Compositionality and Coercion in Categorial Grammar

Paul Buitelaar and Anne-Marie Mineur

Abstract
In this paper we introduce *coercion* in Lambek calculus [7]. Coercion in natural language is studied within GL theory [13] and is named after common usage in programming language design for a similar technique [2] [5].

Essential about coercion is that it changes the compositional semantics, while it leaves the syntactic type unaltered. Because of this, a functor may take an argument that does not fit its requirements, in fact $[a/b, c \vdash a]$ can be a valid inference if coercing c into b is allowed. This is very unusual in Categorial Grammar and it would imply an adjustment to its traditional Fregean view on compositionality.

We also show how coercion relates to recent proposals on disjunctive categories in sign-based Lambek calculus [10], in the sense that coercion relates disjunctive syntactic types through a shared underlying lexical semantics.

1 Introduction

Categorial Grammar (CG) is a compositional theory, in the sense that all information is built up immediately from the composing parts. As observed by several researchers however, such a basic Fregean concern with composition is not enough for a cognitive or computational system [13] [18]. Frequently there is additional information inbetween the composing parts, which is inferred by associative reasoning (e.g.: *metonymy* or *metaphor*).

To account for this kind of inferences in CG some additions have to be made to its traditional way of representing word meaning. We borrow from Generative Lexicon (GL) theory [13] in introducing a richer notion of lexical semantics called *qualia structure* and in giving up the idea of defining the multiple meanings of a word by presenting a fixed enumeration of senses. Fixed enumerations lead to weak compositionality. Instead, GL focuses on generating word senses dynamically by way of *coercion*, which leads to a system of strong compositionality[1] with polymorphism.

2 Coercion

One of the central issues in GL theory is to account for *coercion* in natural language.

1a) John began a book. (NP)
1b) John began *reading* a book. (GP)
1c) John began *to read* a book. (inf S)

1. The reader may be familiar with the definition of strong vs. weak compositionality given in [11] by Morrill and Carpenter who claim their version of weak compositionality to be preferable over traditional Montagovian strong compositionality. Instead, with lexical ambiguity in mind, we argue with [14] by introducing polymorphism, for a modified notion of strong compositionality itself.

A typical example of coercion we see in sentences 1a) through 1c) with three different subcategorization patterns for the verb **begin**. The second argument can either be a nominal phrase (NP), a gerundive phrase (GP) or an infinitival S. Instead of giving three different lexical entries for **begin**, GL relates all three of these patterns to each other by their common underlying lexical semantics.

Not only the GP and the infinitival S represent an event, but also the NP **a book**, although in nature an artifact, implies the event of **reading** or **writing a book**. This is a *metonymic* relation between the lexical semantic object that **book** represents and the two lexical semantic objects that are represented by **read** and **write**. This systematic relation between different denotations Pustejovsky calls *logical polysemy*, following [1] [12].

So, sentence 1a) can actually be reinterpreted as either 1b) or 1c). This process of reinterpretation is called *coercion* [13].

2a) The president of BMW announced a new model. ($NP_{artifact}$)
2b) The president of BMW announced *the production of* a new model. (NP_{event})
2c) The president of BMW announced *that they will produce* a new model. (S_{event})

We see the same pattern in sentences 2a) through 2c) for the verb **announce**. Again, we have a systematic relation between the $NP_{artifact}$ **a new model** in 2a) and the NP_{event} **the production of a new model** in 2b) and the S_{event} **that they will produce a new model** in 2c). By way of coercion we are able to reduce these different subcategorization patterns to a common underlying lexical semantics of an event.

This much about coercion for now, below we will see in more detail how this mechanism actually works in the wider context of GL theory.

3 Qualia Structure and Polymorphism

In GL, lexical semantics is represented by socalled *qualia structure*, which represents different aspects of the denotation of a lexical semantic object. Basically, qualia structure takes the form of a polymorphic type that acts as a sense generator, in the same way as polymorphic types in general are to be seen as type generators. Coercion then is the generation process of the polysemous related senses that are represented in the qualia structure.

