a whole consumes the nominal meaning and produces a (suitably restricted) nominal meaning, as in (47).

(45) [**man**] $\lambda Z.man(Z) : v_{<e>} \multimap r_{<t>}$

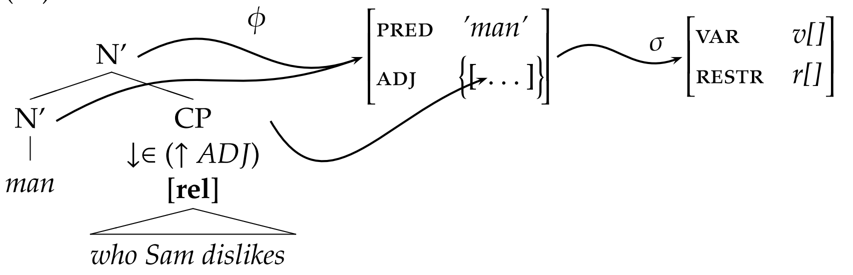
(46) [**rel-who-Sam-dislikes**]
$\lambda Q.\lambda X.person(X) \wedge dislike(Sam, X) \wedge Q(X) : [v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]$

(47) [**man-rel-who-Sam-dislikes**]
$\lambda Q.\lambda X.person(X) \wedge dislike(Sam, X) \wedge Q(X)(\lambda Z.man(Z))$
$\lambda X.person(X) \wedge dislike(Sam, X) \wedge \lambda Z.man(Z)(X)$
$\lambda X.person(X) \wedge dislike(Sam, X) \wedge man(X) : [v_{<e>} \multimap r_{<t>}]$

[7]In (41) we give both the complete paths and the abbreviations naming the pieces of semantic structure, but generally below we give only the abbreviations.

Note that Dalrymple (2001) introduces **[rel]** inside CP as an annotation to C′. This is not crucial, however, and we can recast the analysis of the N′ *man who Sam dislikes* as in (48) — where **[rel]** is associated with CP — without any changes being required to either the f-structure or the meaning side of the meaning constructors. The only changes required are in the associated glue expression, where small changes to the paths to the σ structures are necessary. (The reason this is possible is that semantic derivations using glue are not compositional in the c-structure, of course). The revised definition of **[rel]** is show in abbreviated form in (49); (50) presents the glue expression with abbreviations expanded. This is not crucial to our approach, but it will open up the possibility of a simplification of the treatment we present in the next section.

(48)



(49) **[rel]** $\lambda P.\lambda Q.\lambda X.P(X) \wedge Q(X)$: $[h_{<e>} \multimap g_{<t>}] \multimap [[v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]]$

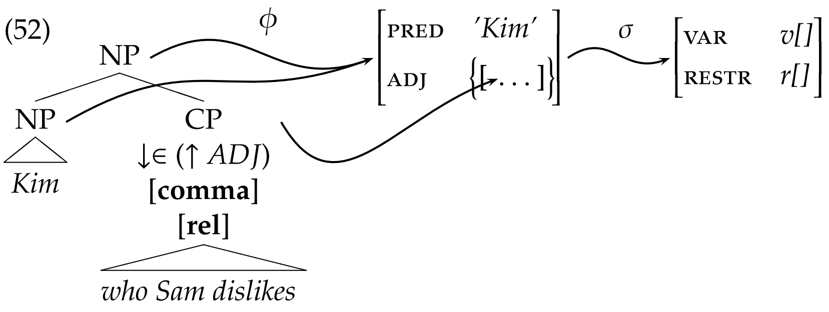
(50) $[\ (\downarrow \text{RELPRO})_\sigma \multimap \downarrow_\sigma\ ] \multimap [[\ (\uparrow_\sigma \text{VAR}) \multimap (\uparrow_\sigma \text{RESTR})\ ] \multimap [\ (\uparrow_\sigma \text{VAR}) \multimap (\uparrow_\sigma \text{RESTR})\ ]]$

4.2 ARCs

We now turn to the analysis of ARCs. Alongside **[rel]** we introduce a meaning constructor **[comma]** in the semantics. We will assume the modification of Dalrymple's approach to English relative clauses made in (48) and (49), so that these meaning constructors are associated with the CP node corresponding to the relative clause itself, which is syntactically integrated as sister of an NP head, as in (52). Notice that, apart from adjunction to N′ vs NP, the only difference between this, and structure given for the restrictive in (48) is the presence of **[comma]** in (52).

