Notice

This paper is the author's draft and has now been published officially as:

Chang Nancy, De Beule Joachim, Micelli Vanessa (2012). Computational Construction Grammar: Comparing ECG and FCG. In Luc Steels (Ed.), *Computational Issues in Fluid Construction Grammar*, 259–288. Berlin: Springer.

BibTeX:

```
@incollection{chang2012computational,
   Author = {Chang, Nancy and De Beule, Joachim and Micelli, Vanessa},
   Title = {Computational Construction Grammar: Comparing ECG and FCG},
   Pages = {259--288},
   Editor = {Steels, Luc},
   Booktitle = {Computational Issues in {Fluid Construction Grammar}},
   Publisher = {Springer},
   Series = {Lecture Notes in Computer Science},
   Volume = {7249},
   Address = {Berlin},
   Year = {2012}}
```

Computational Construction Grammar: Comparing ECG and FCG

Nancy Chang[1], Joachim De Beule[2] and Vanessa Micelli[1]

[1] Sony Computer Science Laboratory Paris, France
[2] Artificial Intelligence Laboratory, Vrije Universiteit Brussel, Belgium

Abstract. This chapter compares two computational frameworks developed over the last decade to support investigations into the emergence and use of language, Fluid Construction Grammar (FCG) and Embodied Construction Grammar (ECG). Both of these representational formalisms are rooted in the construction grammar tradition, sharing basic assumptions about the nature of linguistic units and the crucial role played by contextual factors. Nonetheless, they have arisen from different perspectives and with different goals: FCG was designed to support computational language game experiments that address the evolution of communication in populations of robotic agents, while ECG was designed to support cognitive modeling of human language acquisition and use. We investigate how these differing emphases motivated different design choices in the two formalisms and illustrate the linguistic and computational consequences of these choices through a concrete case study. Results of this comparison sharpen issues relevant to computational construction grammar in general and may hold lessons for broader computational investigations into linguistic phenomena.

1 Introduction

This chapter compares two computational formalisms developed over the last decade: Fluid Construction Grammar (FCG) and Embodied Construction Grammar (ECG). Both formalisms draw broad inspiration from construction grammar and cognitive linguistics, sharing basic assumptions about the nature of linguistic units and the crucial role played by meaning in context. But unlike most other work in this area, both FCG and ECG aspire to provide computational implementations of all proposed linguistic structures and processes. This formalization (or operationalization) requirement reflects an emphasis not just on how linguistic knowledge is *represented* but also on how it is *used*: conceptual and linguistic structures should be seamlessly integrated with processes of language learning and use.

Each formalism is also the centerpiece of a broader scientific framework tackling similar issues, albeit from different perspectives and with different motivations. These may best be captured by examining the core goals and questions driving these respective investigations:

- FCG supports language game experiments that explore answers to the question: How can communication emerge in populations of embodied agents? Its roots are in artificial intelligence, and historically it has been oriented toward artificial languages evolved and acquired by robotic agents. More recently, however, it has begun to address phenomena inspired by natural languages, as exemplified by the case studies in this and other volumes.
- ECG supports cognitive modeling of human language learning and use, within a framework that asks: What is the neural, embodied basis of thought and language? Its roots are in cognitive science and cognitive linguistics, though it is also motivated by psychological, biological and especially developmental considerations.

While these endeavors are theoretically compatible, they have differing orientations and emphases that have shaped their respective formalizations. Some of the resulting differences may be described as superficial notational variations, but others reflect more substantial divergences.

The two formalisms are thus ideal candidates for comparison. In this chapter, we aim to identify the core differences between FCG and ECG, as well as the open research issues suggested by these differences. We center the discussion around a concrete comparison of how the two formalisms realize the key ideas of construction grammar, using a case study that allows a detailed comparison of several lexical and phrasal constructions (Sections 3-5), as well as the processing models (Section 6) associated with each formalism. Section 7 considers how the results of our comparison sharpen issues relevant to computational construction grammar in general, and what lessons they may hold for broader computational investigations into linguistic phenomena.

Shared theoretical and methodological commitments Before turning to our case study, we briefly summarize some basic theoretical commitments shared by the two research frameworks under consideration. Broadly speaking, both are identified with constructional, cognitive and usage-based approaches to language. Constructions (mappings between form and meaning), are taken to be the basic units of language [7**?**], and meanings correspond to particular ways of conceptualizing or construing a situation [8**?**]. Language is also assumed to be inherently embodied, grounded and communicative: language users have sensorimotor capacities that shape their conceptual categories, and they are grounded in particular environments with specific communicative goals.

Most relevantly, both formalisms were designed to support working systems that actually instantiate structures and processes that are elsewhere typically described only discursively. This commitment to supporting language *use* means that it is not sufficient merely to represent linguistic knowledge in formal notation; rather, the processes that interact with that knowledge must also be specified, and considerations related to processing (e.g., search space, storage, efficiency) must guide representational choices at the level of both individual constructions and the formal notation itself.

Linguistic representations in both frameworks are also assumed to interact closely with structures in other domains, including in particular embodied, situational and world knowledge. The two frameworks differ in the details of how such interactions are modeled, and even in how terms like *embodiment* are used.[3] For the pursposes of this chapter, however, we focus on the specifically linguistic knowledge expressed by the two grammatical formalisms and their mechanisms of use. Both frameworks take these to be conceptually distinguishable from the details of sensorimotor representations; world (ontological) knowledge; general mechanisms of inference and belief update; and specific mechanisms of contextual grounding and reference resolution. We will also refrain from addressing in detail how language learning is modeled in each framework, though we will highlight some connections to these topics where relevant.

This chapter is not intended as a comprehensive description of either formalism; this volume and [15] provide a detailed introduction to FCG, and overviews of ECG and its associated research framework can be found elsewhere [3, 6**?**]. But to ground our discussion, in the sections to follow we introduce the notational basics of each, sufficient for discussing the noun phrase *the mouse* (also addressed in [16]). Despite the relative simplicity of this example, the side-by-side comparison it affords helps reveal some fundamental design issues and differences.

2 Informal Constructional Analysis of *the mouse*

A traditional analysis of the phrase *the mouse* might identify a determiner (*the*), a noun (*mouse*), and a noun phrase (NP) combining the determiner and noun in that order. For a construction-based approach, it is crucial to consider the utterance's meaning as well: a speaker uttering "the mouse" is engaging in an act of reference, picking out an individual mouse uniquely identifiable to the hearer in the discourse context. A straightforward constructional analysis might have the structure shown in Figure 1, with three constructions:

- The: The word form *the* constrains the referent to be uniquely identifiable to the hearer in context; other determiners may restrict features like number (*some mice*) or proximity (*these mice*).
- Mouse: The word form *mouse* specifies that the referent's ontological category is a mouse. It might also specify that the referent refers to a single thing (in contrast to the greater quantity specified by *mice*), or that it is animate (or not, in the case of a computer mouse). Other nouns may constrain additional qualities of the referent (e.g., semantic role, gender, countability).
- DeterminedNP: This construction has two constituents, corresponding to the two constructions above. It imposes a word order constraint (*the* must precede *mouse*), and its meaning is a referent in context—in fact the same referent constrained by the two constituents. Here, the relevant constraints

[3] Broadly speaking, embodiment in FCG emphasizes the constraints of using physically embodied agents, while embodiment in ECG emphasizes the constraints of the human sensorimotor system.

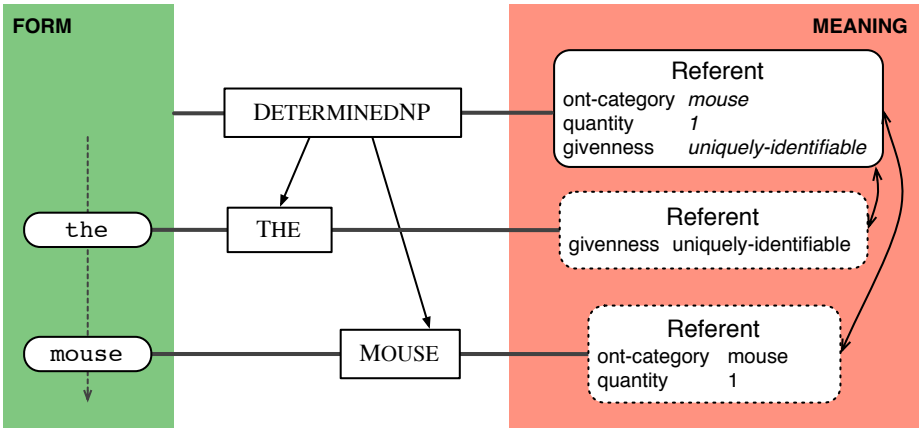

Fig. 1. A graphical depiction of one analysis of example phrase. Constructions (in the center) link the domains of form (left) and meaning (right). Each of the constructions shown here (the DeterminedNP construction and its two constituents, the The and Mouse constructions) contributes to and constrains the particular referent specified by the phrase.

do not conflict; in general, such compatibility or *agreement* in features must be met between determiners and nouns (hence **a mice*, **these mouse*).

