Dynamic typing for lexical semantics. A case study: the genitive construction

Nicholas Asher, Pascal Denis

Dynamic Typing for Lexical Semantics A Case Study: the Genitive Construction

Nicholas ASHER, Pascal DENIS*
The University of Texas at Austin

Introduction

In recent years, there has been a considerable amount of effort in computational work on lexical semantics that seeks to automate the acquisition of disambiguated senses for words. Similar work is less popular in theoretical lexical semantics but still widely practiced throughout the discipline. We believe that this approach is fundamentally mistaken. The only way to build large scale lexicons that are linguistically and practically useful is to use simultaneously the techniques of underspecification and type theory, informed by ontological analysis. These techniques include not only a rich type lattice of simple types but also complex types whose operations of introduction and exploitation (or elimination) must be understood in the form of logical rules. Types put constraints on underspecified lexical meanings or rather *constitute* them. In fact, the semantic types of the lexicon reflect in a simplified manner our common sense ontology as it surfaces in our language. We think not only that ontological analysis is crucial for lexical semantics but that a careful study of the meanings of words also reveals important principles of common sense ontology.

Through the process of semantic composition, words with highly underspecified meanings may acquire more determinate meanings; on our view semantic composition typically adds to the content supplied by the basic lexical elements. We countenance not only deductive, monotonic rules for types but also non-monotonic rules whose conclusions can be overridden by discourse context; these rules and their associated types are intended to capture the preferred meanings of words in a given context.

Such a project, if it succeeds, provides a novel and general view of the lexicon. It will also allow us to make predictions about word meanings in particular contexts. We also hope that it will be possible to acquire information about types and rules automatically by learning from annotated corpora. We summarize our view in the following principles:

- The lexical entry for a word is underspecified but simple—one particular lambda expression that may even include rhetorical information but which is typically underspecified.
- We offset these simple lexical entries with a rich account of types including simple types, functional types and complex types that resemble restricted conjunctions. All variables in the λ-term of a lexical entry are typed.
- The process of composition and predication are type-driven processes, involving not only type checking but also type merging and various coercion operations which are construed as operations of type introduction or exploitation often of complex types.

*We would like to thank Laure Vieu and three anonymous reviewers for their comments on this paper.

- Following the Curry-Howard isomorphism between proofs and types but extending it to complex types, type checking and type merging are understood as rules of introduction and exploitation in a deductive logic. We will also have nonmonotonic rules for type exploitation to capture preferred meanings in a given context. These rules can affect the logical form of the λ-terms.

- The dividing line between lexical and non lexical or encyclopedic knowledge has to do more with how the knowledge is expressed (as types lexically; in a much more expressive formalism encyclopedically). So indeed in terms of content Quine and Fodor ([1], [2]) may be right: there is no distinction in terms of content between lexical and encyclopedic information, though the information may have a privileged epistemic status in virtue of the fact that it is grammaticalized.

This paper illustrates these principles on a case study, namely the genitive construction. We will also relate our work to past efforts, in particular the Generative Lexicon of Pustejovsky. We agree with Pustejovsky that the lexicon should have a generative approach to meaning, but in particular we give this notion of generation a dynamic twist. As already detailed in [3], we don't agree with the formalization of GL of [4], which frankly leaves a lot to be desired. GL is not clear on what lexical information is and on lexical processes, from our perspective. Finally and specifically, we think that the system of types employed in [4] is too restrictive; we will show how in particular the so-called *qualia* have to be replaced with a much more dynamic notion of typing.

1 The type composition logic (TCL)

Lexical entries for us contain semantic information of two kinds. The first is a λ-term with a model theoretic interpretation, while the second is a typing context. A *typing context* for a term t determines an assignment of types to all subterms of t. It is the types that are assigned to the various variables in the λ-term that will affect how it combines with other terms in the composition of a logical form. In earlier semantic theories, the composition process was largely a matter of applying a λ-term to its arguments, and the standard λ-calculus with functional types due to Church sufficed. Our composition logic is more complex. It contains a system of types and rules for introducing and exploiting these types. Our system of types extends the λ-calculus for functional types with complex types like the $\bullet$-types ("dot-types") of Asher and Pustejovsky ([3]), and types that generalize the *qualia* structures that Pustejovsky and others have argued for. As one of us argued in [5], [3], and as already outlined in [4], $\bullet$-types are useful to model many *copredications*, which are constructions where two predications with incompatible typing constraints are made on a single argument as in:

(1) a. Mary picked up and mastered three books on mathematics. [*physical object* and *informational content*]

b. Most cities that vote democratic passed anti-smoking legislation last year. [*population* and *legislative entity*]

The idea is that an object $\bullet$-type $\alpha \bullet \beta$ has two aspects, the aspect of being α and the aspect of being β and that the predication of a property to an object may hold in virtue of that aspect.