(51) Kim, who Sam dislikes, left.

(52)



Recall that Potts uses *comma* to change the type of the relative *who Sam dislikes* so that it becomes a function to *ci* content (type $\langle e, t^c \rangle$), and uses the schema in (34) to ensure that the content of *Kim* is available in both *at-issue* and *ci* domains. We therefore want [**comma**] to consume [**rel-who-Sam-dislikes**] (the meaning of the relative clause) and produce a resource which will consume the meaning of *Kim* (which is of type *e*), and produce a 'tensor' meaning constructor which combines the meanings of *Kim* and the relative clause (a *ci* meaning of type t^c, with roughly the meaning of *Sam dislikes Kim*, formed by applying the relative clause meaning to the meaning of *Kim*). To bring this about we define [**comma**] as in (53).[8]

(53) [**comma**] =
$\lambda P.\lambda Y.[Y, (P(\lambda Z.true))(Y)]$:
$[[v_{\langle e \rangle} \multimap r_{\langle t \rangle}] \multimap [v_{\langle e \rangle} \multimap r_{\langle t \rangle}]] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]]$

Here $[v_{\langle e \rangle} \multimap r_{\langle t \rangle}] \multimap [v_{\langle e \rangle} \multimap r_{\langle t \rangle}]$ is the glue expression associated with a nominal modifier such as a relative clause (a function from noun meanings to noun meanings), using abbreviations introduced above; and l abbreviates $\uparrow_\sigma$ — the σ-projection of the mother NP. On the glue side, then, [**comma**] will consume an RRC-like resource and produce a resource whose glue is of the form $l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]$ — the kind of resource Potts suggested in (37) above; viz a resource which will itself consume an NP meaning ($l_{\langle e \rangle}$) to produce a 'tensor resource' consisting of (a) an NP meaning ($l_{\langle e \rangle}$), and (b) a propositional resource in the *ci* domain ($l_{\langle t^c \rangle}$). On the meaning expression side, [**comma**] is a function that applies to an RRC meaning expression, does some type lowering (this is the purpose of the $\lambda Z.true$ expression), and yields an expression $\lambda Y.[Y, T]$, a function from individuals to a pair of meaning expressions.

The end result will be that we will get a meaning constructor like (54) for *Kim, who Sam dislikes*. In what follows, we will spell out the process by

[8]Our version of [**comma**] is less general than Potts'. This is because we want to build directly on the existing treatment of RRCs. Since semantically RRCs are of common noun modifiers of type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$, and ARCs are NP modifiers (type $\langle e, t\rangle$), this means we have to do some type lowering, and we have chosen to build this in to the definition of [**comma**]. Potts is not concerned with the relation between RRCs and ARCs, and so ignores this. Of course, it is not obvious that combining type lowering with the *ci* to *at-issue* type manipulation in this way is the right thing to do. However, we will not pursue the issue here.

which this result is produced, step by step.

(54) $[Kim, (person(Kim) \wedge dislikes(Sam, Kim) \wedge true)] : l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}$

The RRC-like resource that [**comma**] consumes is shown in (55) — this is just the nominal modifier resource standardly associated with relative clauses (and produced in turn by [**rel**]) — it is in fact exactly what we had for an RRC above, cf. (44c).

(55) [**rel-who-Sam-dislikes**]
$\lambda Q.\lambda X.person(X) \wedge dislike(Sam, X) \wedge Q(X) :$
$[v_{<e>} \multimap r_{<t>}] \multimap [v_{<e>} \multimap r_{<t>}]$

The effect of combining [**comma**] with [**rel-who-Sam-dislikes**] is shown in (56). On the glue side, [**comma**] consumes the resource [**rel-who-Sam-dislikes**] to produce a resource which is a function that will take an entity resource and produce the required tensor resource. The effect of applying the function [**comma**] to the (relative clause meaning) argument is spelled out in the reduction steps in (56), resulting in a lambda function corresponding to a one-place predicate. This in turn will apply to [**Kim**] to produce the tensor resource in (58).