The constructions in the middle of the Figure 1 reflect the phrase's constituent structure, mirroring that of a traditional syntactic parse tree (based on a phrase structure analysis). (See Section 5.3 for an alternative dependency analysis.) However, since these are not just syntactic symbols but constructions, each construction also has a link (shown by horizontal bars) to form (on the left) and meaning (on the right). The form domain contains the relevant word forms, where the dotted arrow indicates the time line (and therefore word order).

The meaning domain contains several structures labeled Referent, each listing features constrained to particular values (where ont-category is an abbreviation for ontological category). Essentially, this structure summarizes any information that is important for determining the actual referent of an expression in the current context. (Determination in both formalisms is further discussed in section 3.3.) The double-headed arrows between the Referents indicate that their values are shared (with values that originate non-locally, i.e. through such a binding, shown in italics). The dashed border of the two Referent structures contributed by the lexical constructions indicates a slightly different relationship than that between the DeterminedNP construction and its Referent; we return to this point below.

As should be apparent, even a noun phrase as simple as *the mouse* involves many representational choices, with reasonable alternatives varying in both the complexity of the structures defined and the generality of the phenomena they account for. Our goal here is not to argue for the particular analysis adopted here as the best or most general one possible; rather, we focus on the basic representational toolkit involved for expressing a variety of concepts and relations and compare those available in the ECG and FCG formalisms.

3 Formalizing Lexical Constructions

Diving now into the formal constructional analysis, we consider in this section how the lexical constructions for our example are defined in each of the two formalisms. We begin with the *mouse* construction shown in Figure 2. Both structures capture a relatively straightforward pairing of form (the orthographic string "mouse") and meaning (the mouse ontological category associated with a referent, whose quantity is additionally specified as 1). They also include grammatical information (e.g., that *mouse* is a singular noun), though they differ in precisely how this information is expressed.[4]

```
;; "mouse" in FCG (template-based)
(def-lex-cxn Mouse-Cxn
  (def-lex-cat Mouse-Cxn
    :sem-cat ((schema ?ref [ReferentDescriptor])
              (quantity ?ref 1))
    :syn-cat  ((schema ?w [WordDescriptor])
              (type Noun)
              (number singular)))
  (def-lex-skeleton Mouse-Cxn
    :meaning (== (ont-category ?ref [mouse]))
    :form (== (orth ?w "mouse"))))
```

```
// "mouse" in ECG
construction Mouse-Cxn
    subcase of Noun
    constructional
        self.number ← singular
    form : Word
        self.f.orth ← "mouse"
    meaning
        evokes ReferentDescriptor as ref
        ref.ont-category ← @mouse
        ref.quantity ← 1
```

Fig. 2. Lexical constructions for *mouse* in FCG (left) and ECG (right).

A few differences in basic format are apparent even at first glance. Roughly speaking, the format of FCG reflects the influence of the Lisp programming language: internal structure is indicated with parenthesized lists employing prefix-list notation, and variable names are marked with a leading question mark. In contrast, the format of ECG reflects the influence of constraint programming

[4] Both distinguish a language-specific *number* categorization (in English, nouns can be either singular or plural) from a more general concept of number, which for clarity will be called *quantity* in the notation.

languages and inheritance-based ontologies: special keywords (in boldface) are used to indicate internal structure and express inheritance relations and other constraints, and dotted slot chains are used to refer to non-local structures.

The sections below take a closer look at the two constructions in Figure 2. To ease comparison, we focus on how each captures the informal linguistic analysis described in the previous section, deferring until Section 6 the details of how constructions are used during language processing.

3.1 Nominal Constructions in FCG

The FCG definition for *mouse* shown on the left in Figure 2 uses the `def-lex-cxn` template for defining lexical units (described in [15]). This notation organizes the various elements of the informal analysis in two parts. First, the `def-lex-cat` clause specifies the linguistic categories (both semantic and syntactic) associated with *mouse*: a variable `?ref` is associated with the schema `ReferentDescriptor` and the quantity 1, and a variable `?w` is associated with the schema `Word`, the type `Noun` and the number `singular`. Second, the `def-lex-skeleton` clause specifies the ontological category (where square brackets on `[mouse]` denote reference to an ontology item) and orthographic string.

In fact, this template is an abbreviation for a more elaborate "native" FCG definition; we show the expanded version in Figure 3. Both styles of FCG definition contain the same information, but the template-based construction omits details shared with other lexical constructions, allowing a more concise definition. As should become apparent, the template-based version is closer to the level of abstraction used in the corresponding ECG definition, but we describe the expanded version here to shed some light on how these structures are used.

The full lexical definition consists of two main sections (or *poles*) separated by a double-headed arrow (`<-`>), corresponding to the meaning (or semantic) and form (or syntactic) domains. Each pole includes two *units*, one named `?top-unit` and one named `?mouse-unit` (where the leading question mark indicates that these are variable names); this latter unit is a *J-unit* (as indicated by the operator `J`). These units specify the constraints and categorizations relevant to each domain, but they differ in the kinds of information they typically contain. J-units generally contain specifically linguistic information, typically expressed using semantic and syntactic categories (in the `sem-cat` and `syn-cat` lists, respectively). Other ("regular") units, like `?top-unit` here) tend to be based on perceptual or cognitive categorizations (here, the ontological category and the perceived word string).[5]

3.2 Nominal Constructions in ECG

We now turn to the right side of Figure 2, which shows a simple ECG construction for *mouse*. The high-level structure of the construction includes three

[5] Regular units and J-units also behave differently during language processing, where the J-operator and other notations (such as the `TAG` and `footprints`) notations play a special role. Language processing will be discussed in more detail in Section 6.1.

```
(def-cxn Mouse-Cxn
 ((?top-unit (TAG ?meaning
                  (meaning (== (ont-category ?ref [mouse]))))
             (footprints (==0 Mouse-Cxn)))
  ((J ?mouse-unit ?top-unit)
             ?meaning
             (sem-cat ((schema ?ref [ReferentDescriptor])
                       (quantity ?ref 1)))
             (footprints (Mouse-Cxn))))
 <-->
 ((?top-unit (TAG ?form
                  (form (== (orth ?w "mouse"))))
             (footprints (==0 Mouse-Cxn)))
  ((J ?mouse-unit ?top-unit)
             ?form
             (syn-cat ((schema ?wd [WordDescriptor])
                       (type Noun)
                       (number Singular)))
             (footprints (Mouse-Cxn))))))
```

Fig. 3. Lexical construction for *mouse* in FCG, expanded form.

blocks, where keywords (in boldface) indicate special terms and structures. The constructional block contains information relevant to the construction as a whole, while the form and meaning blocks (or *poles*) contain information relevant to each domain. These poles are themselves structured, and can be referenced and constrained within the construction. The term self allows reference to the construction being defined, and self.f and self.m refer to the construction's form and meaning poles, respectively.

The MOUSE construction is defined as a *subcase*, i.e. a more specific instance, of the NOUN construction. In fact, ECG constructions are all defined in a multiple inheritance lattice, and a separate lattice defines represent schematic form and meaning categories, or *schemas*. (Both inheritance and schemas will be discussed further in Section 4.) Accessible structures can be constrained to instantiate particular schema categories. Each block contains various constraints that apply to the relevant domain, drawn from a fixed set of possibilities. We highlight a few of the notations that express these constraints:

- Category constraints are indicated with a colon (e.g., the form pole must be a Word), and role-filler constraints of the form x ⟵ y indicate that role (or feature) x is filled by the (atomic) value y (e.g., the form pole is associated with the orthographic string shown).
- The **evokes** ReferentDescriptor **as** ref declaration indicates that there is an instance of category ReferentDescriptor present, accessible using its local name

ref. Subsequent constraints specify that its feature ont-category be filled by @mouse and its quantity set to 1. (Like the square brackets in the FCG definition, the @ symbol indicates reference to an external conceptual ontology.)

