A comparison between Fluid Construction Grammar and Sign-Based Construction Grammar

Remi van Trijp
Sony Computer Science Laboratory Paris

Construction Grammar has reached a stage of maturity where many researchers are looking for an explicit formal grounding of their work. Recently, there have been exciting developments to cater for this demand, most notably in Sign-Based Construction Grammar (SBCG) and Fluid Construction Grammar (FCG). Unfortunately, like playing a music instrument, the formalisms used by SBCG and FCG take time and effort to master, and linguists who are unfamiliar with them may not always appreciate the far-reaching theoretical consequences of adopting this or that approach. This paper undresses SBCG and FCG to their bare essentials, and offers a linguist-friendly comparison that looks at how both approaches define constructions, linguistic knowledge and language processing.

Keywords: formalization, computational construction grammar, constructional language processing

1. Introduction

In his seminal work on Cognitive Grammar, Ronald Langacker (1987: 42) wrote:

> We can agree that a linguist should make his description as precise and explicit as possible at every stage of analysis. We can further agree that at some point in its evolution a linguistic theory must receive appropriate mathematical expression.

Now, a quarter of a century later, that time has come for cognitive linguistics, in which many researchers are looking for ways to make their analyses formally explicit. Fortunately, there have been some exciting developments to cater for this demand, most notably in *Sign-Based Construction Grammar* (SBCG; Boas & Sag 2012) and *Fluid Construction Grammar* (FCG; Steels 2011b, 2012). On the downside, however, the formalisms used by SBCG and FCG take time and effort to

Constructions and Frames 5:1 (2013), **88–116**. DOI 10.1075/cf.5.1.04van
ISSN 1876–1933 / E-ISSN 1876–1941

master, and it may not always be clear from the outset what the far-reaching consequences are of adopting this or that approach.

One particular misconception is that analyses in SBCG and FCG are easily interchangeable because both theories use *feature-value pairs* for representing linguistic knowledge, and *unification* for combining these feature-value pairs. The truth is that the notions of 'feature-value pairs' and 'unification' have been developed in a broad array of scientific fields, ranging from computational linguistics to knowledge representation and theorem proving (Shieber 1986). So rather than 'unifying' all formalisms that make use of these notions, feature structures and unification have turned out to be powerful, yet theory-independent tools that can be adopted in many different ways depending on one's research objectives.

The goal of this paper is therefore to look beyond this superficial connection between SBCG and FCG and offer a linguist-friendly comparison.[1] The richness of both theories precludes any in-depth coverage of either approach in a single paper, so I stripped down the two approaches to their bare essentials in order to reveal the often deep-running theoretical divergences between them. It should also be noted that I am not a neutral observer in this comparison, as I operationalize all my own work in Fluid Construction Grammar. My aim is however not to argue which approach is 'best', but rather to help the uninitiated linguist to choose which of the two better suits his or her research objectives.

2. Scientific objectives and historical roots

So what are those research objectives? In a nutshell, SBCG embraces a *generative* conception of linguistic theory, whereas FCG adopts a *cognitive-functional* approach.

The goal of a generative grammar is to distinguish well-formed utterances from ungrammatical ones, and to assign a 'correct' structure to the well-formed utterances through a model of linguistic competence (= *generation*). The words "generative" and "generation" should not be confused with actual language production, as Chomsky (1965: 9) has repeatedly emphasized:

1. Sag (2010) has called SBCG a variant of construction-based HPSG that integrates insights from Berkeley Construction Grammar (BCG; Fillmore 1988). However, HPSG and BCG remain different theories in significant ways, so this paper's discussion on SBCG does not automatically hold for those two approaches (see e.g. Müller 2010: 248–253, for a comparison of SBCG to HPSG; and Michaelis 2013 for a comparison between SBCG and BCG). Another formal approach, Embodied Construction Grammar (ECG), is left out of the discussion because a comparison between ECG and FCG has already been presented by Chang, De Beule, & Micelli (2012), and because Müller (2010: 253–257) has already shown how ECG can be translated into HPSG.