(56) [**comma-rel-who-Sam-dislikes**]
$\lambda P.\lambda Y.[Y, (P(\lambda Z.true))(Y)]([\textbf{rel-who-Sam-dislikes}]) : l_{\langle e\rangle} \multimap [l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}]$
$\lambda Y.[Y, ([\textbf{rel-who-Sam-dislikes}](\lambda Z.true))(Y)] : l_{\langle e\rangle} \multimap [l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}]$
$\lambda Y.[Y, (\lambda Q.\lambda X.person(X) \wedge dislikes(Sam, X) \wedge Q(X)(\lambda Z.true))(Y)] : l_{\langle e\rangle} \multimap [l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}]$
$\lambda Y.[Y, (\lambda X.person(X) \wedge dislikes(Sam, X) \wedge \lambda Z.true(X))(Y)] : l_{\langle e\rangle} \multimap [l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}]$
$\lambda Y.[Y, (\lambda X.person(X) \wedge dislikes(Sam, X) \wedge true)(Y)] : l_{\langle e\rangle} \multimap [l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}]$

(57) [**Kim**] $Kim : l_{\langle e\rangle}$

(58) [**Kim-comma-rel-who-Sam-dislikes**]
$\lambda Y.[Y, (\lambda X.person(X) \wedge dislikes(Sam, X) \wedge true)(Y)](Kim) : l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}$
$[Kim, (\lambda X.person(X) \wedge dislikes(Sam, X) \wedge true)(Kim)] : l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}$
$[Kim, (person(Kim) \wedge dislikes(Sam, Kim) \wedge true)] : l_{\langle e\rangle} \otimes l_{\langle t^c\rangle}$

Thus, corresponding to *Kim, who Sam dislikes,* we have, on the meaning side, a pair of meanings corresponding to *Kim* and the proposition that Kim is a person and Sam dislikes Kim. On the glue side, we have a 'tensor' resource consisting of two resources, one in the *at-issue* dimension, and one in the *ci* dimension.

A further step is now required. In order to deal with these resources separately, we will need a new inference rule, which we will call *at-issue-ci-split* (*ACiS*), to separate the two parts of the tensor resource so that they may be used separately in subsequent proof steps. The rule is formulated as in (59); it is inspired by the Context Split rule of Dalrymple (2001, 297). In (60) we show a proof using it.

(59) $$\frac{[M, M'] : R_e \otimes R_{t^c}}{M : R_e \quad M' : R_{t^c}} \text{ ACiS (at-issue-ci-split)}$$

(60)

$$\frac{\begin{array}{c} \lambda Y.left(Y) : \\ l_{\langle e \rangle} \multimap s_{\langle t \rangle} \end{array} \quad \dfrac{[Kim, (person(Kim) \wedge dislikes(Sam, Kim) \wedge true)] : l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}}{Kim : l_{\langle e \rangle} \quad (person(Kim) \wedge dislikes(Sam, Kim) \wedge true) : l_{\langle t^c \rangle}}\ \text{ACiS}}{left(Kim) : s_{\langle t \rangle} \quad (person(Kim) \wedge dislikes(Sam, Kim) \wedge true) : l_{\langle t^c \rangle}}$$

If, following Potts, we assume there are no functions from *ci* types, then this will ensure that the content of an ARCs will not be in the scope of any normal operator. However, we must still find a way of integrating the *at-issue* and *ci* content, since both are relevant for truth conditions. Recall that Potts (2005) dealt with this by having a principle which collects all *ci* content from the semantic parsetree (cf (35)). Our goal now is to find a resource sensitive way of dealing with this issue.

The simplest approach is to introduce an '!' ('of course') meaning constructor on the root S, which will (iteratively, and non-deterministically) select *ci* propositions and combine them with the main clause content. Suppose, for example, that the 'start' rule for the grammar is associated with a [**root-cp**] meaning constructor, as in (61), where [**root-cp**] is defined as in (62).

(61) $Root \rightarrow \begin{array}{c} CP \\ \textbf{[root-cp]} \end{array}$

(62) [**root-cp**] $\lambda q.\lambda p.(p \wedge q) : ![Some_{t^c} \multimap [\uparrow_\sigma \multimap \uparrow_\sigma]]$

The idea is that this meaning constructor can be applied as many times as required to consume resources of type t^c, 'and'-ing them together and then combining them with the main clause content.