In short, the construction indicates that the MOUSE-CXN is a kind of NOUN; asserts constraints on the constructional (or grammatical) number feature and the particular orthographic form; and evokes a ReferentDescriptor of a specified ontological category and quantity.

For comparison, it may be useful to see how the ECG construction definition is expanded during processing. Figure 4 shows the corresponding feature structure representation. Note that the feature structure for Mouse-Cxn also contains two subsidiary feature structures for the Word and ReferentDescriptor categories mentioned in the definition. As discussed in Section 4.2 and illustrated in Figure 6 below, these additional schematic structures are also defined as part of the ECG formalism.

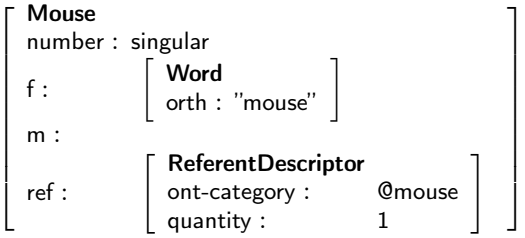

Fig. 4. Feature structure corresponding to the ECG *mouse* construction.

3.3 Determiners in ECG and FCG

The basic lexical constructions just defined are easily modified for the determiner *the*. In accord with earlier accounts (e.g. [**?**]), we assume that determiners provide cues that help a hearer identify a referent in the discourse context. Thus, like the *mouse* construction, they constrain a referent, but instead of specifying its ontological category, they typically constrain features like number, gender and proximity. In the case of *the*, the referent is asserted to be uniquely identifiable.

The constructions in Figure 5 are structurally similar to the *mouse* constructions defined above. Each specifies that *the* is a Determiner (as the syntactic category `(type Determiner)` in FCG and the **subcase of** Determiner constraint in ECG); each specifies the relevant orthographic string; and each specifies a value for the referent's givenness feature (in FCG using the predicate `(givenness ?ref uniquely-identifiable)`, and in ECG with the constraint ref.givenness ⟵ uniquely-identifiable).

The two formalisms differ slightly in how they handle the feature of number. The FCG definition adds a number category to its list of syntactic categories,

```
;; "the" in FCG (template-based)
(def-lex-cxn The-Cxn
  (def-lex-cat The-Cxn
    :sem-cat ((schema ?ref [ReferentDescriptor]))
    :syn-cat  ((schema ?w [WordDescriptor])
               (type Determiner)
               (number ?number)))
(def-lex-skeleton The-Cxn
    :meaning (== (givenness ?ref uniquely-identifiable))
    :form (== (orth ?w "the"))))
```

```
// "the" in ECG
construction The-Cxn
   subcase of Determiner
   form : Word
      self.f.orth ← "the"
   meaning
      evokes ReferentDescriptor as ref
      ref.givenness ← uniquely-identifiable
```

Fig. 5. Lexical constructions for *'the'* in FCG (left) and ECG (right).

whose value remains underspecified (as indicated by the variable `?number`). The ECG definition does not mention number explicity. Note that ECG definitions for determiners like the plural *some* or singular *a*) would include a constructional block with a constraint on the number feature, thus more closely resembling the *mouse* construction.

4 A First Comparison

The parallel definitions of lexical constructions in FCG and ECG given in Section 3 are based on the same linguistic analysis and designed to maximize similarity across the two formalisms. It is not entirely surprising, then, that they have much in common: each represents the basic forms and meanings involved, as well as additional grammatical information associated with reference, as expressed by common nouns (like *mouse*) and determiners (like *the*).

But these examples also exhibit some striking differences. Perhaps the most important distinction between the formalisms is the treatment of categories and inheritance. ECG structures are all defined within *inheritance lattices* specifying constructional and other relationships. Many ECG notations thus allow reference to other existing structures, for example to inherit features and values, or to assert values or bindings on connected structures. FCG constructions, on the other hand, rely on category lists associated with each domains; each construction is thus relatively stand-alone, and defined independently of other structures that may contain similar information. In the sections below we discuss several representational consequences of this fundamental difference.

4.1 Inheritance and Categorization

Categories play a prominent role in most linguistic theories: they capture generalizations about shared structure and behavior across different linguistic units.

Part of speech categories, semantic (or thematic) roles, lexical subcategorization types, speech act types, and phonological categories are all well-established ways of carving up various linguistic domains into groups exhibiting similar properties. Both ECG and FCG allow such categories to be expressed, but they differ in the approaches taken, as well as the degree to which the relationships among categories is made explicit.

Inheritance hierarchies in ECG. The ECG approach to categories is based on inheritance networks, where shared properties are expressed at the highest level of generalization possible and inherited by subsidiary categories and instances. That is, ECG constructions are defined (using the **subcase of** relation) within a multiple inheritance hierarchy (or lattice); structures and constraints defined in a *base* (or *parent*) construction are inherited by and accessible to their subcases, and thus need not be explicitly specified. The subcase can impose additional constraints, or refine existing ones.

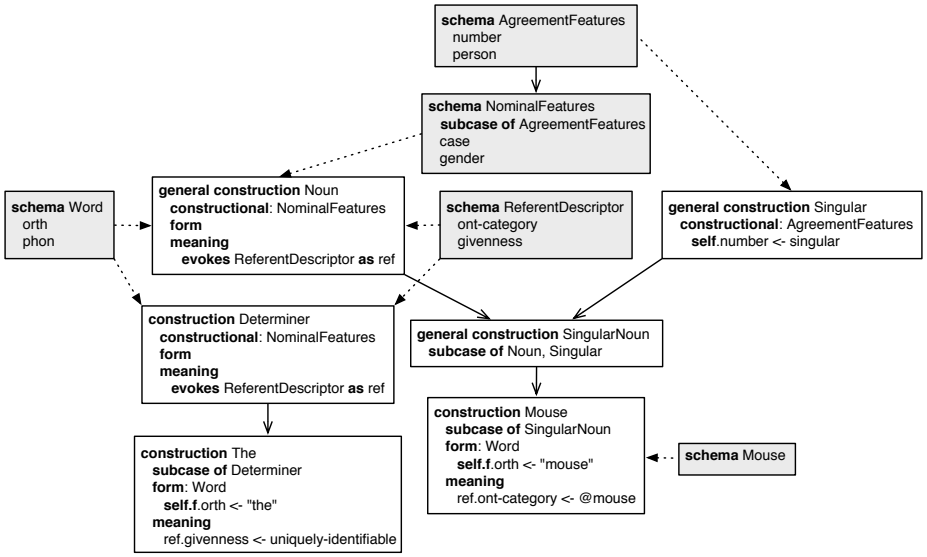

Fig. 6. A portion of the ECG construction lattice for the lexical items in the example, showing both constructions (white boxes) and schemas (shaded boxes). (Solid arrows indicate subcase (or inheritance) relations, while dotted lines indicate other links through the constructional, form or meaning domains.)

A fragment of the constructional lattice relevant to the lexical constructions in our example is shown in Figure 6, where both the Mouse and The constructions have been redefined to exploit inheritance. Focusing on the white construction boxes, we can see that this version of the Mouse construction is

defined as a subcase of the SINGULARNOUN construction, which in turn is a subcase of both NOUN and SINGULAR. These abstract ancestral constructions—marked **general** to indicate their lack of concrete form constraints—contain some constraints shared across noun constructions (the **evokes** statement and number ⟵ singular constraint), leaving only the most specific constraints for the MOUSE construction.

These examples illustrate how certain linguistic generalizations can be concisely captured through inheritance. Though not shown, the construction for the irregular plural *mice* inherits from the PLURAL and PLURALNOUN constructions, which are analogous to SINGULAR and SINGULARNOUN, respectively. Similarly, determiners that specify number (such as *these* or *a*) are defined as subcases of both DETERMINER and the appropriate number-specifying construction.[6]

Categories in FCG. Categories are a fundamental notion in FCG, expressed as predicates in the `sem-cat` and `syn-cat` lists. Constructions that have such predicates in common implicitly form a category: hence both *mouse* and *the* are associated with the syntactic category `schema`, which is further specified to be a `WordDescriptor`, and they also both include the syntactic category `number`.