Preferential readings of similar typing conflicts have motivated Pustejovsky's introduction of *qualia*, for which Asher and Pustejovsky provide a complex type similar to the $\bullet$-type, the so-called *qualia constructs* (or QCs). Qualia are also an Aristotelian idea; roughly Pustejovsky's qualia for any object correspond to the four metaphysical $\alpha\iota\tau\iota\alpha$ or explanatory

causes of an object's existence. While we are Aristotelians with respect to restricted predication, we believe that the lexicon is not restricted to encoding Aristotelian causes of objects. We will move away from *qualia* here, and so we will just introduce this type via its special operator and call it a ⊗-type ("product-type").

The •-type and ⊗-type resemble conjunctive types and our natural deduction rules for exploiting them and introducing them will resemble something like conjunction elimination and introduction. But our rules are quite a bit more complicated than the introduction and elimination rules for simple conjunctive types, as they add material to logical form, as well as revise the types of variables. Besides these rules, we'll assume the presence of a type hierarchy with a subtyping relation $\sqsubseteq$ that defines a partial order on the set of types and a greatest lower bound operation $\sqcap$ on the set of types. $\sqcap$ has the usual properties—e.g., idempotence, commutativity, and $\alpha \sqsubseteq \beta$ iff $\alpha \sqcap \beta = \alpha$. We will capture incompatibility between types in terms of their common join, $\perp$. Our type language takes as fundamental the notion of a term together with a typing context that our rules can revise or extend. We note by $c(t : \alpha)$ that the type assignment $t\colon \alpha$ is in the typing context c, by $c * (t : \alpha)$ that the context c is revised in the assignment $t : \alpha$ (which boilds to simply adding this assignment, $c + (t : \alpha)$, if t is not defined in c).

We assume there are *simple* types and two sorts of complex types, •- and ⊗-types as well as their associated functional types (e.g. $\alpha \multimap \beta$) corresponding to λ-expressions:

(2) a. Primitive Types: e the general type of entities and $\underline{t}$ the type of truth values. Below σ, τ range over all simple types, the subtypes of e as well as e and $\underline{t}$.

b. Functional Types: If σ and τ are types, then so is $(\sigma \multimap \tau)$

c. Dot Types: If σ and τ are types, then so is $(\sigma \bullet \tau)$

d. Product Types: If σ and $\tau_1, \ldots \tau_n$ are types, then so is $(\sigma \otimes_{R_1,\ldots R_n} (\tau_1 \ldots \tau_n))$ (where R_i is a relation over (σ, τ_i)).

Our Type Composition Logic (TCL) combines the usual λ-terms familiar from compositional semantics with a set of type assignments, of the form $x\colon \sigma$, $x : \sigma \bullet \tau$, or $x\colon (\sigma \otimes_{R_1,\ldots R_n} (\tau_1 \ldots \tau_n))$. Constraints on types will also be available; for instance, we may need to know that σ is a subtype of τ, something we express as $\sigma \sqsubseteq \tau$ or that two types are compatible, which we write as $\sigma \sqcap \tau \neq \perp$.

Let us add a little more here regarding the new ⊗-types. Like Asher and Pustejovsky's QCs, but unlike •-types, these new types are "asymmetric". That is, a complex type like $x\colon (\sigma \otimes_{R_1,\ldots R_n} (\tau_1 \ldots \tau_n))$ combines on its left an *intrinsic* type σ that defines the nature of the thing denoted by the variable and on its right a set of *dependent* types $\tau_1 \ldots \tau_n$ corresponding to eventualities that are conventionally (but defeasibly) associated to that thing. Each of these dependent types τ_i is associated with a particular relation R_i that encodes the generic explanatory cause linking τ_i to σ. The main point of departure from [4] and [3] lies in that these relations R_i will not be restricted to *qualia* roles, as was the case with QCs.

Following [3], we also make sure that some minimal information about syntactic structure can be exploited in our rules; for instance, we will have a formula *head* (ψ), where ψ is a term telling us that ψ is derived from some projection of the head or the head itself of the syntactic structure whose meaning we are currently trying to build up. They advance the following principle:

(3) **Head Typing Principle:** Given a compositional environment X with constituents A and B, and type assignments $A\colon \alpha$ and $B\colon \beta$ (in the type contexts for A and B, respectively) that clash, if A is the syntactic head in the environment, then the typing of A must be preserved in any composition rule for A and B to produce a type for X.

We now introduce the rules for the TCL. As usual a λ-expression denotes a functional type, i.e., a $\alpha \multimap \beta$ type. Such rules should be understood as reduction rules, thereby giving rise to equivalent term expressions. The first rule is *Application* and corresponds in terms of the type calculus to a rule of Modus Ponens for $\multimap$. *Application* is defined in terms of a context, *c*, which provides typing assignments to both the variable in the applicand and the argument.