The main question is how the relevant *ci* resources are to be located , i.e. how $Some_{t^c}$ should be defined in (62). This is straightforward. The following functional uncertainty expression, which will pick out meaning resources of type t^c associated with any piece of f-structure, will do the job, giving (64) as the full definition of [**root-cp**].

(63) $(\downarrow GF*)\sigma_{t^c}$

(64) [**root-cp**] $\lambda q.\lambda p.(p \wedge q) : ![(\downarrow GF*)\sigma_{t^c} \multimap [\uparrow_\sigma \multimap \uparrow_\sigma]]$

With respect to the example in hand, one solution for the functional uncertainty expression will be the resource of type t^c associated with *Kim, who Sam dislikes*, namely [**Kim-comma-rel-who-Sam-dislikes**], as in (58). We can procede as follows. First [**root-cp**] applies to this, as in (65), the result can then be applied to the *at-issue* content to produce the interpretation of the whole main clause, as in (66).

(65) [**root-cp**] ([**Kim-rel-who-Sam-dislikes**])
$\lambda q.\lambda p.(p \wedge q)(person(Kim) \wedge dislikes(Sam, Kim) \wedge true)$
$\lambda p.(p \wedge person(Kim) \wedge dislikes(Sam, Kim) \wedge true) : [\uparrow_\sigma \multimap \uparrow_\sigma]$

(66) [**root-cp-Kim-rel-who-Sam-dislikes**] ([**left-Kim**])
$\lambda p.(p \wedge person(Kim) \wedge person(Kim) \wedge dislikes(Sam, Kim) \wedge true)(left(Kim))$

$$left(Kim) \wedge person(Kim) \wedge person(Kim) \wedge dislikes(Sam, Kim) \wedge true :\uparrow_{\sigma}$$

This gives us the desired interpretation for the whole clause, but it deviates from the strict Pottsian approach, because [**root-cp**] must clearly be a function from expressions of type t^c, which are otherwise forbidden. However, since this should be the only such deviation, it is perhaps not problematic.

To summarise: we have shown how a Potts' style approach to ARCs can be implemented in a resource sensitive way, based on the standard LFG analysis of RRCs. Apart from a straightforward, and empirically harmless, reformulation of Dalrymple's treatment of RRC's, we have taken over Potts' type theory, and introduced a new meaning constructor [**comma**], and an inference rule, *ACiS*, to extract *ci* content, and a further new meaning constructor, [**root-cp**] to integrate it. We have the following two rules for RRCs and ARCs respectively:

(67) a. $N' \rightarrow N' \quad \underset{[\mathbf{rel}]}{\overset{CP}{\downarrow \in (\uparrow ADJ)}}$ b. $NP \rightarrow NP \quad \begin{array}{c} CP \\ \downarrow \in (\uparrow ADJ) \\ [\mathbf{rel}] \\ [\mathbf{comma}] \end{array}$

The special restrictions on ARCs (viz that they are finite and +WH) can easily be expressed by adding the annotations (↑TENSE), and (↑WH=$_c$+) to (67b). Notice that we can use exactly the same rules to describe the internal structure of ARCs and RRCs.

We can, however, simplify things further. Given that [**comma**] and [**rel**] are associated with the same node in (67b), we can fold them together, to produce something simpler. If we introduce the meaning constructor [**rel**$_{arc}$], as in (68), we can dispense with the trivial (though harmless) *true* conjunct used above, and procede in fewer steps.[9]

[**rel**$_{arc}$] consumes the one-place predicate [**who-Sam-dislikes**], in (69), and produces the tensor resource (70). This in turn consumes [**Kim**] to produce the tensor resource corresponding to $[Kim, person(Kim) \wedge dislike(Sam, Kim)]$ exactly as before, see (72).

(68) [**rel**$_{arc}$] $\lambda P.\lambda Z.[Z, P(Z)] : [h_{<e>} \multimap g_{<t>}] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]]$
(69) [**who-Sam-dislikes**]
$\lambda Y.person(Y) \wedge dislike(Sam, Y) : h_{<e>} \multimap g_{<t>}$
(70) [**rel**$_{arc}$**-who-Sam-dislikes**]
$\lambda P.\lambda Z.[Z, P(Z)] \; (\lambda Y.person(Y) \wedge dislike(Sam, Y))$
$\lambda Z.[Z, \lambda Y.person(Y) \wedge dislike(Sam, Y)(Z)]$
$\lambda Z.[Z, person(Z) \wedge dislike(Sam, Z)] : l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]$
(71) [**Kim**] $Kim : l_{\langle e \rangle}$
(72) [**Kim-rel**$_{arc}$**-who-Sam-dislikes**]

[9]Our main reason for introducing this simplification is to simplify the discussion in the following section. It is in no way essential. In fact, on may wonder if it is not retrograde from a theoretical point of view, because it takes us further from Potts' idea of a very general statement of a single operation for type manipulation.