Inheritance networks like those of ECG have not yet been much explored in FCG: there is no explicit notion of inheritance for constructions, meaning or form components, or semantic and syntactic categories. Recent developments, however—such as the use of templates [15] and distinctive feature matrices [17]—can be seen as moving in this direction.[7] Templates, for example, provide a means of capturing similarities across constructions, allowing a more concise, uniform declaration of constructions (as illustrated by the alternate definitions for *mouse* above). Note, however, that templates currently serve mainly as an abbreviation: they do not specify inheritance *relationships*. That is, there is no mechanism for allowing one construction to refer to or inherit from another, and more general lexical categories like `Noun` are not themselves defined as structures that can inherit features.

Of course, the use of templates in FCG is relatively recent and their precise form is still under development. Thus it may be possible to extend the template approach (as proposed in Section 5) to exploit the benefits of inheritance and type hierarchies. These benefits become especially important as grammars grow larger: keeping track of the various dependencies between constructions for any non-trivial language phenomenon is a tedious undertaking. Adopting approaches based on inheritance would enable more concise grammars that reduce errors.

[6] Inheritance is only one way of capturing complex category structure of the kind described by [**?**], and the particular multiple inheritance lattices used by ECG are intended only as an approximation. While current versions of ECG allow overriding of inherited constraints and the incorporation of some probabilistic information, further refinements would be needed to handle categories including scalar and continuous quantitative values, prototype structures and other aspects of human categorization.

[7] See also [4], which describes FCGlight (a core subset of FCG that uses of lattices of constructions); and [1].

4.2 Form and meaning representations

The lexical examples we have seen also illustrate different approaches to representing the domains of form and meaning. Organizationally, FCG distinguishes specifically linguistic categorizations (as listed in `sem-cat` and `syn-cat`) from the concrete forms and meanings taken to be based on perceptual or cognitive categorizations (associated with the `meaning` and `form` parameters. ECG constructions do not explicitly represent this difference in the notation itself (except for the use of `@` to denote ontological categories).

A more important difference lies in how these categories are represented. As noted before, all categorizations (linguistically specific or not) in FCG are expressed in predicate-argument format, and are independently defined as part of each relevant consruction. The particular style of semantic representation can vary; though the examples shown in this section have a declarative flavor (following the informal analysis presented in Section 2), other studies show how it is also possible to adopt procedural semantics [14] or frame semantics [**?**].

In ECG, both forms and meanings are represented using a special-purpose *schema* formalism, similar to that used for constructions and also defined within an inheritance lattice (see Figure 6 for some examples, shown in shaded boxes). ECG schemas resemble depictions of *semantic frames* [**?**] and *image schemas* in the literature, and are similarly used to bring together a set of associated and interdefined *roles* or *features* comprising a complex concept. The roles defined in a schema can be referred to and constrained by other schemas and constructions. Hence, both lexical constructions assert form constraints on the orth role of the Word schema, as well as meaning constraints on the roles of the ReferentDescriptor.

As with constructions, we see that ECG emphasizes the interdefined nature of constructions and their associated forms and meanings. Separately defined schemas capture various linguistic generalizations and expectations, allowing brevity in definitions and enforcing some consistency across constructions. FCG constructions, meanwhile, each independently define their relevant predicates, which are therefore less constrained. This tradeoff—between explicit representation of generalizations on the one hand, and freedom of expression on the other—will manifest itself in several other ways to be discussed.

4.3 Constructional Features and Grammatical Categories

The two formalisms differ, finally, in how certain kinds of grammatical information are treated. Specifically, while all categories and constraints in FCG must be in either the meaning or form pole, ECG allows some information to be expressed in the constructional lattice:

- Constructional inheritance: Lexical and grammatical categories (like noun and verb) can themselves be represented as constructions and associated with specific roles and values. Thus, the Mouse construction can be defined as a subcase of the SingularNoun construction, inheriting relevant properties it may share with other singular nouns.

– Constructional features: The constructional domain itself can be defined as having particular features, often inherited from ancestral types. In Figure 6, AgreementFeatures and NominalFeatures are schemas in the constructional domain that list various grammatical features. These are not strictly about either the form or meaning domain; rather, they are associated with the constructional connection between the two.

In both cases above, the equivalent information can be expressed in FCG but must be explicitly included in every constructional definition (unless the template system could be extended to do this, though this would be a non-trivial modification).

In each formalism, it remains largely at the discretion of the grammar writer how to decide precisely which features ought to be defined and what domain they belong in.

5 Constituent Structure and Agreement

We now turn our attention to the DeterminedNP construction. This construction combines a determiner and a noun into a larger phrasal unit. Combining smaller units into larger chunks and phrases and thus exhibiting hierarchical *constituent structure* is a defining feature of grammatical constructions. Like lexical constructions, such constructions can impose constraints in both the form and meaning domains, such as word order (form) or role-filler bindings (meaning). They may also enforce compatibilities, or *agreement*, across constituents. Both ECG and FCG have ways of introducing constituents, specifying relational constraints and enforcing agreement.

5.1 Determined NPs in ECG

The DeterminedNP construction in Figure 7 shows how determined noun phrases with constituent structure might be defined in ECG. The intuition behind the analysis is that such phrases draw on both determiners and nouns to provide crucial information for constraining an act of reference, resulting in a single larger unit (as in our informal analysis). Like lexical constructions, phrasal constructions have a form and meaning pole; they also, however, have a **constructional** block within which constructional **constituents** as well as additional **constraints**—for example, to enforce agreement—are specified.

Here, the two constituents have local variable names det and nom, and they are typed respectively as Determiner and Noun. These constituent names allow simple access to their respective form and meaning poles. In the form block, their form poles (det.f and nom.f) are specified as coming in a particular order (expressed using the before relation). The meaning of the overall expression is itself constrained to be a ReferentDescriptor—which, recall, is also the type of the ref argument evoked in the meanings of each of the two constituents. That is, all three (the composite structure and each of the two constituents) have

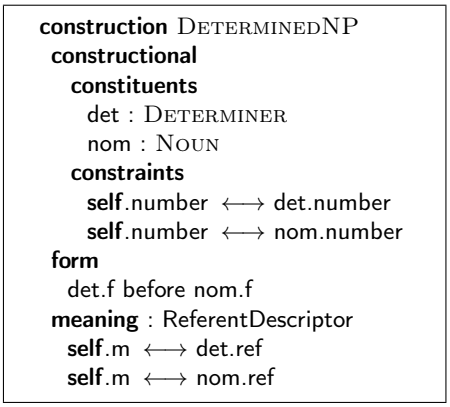

Fig. 7. A complex DeterminedNP construction.

accessible ReferentDescriptors. Thus, the last two constraints simply identify the evoked referents of the constituents with the meaning of the overall phrasal construction. Note that these constraints enforce all the roles defined within the ReferentDescriptor structures to have the same value, including the ont-category, givenness, and quantity roles. This can be seen as a kind of semantic agreement: the referent that all of these constructions describes is the same, and therefore the constraints that apply to it as also the same.

Turning to the constructional domain, we see another kind of agreement enforced by the constraints in the constructional block. These simultaneously encode agreement between the two constituents' number features (notated here as det.number and n.number) and ensure that this value is shared by the noun phrase (**self**.number). All of these require that these constructions are typed so that they have an accessible number feature (in our analysis, their constructional poles are all constrained to be of type NominalFeatures). This agreement might be seen as the more prototypical grammatical agreement, based on explicitly grammatical features like *number* that may not have any basis in the meaning domain alone.[8]

5.2 Determined NPs in FCG

An account of how determined noun phrases might be handled in FCG using phrasal construction templates is given in [15]. This section provides an alternative analysis that is as close as possible to the one given for ECG, while remaining

[8] Of course, it is not always possible to draw a neat boundary between the semantic and constructional domains, especially with respect to a linguistically oriented schema like ReferentDescriptor.