(4) **Application:**

$$\frac{\lambda x\phi[t], \quad c(x\colon \alpha, t\colon \alpha)}{\phi[t/x], c}$$

The contexts that accompanies the rule of *Application* and other operations may be updated or combined, as the result of a rule being applied. We will refer to this second rule as *Merging Contexts* (noted with the symbol "^").

(5) **Merging Contexts:**

$$\frac{\lambda x\phi, c[t, c']}{\lambda x\phi[t], (c \hat{} c')}$$

For applications whose typing contexts are not locally satisfied, but are consistent with the types available in the type semilattice from the lexicon (cf. [4], [6]), we have the rule of *Type Accommodation*. This third rule covers what results in type unification, in which a supertype can unify with a subtype yielding the subtype as the result.

(6) **Type Accommodation:**

$$\frac{\lambda x\phi[t], c(x\colon \alpha, t\colon \beta), \ \alpha \sqcap \beta \neq \bot}{\lambda x\phi[t], c * (x, t\colon \alpha \sqcap \beta)}$$

These are the basic rules for computing with types. We will also assume more specific rules for complex $\otimes$-types which are relevant for the analysis of the genitive construction. We will discuss these additional rules in section (4).

2 A quick comparison with "classic" GL

In [3], it was shown how these rules can be used to model dynamic type shifts in copredication and coercion phenomena. These are phenomena already introduced in classic GL but our approach is quite different. To give a quick overview of the differences, we need to recall that in classical GL, there are four types of information:

1. **Argument Structure**: Specification of number and type of logical arguments.
2. **Event Structure**: Definition of the event type of a lexical item. Sorts include STATE, PROCESS, and TRANSITION.
3. **Qualia Structure**: A structural differentiation of the predicative force for a lexical item.
4. **Lexical Inheritance Structure**: Identification of how a lexical structure is related to other structures in the type lattices.

Our approach makes use of argument structure and event structure, but it understands these as types and the relations between types determine the lexical inheritance structure. GL originally only countenanced *qualia* though in [4] dot objects are also introduced.

The *qualia* structure of a word specifies four aspects of its meaning:

- CONSTITUTIVE: the relation between an object and its constituent parts;
- FORMAL: that which distinguishes it within a larger domain;
- TELIC: its purpose and function;
- AGENTIVE: factors involved in its origin or "bringing it about".

In way of illustration, below is the *qualia* structure for the word *door*:

(7)
$$\begin{bmatrix} \text{CONST} : \textbf{aperture(y)} \\ \text{FORMAL} : \textbf{physobj(x)} \\ \text{TELIC} : \textbf{walk\_through(P,w,y)} \\ \text{AGENTIVE} : \textbf{make(T,z,x)} \end{bmatrix}$$

A quick look at this *qualia* structure reveals some obvious technical problems of classical GL. Formulas rather than actual types are taken to be the values of the various *qualia* features. Classic GL's use of AVMs and other tools of unification indicates that the construction of a logical form should proceed by unification, but it's unclear how unification is to proceed on these structures. We can unify AVMs but how are we supposed to use the information in one of the *qualia* to interpret (8) in the sense that *enjoy the book* is preferably interpreted as *enjoy reading the book*?

(8) Sheila enjoyed {the book/the sonata/strong coffee}

In classic GL, there's no clear answer to our question. In contrast in the current system, the information within these formulas is added to the logical form by using one of the $\otimes$-Exploitation rules. That is, in our system complex type exploitation results in the addition of information to logical form. Our system also makes clear that what drives the explanation of the phenomena of logical metonymy are type mismatch and then type exploitation or introduction to correct the mismatch, whereas that's not at all clear in classic GL. Our approach also makes clear that *qualia* are just a special case of complex types, which leads us to the next criticism—that either the *qualia* are far too restrictive a set of relations to be really useful in determining the meaning of sentences like (8) or the relations that are associated with the *qualia* are too vague to be useful. While problems of overgeneration with *qualia* have been noticed (see e.g. [7]), our point is that they don't postulate a rich enough system of relations for interpreting many constructions. One type of construction that we help us put our argument to the test is the genitive construction.

3 The genitive construction

The genitive construction offers us a nice point of comparison to test how our more flexible typing system fares in contrast to the "classic" GL. We will in particular look at the account proposed by [8] that explicitly appeals to GL.

Let's start with some simple English examples of the genetive constructions:

(9) a. Bill's mother
b. Mary's ear
c. The girl's car

Interpreting phrases like (9) requires us to compute a *relation* between the two nominal referents that are introduced, respectively, by the specifier NP (aka *genitive NP* or *possessum*) and the head noun (aka the *possessor*) (see e.g. [9]). The main problem is that this relation is often not directly specified by the grammar, and as a result these constructions give rise to many

interpretations (i.e., many different relations can be inferred). For instance, depending on the context, *the girl's car* can be the car *owned*/*driven*/*dreamt about*/*designed*. . . by the girl. This great diversity of meanings has led some authors like [10] to the view that *any* relation was indeed possible.