3.1 Qualia Structure

GL presumes a relational model of lexical semantics [3], which represents word meaning not by decomposing it into primitives as in lexical decomposition models (all of which go back on [6]), but by showing the relations between lexical semantic objects. These relations are to be seen in a formal way, constituting a semantic network, through which reasoning paths connect lexical semantic objects to each other.

Essentially, qualia structure represents four different relations that lexical semantic objects can have with each other. More specifically, GL distinguishes *hyponymy* (the traditional is-a relation in AI), *meronymy* (the part-whole relation), *causality* and *purpose*, each of which are given different role names: the FORMAL, CONSTITUTIVE, AGENTIVE and TELIC roles.

Examples for each in the case of **book** are: a book can also be seen as a **physical object**[2], a book has **pages**, a **cover** and more, the existence of a book is caused by **writing** it and its purpose is to be **read**.

We can represent this in a constraint-based fashion [17]:

$$\begin{bmatrix} \textbf{book} & \\ & \\ \text{QUALIA\_STRUCT} = & \begin{bmatrix} \text{FORMAL} = \textit{phys\_obj} \\ \text{CONSTITUTIVE} = [\textit{pages}, \textit{cover}, ...] \\ \text{AGENTIVE} = \textit{write} \\ \text{TELIC} = \textit{read} \end{bmatrix} \\ & \end{bmatrix}$$

Figure 1: Qualia Structure for *book*

In the case of **model** the FORMAL role would be filled by an artifact, the CONSTITUTIVE is a collection of subparts, the AGENTIVE role is filled by a collection of prototypical 'create' events and finally the TELIC role by a collection of 'show' events, because this seems to be the typical purpose of a **model**.

$$\begin{bmatrix} \textbf{model} & \\ & \\ \text{QUALIA\_STRUCT} = & \begin{bmatrix} \text{FORMAL} = \textit{artifact} \\ \text{CONSTITUTIVE} = [\textit{front}, \textit{back}, ...] \\ \text{AGENTIVE} = [\textit{design}, \textit{produce}, ...] \\ \text{TELIC} = [\textit{display}, \textit{show}, ...] \end{bmatrix} \\ & \end{bmatrix}$$

Figure 2: Qualia Structure for *model*

An additional word has to be said about the nature of the lexical semantic objects we are talking about. As distinguished by Weinreich [20] there are *contrastive* and *complementary* senses. Contrastive senses are for instance the very different meanings of the word **bank**, the *riverside* meaning is unrelated in every respect to its *financial institution* meaning. Complementary senses on the other hand are not contrasting each other, but are related by *logical polysemy* as explained above.

So, the lexical semantic objects we talk about are particular *contrastive* senses of words, like for instance the *riverside* meaning of **bank**. The qualia structure then represents the different *complementary* senses of that particular *contrastive* sense.

In the case of **model** we concentrate on its *contrastive* sense of a *(proto)type*, thereby discarding other *contrastive* senses like that of a *fashion model*.

2. Sorts like *artifact* and *phys_obj* are not to be seen as primitives but as labels for clusters of lexical semantic object types. The main objective of lexical semantic study is dividing up the world in clusters like these and thereby generalizing over individual lexical semantic objects to get to linguistically informed categories.

3.2 Parametric Polymorphism

GL is a type driven theory of linguistic analysis. In this context, qualia structure is best seen as a polymorphic type with the different qualia roles acting as parameters on the proper interpretation of the lexical semantic object that the qualia structure represents. This is an instance of what is called *parametric* polymorphism in the programming languages literature [2] [5] [8]. An example is the generic qualia structure for artifacts.