within the limits of what is currently possible in FCG. Besides enabling a more detailed comparison, this variation is also intended to add another perspective to the relatively recent development of FCG templates that may help shed light on the benefits and drawbacks of different approaches.

```
;; DeterminedNP (template-based definition)
(def-phrasal-cxn DeterminedNP
   :syn-cat ((type DeterminedNP) (number ?number))
   :sem-cat ((schema ?ref [ReferentDescriptor]))
   :constituents
   ((lex-cxn ?Determiner
     :syn-cat ((type Determiner) (number ?number))
     :sem-cat ((schema ?ref [ReferentDescriptor]))
     :meaning ((givenness ?ref ?givenness))
     :form ((orth ?det ?orth-det)))
    (lex-cxn ?Noun
     :syn-cat ((type Noun) (number ?number))
     :sem-cat ((schema ?ref [ReferentDescriptor]))
     :meaning ((ont-category ?ref ?ont-cat))
     :form ((orth ?N ?orth-N))))
   :form ((before ?det ?N)))

;; DeterminedNP (expanded definition)
(def-cxn DeterminedNP
   ((?top-unit (footprints (==0 DeterminedNP))
               (sem-subunits (?Determiner ?Noun)))
    (?Determiner
       (meaning (== (giveness ?ref ?giveness)))
       (sem-cat (==1 (schema ?ref [ReferentDescriptor]))))
    (?Noun
       (meaning (== (ont-category ?ref ?ont-cat)))
       (sem-cat (==1 (schema ?ref [ReferentDescriptor]))))
    ((J ?DeterminedNP ?top-unit (?Determiner ?Noun))
       (sem-cat ((schema ?ref [ReferentDescriptor])))))
    <-->
   ((?top-unit
       (TAG ?form (form (== (before ?det ?N))))
       (footprints (==0 DeterminedNP))
       (syn-subunits (?Determiner ?Noun)))
    (?Determiner
       (form (== (orth ?det ?orth-det)))
       (syn-cat (==1 (type Determiner)
                  (number ?number))))
    (?Noun
       (form (== (orth ?N ?orth-det)))
       (syn-cat (==1 (type Noun) (number ?number))))
    ((J ?DeterminedNP ?top-unit (?Determiner ?Noun))
       ?form
       (syn-cat ((type DeterminedNP) (number ?number))))))
```

Fig. 8. DeterminedNP construction in FCG, both template and expanded versions.

Concretely, we define a phrasal construction for determined noun phrases in Figure 8, where the first definition expands (using the appropriate template code for `def-phrasal-cxn`) to the second. Like the lexical constructions, the `DeterminedNP` construction has both meaning and form components, each including both regular units (`?top-unit`, `?determiner`, and `?noun`) and a J-unit. The basic idea of constituency is captured by the specification that the `?top-unit` lists the other two regular units in its `subunits`, corresponding respectively to a determiner and a noun (again, in both the meaning and form domains).

On the syntactic side, the construction furthermore specifies the constraint `(before ?det ?N)` on the word order of its constituents. On the semantic side it requires that the `ReferentDescriptor` schemas of both constituents are the same (through a variable equality). Agreement in number is also achieved through variable equalities.

5.3 Comparing Complex Constructions

The two approaches to representing complex constructions demonstrated in the preceding sections are both capable of expressing constituency, word order and agreement. They differ, however, in several key respects.

First, as elsewhere, the ECG formalism avails itself of type lattices for both constructions and schemas. Thus, various constraints require that relevant features are defined and accessible for a given structure (i.e., a slot chain like det.number implies that det is defined as having inherited a number role). This stands in sharp contrast with FCG, which does not require typing of this kind: previously unspecified features are added during processing if not already defined, and only if it is explicitly indicated that this should be the case.

The less type-constrained approach of FCG may be seen as a double-edged sword: while it leaves more freedom of choice, it also requires that the grammar writer maintain the soundness of his or her grammars and ensure that the relevant semantic and syntactic categories of constituent units are percolated properly to newly created units. For instance, although the `?DeterminedNP` in the DeterminedNP construction unit is specified as an instance of the `ReferentDescriptor` schema, neither its `givenness` or `ont-category` values are specified. These could be inferred from its constituents, and the template could perhaps be changed to do this automatically, but again, doing so would be far from trivial. In contrast, ECG makes some structural assumptions that allow certain constraints to be succinctly stated, though possibly at the cost of flexibility. Thus the identification of the various ReferentDescriptors allows all their roles to be bound with one constraint, both across constituents and with the overall resulting construction.

Second, the two formalisms allow somewhat different options with respect to how particular kinds of features are expressed. As noted earlier, grammatical features and categories are typically expressed in the constructional domain in ECG (though as demonstrated above, agreement can also be enforced just in the form or meaning domain). As with the lexical constructions, FCG tends to express such grammatical information by including it as a syntactic category.

This difference may not ultimately affect expressive power, but it does reflect different theoretical views of particular linguistic concepts.

6 Processing

Both frameworks under comparison are committed to the idea that grammatical formalisms should do more than just describe language: they should also support processes of language use. In this section we compare how processes of language use interact with the linguistic representations encoded by the two grammatical formalisms we have described.

6.1 Parsing and Production in FCG

The FCG formalism was designed to support processes of both parsing (mapping from form to meaning) and production (mapping from meaning to form). These processes have been described in detail elsewhere; we review them briefly here.

The internal structure of the FCG construction reflects a number of symmetries in how the different components are used during language processing, depending on whether parsing or production is performed. In particular:

- Each pole corresponds to the input to one process and to the output of the other: an utterance representation in the form pole is the input to parsing and the output of production, and vice versa for meaning representations in the meaning pole.
- The regular units and J-units together specify constraints and categorizations relevant to each domain, but they behave differently with respect to the *match* and *merge* operations at the core of language processing in FCG. Briefly, matching is used to test whether a transient structure fits (i.e., matches) a given construction; it thus acts as a filter on applicable constructions; merging then effects the construction's application and contributes additional information. Regular units in the input pole are matched, and thus are typically used to select which constructions to apply. J-units and regular units in the output pole are merged and thus provide additional constraints and categorizations.

In both processes, constructions operate on a *transient linguistic structure*. Before processing starts, the transient structure is initialized with the meaning that needs to be verbalized (production), or with the form that needs to be parsed (parsing); this structure is then gradually transformed to the desired output structure. We describe both processes for our example below.

Parsing. The aim of the parsing process is start from an empty meaning and gradually add content until a full meaning specification (and a complete parse) is achieved. The initial transient structure for parsing *the mouse* is as follows:

```
((top))
<-->
((top (form ((orth W1 "the") (orth W2 "mouse")
             (before W1 W2)))))
```

The form side of the linguistic structure specifies a single constituent named `top`, containing three predicates that together fully specifying the string "the mouse". These predicates include the constants `W1` and `W2` representing the actual words.

The application of the `mouse-cxn` to this structure is licensed through the *match* and *merge* operations mentioned above. The syntactic side of the construction is matched against the syntactic side of the transient structure (excluding J-units), resulting in a set of bindings for the variables in the construction. The features marked by the special `TAG` operator (e.g., the meaning and form features in the MOUSE-CXN of Figure 2) cause the tag-variable (e.g., `?meaning` and `?form`) to be bound to the matched set of feature values.

In the example, the MOUSE-CXN is triggered by the predicate `(== (orth ?word "mouse"))` in its top unit, where the includes operator `==` indicates that other components besides the specified meaning are also allowed. The construction therefore matches the initial structure with the following bindings: '[?top-unit/top, ?form/(form ((orth W1 "mouse"))),?w/W1].[9]

The merge operation then results in a new, modified transient linguistic structure that is the union of the matched structure with the additional constraints specified in the construction. Merging in FCG thus roughly corresponds to unification in ECG and other unification- or constraint-based formalisms. Given the match-bindings obtained earlier, both sides of the MOUSE-CXN also merge with the transient linguistic structure.

In parsing, the resulting structure is as below:

```
((top (subunits (mouse-unit)))
 (mouse-unit
      (meaning  ((ont-category ?ref-1 [mouse])))
      (sem-cat  ((schema ?ref-1 [ReferentDescriptor])
                 (quantity ?ref-1 1)))))
<-->
((top (form ((orth W1 "the") (before P1 W1 W2)))
      (subunits (mouse-unit)))
 (mouse-unit
      (form     ((orth W2 "mouse")))
      (syn-cat  ((schema W2 [WordDescriptor])
                 (type Noun)
                 (number singular)))))
```

Note that the transient structure now includes a new unit named `mouse-unit` on each side, corresponding to the constructional J-unit. This new unit includes

[9] If variable '$?x_1$' is bound to value X_1, and variable '$?x_2$' to X_2 etc., this is denoted as $[?x_1/X_1, ?x_2/X_2, ...]$.

the tagged form predicate matched by the `mouse-cxn`, and its meanings have additional semantic categories as specified by the `mouse-cxn`)

Production. Production in FCG is entirely analogous to parsing, but with the role of the poles reversed. The initial structure for producing our example is as follows:

```
((top (meaning
       ((ont-category Ref [mouse])
        (giveness Ref uniquely-identifiable)))))
<-->
((top))
```

Similar to the parsing situation, the semantic side of the MOUSE-CXN matches the semantic side of the initial transient linguistic structure above because the construction requires only that there is a unit, with variable name (`?top-unit`), which contains a `meaning` feature including the component `(ony-category ?ref [mouse])`. Matching results in the bindings: '[?top/top, ?meaning/(meaning ((ont-category Ref [mouse]))), ?ref/Ref]'.