A more nuanced view emerges from more recent works, especially that of Partee and Borshev (henceforth, P&B, e.g., [11, 12]) and Jensen and Vikner (henceforth, J&V, e.g., [8]). These authors show in particular that the semantic properties of the head noun both play a decisive role in constraining the different available interpretations; they also try to to model how pragmatic information gets into the picture. That the (syntactico-)semantic properties of the head are driving the interpretation is obvious with deverbals (e.g. *gift*, *destruction*) and relational nouns in general (e.g., *mother*, *edge*, *captain*). Thus, example (9a) seems to license a single interpretation (i.e., $mother\_of(x, bill)$), one where the relation is indeed *intrinsic* to the head noun.

These authors actually argue that this claim extends to many other, monadic nominals: these are *coerced* into relational denotations. For instance, nouns denoting *body parts* like *ear* give rise in (9b) to a strongly preferred interpretation, in which the denotation of the head noun is interpreted as a *part_of* the denotation of the genitive NP. Things are a little less clear with nominals like *car*, but there is still a clear sense that, out of explicit contexts, some relations (like *owned* or *driven*) are more preferred than others (say, *dreamt_about*); that is, one would probably want to infer the former relations as defaults.

[8] are probably the most explicit in terms of how lexical semantics of the head noun is exploited during the interpretation of the genitive. Assuming Pustejovsky's GL as their background, they argue that the default relations found in the genitive constructions are provided by the different *qualia roles* associated with the head noun. More precisely, the idea is that the genitive NP, acting as the functor, is able to type-coerce the (monadic) denotation of head noun into a relation: crucially, this relation corresponds to one of *qualia* roles in the lexical entry of the noun. This approach has *prima facie* some empirical appeal, for one can indeed find examples corresponding to most of the *qualia* roles:

(10) a. FORMAL: The car's design

b. TELIC: Mary's book (i.e., the book *read by* Mary)

c. AGENTIVE: Bill's cake (i.e., the cake *cooked by* Bill)

However, this approach has a number of problems. The main one is that the *qualia* are simply not a reliable guide for inferring the genitive relation. For one thing, we can't find any plausible example of a true CONSITUTIVE relation between a head and the genitive NP; there are lots of genitives that involve material constitution—e.g., *the car's metal*, *the sail's fabric*, *the computer's motherboard*, but in these examples the quale constitution is introduced by the genitive NP not the head. Trying to deal with this problem in the framework of GL has *de facto* an undesirable consequence, since it supposes either a relaxation of the Head Principle or an *ad hoc* redefinition of the CONST role; J&V opt for that latter alternative proposing that the CONST *quale* be interpreted as the *part-whole* relation.

This brings us to another point, namely that J&V are often led to adopt an extremely liberal view of the *qualia*. For instance, they assume that the same *qualia* role (namely, the CONST role) is responsible for explaning both the interpretation of genitive like *The girl's nose* (clearly a case of *part-whole* relation) and that of *The girl's team* (clearly, a *set-membership*). This makes the CONST role very unclear. For one thing, there is first a directionality problem: the *part-whole* relation is between the head N and the genitive NP, while the *set-membership* relation goes the other around (in potential violation of the head principle). There is a more crucial problem: these two relations *part-whole* and *set-membership* are rather different in

formal terms (e.g., the former is clearly transitive, while the latter is not). Under our analysis, nouns like *nose* and *team* having different intrinsic types (a *body-part* and a *group*, respectively) and therefore dependent types; and so, they will naturally give rise to different types of relations.

Another argument against *qualia* comes from morphologically rich languages that have grammaticalized certain genitive relations. Crucially, none of these languages have grammaticalized *qualia* roles therein. In Basque for instance, there are two genetive postpositions, namely an unmmarked *-(a)ren* suffix and a marked *-ko* suffix that specifically encodes (spatial) *localization* and which thus requires that the type of the object denoted by the nominal in this genitive case be a location (see [13]). Below is a minimal pair from Basque that gives an illustration of the two genitives:

(11) a. liburu**ko** argazkia
book-ko photo-def
'the photo from/in the book'

b. liburu**aren** argazkia
book-aren photo-def
'the photo of the book'

Crucially, the relation of localization is not part of the *qualia*.

Another illustration of the problem of using *qualia* is that, as it stands, the *qualia* fail to provide the relation *own* (i.e., material possession) that one finds (preferred) with most physical objects (e.g., *John's car*) in many different languages. To avoid this problem, J&V have to resort to a multi-purpose (and therefore rather vague) relation, namely CONTROL, which encompasses the TELIC role but also the relation of ownership, and basically everything that is not provided by the *qualia*. This again weakens their position. Even worse is the fact that the authors later on appeal to the TELIC role to account for the modification case with *favorite*.