$$\begin{bmatrix} \textbf{artifact} & \\ \textsc{qualia\_struct} = & \begin{bmatrix} \textsc{formal} = \textit{entity} \\ \textsc{constitutive} = \textit{artifact} \\ \textsc{agentive} = \begin{bmatrix} \textbf{create\_event} & \\ \textsc{arg1} & = \textit{animate} \\ \textsc{arg2} & = \textit{artifact} \end{bmatrix} \\ \textsc{telic} = \begin{bmatrix} \textbf{event} & \\ \textsc{arg1} & = \textit{animate} \\ \textsc{arg2} & = \textit{artifact} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3: Qualia structure for artifacts

This lexical semantic object type is polymorphic between the simple types (or sorts) entity and artifact, denoted by the FORMAL and CONSTITUTIVE roles and the function[3] type animate $\rightarrow$ (*artifact* $\rightarrow$ *event*), denoted by the AGENTIVE and TELIC roles.

3.3 Implicit and True Coercion

Parametric polymorphism deals with complex types, constructed from more simple ones by recursion. By way of context dependent reasoning the proper type will be inferred from such a parametric polymorphic type. This is what Pustejovsky calls *true coercion* [15], to distinguish it from socalled *implicit coercion*[5], which depends on a subset relation between types.

An example of *implicit coercion* is the subset relation between the sorts create_event and event. It may be obvious that every create_event at the same time is to be seen as an event. This relation between types or sorts is formalized by a partial ordering:

$$\text{create\_event} \leq \text{event}$$

True coercion extends traditional Fregean composition by introducing a form of associative reasoning that builds on the lexical semantics (qualia structure) of an argument in a dynamic way. That is, not a priori, but during the derivation, taking context into account (requirements of the functor). GL considers the role of the argument in this not a passive one, but on the contrary as active. The argument as well as the functor take a role in their

3. Note that before we filled this qualia role with a mere sort, here we fill it with a fullblown function. Sorts are then to be seen as abbreviations for functions like these.

mutual interpretation. This is what Pustejovsky calls *cocomposition* [13]. In the next section we will give a few examples of such associative reasoning.

4 Some Examples of Coercion

As you may recall, in sentence 2a), which is repeated here, the $\mathrm{NP}_{artifact}$ **a new model** has to be coerced into an event, in order to satisfy the requirements of the verb **announce**.

2a) The president of BMW announced a new model.

The artifact denotation of **model** is coerced into an event by focussing on the AGENTIVE or TELIC roles in its qualia structure, where we find a collection of lexical semantic objects of sort create_event and show_event respectively. By *implicit coercion*, these are automatically also of sort event as desired.

$$\left[\begin{array}{ll}\textbf{model} & \\ \textsc{qualia\_struct} = & \left[\begin{array}{l}\textsc{formal} = \textit{artifact} \\ \textsc{constitutive} = [\textit{front}, \textit{back}, ...] \\ \textsc{agentive} = [\textit{design}, \textit{produce}, ...] \\ \textsc{telic} = [\textit{display}, \textit{show}, ...]\end{array}\right] \\ & \end{array}\right]$$

Figure 2: Qualia Structure for *model* – repeated

A more complicated example of coercion we find in sentence 3).

3) BMW announced a new model.

The verb **announce** requires an animate phrase in its first argument position, so the artifact denotation of **BMW** has to be coerced into an animate denotation. We can do this for instance by focussing on the people that are involved in the *production* of these *artifacts* (BMW cars). That is, we can construct a type path from the type **BMW**, through the create_event in its AGENTIVE role to the animate type in the ARG1 role we find there.

$$\left[\begin{array}{ll}\textbf{BMW} & \\ \textsc{qualia\_struct} = & \left[\begin{array}{l}\textsc{formal} = \textit{artifact} \\ \textsc{constitutive} = \textit{artifact} \\ \textsc{agentive} = \left[\begin{array}{ll}\textbf{produce} & \\ \textsc{arg1} & = \textit{animate} \\ \textsc{arg2} & = \textit{artifact}\end{array}\right] \\ \textsc{telic} = \textit{event}\end{array}\right] \\ & \end{array}\right]$$

Figure 4: Qualia structure for *BMW*

In the remaining part of this paper we will investigate how coercion works out in a CG framework. We will show that an extended notion of Fregean composition can be introduced into CG with only a slight adjustment to its basic apparatus.