$\lambda Z.[Z, person(Z) \wedge dislike(Sam, Z)](Kim)$
$[Kim, person(Kim) \wedge dislike(Sam, Kim)] : l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}$

This can be split, and the parts combined with the *at-issue* content exactly as before.

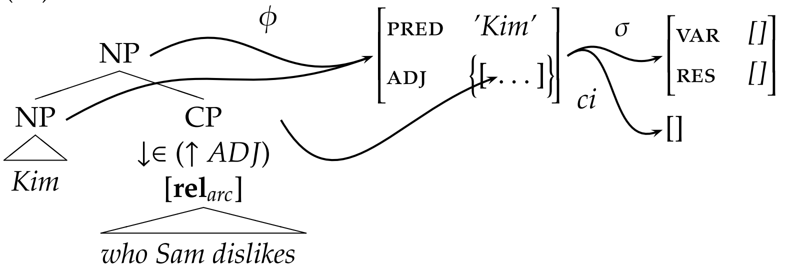
With or without this simplification, this is a promising result — it suggests that we can directly incorporate Potts's approach, and his analyses of particular phenomena, in a straightforward and simple manner into the LFG framework. In the following section we consider an alternative which avoids the multiplication of types characteristic of the Potts' approach.

5 An Alternative Implementation

The previous section built directly on Potts' idea of handling supplementary/appositive meanings by introducing non-standard types, and apparatus for manipulating them.

A potential alternative in the LFG architecture would be to use the projection architecture directly, introducing an additional projection for *ci* content, the idea being that the *ci* content would be distinct only in terms of which projection it is associated with. This will permit us to dispense with Potts' non-standard *ci* types. The architecture which this approach involves is exemplified in (73) — in addition to the σ projection from f-structure to semantic structure (we might now say *'at-issue'* semantic structure), we introduce a parallel projection, the *ci*-projection, from f-structure to '*ci*-structure'. In this section we outline a formalization of this alternative.

(73)



We start from the [**rel**$_{arc}$] resource introduced above and repeated here as (74) for convenience. Recall that this consumes the resource corresponding to a relative clause and produces a function from the meaning of the head NP to the tensor resource:

(74) [**rel**$_{arc}$] $\lambda P.\lambda Z.[Z, P(Z)] : [h_{<e>} \multimap g_{<t>}] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]]$

The new idea is that the *ci* meaning should be associated with a separate projection, hence we replace (74) by (75): (76) spells out the paths to the f-structure in full.

(75) [**rel**$_{arc}$] $\lambda P.\lambda Z.[Z, P(Z)] : [h_{<e>} \multimap g_{<t>}] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes \uparrow_{ci}]]$

(76) $[\textbf{rel}_{arc}]\ \lambda P.\lambda Z.[Z, P(Z)] : [\ (\downarrow \text{RELPRO})_\sigma \multimap \downarrow_\sigma\ ] \multimap [\ \uparrow_\sigma \multimap [\uparrow_\sigma\ \otimes\ \uparrow_{ci}]\ ]$

Careful comparison of (75) with (74) will reveal exactly one difference — where in the former, the second part of the tensor resource is an resource of type t^c, in the latter it is a resource associated with the *ci* projection, $\uparrow_{ci}$. We have not indicated its type, but it is the standard type *t*. This, with some obvious and straightforwardly related modifications, is almost the only thing we need to change to implement the approach in the projection architecture.

The derivation proceeds as before, only the final result is different. The newly defined meaning constructor [**rel**$_{arc}$] consumes the meaning of *who Sam dislikes* to produce the meaning of an ARC as shown in (79). The resulting meaning constructor is a function from an NP meaning (that of the host NP of the relative clause) to a tensor resource corresponding to this NP meaning and a (propositional) meaning in the *ci* projection.