> To avoid what has been a continuing misunderstanding, it is perhaps worthwhile to reiterate that a generative grammar is not a model for a speaker or hearer. It attempts to characterize in the most neutral possible terms the knowledge of the language […]. When we speak of a grammar as generating a sentence with a certain structural description, we mean simply that the grammar assigns this structural description to the sentence.

The word "generative" should also not be confused with its more technical sense in formal grammar, where "generative-enumerative" grammars are contrasted with "model-theoretic" grammars (see Pullum & Scholz 2001 for more details).

The goal of a cognitive-functional grammar, on the other hand, is to explain how speakers express their conceptualizations of the world through language (= *production*) and how listeners analyze utterances into meanings (= *parsing*). Cognitive-functional grammars therefore implement both a competence and a processing model.

Since both the generative and cognitive-functional approaches cover large families of linguistic theories, it is useful to trace back the history of ideas that led to SBCG and FCG to see more clearly where the roots of both theories are situated. This history is illustrated in Figure 1.

2.1 Historical roots of SBCG: Modern CFGs

Through three of the most influential publications in the history of linguistics, Chomsky (1956, 1957, 1958) established the foundations of generative grammar, which aims to describe a speaker's *linguistic competence* in the form of a model that is able to *generate* all of the well-formed utterances of a language (and only these). One of the crucial ideas of generative grammar is the notion of a *context-free phrase structure grammar* (also called *phrase structure grammar* or *context-free grammar*, and commonly abbreviated as *CFG*). Example (1) shows a simple CFG:

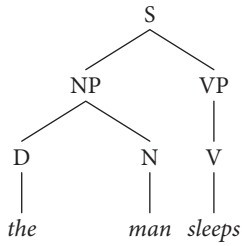
(1)

S	→	NP VP	N	→	*ideas*
NP	→	(D)N	N	→	*bagels*
VP	→	V (NP)	V	→	*sleeps*
D	→	*a*	V	→	*kick*
D	→	*the*	V	→	*ate*
N	→	*man*	…		

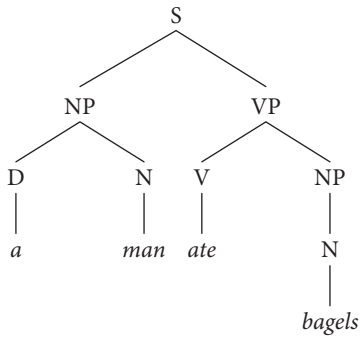
The term *context-free* comes from the fact that the rules of a CFG can be applied regardless of the context of their left-hand side. In the original CFGs, those rules were *rewrite rules* where the non-terminal symbol (i.e. a category) on the left of the arrow is rewritten as the ordered list of terminals (i.e. lexical items) and non-

terminals on the right of the arrow. The CFG in (1) can thus be used to randomly generate a series of structures such as the ones in example (2).

(2) a.



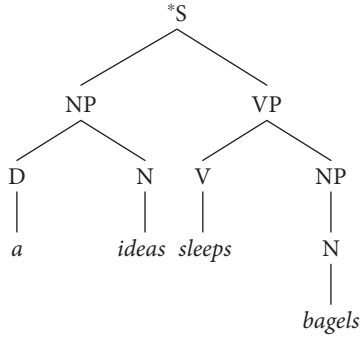
b.



2.1.1 *SBCG as a non-derivational generative grammar.*

One of the problems of the original CFGs is that they easily run into problems of overgeneration, such as example (3).

(3)



In order to overcome this and other limitations of CFGs, Chomsky (1956) introduced the notion of *transformational grammars*. The backbone of such a grammar is a small kernel of basic structures that can be generated by CFGs, but additionally

there is a set of transformation rules, such as passivization, that can be applied to those kernel structures to derive more complex sentences.