5 Sign-based Lambek Calculus

According to Frege [4] the meaning of a compound expression is a function of the meaning of the composing parts (the compositionality principle). That implies that the entire meaning of the compound expression is to be derived from the meaning of the composing parts, and no external semantic rules should be necessary.

The extreme lexicalism that CG adheres to fits very well in the context of compositionality. More recent versions of CG store combinatorial information (types) in the lexicon, as well as orthographical and logical properties, so that the calculus operates on all dimensions simultaneously, in an exclusively type-driven fashion. Not only is the meaning independent of external rules, the same holds for the combinatorical properties as well as the prosodical behaviour. A typical sign in such a system contains a representation of type, semantics and structure, $\langle A, \mathsf{t}, \sigma\rangle$, for combinatorics, logical form and prosodics, respectively. In a sign-based Lambek calculus [9] the rules for axiom [Ax], application [L] and abstraction [R] are as in figure 5.

$$[Ax]\frac{}{\langle A, \mathsf{t}, \sigma\rangle \vdash \langle A, \mathsf{t}, \sigma\rangle}$$

$$[L/]\frac{T \vdash \langle B, \mathsf{u}, \tau\rangle \quad \Gamma(\langle A, \mathsf{tu}, \sigma + \tau\rangle) \vdash C}{\Gamma(\langle A/B, \mathsf{t}, \sigma\rangle, T) \vdash C} \qquad [R/]\frac{\Gamma(\langle B, \mathsf{v}, \tau\rangle) \vdash \langle A, \mathsf{u}, \sigma + \tau\rangle}{\Gamma \vdash \langle A/B, \lambda \mathsf{v}.\mathsf{u}, \sigma\rangle}$$

$$[L\backslash]\frac{T \vdash \langle B, \mathsf{u}, \tau\rangle \quad \Gamma(\langle A, \mathsf{tu}, \tau + \sigma\rangle) \vdash C}{\Gamma(\langle B\backslash A, \mathsf{t}, \sigma\rangle, T) \vdash C} \qquad [R\backslash]\frac{\Gamma(\langle B, \mathsf{v}, \tau\rangle) \vdash \langle A, \mathsf{u}, \tau + \sigma\rangle}{\Gamma \vdash \langle B\backslash A, \lambda \mathsf{v}.\mathsf{u}, \sigma\rangle}$$

Figure 5: Derivation rules for axiom, application and abstraction

As can be seen, the resolving of, or abstracting over an argument position not only involves the type of the sign (A/B, A, B), but also its logical representation (t, u, v) and its structure (τ, σ). Moreover, all operations take place in parallel, and only with the given information – no external rules are required.

As nice and elegant as this approach is, it also has a few drawbacks. The way it is now, it leaves very little room for a wider interpretation than just what is given. No flexibility, no room for new meanings, no associative interpretations are allowed. The only way to account for polymorphism that has been given so far, is by simply listing the disjunctive set of possible categories, with their fixed meanings attached to them [10].

5.1 Conjunction and Disjunction

In this section we will discuss the technique that Morrill uses for polymorphic types. Morrill's approach enables us to bundle the superficial differences in the realizations of words that are otherwise strongly related in meaning and form. 'To be' is one example: this may either require a noun phrase or an adjectival phrase, with the lambda term changing accordingly.

'To paint' is another example: this does not impose any different syntactic requirements to its arguments, but it may either mean to cover something with paint ('to paint a wall') or to make a picture of something ('to paint a portrait').