(77) $[\textbf{rel}_{arc}]\ \lambda P.\lambda Z.[Z, P(Z)] : [h_{<e>} \multimap\ g_{<t>}] \multimap\ [l_{\langle e\rangle}\ \multimap [l_{\langle e\rangle} \otimes\ \uparrow_{ci}]]$

(78) [**who-Sam-dislikes**]
$\lambda Y.person(Y) \wedge dislike(Sam, Y) : h_{<e>} \multimap\ g_{<t>}$

(79) [**rel$_{arc}$-who-Sam-dislikes**]
$\lambda P.\lambda Z.[Z, P(Z)]\ (\lambda Y.person(Y) \wedge dislike(Sam, Y))$
$\lambda Z.[Z, \lambda Y.person(Y) \wedge dislike(Sam, Y)(Z)]$
$\lambda Z.[Z, person(Z) \wedge dislike(Sam, Z)] : l_{\langle e\rangle}\ \multimap [l_{\langle e\rangle} \otimes\ \uparrow_{ci}]$

(80) [**Kim**] $Kim : l_{\langle e\rangle}$

(81) [**Kim-rel$_{arc}$-who-Sam-dislikes**]
$\lambda Z.[Z, person(Z) \wedge dislike(Sam, Z)](Kim)$
$[Kim, person(Kim) \wedge dislike(Sam, Kim)] : l_{\langle e\rangle} \otimes\ \uparrow_{ci}$

As before, we need to split the tensor resource, so that the resource associated with *Kim* can be consumed separately by composition steps accessing resources in σ structure. The reformulation of the splitting rule is straightforward. The resources we want to split are now characterised by the projection they are associated with, rather than their type (we have changed the name of the rule accordingly):

(82) $$\frac{[M, M'] : R_\sigma \otimes\ R_{ci}}{M : R_\sigma \quad M' : R_{ci}}\ \text{SCiS (sigma-ci-split)}$$

We also need to be able to re-integrate content from σ- and *ci*-projections. The most straightforward approach involves a trivial reformulation of the [**root-cp**] meaning constructor: what we now want is that [**root-cp**] picks up resources of type *t* associated with the *ci*-projection of various pieces of f-structure, and 'and's them together with the content of the σ projection:

(83) $[\textbf{root-cp}]\ \lambda q.\lambda p.(p \wedge q) : ![(\downarrow GF*)ci_t \multimap\ [\uparrow_\sigma \multimap \uparrow_\sigma]]$

This is a straightforward re-formulation of the definition we gave in the previous section, and we will not repeat the discussion.[10]

[10] To ensure that *ci* content does not appear in the scope of other operators, we must also

However, once we have dispensed with the need to associate *ci* content with Potts' non-standard types, there are other possibilities. In particular, we can avoid the use of the '!' 'of course' meaning constructor. We will briefly outline how.

To begin with, we will introduce the abbreviation $\uparrow_{R\sigma}$ to designate the σ-projection of the root f-structure. It can be defined by the inside-out functional uncertainty expression (84), in which the off-path constraint ensures that the solution to the functional uncertainty statement is not the value of any attribute.[11]

(84) $\left(\underset{\neg \leftarrow}{GF}\ GF{*} \uparrow \right)_{\sigma}$

Now suppose that instead of being of type t, as we have assumed, *ci* resources are of type $\langle t, t \rangle$: specifically, functions from the σ projection of the root f-structure to the σ projection of the root f-structure (i.e. from $\uparrow_{R\sigma}$ to $\uparrow_{R\sigma}$).

The general form of a *ci* meaning constructor would be (85), where **CI** is the content specific to the particular construction. So, for *Kim, who Sam dislikes* it might be as in (86).

(85) $\lambda p.(p \wedge \mathbf{CI}) : \uparrow_{R\sigma} \multimap \uparrow_{R\sigma}$

(86) **[Kim-rel$_{arc}$-who-Sam-dislikes]**
$\lambda p.(p \wedge \mathit{person}(\mathit{Kim}) \wedge \mathit{dislike}(\mathit{Sam}, \mathit{Kim})) : \uparrow_{R\sigma} \multimap \uparrow_{R\sigma}$

Since the ordinary (*at-issue*) meaning associated with the root f-structure of *Kim, who Sam dislikes, left* is:

(87) **[left-Kim]** $\mathit{left}(\mathit{Kim}) : \uparrow_{R\sigma}$

The 'full-interpretation', involving reintegration of the *ci* content, will be obtained by applying each *ci* meaning constructor to the normal *at-issue* meaning constructor associated with the root f-structure:

(88) **[Kim-rel$_{arc}$-who-Sam-dislikes]**
$\lambda p.(p \wedge \mathit{person}(\mathit{Kim}) \wedge \mathit{dislike}(\mathit{Sam}, \mathit{Kim})) : \uparrow_{R\sigma} \multimap \uparrow_{R\sigma}$
(89) **[Kim-rel$_{arc}$-who-Sam-dislikes]** (**[left-Kim]**)
$\lambda p.(p \wedge \mathit{person}(\mathit{Kim}) \wedge \mathit{dislike}(\mathit{Sam}, \mathit{Kim}))(\mathit{left}(\mathit{Kim}))$
$\mathit{left}(\mathit{Kim}) \wedge \mathit{person}(\mathit{Kim}) \wedge \mathit{dislike}(\mathit{Sam}, \mathit{Kim}) : \uparrow_{R\sigma}$

stipulate that no ordinary meaning constructor (i.e. other than [**root-cp**]) is permitted to access the *ci*-projection.

[11] We could have used this idea of directly accessing the root f-structure with the Pottsian approach we considered in Section 4.2. If we have b., rather than a., as the glue expression associated with [**comma**], then the glue expression for [**Kim-comma-rel-who-Sam-dislikes**] will be $[l_{\langle e \rangle} \otimes \uparrow_{R\sigma}]$, and the *ci* content will be directly associated with the root f-structure. Nothing else would need to change.

a. $[[v_{\langle e \rangle} \multimap r_{\langle t \rangle}] \multimap [v_{\langle e \rangle} \multimap r_{\langle t \rangle}]] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes l_{\langle t^c \rangle}]]$
b. $[[v_{\langle e \rangle} \multimap r_{\langle t \rangle}] \multimap [v_{\langle e \rangle} \multimap r_{\langle t \rangle}]] \multimap [l_{\langle e \rangle} \multimap [l_{\langle e \rangle} \otimes \uparrow_{R\sigma}]]$

We will not pursue this here, because associating all *ci* content with the syntactic root is a departure from Potts' approach. It is not clear if it would have empirical consequences.

This resource is available both as the final interpretation of the whole clause, or as input to be consumed by any other *ci* resources that may exist.

6 Conclusion and Further Work

Potts' approach provides an interesting account of the semantics of *inter alia* supplementals (including ARCs) and expressives by introducing additional semantic types alongside the normal types used to capture *at-issue* content. We have explored how the central insights of this approach can be expressed within the LFG formalism. We have suggested two ways: one more or less directly encodes Potts idea, including his non-standard types. Our contribution here has been to fill out Potts' sketch of how his approach could be made compatible with a resource sensitive approach, and show that the idea is workable. We have also suggested an alternative which uses only standard types, and which exploits the projection architecture of LFG. We leave open the question whether there is any empirical basis for favouring one approach over the other.

References

Doug Arnold. Non-Restrictive relatives are not orphans. *Journal of Linguistics*, 43(2):272–309, 2007.

Ash Asudeh. *Resumption as Resource Management*. PhD thesis, Stanford University, 2004.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. *A Grammar Writer's Cookbook*. CSLI Pubs., Stanford, CA, 1999.

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

Mary Dalrymple. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, New York, 2001.

Christian Fortmann. On Parentheticals (in German). In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG '05 Conference*, Stanford, CA, 2005. CSLI Pubs.

Christian Fortmann. The complement of *verba dicendi* parentheticals. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG '06 Conference*, University of Konstanz, Germany, 2006.

Jesse A. Harris and Christopher Potts. Perspective-shifting with appositives and expressives. *Linguistics and Philosophy*, 32(6):523–552, 2010.

Eric Potsdam. English verbal morphology and VP ellipsis. In *Proceedings of the 27th Meeting of the North East Linguistic Society*, pages 353–368, Amherst, Mass, 1997. GLSA, University of Massachusetts at Amherst.

Christopher Potts. *The Logic of Conventional Implicatures*. Oxford University Press, Oxford, 2005.

Anthony R. Warner. *English Auxiliaries: Structure and History*. Cambridge University Press, Cambridge, 1993.