In the seventies and eighties, however, there came a strong reaction against such transformational grammars because they made predictions that were incompatible with data on linguistic performance. Whereas most transformationalists considered such data to be irrelevant, an important group of scholars argued that a psychologically plausible competence grammar must be compatible with models of performance (see Sag et al. 1986 for a joint vision statement). This movement of *non-derivational generative grammars* (see Figure 1) included, among others, *Generalized Phrase Structure Grammar* (Gazdar, Klein, Pullum, & Sag 1985), *Lexical-Functional Grammar* (LFG; Kaplan & Bresnan 1982), *Head Grammar* (HG; Pollard 1984) and *Head-Driven Phrase Structure Grammar* (HPSG; Pollard & Sag 1994). SBCG is a direct descendant from HPSG (and its predecessors HG and GPSG), of which two important characteristics are relevant for our discussion:

1. They incorporate a 'modern' phrase structure grammar (Blevins & Sag 2013) in which the traditional rewrite rules are reinterpreted as "static constraints" (see Section 3.1).[2]
2. Non-terminal symbols are no longer treated as atomic units, but as complex feature-value pairs that can be combined with each other through unification. Unification, which was originally proposed by Martin Kay (1979) who needed a powerful tool for machine translation, has been adopted in some form or the other by virtually all non-derivational grammar formalisms because it is *monotonic*. That is, unification can only combine feature-value pairs that are compatible with each other, hence there are never structural changes.

2.1.2 *The heritage of BCG.*

The seventies and eighties also saw the birth of Construction Grammar (Fillmore 1988; Kay & Fillmore 1999), which has now become known as Berkeley Construction Grammar (BCG). The foundations of BCG can be traced back directly to Fillmore's influential *Case Grammar* (Fillmore 1968; Östman & Fried 2004). BCG soon united several researchers who broke with some of the central methods of Chomskyan linguistics, such as its treatment of semantics and the strict divide between core and periphery.

Under the impulse of Paul Kay, BCG was seeking to strengthen its formal foundations as well. BCG joined the movement of non-derivational grammars that use

2. An important observation is that HPSG, despite its name, is not a phrase structure grammar even in this modern sense because it does not restrict itself to the locality constraints of CFGs and because it allows other levels of linguistic organization that are decoupled from phrase structure (e.g. Reape 1994).