With the aid of the conjunction and disjunction operators, $\vee$ and $\wedge$, respectively, it is possible to join various appearances of a polymorphic word. As can be seen in the examples given above there are different levels on which a word can be polymorphic, either in combinatorical requirements or in meaning, a property that is mirrored nicely in the rules for conjunction and disjunction (see figure 6).

$$[L\wedge_a]\frac{\Gamma(A) \vdash C}{\Gamma(A \wedge B) \vdash C} \qquad [L\wedge_b]\frac{\Gamma(B) \vdash C}{\Gamma(A \wedge B) \vdash C}$$

$$[R\wedge]\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

$$[L\vee]\frac{\Gamma(A) \vdash C \quad \Gamma(B) \vdash C}{\Gamma(A \vee B) \vdash C}$$

$$[R\vee_a]\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad [R\vee_b]\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$$

Figure 6: Derivation rules for conjunction ($\wedge$) and disjunction ($\vee$)

Conjunction essentially means that if a given type A (or B) in a sequent Γ is sufficient to yield a single type C then it doesn't hurt to add B (or A) and thus have A$\wedge$B in that same context Γ. This will yield that same result C. If on the other hand a sequent Γ can yield A as well as B, then it yields A$\wedge$B as well.

Disjunction means that if both A and B are by themselves sufficient in a given sequent Γ to yield C, then Γ can also yield C with A$\vee$B. On the other hand, if a sequent Γ can yield A (or B), then it can yield A$\vee$B as well.

From these rules follows that operators behave differently when required or when given[4], and what is more, that conjunction and disjunction behave in exactly opposite ways. This means that any kind of behaviour can be described. A pairing of two types, one of which will turn out to be redundant, may either occur as given when connected by the $\wedge$-operator, or as required when connected by the $\vee$-operator. Similarly, a pairing of two types that will both end in a sensible lambda-term, will be connected by the $\vee$-operator when given, and by the $\wedge$-operator when required. According to Morrill [10], however, "it appears that application to linguistic derivation requires just *use* of conjunction, and *proof* of disjunction." Or, in other words, we want only to be able to describe the case where in the course of the derivation one type turns out to be sufficient. From a categorial perspective, it is ofcourse of little use to have a pair of types that should be equivalent because they should fit in the same context. For semantic reasons, however, one might consider using them nonetheless.

What this approach does, is allow us to pair alternative rigid typings of orthographically

4. Positive and negative occurrence, respectively, or in the antecedent and in the succedent, or as a rule of use and a rule of proof, or subcategorized-for category and argument; this latter definition is helpful but a bit too narrow.

identical realizations of a word. No dynamic interpretation is possible, just the choice between two[5] given semantical representations. This still enforces us to add to our lexicon every new shade of meaning we might find while using it in natural language applications.

Obviously, our solution circumvents this problem. What we suggest is to incorporate the approach that has been suggested in GL theory, and embed it in the strict formalism that CG employs. Adding the lexical semantic information is not the hardest bit. As we shall see in subsection 5.2, the construction of the CG-sign easily allows expanding with extra information.

5.2 Sign-based Types

In order to distinguish more subtle nuances than just syntactic types, features have been used to indicate sorts like natural gender, number and person. By introducing features we advance from a propositional approach to a predicational approach. As with predicational logic, the formulae we use are not simply p and q, but instead are more complex and consist of a predicate with some N arguments: $p(arg_1, \ldots, arg_n)$. With this extension, our basic types are no longer N, NP and S, but N(*number, person, gender, sex, case,*...), NP(*definiteness, number, person, gender, sex, case,*...) and S(*tense, aspect,*...). More detailed requirements can be made.

The laws of existential and universal quantification can ofcourse be applied to the arguments of these predicate-argument constructions. The presence of the existential quantifier enables us to generalize over all instantiations a variable might have, and so to abstract over specific instantiations. The properties of existential quantification may be used for coordination of unlike categories: "men and women", for instance, which requires generalization over gender features. We will omit the rules here. Of greater general use is the universal quantifier, which quantifies over all possible instantiations of the variable and hence may be replaced by any instantiation (see figure 7). This is exactly what happens with the normal instantiation of a variable under unification, and for strictly syntactical features unification is all we need. For purely syntactic features we can assume the "standard" notation of unification, and use free variables instead of quantified-over ones.