These different points seem to make it clear that the *qualia* roles are in a way arbitrary; they don't seem to have any privileged epistemic status. We suggest that we dispense with the *qualia* altogether, replacing it with a more dynamic notion of typing.

Before turning to that, a number of points about the genitive constructions are still in order. First, *contra* previous accounts, it seems that constraints on the interpretation of the genitives not only come from the head:

(12) a. The rapist's victims

b. The designer's car

c. The composer's concerto

d. A mother's boy

These data suggest that both the type of the complement/modifier can help determine the genitive relation.

Another way we will move away from previous accounts is in assuming that the relation intrisically provided by relational nouns can also, under certain circumstances, be overridden:

(13) [Context: Picasso and Raphael both painted a mother with a child.]
Picasso's mother is bright and splashy —a typical cubist rendition. It makes Raphael's mother look so somber.

(14) Brancusi's kiss has recently been bought by the Philadelphia Museum of Art.

These data in conjunction with a static view of complex types like that enshrined in the notion of *qualia* lead us to conclude that genitive NPs always act as modifiers; they are never arguments of the head noun (see also discussion in [12]). However, we reject a static view of complex types. As we'll see below, the Head Principle and the view that genitive NPs are arguments is compatible with a more flexible system of types.

4 Our account of the genitive construction

4.1 Basic story

The previous section led us to conlude that *qualia* were not a reliable guide for inferring the genitive relation. In our account, *qualia* will be replaced by ⊗-types of the form ($\sigma \otimes_{R_1,\ldots R_n} (\tau_1 \cdots \tau_n)$). As noted, these types are asymmetric: they have the intrinsic type σ associated with the variable on the left and a set of dependent types $\tau_1 \cdots \tau_n$ are located on the right. The idea is that these dependent types are exploited when the functor type-clashes with the intrinsic type. Each of the dependent type τ_i is associated with a relation R_i, that will in turn serve as our genetive relation.

The basic ⊗-Exploitation rules are given in (15) and (**??**). These rules will have to be defeasible (this is noted with the prefix "/" on the conclusion), this corresponds to the fact that contextual factors can always override lexical inference (see section 4.4). A key feature of the ⊗-exploitation rules for our concern is that they introduce material that changes the arity of some predicate variable—replacing a property expression with a relational one at logical form. [3] argue that the relation links a variable with the complex type together with one of the dependent types that serves as an argument for a predicate. Crucially, however, they claim that the head type does not change, in keeping with the Head Typing Principle. This means that these type shifts are very local.

(15) ⊗-**Exploitation:**

$$\frac{\{\lambda P\phi(P(x)),\ c(P\colon \alpha \otimes_{R_i} (\ldots\beta_i\ldots) \multimap \gamma)\}[\psi, c'(\psi\colon \delta \multimap \gamma)], \delta \sqcap \alpha = \bot, \delta \sqcap \beta_i \neq \bot}{/\{\lambda P\phi[\frac{(R_i(y,x)\wedge P(y))}{P(x)}], c * y\colon \beta_i \sqcap \delta\}[\psi, c']}$$

(16) ⊗-**Exploitation**^{TS}

$$\frac{\{\lambda\mathcal{P}\phi c(\mathcal{P}\colon (\delta \multimap \gamma) \multimap \delta)\}[\lambda P\psi(P(x)),\ c'(P\colon [\alpha \otimes_{R_i} \beta] \multimap \gamma)], \delta \sqcap \alpha = \bot, \delta \sqcap \beta \neq \bot}{/\{\lambda\mathcal{P}\phi[\lambda P\psi\frac{\exists y(R_i(y,x)\wedge P(y))}{P(x)}, c * (x\colon (\alpha \otimes_{R_i} \beta), y\colon (\beta \sqcap \delta))]\}}$$

Sometimes, however, changing the types in this way interacts with the compositional semantics derived from the syntax/semantics interface, which may force the existential closure of some of the λ-abstracted variables in the new relational expression. This occurs in examples of logical metonymy where ⊗-Exploitation should convert the complement of *enjoy* from a simple DP like *the book* into something like *reading the book*. The syntactic structure for such a complement of *enjoy* is some projection of *I* as argued for in [14] –which on a compositional semantics for nominals like that in [14] must existentially bind the event variable introduced by the ⊗-Exploitation rule and the gerund *reading*. In any case we can expect that in many of the *qualia* motivating constructions there will be existential closure of variables introduced by ⊗-Exploitations within some complete functional complex, typically DP or IP, but also partial closure of other maximal projections like NP as in a *quick cigarette*, where an ⊗-Exploitation would transform $\lambda x(cigarette(x))$. But it appears that in the genitive, the closure principle is optional; in fact the empty head determiner will not only serve to bind

an extra variable as we would expect in X' theory, but also introduce a relation abstract. The reason for this is simple: genitive constructions, like noun-noun compounds and bridging descriptions require a relation to be determined between the variable introduced by the DP in genitive case and the variable that is introduced by the whole DP.