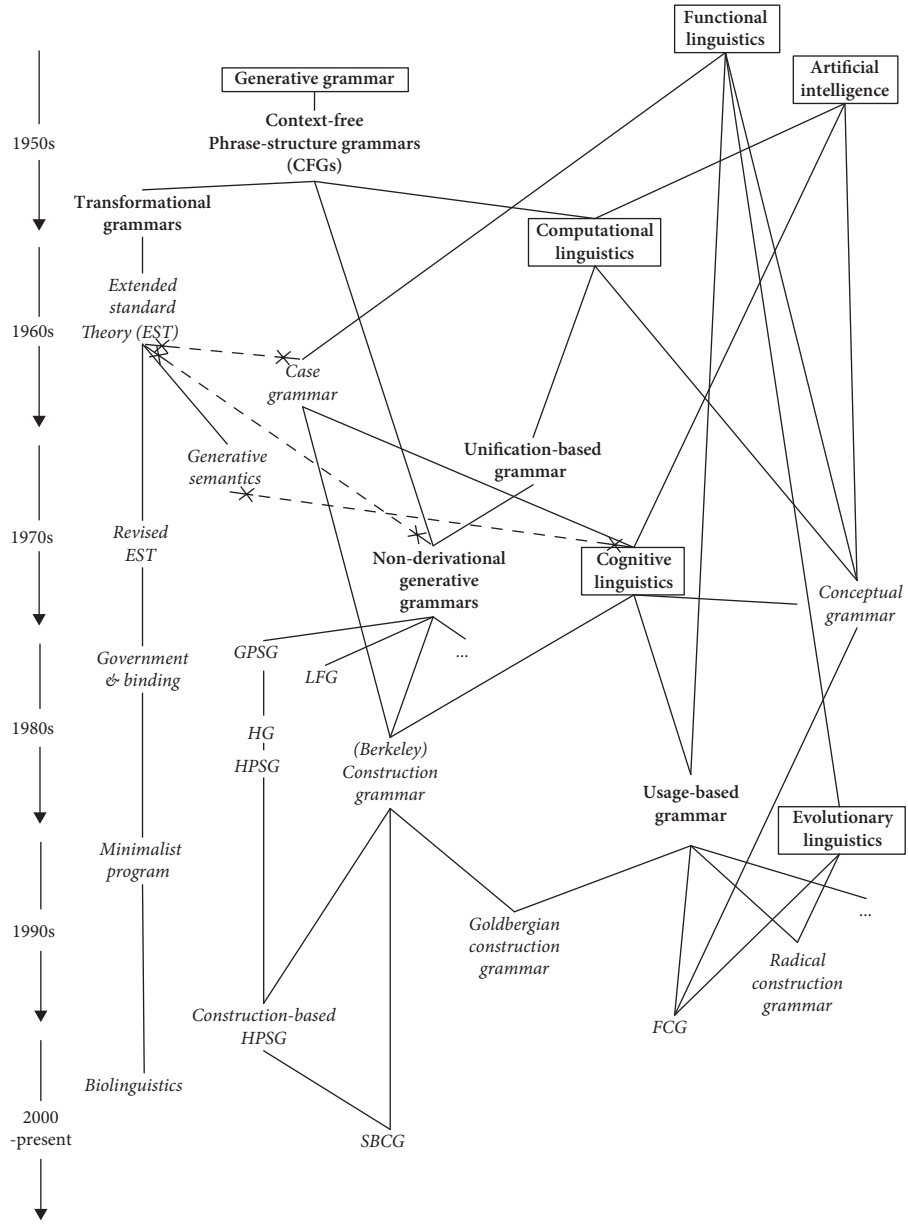


Figure 1. A partial history of linguistics situates SBCG in a family of generative grammars, whereas FCG's heritage lies in cognitive-functional approaches to language. A dotted line in the Figure stands for an important rupture with an ancestor. For BCG, only its generative dimension is displayed, but a lot of work in this framework is also cognitive-functional.

unification as their main device for ensuring monotonic analyses, which facilitated the exchange with the aforementioned theories. While the original BCG is still an active and evolving line of research today (e.g. Ohori 2005; Fried 2009), the discussions between HPSG and BCG that started in the late 80s would ultimately lead to SBCG, a theory that aims to "expand the empirical coverage of HPSG, while at the same time putting BCG on a firmer theoretical footing" (Sag 2012: 70).

2.2 Historical roots of FCG: Artificial intelligence

One of science's little ironies is that the field of Artificial Intelligence (AI) was founded at the famous Dartmouth conference in 1956 (Russel & Norvig 2003: 17), which happened to be the same year in which Chomsky (1956) launched transformational grammar. The irony is that an AI approach to language is by definition a challenge to 'hardcore' Chomskyans: all work in AI is supported by *computational* implementations, which in the case of language involves *processing models* for parsing and production. Because a particular branch of Chomskyans dismissed processing models as being irrelevant for linguistic theory, several AI researchers (such as Ron Kaplan, Marvin Minsky, Roger Schank, Eric Wanner, and Terry Winograd) were heavily attacked in the 70s.[3] Consider the following criticism of AI by Dresher and Hornstein (1976: 377):

> [W]orkers in AI have misconstrued what the goals of an explanatory theory of language should be […] Not only has work in AI not *yet* made any contribution to a scientific theory of language, there is no reason to believe that [it …] will *ever* lead to such theories, for it is aimed in a different direction altogether.

Early work in cognitive linguistics was, however, directly inspired by the AI approach (Lakoff & Thompson 1975: 296), and it is also in the field of AI in the seventies that the first predecessor of Fluid Construction Grammar emerged: *Conceptual Grammar* (Steels 1978a, b). Conceptual Grammar not only drew inspiration from AI and early cognitive linguistics, but also brought in a strong flavor of functional linguistics (which was firmly rooted in Europe, see, among others, Tesnière 1959; Halliday 1973; Dik 1978) — a mix of influences that reappears in FCG today.