Merging then results in the following modified structure:

```
  ((top (meaning ((giveness R uniquely-identifiable]))
        (subunits (mouse-unit)))
   (mouse-unit
        (meaning  ((ont-category R [mouse])))
        (sem-cat  ((schema R [ReferentDescriptor])
                   (quantity R 1)))))
  <-->
  ((top (subunits (mouse-unit))
   (mouse-unit
        (form     ((orth ?word-1 "mouse")))
        (syn-cat  ((schema ?word-1 [WordDescriptor])
                   (type Noun)
                   (number singular))))))
```

As in parsing, all constraints in the construction missing in the initial transient structure have been added, and the new `mouse-unit` includes in its meaning the tagged component corresponding to the matched meaning of `mouse-cxn`.

Although not an issue for our simple example, many problems can arise in the search for matching constructions during processing and the selection of the best set of constructions to apply. FCG provides several mechanisms for coping with these challenges. These include the use of different goal tests (such as reentrance, in which a produced utterance is re-parsed using the current grammar to test for interpretability); the option of continuing processing if an analysis is insufficient; and the possibility of associating conventionality and preference scores with constructions as heuristics for guiding search. These strategies are explained in more detail elsewhere [15].

6.2 Constructional Analysis in ECG

In this section we briefly summarize how ECG constructions support language comprehension, as implemented by the construction analyzer described by [2]. The term *constructional analysis* as used here is analogous to *parsing* in FCG: it is the constructional analogue to syntactic parsing—that is, the identification of which linguistic structures are instantiated in a particular utterance—where the structures crucially include semantic information.

Overview. The input to constructional analysis is an ECG grammar (including both schema and construction lattices), along with the utterance to be analyzed, and (optionally) a situation description. All schemas and constructions are first translated into a feature structure representation, ensuring that all inherited roles, constituents and constraints are included. Earlier we showed the feature structure for *mouse* in Figure 4; Figure 9 shows a feature structure version of the DeterminedNP construction.

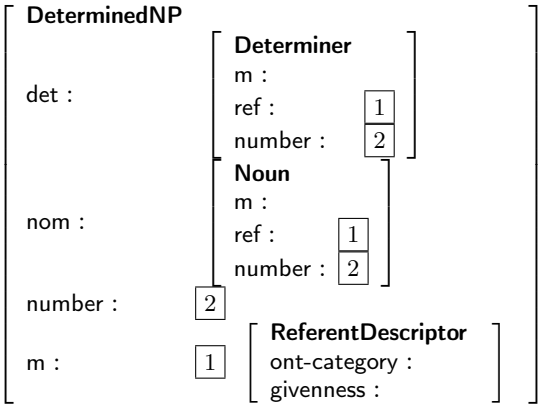

Fig. 9. Translation of the DeterminedNP construction into a feature structure. The features shown correspond to the two constituents (det and nom), the number constructions feature and the meaning pole m. Identification bindings are represented using boxed index numbers, where indices with the same number are bound to the same value.

The analyzer processes utterances from left to right, incrementally building up an *analysis graph* (a set of constructional instances or *constructs* linked by constituency relations) and a *semantic specification*, or *semspec* (a graph of the meaning schemas associated with all constructs in the analysis). In the broader research context for which ECG was developed, this semspec is an intermediate structure whose purpose is to support two connected language understanding processes: (1) contextual *resolution*, which grounds this interpretation in the situational context; and (2) embodied *simulation*, which draws on richer embodied

structures to yield further context-sensitive inferences [**?**]. Here we focus on how the semspec is built up during analysis.

The construction analyzer described by Bryant (2008) uses unification as the basic mechanism for composing constructions and verifying that their constraints are consistent, where both constructions and schemas are represented as typed feature structures with unification constraints as specified by the ECG formalism. But the search for the best analysis also exploits many heuristics to improve efficiency, limit search and approximate aspects of human language processing, including:

- Incremental interpretation: the analyzer allows incremental left-to-right interpretation of the utterance. To do this, it employs left-corner parsing techniques [9] to keep track of competing analyses and update their scores, where partially matched subportions of complex constructions provide top-down expectations about which constructions may next be encountered.
- Best-fit interpretation: the analyzer defines a quantitative heuristic for combining information from disparate domains, ranking candidate interpretations, and guiding parsing decisions. The implementation is a Bayesian probabilistic model that integrates information affecting the likelihood of the analysis (e.g., lexical and constructional frequencies; the likelihood that one construction has another as a constituent; and the likelihood that a schema has a particular kind of filler in a given role).
- Partial interpretation: the analyzer produces partial analyses even when the input utterance is not covered by the grammar or is missing constituents. An extension to the analyzer permits analyses with omitted constituents (as often encountered in, for example, Mandarin) by integrating the score of an interpretation with the results of the contextual resolution process.

In sum, the analyzer is consistent with the constructional view, drawing on all available information at every step to ensure that syntactic, semantic and constructional constraints are satisfied. Crucially, the early incorporation of semantic, pragmatic and statistical constraints can dramatically reduce the search space that may result from purely syntactic approaches.

Example. We consider the simple case of analyzing the input sentence string "the mouse" given a grammar containing just the constructions defined earlier. Following the left-corner parsing algorithm, the analyzer maintains a stack of all the constructs (instances of constructions) recognized so far, all labeled as incomplete or complete; incomplete constructs are also annotated with which constituents still remain to complete it. Processing unfolds in several steps:

- The analyzer reads in the first word "the" and retrieves the THE construction based on the orthographic form and adds it to the current stack of available constructs. Since it has no constituents remaining, it is recorded as complete.
- All constructions of which THE is a subcase (here, just the DETERMINER construction) are added to the stack as complete.

- Because the first constituent of the DeterminedNoun construction is typed as a Determiner, it is placed on the stack and a constituent binding (between the The construction and its first constituent) is attempted. The binding is successful, resulting in a partial semspec consisting of the Determiner construction's evoked ReferentDescriptor being bound with the ReferentDescriptor of the DeterminedNoun construction; this structure is also specified having a givenness status of uniquely-identifiable. The DeterminedNoun construction has now scanned past the Determiner constituent.
- The analyzer reads in the second word "mouse" and retrieves the Mouse construction. Similar to before, it places Mouse as well its parent construction type Noun on the stack as complete.
- The DeterminedNoun construction placed on the stack earlier successfully scans its next unfulfilled constituent, adding the constituent binding to its nom constituent and updating the semspec with the relevant bindings.
- The DeterminedNoun is now marked as complete; since the utterance string has been exhausted, it has been successfully parsed.

Figure 10 shows the output of the ECG constructional analyzer on our simple example of *the mouse*. In fact, this structure shows both (a portion of) the analysis graph and its associated semspec. (The form domain is not shown.) In brief, each large box corresponds to an instantiated construction or schema, shown with its constituents or roles. Thus, the box labeled DeterminedNP has as top-level features its two constituents det and nom, its constructional number feature, its meaning pole m and the evoked ReferentDescriptor ref. The boxed numbers indicate shared values, many referring to structures not shown in this reduced figure, but note that the boxed $\boxed{2}$ indicates that ReferentDescriptor is shared in several places: the overall meaning of the DeterminedNoun construction, its ref slot as well as those of its two constituents.

Though beyond the scope of the example, it may be illuminating to take the analysis above a few steps further. Once the DeterminedNP has been successfully matched, various constructions with an initial constituent matching that construction would be added to the stack for consideration. These include constructions corresponding to a variety of possible completions, including *The mouse ran*, *The mouse ran past*, *The mouse ran past the barn* and even *The mouse ran past the barn fell*. Depending on the actual input, these constructions would differ in, for example, how much of the form they account for, the semantic likelihood of the associated semspec, and the constructional likelihood. The combined scores is used to prune the set of candidate parser actions and select the best one.