Consequently, we need a quite particular version of ⊗-Exploitation that takes care of the clash between a monadic property term and a construction that requires a relation term:

(17) **⊗-Exploitation^R:**

$$\frac{\{\lambda P\phi, c(P\colon (\delta \multimap (\epsilon \multimap \gamma)))\}[\lambda v\psi(v), c'(v\colon \alpha \otimes \beta)], \alpha \sqcap \delta \neq \bot, \gamma \sqcap \delta' \neq \bot, R_\beta(u,v) \rightarrow u : \alpha \otimes \beta \wedge v : \gamma}{\{\lambda P\phi, c\}[\lambda w \lambda v[\psi(v) \wedge R_\beta(v,w)]], c * w\colon \alpha \sqcap \delta, v : \gamma \sqcap \delta'}$$

We will also need an ⊗-Introduction rule to handle certain interpretations of the genitive construction. It was a contention in [3] that such a rule was needed; here we provide empirical evidence for it. We will depart from Asher and Pustejovsky's format for •-introduction rules, and simply write a rule for revising type contexts. Thus, this rule isn't ampliative at all and does not affect the logical form of a construction, making it quite simple to state.

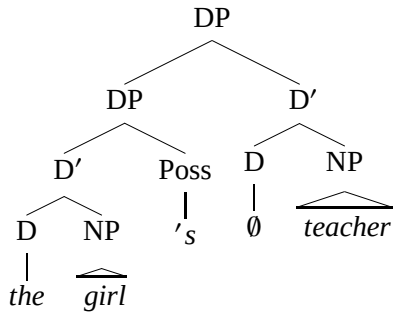
(18) **⊗-Introduction:**

$$\frac{(\lambda P\phi,\ c(P\colon ((\delta \otimes \beta) \multimap (\epsilon \multimap \gamma))))\ \ [\lambda v\psi(v)],\ c'(v\colon \alpha), \alpha \sqcap \delta \neq \bot}{(\lambda P\phi, c)[\lambda v\psi(v)], c * (v\colon \alpha \otimes \gamma), \text{provided } R_\beta(x,y) \rightarrow R_\gamma(y,x)}$$

4.2 A little syntax

As mentionned, the syntax we will assume for the genitive construction follows a DP analysis (cf. [15]; see also [14]) in which the construction is headed by an empty functional D head, which assigns (genitive) case to the Spec, DP. This is shown below for *the girl's teacher*:

(19)



Previous accounts like J&V and P&B and use a much simpler syntax, but one which fails to predict quantificational facts.

4.3 Interpreting the genitive

Our basic case involves a deverbal, relational noun in head position; e.g. as in *the girl's teacher*. This is the simplest case, since the relation is intrinsic to the head N (i.e., *teach*). Below are the bits of denotation and the compositional derivation –types are left out for now:

(20) a. $[\![\emptyset]\!] = \lambda R \lambda \Phi \lambda S\,(\Phi(\lambda y \exists! x(R(y)(x) \wedge S(x))))$

b. $[\![teacher]\!] = \lambda y \lambda x(teach(x,y))$

c. $[\![\emptyset]\!]([\![teacher]\!]) = \lambda \Phi \lambda S\,(\Phi(\lambda y \exists! x(teach(x,y) \wedge S(x))))$ (Application and β-reductions)

d. $[\!['s]\!] = \lambda X.X$

e. $[\![the\ girl]\!] = [\![the\ girl's]\!] = \lambda P(\exists! z(girl(z) \wedge P(z)))$ (Application and β-reduction)

f. $[\![the\ girl's\ teacher]\!] = \lambda S\,(\exists! z(girl(z) \wedge \exists! x(teach(x,z) \wedge S(z))))$ (Application and β-reductions)

We now move to a case involving a monadic nominal like *nose* in head position, as in:

(21) Every girl's nose had been pierced.

This type of cases will be be handled using our ⊗-Exploitation^R rule. This rule in effect formally models the meaning shift from a property to a relation which is directly dependent on the complex type of individual denoted by the property: in the case of *nose* (as for other body parts), the type we exploit is PHYS-OBJ ⊗ BODY-PART:

(22) a. $[\![\emptyset]\!] = \lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x)))), \langle x\colon e, y\colon e\rangle$

b. $[\![nose]\!] = \lambda z(nose(z)),\ \langle z\colon$ PHYS-OBJ ⊗ BODY-PART$\rangle$

c. $[\![\emptyset]\!]([\![nose]\!]) = \lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x)))[\lambda z(nose(z))]\langle x\colon e, y\colon e, z\colon$ PHYS-OBJ ⊗ BODY-PART$\rangle$ (Application and Merge)

d. $\lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x)))[\lambda v\lambda y(nose(y) \wedge body-part(y,v))]\langle x\colon e, y\colon$ PHYS-OBJ ⊗ BODY-PART$, v :$ ANIMAL$\rangle$ (⊗-Exploitation^R)

e. $\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(nose(x) \wedge part-of(x,y)) \wedge S(x)))\langle y\colon$ PHYS-OBJ ⊗ BODY-PART$, x :$ ANIMAL$\rangle$ (β-reductions)

f. $[\![every\ girl's]\!] = \lambda P(\forall z(girl(z) \rightarrow P(z)))\langle z :$ HUMAN$\rangle$

g. $[\![every\ girl's\ nose]\!] = \lambda\Phi\lambda S(\Phi(\lambda y\exists!x(nose(x) \wedge part-of(x,y)) \wedge S(x)))[\lambda P(\forall z(girl(z) \rightarrow P(z)))]\langle x\colon$ PHYS-OBJ ⊗ BODY-PART$, y :$ ANIMAL$, z :$ HUMAN$\rangle$ (Application and Merge)

h. $[\![every\ girl's\ nose]\!] = \lambda S(\forall z(girl(z) \rightarrow \exists!x(nose(x) \wedge part-of(x,z)) \wedge S(x)))\langle x\colon$ PHYS-OBJ ⊗ BODY-PART$, z :$ ANIMAL$\rangle$ (β-reductions and Accomodation)

We now tackle a case where the nominal head has a simple type as in:

(23) The girl's rock

This is an example where we have to use a ⊗-Introduction. But what introduces a complex ⊗-type in this genitive construction? We hypothesize that it is the empty determiner itself. But in contradistinction to the Head Typing Principle, this type can be overridden either by a complex type either in the NP or in the DP in SPEC position. The empty determiner posits a default according to which the first argument of the relation abstract is of the type $e \otimes$ POSS where POSS stands for the relation of possession (i.e., ownership). Let's consider the derivation here. We'll only put in the one, complex default type for one variable in the determiner, and we'll assume POSS' to be the converse of POSS; i.e., $POSS(x,y) \rightarrow POSS'(y,x)$ and that $POSS'(x,y) \rightarrow (y : e \otimes$ POSS $\wedge\ x : e)$.

(24) a. $[\![the\ girl]\!] = \lambda P(\exists!z(girl(z) \wedge P(z))), \langle z :$ HUMAN$\rangle$

b. $[\![\emptyset]\!] = \lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x)))), \langle /y : e \otimes$ POSS$\rangle$

c. $[\![rock]\!] = \lambda v rock(v), \langle v\colon$ PHYS-OBJ$\rangle$

d. $[\![\emptyset]\!]([\![rock]\!]) = \lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x))))[\lambda v rock(v)], \langle /y : e \otimes$ POSS$, z :$ HUMAN$\rangle$ (Application and Merge)

e. $\lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x))))[\lambda v rock(v)], \langle /y : e \otimes$ POSS$, v\colon$ PHYS-OBJ ⊗ POSS$\rangle$ (⊗-Introduction)

f. $\lambda R\lambda\Phi\lambda S(\Phi(\lambda y\exists!x(R(y)(x) \wedge S(x))))[\lambda v\lambda u rock(v) \wedge owns(v,y)], \langle /y : e \otimes$ POSS$, v\colon$ PHYS-OBJ ⊗ POSS$, u : e\rangle$ (⊗-Exploitation^R)

The rest of the derivation proceeds as before except that the default type for *y* is overridden by the hard information that our type introduction rule now gives us. And then when we apply the determiner more specific type information will affect the second argument of POSS'. The final result we get is:

(25) $\lambda S(\exists!z(girl(z) \wedge \exists!x(rock(x) \wedge owned-by(x,z) \wedge S(x)))), x : \text{PHYS-OBJ} \otimes \text{POSS}, x : \text{HUMAN}.$

The question is how do we get more specific interpretations for the relational abstract *R* when those are given by the DP in SPEC position. It should be noted that some of these are uninterpretable—for instance, *the child's woman*, except as cases of possession, even though *child* is relational and arguably *woman* is not. Other putative examples, however, are *the artist's object*, which we think can mean the object created by the artist. *Object* is not plausibly relational; when it occurs in the genitive construction with another noun that introduces a λ-bound variable of simple type, then the only relation that seems to hold between the two variables is POSS. But in this case, we get a different interpretation. How is this possible? What needs to happen, we think, is that somehow the typing information from the DP in SPEC has to affect the interpretation of *R* in the possessive determiner; and the only way we can do this in our framework is to relax the order in which functional applications are made, as argued for in [14]. At this point our type logic becomes polymorphic. We don't go into details but essentially this means that we can "freeze" the application of the determiner meaning to the head NP and go ahead and combine the meaning of the DP in SPEC with that result. This would give us the more specific typing information from the DP onto the relational abstract. And this in turn will allow us to construct a more specific $\otimes$-type via $\otimes$-Introduction for the λ-bound variable in the NP. And then we can proceed as above to get a relational meaning for the NP via $\otimes$-Exploitation^R. Our final logical form for (26) is (26'):