During the 80s and early 90s, Steels moved on to different domains in AI, such as expert systems (Steels 1990), and he became one of the pioneers in

3. AI and computational linguistics also played a significant role in the history of the non-derivational generative grammars as described in Section 2.1. I did not focus strongly on this part of SBCG's historical roots because the theory is more firmly situated in a tradition of mathematical linguistics and it currently has no complete computational embodiment yet; see Section 5.1.

behavior-based robotics (Steels & Brooks 1995). The behavior-based approach has caused a major paradigm shift in robotics: instead of pre-programming intelligent behaviors, the new paradigm investigates which mechanisms are minimally required for more complex sensori-motor intelligence to *emerge spontaneously*. Steels (1995, 2000) soon realized that language is a *complex adaptive system* (CAS) as well, and since the mid-nineties, he and his collaborators have applied this insight to explore the origins and evolution of language (Steels 2011c).

The CAS view on language radically changes what a grammar formalism should operationalize. Instead of trying to account for 'the' grammar of an abstracted, ideal speaker-listener (Chomsky 1965), the CAS approach tries to account for the emergence of language as the result of local interactions between language users from a heterogeneous speech population. This means that the formalism needs to be able to handle utterances that are not well-formed, conventions with different degrees of entrenchment, and so on.

Fluid Construction Grammar thus grew out of the need for more sophisticated representation and processing techniques to support such a conception of language. The earliest implementations of FCG date from the end of the nineties (Steels 1998), and it soon became clear that FCG embodied many of the viewpoints that were advocated in construction grammar, particularly the streams that emphasized the cognitive and functional dimensions of CxG (among others Croft 2005; Geeraerts 2008; Goldberg 1995; Östman & Fried 2004). A significant effort was, therefore, made to make the FCG-system also relevant for any work in linguistics from a constructionist point of view (Steels 2011b).

3. How are constructions defined?

As the name "construction grammar" suggests, all constructionist approaches to language assign a central role to *constructions*, which are informally defined as conventionalized pairings of meaning and form. This section compares how SBCG and FCG operationalize this definition.

3.1 Static constraints in SBCG

In SBCG, constructions are "descriptions that license classes of linguistic objects" (Sag 2012: 72). These descriptions take the form of static constraints, which means that they are defined once and then remain true for all the instances of the types of linguistic objects they constrain. SBCG does not have a single representation for constructions, but instead distinguishes between constructions that operate on

lexical items (listemes, lexical class constructions, derivational constructions and inflectional constructions) and combinatoric constructions.

3.1.1 *Constructions that license words*

Words are licensed in SBCG by listemes, lexical class constructions, derivational constructions, and inflectional constructions. A listeme can be considered as a generalized lexical entry (including multiword expressions). The listeme in example (4) defines the lexical entry of the proper noun *Kim* (Sag 2012: ex. 49).

(4)
$$\begin{bmatrix} \textit{pn-lxm} \\ \text{FORM} \quad \langle \textit{kim} \rangle \end{bmatrix}$$

The *Kim*-listeme is very concise because its other properties are inherited from other constructions, which is made possible through a *type inheritance hierarchy* (see Section 4.1). Here, the *Kim*-listeme is declared to be of type *pn-lxm* ('proper-noun-lexeme'). A second kind of lexical constructions, *lexical class constructions*, define the feature-value pairs that are associated with such types. Example (5) defines the *proper noun construction* (Sag 2012: Figure 9).

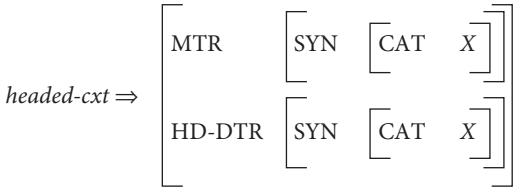
(5)
$$\textit{pn-lxm} \Rightarrow \begin{bmatrix} \text{FORM} & L \\ \text{SYN} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \textit{noun} \\ \text{SELECT} \quad \textit{none} \\ \text{XARG} \quad \textit{none} \end{bmatrix} \\ \text{VAL} & \lozenge \\ \text{MRKG} & \textit{def} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{IND} & i \\ \text{FRAMES} & \lozenge \end{bmatrix} \\ \text{CNTXT} & \begin{bmatrix} \text{BCKGRND} & \left\langle \begin{bmatrix} \textit{naming-fr} \\ \text{ENTITY} \quad i \\ \text{NAME} \quad L \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