$$[L\sqcap]\frac{\Gamma(\psi : \sqcap vA) \vdash C}{\Gamma(\psi : A[v \leftarrow t]) \vdash C} \qquad\qquad [R\sqcap]\frac{\Gamma \vdash \phi : A}{\Gamma \vdash \phi : \sqcap vA}$$

Figure 7: "Normal" universal quantification over features – instantiation

Note that typically semantic information like sex (as opposed to grammatical gender) is also noted in this part of the sign, instead of in the lambda term. The type contains *all* the information that is important to the combinatorial properties of the lexical unit, whether syntactic or semantic. Any feature that does not only have combinatorical importance but semantic relevance as well, could be copied to the lambda term and thus remain visible in the eventual result. This can be done by distinguishing a semantically potent universal quantifier, as in figure 8. When instantiating a bound variable with a certain value t, then this value t will also be applied to the lambda term. When introducing a universal quantifier over some feature, this same feature in the lambda term will be bound by a lambda operator. We will call this *feature jumping*.

5. Two representations, or more; it is ofcourse possible to pair complex types, and hence obtain the equivalence of a n-ary branching con-/disjunction.

$$[L\bigwedge]\frac{\Gamma(\psi : \bigwedge vA) \vdash C}{\Gamma((\psi\ t) : A[v \leftarrow t]) \vdash C} \qquad [R\bigwedge]\frac{\Gamma \vdash (\phi : A)}{\Gamma \vdash (\lambda v.\phi) : \bigwedge vA}$$

where v is not free in ϕ

Figure 8: Semantically active universal quantification over features

6 Coercion in Sign-based Lambek Calculus

The lexical representation in GL is rich, all aspects of a concept are present. This is an important point to note, because it means that the signs do contain all the information that will eventually constitute the meaning of the sentence, be it that this information may not be present on the surface, but has to be derived by associative reasoning. In the next subsections we will first see how CG is extended with a qualia structure, followed by an extension of the CG machinery, that loosens the restrictions on derivability and takes care of the semantic representation.

6.1 Qualia structure in the CG sign

As can be seen in figure 9, to add a qualia structure to the CG sign is rather straightforward. Qualia structure is simply a complex feature in the basic type.

$$\mathrm{NP}\left(\begin{array}{ll} \text{person, number, qualia (} & \text{formal : Qf,} \\ & \text{constitutive : Qc,} \\ & \text{telic : Qt,} \\ & \text{agentive : Qa)} \end{array}\right)$$

Figure 9: Qualia structure in the CG sign

In order to keep our derivations legible, we will use abbreviations of these types. In CG-format we encode **announce** and **model** as follows: **announce** basically has type (NP\S)/(NP∨S), and **model** has type NP. As can be seen here, not all qualia roles need to be instantiated, especially in the case of requirements.

model

NP(3rd,sing,np1Q)

np1Q = formal : artifact,
constitutive : artifact,
telic : create_event,
agentive : show_event

announce

(NP(3rd,sing,np2Q)\S(finite,s1Q))/(NP(3rd,sing,np3Q)∨S(finite,s2Q))

$$\begin{array}{lll} np2Q = & \text{formal : animate,} \\ & \text{const : Qc1,} \\ & \text{telic : Qt1,} \\ & \text{agentive : Qa1} \\ np3Q = & \text{formal : event,} \\ & \text{const : Qc3} \end{array} \qquad \begin{array}{lll} s1Q = & \text{formal : event,} \\ & \text{const : Qc2} \\ s2Q = & \text{formal : event,} \\ & \text{const : Qc4} \end{array}$$

6.2 A new axiom

We handle syntactic polymorphism by multiple listings, but we do want to handle polymorphism in semantics by way of coercion. Hence we do not change our notion of derivability, but of unifiability. In a sequence of syntactic types $A/B, C \vdash A$, C can derive B iff they are of the same category, and if the features of type C either unify with, or license coercion to the features of type B. The first case is standard, the second case is more complicated. A set of features licenses coercion to another type if any of its qualia roles can form the bridge between C and B. The rule we propose in figure 10 subsumes both cases.