The constructional analyzer has been applied to a variety of linguistic phenomena, including modeling families of related argument structure constructions [5], early Mandarin constructions [11] and Hebrew morphological constructions [13]. Besides serving as a platform for linguistic analysis, it has also been applied as a psycholinguistic model of reading time data [2], and versions of the analyzer have been integrated in models of child language acquisition [3, 11]. Ongoing research has integrated ECG representations of mental spaces and metaphor

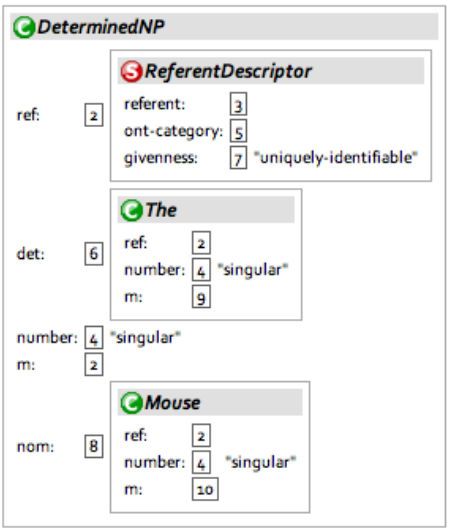

Fig. 10. The semantic specification resulting from analyzing "the mouse".

into the constructional analysis process (Feldman & Gilardi, In prep.), similar to earlier proposals [12**?**].

6.3 Two Processing Models

Major differences between FCG and ECG can be seen in their processing models. Although in some ways not surprising, given their different goals, it is nevertheless interesting to compare how the two frameworks handle essentially the same input. The comparison is most direct for the comprehension models: the two formalisms are subject to the same high-level requirements that they must select candidate constructions, check whether they fit with the current transient structure (in FCG) or partial analysis (in ECG), and choose the overall best set of such constructions. Where they differ is in what kinds of information are available for each of these steps, and what criteria they use for making decisions and prioritizing their respective searches. We discuss these differences below.

Bidirectionality The most obvious difference between the two formalisms is defined by an absence: ECG currently lacks an implemented model of processing, and cannot thus be said to provide a full model of both sides of the usage coin. This asymmetry is due in part to the focus in ECG on building cognitively plausible models, since the preponderance of psycholinguistic evidence is in comprehension. In principle, however, ECG grammars are declarative sets of

constraints that state relationships between form and meaning that should hold in production just as in comprehension. Thus, it is possible that production in ECG would draw on similar data structures and processes as used in comprehension (best-fit combination of evidence, prioritization based on semantic and cognitive heuristics, etc.). One hypothesis is that production may employ the same grammar structures (i.e., construction and schema definitions) as in comprehension, but with different usage statistics.

As noted earlier, the structural components of FCG constructions are used differently in parsing and production: in parsing, regular syntactic units are matched and J-units are merged, whereas in production the regular semantic units are matched, and J-units are merged. FCG thus makes a more specific claim about the relation between parsing and production than ECG can yet make—and it may well be that a working production model for ECG would make quite different kinds of claims, especially if (as conjectured above) the declarative structures of ECG are able to support both kinds of processes.

Matching, Merging and Unification Both parsing and production in FCG rely on the distinction between matching and merging: the match performs the check or filter on candidate constructions, and the merge causes additional constraints from applicable constructions to be unified into the transient structure. Comprehension in ECG similarly involves determining which constructions out of the whole grammar apply, and then using unification to produce the interim semantic specification. The first step employs some heuristics to reduce the set of possible constructions: specific forms (typically orthographic strings) observed in the utterance are directly associated with constructions including those forms; and (as described earlier) the combination of left-corner parsing with constructional types restricts the set of candidate constructions at any given stage of processing. Constructions passing through this initial filter must still be tested for whether they can be unified with the analysis in progress; successful unification at this stage acts as both a filter on candidate constructions and the mechanism for combining all relevant constructional constraints. It thus corresponds to both matching and merging in FCG.

Essentially, both frameworks have devised different strategies to cope with the computational expense of unification. In FCG, the split between simpler matching heuristics and more expensive merging operations allows some degree of pre-optimization. In ECG, costly unification is part of the matching process, but the availability of a lattice of constructional types helps to compensate for that expense by supplying useful heuristics that restrict the search space of constructions.

Note, however, that the increased efficiency afforded by the use of inheritance in ECG may come at a price: changes to the grammar have potentially wide-ranging effects, and may necessitate costly measures to ensure consistency in the inheritance network. While such changes can be restricted to those that have minimal impact on the network, in general the cost of maintaining consistency may make inheritance impractical for situations in which grammars are liable

to undergo frequent changes. But it is precisely situations involving dynamically changing grammars that are the overriding (and titular) concern in FCG.

It should also be mentioned that it is possible in FCG to skip the matching and directly apply merging. Although this strategy has not yet been fully investigated, initial results indicate that it indeed leads to an explosion of the search space. But it also facilitates less restricted and hence more creative usages of constructions, which may be useful both for learning and for achieving robust handling of unfamiliar input.

Structural flexibility At first glance the two formalisms differ in the surface impressions they make. In both content and appearance, ECG is heavily influenced by work in frame semantics and cognitive linguistics. The notation itself, though integrated with processes of language learning and use, is expressed in a constraint language that avoids explicitly procedural information. ECG's schema and construction definitions act as data structures that are created, used and altered by the language learning and comprehension models. FCG, on the other hand, stems more directly from work in artificial intelligence and symbolic programming. Though FCG constructions include many declarative constraints, they also have operations that are closely tied to aspects of processing (especially when shown in their expanded form). In a way, FCG constructions act like programs that transform a transient linguistic structure as data.

It is difficult, however, to draw too fine a line between declarative and procedural aspects of each formalism. Both formalisms are, of course, data in the sense that a separate processing engine ultimately controls their execution, verifying that the constraints they specify hold or performing the actions they entail. By the same token, the various notations employed by each formalism have direct effects on processing and thus include procedural information in that sense. Hence, despite the surface differences, most of the constraints expressed in each language can be seen in both lights.

One possible way in which FCG may exhibit a more procedural orientation is provided by TAGs and J-units. With TAGs, a construction can explicitly instruct the match process to remember a binding for a tag variable. Such a variable acts as a local variable in standard programs, and serves to temporarily store a value until it is needed again later, e.g. in a J-unit. J-units in turn serve to provide explicit instructions to the merge process. They specify how to change constituent structure and, in combination with tags, and how to move information between constituents.

Again, however, such effects could be viewed in terms of a set of constraints on the intermediate and final structures involved. The crucial observation here might center not on how these effects are described, but instead on what they do—in particular, the fact that they change constituent structure in this manner. As noted earlier, each FCG constructions specifies precisely the effects (agreement, categories, etc.) that apply to the resulting transient structure, which means that they are relatively free to change the internal structure. As a result, the constituent structure at the end of processing may be difficult to infer di-

rectly from constructional definitions. In ECG, on the other hand, constituent structure generally follows that declared in the constructional domain; while the semantic structure need not mirror this structure, the final constructional structure is related relatively directly to that specified in construction definitions. This difference reflects yet again the tendency toward flexibility and freedom in FCG, versus the importance of motivated constraints in ECG.

Cognitively motivated processing Last, but not least, the two processing models differ markedly in the phenomena they target. FCG models are designed to be *functional*: they are intended to satisfy the input and output constraints of communication systems, often those exemplified by particular human linguistic phenomena. The particular processing mechanisms involved are not, however, intended to reflect the implementations of language processing in the human brain. In contrast, ECG's language comprehension model is specifically designed to be cognitively plausible: not only does it fulfill the basic task of identifying constructions instantiated by an utterance and producing the corresponding interpretations, but it does so in a way that reflects the robust, incremental and best-fit nature of human language processing. It thus exploits many efficiencies that come from psycholinguistic evidence about online sentence processing.

Recent work by Wellens (this volume) explores how usage-based networks of FCG constructions can be used to facilitate processing. Such work takes a step in the direction of cognitively motivated processing. Heuristics used to guide search during language processing may also capture some cognitively motivated factors, but thus far little work has been done to fully exploit the potential for learning from the constraints of human language processing in FCG.

7 Discussion and Outlook

The preceding sections describe two computational formalisms that implement ideas from construction-based approaches to grammar. Relative to the range of approaches in the literature, the similarities between the two formalisms and their associated research frameworks far outnumber their differences. Both include notational means of expressing constructional mappings between form and meaning, and both provide the basic representational toolkit for representing categories, agreement and constituent structure.