As can be seen, a lexical class construction is an implicational constraint (indicated by the right arrow). The details of the construction are not important for our comparison. What matters here is that the above definition simply means that if a feature structure is of type *pn-lxm*, then it has to be compatible with the feature description defined in the construction. Other constructions that license words are derivational, inflectional and postinflectional constructions. These three kinds of constructions allow new lexemes to be built (or derived) from other lexical items (Sag 2012: 115–127).

3.1.2 *Combinatoric constructions as local constraints*

SBCG subscribes to a structuralist tradition that assumes that semantics can be directly read off a syntactic tree. This is most clear in its combinatoric constructions, which define how signs are allowed to combine into more complex signs by imposing constraints on *local tree configurations* (i.e. a parent node and its *immediate* children, or a "mother sign" and its "daughters" in SBCG terminology). For example, the *headed construction* shown in example (6), taken from Sag (2010: ex. 39), constrains the mother sign of a headed construct to have the same syntactic category as its head daughter. This construction thus imposes the 'X' as in X-bar theory.

(6)



The locality of SBCG constructions is perhaps the most salient break between SBCG and its predecessors HPSG and especially BCG, which argues for constructions that are able to access information at any depth of a linguistic structure (Fillmore 1988). So how does SBCG tackle the numerous nonlocal dependencies in language, such as topicalization (7) and WH-questions (8)?

(7) The ball the boy hit.
(8) What did the boy hit?

SBCG analyzes such nonlocal dependencies as a chain of local dependencies (Sag 2010), a solution that was pioneered by Gazdar (1981). More specifically, SBCG constructions may encode that there is a "gap" in the local structure of a phrase, and then percolate information about the "extracted element" up to the place where a "filler" for that gap can be found:

(9) [The ball $_{\text{FILLER}}$] the boy hit $\_\_\_\__{\text{GAP}}$.
(10) [What $_{\text{FILLER}}$] did the boy hit $\_\_\_\__{\text{GAP}}$?

3.2 Dynamic meaning-form mappings in FCG

In Fluid Construction Grammar, the task of a construction is to transduce meanings and functions into the forms that are conventionally associated with them (production), and vice versa, to transduce forms into their meanings and functions (parsing). FCG has a single representation for all constructions: a bidirectional mapping between meaning/function and form.

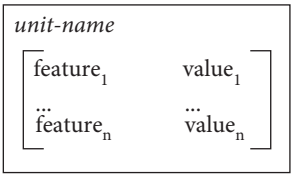
3.2.1 *The actual constructions*

Technically speaking, FCG represents linguistic information as a mapping between a *semantic pole* (or *left pole*) and a *syntactic pole* (or *right pole*). If such a mapping concerns information about an actual utterance, it is called a *transient structure*. The word "transient" refers to the fact that the utterance structure is not a pre-existing structure, but that it is built during processing. At the beginning of a processing task, the transient structure only contains the conceptualization that needs to be expressed (production) or the utterance that needs to be analyzed (parsing). During processing, the transient structure gradually grows more and more elaborate as more information about the utterance is added to it.

Information can be added to the transient structure by *constructions*. A construction is, just like a transient structure, a mapping between a semantic and a syntactic pole. Constructions, however, capture conventionalized meaning-form mappings, hence they are stored in the linguistic inventory. A construction also has an open-ended list of attributes where additional information about the construction can be kept, such as its type and token frequency, its degree of entrenchment and network-links to other constructions (see Section 4.2).