$$[Ax]\frac{}{\langle\ \text{X(F}_1,\ldots,\text{F}_n\text{, qualia(formal : t}'\vee\ \ \text{constitutive : t}'\vee\ \ \text{agentive : t}'\vee\ \ \text{telic : t}'\text{) : Sem}\ \rangle \vdash \langle\ \text{X(F}_1,\ldots,\text{F}_n\text{, qualia(formal : t)) : t}'\text{(Sem)}\ \rangle}$$

$$\text{iff t}' \leq \text{t}$$

Figure 10: Extended axiom

In this rule, not only the formal role of the antecedent type can satisfy the requirements that are made by the functor – a case which is included in the identity rule –, but so can all the other qualia roles, as long as they are subsumed by the formal role of the succedent type. For this we need a sortal hierarchy that indicates the position a sort could have in the hierarchy. As discussed in subsection 3.3 this hierarchy is described by partial ordering.

Another important part of this approach lies in the semantic representation. As has been demonstrated in section 5.2 it is possible by way of *feature jumping* to transfer combinatorial information to the logical representation, and in that manner reflect the importance of meaning in combinatorics. Here the one qualia role t′ that satisfies the requirements of the axiom is applied to the lambda term 'Sem' of the succedent category.

Note that our new axiom only describes the situation where the succedent type coerces the antecedent type, i.e. functor-argument coercion. The reverse is ofcourse conceivable, both for the coercion and for the axiom, but we will not go into that matter now.

7 A Sample Derivation

If we look at a fragment of the derivation of 'began the book', in figure 11, we see that one possible reading of this sentence is *began(write_event(the_book))*. This reading is derived by selecting the agentive role from the qualia structure of 'the_book'. The agentive role is a *write_event*, which is a subsort of the required *event*. This features jumps to the semantic level, and applies to the semantics of the argument.

$$\dfrac{\dfrac{}{\langle NP_3 : \text{the\_book}\rangle \vdash \langle NP_2 : Sem_1\rangle}[\text{Ax}] \quad \dfrac{\vdots}{\langle NP_1\backslash S_1 : \text{began}(\text{Sem}_1)\rangle \vdash \langle NP\backslash S) : Sem_2\rangle}}{\langle (NP_1\backslash S)/NP_2 : \text{began}\rangle, \langle NP_3 : \text{the\_book}\rangle \vdash \langle NP\backslash S : Sem_2\rangle}[\text{L}/]$$

$$\begin{aligned}
\text{NP}_1 &= \text{NP(qualia (} \quad \text{formal : animate))}\\
\text{NP}_2 &= \text{NP(qualia (} \quad \text{formal : event))}\\
\text{NP}_3 &= \text{NP(qualia (} \quad \text{formal : phys_obj}\\
&\qquad\qquad\qquad\quad\ \text{constitutive : [pages, cover,...]}\\
&\qquad\qquad\qquad\quad\ \text{agentive : write_event}\\
&\qquad\qquad\qquad\quad\ \text{telic : read_event))}\\
\text{S}_1 &= \text{S(qualia (formal : event))}
\end{aligned}$$

$$\begin{aligned}
Sem_1 &= \text{write_event(the_book)}\\
Sem_2 &= \text{began(write_event(the_book))}
\end{aligned}$$

Figure 11: Fragment of a derivation of 'began the book'

An alternative reading can be obtained if we select the telic instead of the agentive role; this too is a subsort of sort *event*. In principle there are many more actions that cause the book's existence (edit, typeset, proofread, print, distribute, etc.), also there are many more actions that can typically be done with a book (study, copy, read aloud, etc.) and all these may occur in the qualia structure of 'book'. All these will yield fine and plausible readings. It depends of the context which reading is to be prefered. But that decision is beyond the scope of this paper.