(26) The artist's object

(26') $\lambda S(\exists!x(artist(x) \wedge \exists!y(object(y) \wedge created-by(y,x) \wedge S(y)))$

4.4 Contextually Sensitive Typing

The examples we've looked at so far all use complex typing information that we think is plausible to suppose in the lexicon, though it goes far beyond the sort of information stored in the *qualia* of classic GL. However, it is well-known that discourse can affect typing and that these types can propagate through the discourse. Here's a typical example:

(27) a. All the children were drawing fish.

b. Suzie's salmon was blue.

Here we need to draw on a theory of discourse structure like SDRT, which would posit that (27a) and (27b) stand in an *Elaboration* relation, or even *Instance* ([14]). But this will be so, only if Suzie's salmon is interpreted in a very particular way; namely, the salmon is the fish that Suzie is drawing. Without this link, we cannot make any clear discourse link between the two sentences, threatening the discourse's coherence. So how does this work? *salmon* introduces a variable whose subtype is that of the variable introduced *fish*—i.e., SALMON $\sqsubseteq$ FISH. But *fish* has had it's type changed in the context because it is an argument of *draw*. *draw* coerces *fish* into a complex type, one where we have FISH$\otimes$PICTURE-OF. The discourse context dictates that *salmon* inherits this complex type, this inheritance being crucial to computing the discourse relation of *Instance*. This complex type now helps specify the relation between Suzie and her salmon, but by itself it doesn't suffice, because Suzie is not the picture of the salmon. But Suzie is an agent and agents, among other things, by default make things. That is, the variable introduced by *Suzie* is also an $\otimes$-type: a PERSON$\otimes$ MAKER-OF type. The discourse context also facilitates the selection of this particular activity associated with agents, since *Instance* requires that Suzie be one of the children, all of whom are making pictures. The clash between the intrinsic type of the variable associated with *Suzie* and the typing on the

variable associated with the picture induces another ⊗-Exploitation, this time linking Suzie as a maker of the picture of the fish. By performing ⊗-Explotation twice, we get that there is a picture of a fish, which Suzie is making, therefore avoiding type clashes incoherent discourse.

This example shows several things. First, a simple, static typing system like the *qualia* in classic GL cannot possibly do justice to the phenomena. Second, lexical semantics has to be integrated with a discourse semantics in order to account for the incredible complexity and context sensitivity of predication. We hope that our system of flexible typing will make inroads on these thorny issues.

References

[1] W. V. O. Quine. *Word and Object*. MIT Press, 1960.

[2] Jerry Fodor and Ernest Lepore. The emptiness of the lexicon, critical reflections on J. Pustejovsky's The Generative Lexicon. *Linguistic Inquiry*, 29(2):269–288, 1998.

[3] N. Asher and J. Pustejovsky. The metaphysics of words in context. University of Texas at Austin and Brandeis University, available from `nasher@bertie.la.utexas.edu`, 2001.

[4] J. Pustejovsky. *The Generative Lexicon*. MIT Press, 1995.

[5] N. Asher and A. Lascarides. The semantics and pragmatics of metaphor. In P. Bouillon and F. Busa, editors, *The Language of Word Meaning*, pages 262–289. Cambridge University Press, 2001.

[6] A. Copestake and E. J. Briscoe. Semi-productive polysemy and sense extension. *Journal of Semantics*, 12(1):15–67, 1995.

[7] D. Godard and J. Jayez. Towards a proper treatment of coercion phenomena. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL 1993)*, pages 168–177, 1993.

[8] Carl Vikner and Per Anker Jensen. A semantic analysis of the english genitive. interaction of lexical and formal semantics. *Studia Linguistica*, 56:191–226, 2002.

[9] Chris Barker. *Possessive Descriptions*. CSLI, Stanford, CA, 1995.

[10] Edwin Williams. The NP-cycle. *Linguistic Inquiry*, 13(277295), 1982.

[11] Vladimir Borschev and Barbara H. Partee. Genitive modifiers, sorts, and metonymy. *Nordic Journal of Linguistics*, 24(2):140–160, 2001.

[12] Barbara Partee and Vladimir Borshev. Genitives, relational nouns, and argument-modifier ambiguity. In C. Fabricius-Hansen E. Lang C. Maienborn, editor, *Modifying Adjuncts*, Interface Explorations. Mouton de Gruyter, 2003.

[13] Michel Aurnague. Basque genitives and part-whole relations : typical configurations and dependences. Technical report, Université Toulouse-Le Mirail, 1998. 50p.

[14] N. Asher. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers, 1993.

[15] Steven Abney. *The English Noun Phrase in Its Sentential Aspect*. PhD thesis, MIT, 1987.