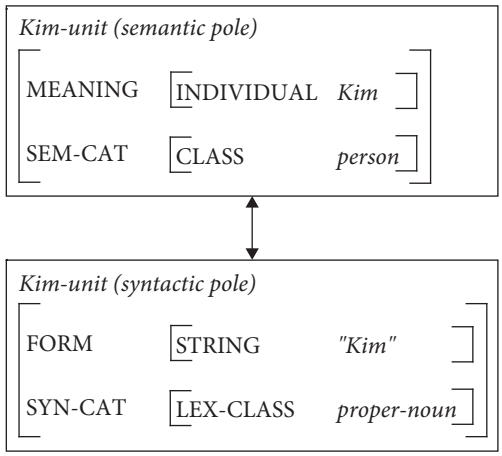
The semantic and syntactic poles of constructions and transient structures are represented as a flat list of *units*. A unit can be thought of as a labeled box that encapsulates feature-value pairs, as depicted in (11).

(11)



A unit can occur exclusively on the semantic or syntactic pole, but usually, the same unit-name will occur on both poles to indicate a coupling between semantic and syntactic features. Suppose that the FCG-system needs to parse the utterance *Kim saw Pat*, and that the *Kim*-construction has added a unit for *Kim* to the transient structure. The coupling of the semantic and syntactic *Kim*-unit is shown in example (12). For reasons of space, the semantic *Kim*-unit is shown on top and the syntactic *Kim*-unit at the bottom.

(12)



Remember that in SBCG, the *Kim*-listeme was *typed* as a proper-noun-lexeme, and that it *inherited* relevant information from the lexical class construction of example (5), which defined "once and for all the properties that distinguish proper nouns from other lexical classes" (Sag 2012: 108). In other words, the SBCG construction acted as a type constraint. FCG, on the other hand, uses *untyped* feature structures *without* inheritance because the set of types and their appropriate

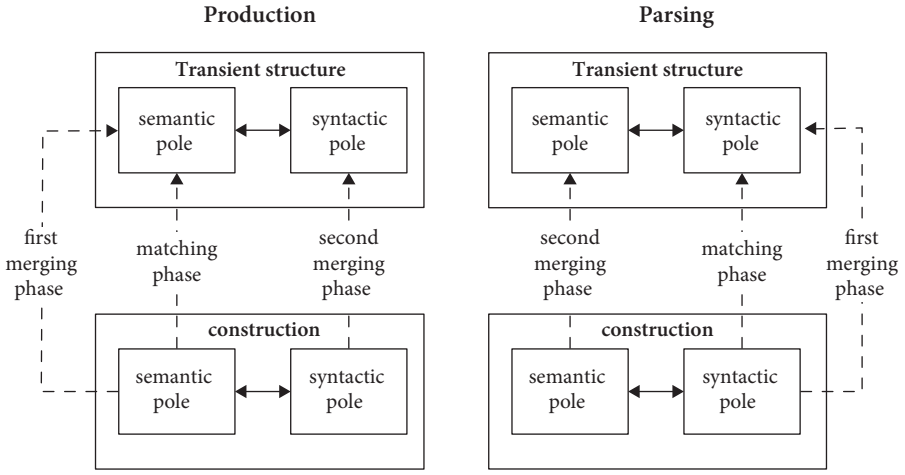
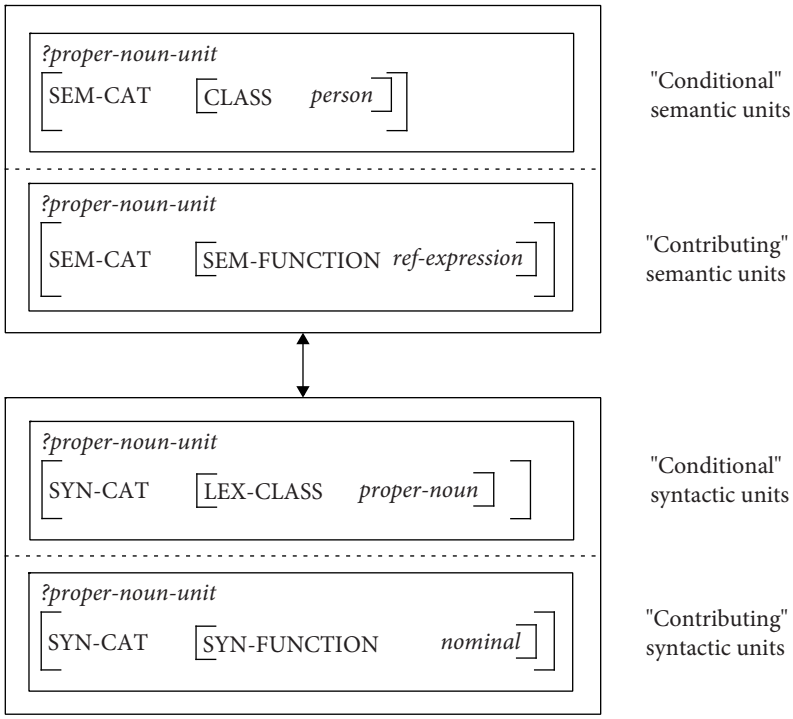