In addition to these theoretically inspired commitments, the two frameworks also share many methodological assumptions. Both formalisms aim to build working systems that not only describe but in fact instantiate the structures and processes proposed. Unlike many other approaches, they do not stop at describing linguistic knowledge in formal notation but rather offer models of how they are actually used in communication. Processing considerations have thus shaped both formalisms.

The many shared qualities discussed in the preceding sections might be seen as independent requirements for any computational construction grammar formalism. Their differences are perhaps even more revealing of the specific issues

that computational implementations of construction-based grammar must face; we summarize some of these below.

7.1 Freedom of expression

Perhaps the main recurring theme in this comparison has involved the relatively restricted nature of ECG as compared to FCG: in general, ECG allows a more restricted set of notational possibilities, as exemplified by its inclusion of a schema formalism for expressing constructional form and meaning; its limited set of expressible constraints; its stronger assumptions about (some) structural parallels between the form and meaning domains; and the relative monotonicity of the internal structures built up during processing.

By contrast, FCG has been designed to be as open-ended as possible, allowing grammar-writers (and, not coincidentally, evolving agents) a free hand in exploring different styles of representation and strategies for achieving successful communication. This freedom is apparent not only in the broad array of representational devices allowed in form and meaning, but also in the choice of syntactic and semantic categories; the flexible independence of units in the form and meaning domains; and the possibility of fundamental alterations of constituent structure allowed during processing.

These fundamental differences beg the question: how much freedom of expression is enough, and can you have too much? These questions must, of course, be posed relative to the kinds of phenomena the respective formalisms are intended to account for. While ECG is a simpler formalism, it has thus far proven sufficiently expressive for its purposes—to wit, capturing linguistic insights, accounting for psycholinguistic evidence, and being learnable in a developmentally plausible way. It has not, of course, been deployed in the context of language evolution experiments, so it is as yet unclear whether its restricted set of possibilities would give rise to the same unbounded range of communicative creativity, or allow the degree of representational fluidity, fostered by FCG. On the other hand, to the extent that FCG has interest in the specific questions of human communication, it would be worthwhile to find concrete, realistic cases in natural language that demand the amount of freedom and corresponding complexity afforded by FCG.

7.2 Structure and process

A related issue concerns the directness of the connection between the structures appearing in the formalism and the particular procedures employed during processing. The structure of ECG definitions specifies constraints on the function of processing, but it does not specify precisely how the analyzer proceeds. This affords it a certain amount of stability across particular implementations of processing. The same structures are also intended to be useful for both language comprehension and language production, though a concrete implementation of the latter will be necessary before this idea can be explored and validated.

FCG constructions are not only useful in both language production and comprehension, but their internal structure also reflects some more specific claims FCG makes about the relation between those two processes. In particular, the ways in which different kinds of constraints are expressed (e.g., whether used for matching or merging) correspond directly to the (symmetric) ways in which they are used in these both processing modes.

These differences raise the question of whether and how directly the declarative constraints relevant to each construction can be abstracted from processes of use. A related question is how and whether such construction content must change in order to satisfy the constraints of both kinds of processing. Perhaps FCG's experience in this area could lead to predictions about how these questions would be answered for ECG.

7.3 Benefits and drawbacks of inheritance

The organization of FCG and ECG grammars reflects a major representational difference between the two formalisms. ECG makes explicit use of constructional and schematic inheritance relationships, as expressed by type lattices. Such relations capture various linguistic generalizations and naturally lead to more concise and coherent grammars. Not coincidentally, they are also more reminiscent of the kinds of linguistic structures typically proposed by cognitive linguists (and hence perhaps easier for them to understand), and exploit object-oriented design principles from computer science.

FCG grammars have mechanisms for achieving some of the effects of inheritance, such as the templates to notate shared structure. Some FCG grammars also employ frame-based ontologies that are comparable to the schema hierarchies of ECG (see for instance [10]). And, as mentioned, some relations implicit from the use and interaction of constructions during processing may be captured in constructional dependency networks, thus making constructional relations a directly usage-based matter. On the whole, however, FCG has not yet employed explicit notions of inheritance.

As noted earlier, this difference is consistent with FCG's emphasis on the independence of constructions, and the need for making small, local changes: it is relatively easy to change the flow of processing by modifying existing constructions or by adding new constructions to the constructicon (though it may be difficult to predict their consequences). By contrast, in grammars like ECG that employ extensive inheritance relations, small changes may affect a large number of constructions, necessitating more complicated measures for maintaining consistency or re-initializing to reflect updates. The effects of these differences on processing depend, of course, on particular implementational choices, and how much change and fluidity is necessary. The arena of language learning, though not discussed in the current chapter, may offer the best domain for exploring these questions: both formalisms have associated models of learning, though ECG's is developmental while FCG's is evolutionary; the parallels and differences between these endeavors should reward further investigation.

7.4 Different formalisms for different goals

Many of the differences between FCG and ECG reflect their respective backgrounds and priorities. ECG was intended from the start as a theory of human cognition, embracing the foundational ideas of cognitive linguistics. Its goal has been to capture patterns of human categorization and processing, while expressing linguistic and conceptual generalizations. The roots of FCG in artifical language evolution have given it a more dynamic and fluid view of linguistic representations, which crucially requires a certain amount of independence among representations. The developmental path of each formalism reflects these biases and accounts for many of the phenomena we have illustrated here.

But stepping back, it should be clear that these different approaches provide complementary perspectives on the same underlying phenomena of embodied, situated communication. Though they ask different versions of the question—emphasizing, respectively, the constraints imposed by human cognition, versus the freedom to evolve diverse communicative strategies—they nevertheless both provide important ways of framing any complete approach to modeling language structure, use and acquisition. Perhaps most significantly, the fact that both formalisms have concrete implementations of their various structures and processes gives them an additional dimension of considerations not typically available for non-implemented grammatical theories and permits a much more nuanced comparison of approaches than would otherwise be possible. Only when such computationally precise descriptions are available can issues like those raised here be recognized and explored across the broader field of constructional approaches to grammar.

Bibliography

[1] Bleys, J., Stadler, K., De Beule, J.: Search in linguistic processing. In: Steels, L. (ed.) Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[2] Bryant, J.: Best-Fit Constructional Analysis. Ph.D. thesis, Computer Science Division, University of California at Berkeley (2008)
[3] Chang, N.: Constructing grammar: A computational model of the emergence of early constructions. Ph.D. thesis, Computer Science Division, University of California at Berkeley (2008)
[4] Ciortuz, L., Saveluc, V.: Fluid Construction Grammar and feature constraints logics. In: Steels, L. (ed.) Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[5] Dodge, E.: Conceptual and Constructional Composition. Ph.D. thesis, University of California at Berkeley (2010)
[6] Feldman, J.A.: From Molecule to Metaphor: A Neural Theory of Language. MIT Press, Cambridge, MA (2006)
[7] Goldberg, A.E.: Constructions: A Construction Grammar Approach to Argument Structure. University of Chicago Press (1995)

[8] Langacker, R.W.: Foundations of Cognitive Grammar, Vol. 1. Stanford University Press (1987)
[9] Manning, C., Carpenter, B.: Probabilistic parsing using left-corner language models. In: Proceedings of the 5th International Workshop on Parsing Technology (1997)
[10] Micelli, V.: Field topology and information structure: A case study for German constituent order. In: Steels, L. (ed.) Computational Issues in Fluid Construction Grammar. Springer Verlag, Berlin (2012)
[11] Mok, E.: Contextual Bootstrapping for Grammar Learning. Ph.D. thesis, Computer Science Division, University of California at Berkeley (2008)
[12] Mok, E., Bryant, J., Feldman, J.: Scaling understanding up to mental spaces. In: Proceedings of the 2nd International Workshop on Scalable Natural Language Understanding (ScaNaLU-2004). Boston, MA (2004)
[13] Schneider, N.: Computational cognitive morphosemantics: modeling morphological compositionality in hebrew verbs with embodied construction grammar. In: Proceedings of the Berkeley Linguistics Society (2010)
[14] Spranger, M., Loetzsch, M.: Syntactic indeterminacy and semantic ambiguity: A case study for German spatial phrases. In: Steels, L. (ed.) Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[15] Steels, L. (ed.): Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[16] Steels, L.: A first encounter with Fluid Construction Grammar. In: Steels, L. (ed.) Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)
[17] van Trijp, R.: Feature matrices and agreement: A case study for German case. In: Steels, L. (ed.) Design Patterns in Fluid Construction Grammar. John Benjamins, Amsterdam (2011)