8 Conclusions

In this paper we have shown the impact of handling coercion phenomena within CG. It may be obvious from our presentation that CG has to deal with these natural language facts. Our proposal is to handle these within the framework of sign-based Lambek calculus. For CG this means a step towards a more natural approach to lexical semantics. Also, it means a larger coverage of linguistic data: thanks to the less restricted view on compositionality as well as derivability, a more accurate judgement can be given on whether a sentence is correct, and what its semantic representation might be.

CG does not need to concern itself with the question where the knowledge in a qualia structure comes from. It can simply take a given qualia structure for granted. GL itself however is concerned with this question and proposals have been made to derive qualia structures by way of Machine Learning [16] [19].

Acknowledgements

We would like to thank (in alphabetical order) Michael Moortgat, Glyn Morril, James Pustejovsky, Marc Verhagen, Kees Vermeulen, Koen Versmissen and our audience at the Amsterdam Colloquium for useful comments and discussion.

References

[1] J. Apresjan. Regular polysemy. *Linguistics*, (142), 1973.

[2] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17:22–29, 1985.

[3] M. Evens. *Relational Models of the Lexicon.* Cambridge University Press, 1988.

[4] L. Gamut. *Logic, Language and Meaning. Volumes I and II.* University of Chicago Press, Chicago, 1991.

[5] C. Gunter. *Semantics of Programming Languages.* MIT Press, 1992.

[6] J. Katz and J. A. Fodor. The structure of a semantic theory. *Language*, (39):170–210, 1963.

[7] J. Lambek. On the calculus of syntactic types. In R. Jakobson, editor, *Structure of language and its mathematical aspects, Proceedings of the Symposia in Applied Mathematics XII*, pages 166–178. American Mathematical Society, 1961.

[8] H. Leiss. Polymorphic constructs in natural and programming languages. In J. van Eyck, editor, *JELIA90 Logic in AI. Lecture Notes in Artificial Intelligence 478.* Springer Verlag, Berlin, 1990.

[9] M. Moortgat. Labelled deductive systems for categorial theorem proving. Technical Report OTS-WP-CL-92-003, Onderzoeksinstituut voor Taal en Spraak, Utrecht, 1992.

[10] G. Morrill. Type-logical grammar. Technical Report OTS-WP-CL-92-002, Onderzoeksinstituut voor Taal en Spraak, Utrecht, 1992.

[11] G. Morrill and B. Carpenter. Compositionality, implicational logics and theories of grammar. *Linguistics and Philosophy*, 13, 1990.

[12] G. Nunberg. The non-uniqueness of semantic solutions: Polysemy. *Linguistics and Philosophy*, 3, 1979.

[13] J. Pustejovsky. The generative lexicon. *Computational Linguistics*, 17(4), 1991.

[14] J. Pustejovsky. Linguistic constraints on type coercion. In P. Saint-Dizier and E. Viegas, editors, *Computational Lexical Semantics.* Cambridge University Press, 1994.

[15] J. Pustejovsky. Semantic typing and degrees of polymorphism. In Martin-Vide, editor, *Current Issues in Mathematical linguistics.* Elsevier, 1994.

[16] J. Pustejovsky, S. Bergler, and P. Anick. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2), 1993.

[17] S. Shieber. *Constraint-Based Grammar Formalisms. Parsing and Type Inference for Natural and Computer Languages.* MIT Press, Cambridge, MA, 1992.

[18] J. F. Sowa. Logical structures in the lexicon. In J. Pustejovsky and S. Bergler, editors, *Lexical Semantics and Knowledge Representation. Lecture Notes in Artificial Intelligence 627*, pages 577–613, Berlin, 1992. Springer Verlag.

[19] T. S. Waterman. Structural methods for lexical/semantic patterns. In B. Boguraev and J. Pustejovsky, editors, *Proceedings SIGLEX Workshop: Aquisition of Lexical Knowledge from Text*, pages 128–142. ACL, Ohio State University, 1993.

[20] U. Weinreich. Webster's third: A critique of its semantics. *International Journal of American Linguistics*, 30:405–409, 1964.