Figure 2. FCG constructions can be applied in both production (shown on the left) and parsing (shown on the right) without requiring a separate parsing or production algorithm. In production, the conditional units of the semantic pole are matched against the semantic pole of the transient structure. If matching is successful, the remaining units are merged with the transient structure in two phases. In parsing, constructional knowledge is simply applied in the opposite direction.

values is assumed to be open-ended and dynamically changing over time. So in order to distinguish itself as a proper noun, the *Kim*-construction either needs to redundantly store its own copy of the properties of a proper noun, or it needs to "collaborate" with a *proper-noun-construction*.

How can constructions collaborate with each other? Simply put, the information that one construction adds to the transient structure may "trigger" the application of another one that can then further elaborate on that information. To understand how constructions do that, it is necessary to know how FCG applies constructional knowledge. Let's illustrate this process through a simplified *proper-noun-construction* as shown in example (13).

(13) *Proper-noun-construction*



As can be seen, the *proper-noun-construction* looks very similar to a transient structure. One difference, however, is that the construction distinguishes between so-called "conditional units" (shown above the dotted line in both poles) and "contributing units" (shown below the dotted line in both poles). Depending on the direction of processing, the conditional units of either the semantic or syntactic pole are used to check whether or not a construction might have relevant informa-

tion to contribute. If so, the construction will try to add all of its information to the transient structure.

Let's look at parsing first. Parsing is a process that goes from form to meaning, so here the conditional units of the syntactic pole of the construction act like a set of constraints that need to be satisfied before the construction is actually applied. Technically, the FCG-system will try to *match* these conditional units against the units in the current transient structure. Here, the *proper-noun-construction* requires any unit that contains the feature-value pair [SYN-CAT [LEX-CLASS *proper-noun*]]. The FCG-system indeed finds a matching unit: the *Kim-unit* that was shown in example (12). Since matching was successful, the construction will add all its information to the transient structure.

Production works entirely the same but goes in the other direction. Since production goes from meaning to form, this time the conditional units of the semantic pole must be matched against the transient structure to test whether the construction might have relevant information to contribute. This bidirectional application process of an FCG-construction is summarized in Figure 2.

In sum, by categorizing *Kim* as a proper noun, the information that was contributed to the transient structure by the *Kim-construction* has triggered the application of the *proper-noun-construction*. Both constructions thus collaborated to provide the information that in an SBCG grammar would be obtained through inheritance. But whereas inheritance provides an elegant solution for defining static constraints, the FCG approach is easier and more flexible for exploring variation and language change. Suppose, for instance, that there is a variant of the *proper-noun-construction* that states that the proper noun should be preceded by a determiner. Such variation is attested in some languages such as Dutch (including my own dialect). The two variants would both be triggered and thus compete with each other for adding their information to the transient structure.

Another important feature of FCG is that the formalism does not impose any restrictions on which set of feature-value pairs can have their own unit status or which relations there might be between units. In fact, the units offer an efficient way of accessing information that abstracts away from the kind of analysis that is assumed (e.g. phrase structure, dependency relations, and so on). As can be seen in example (13), a construction also does not use the actual name of a unit, but instead uses a variable (indicated by a question mark), so the variable *?proper-noun-unit* can be bound to any unit in the transient structure that it matches with, regardless of that unit's position in the structure of an utterance.

In other words, there is no locality constraint imposed in FCG: the unit structure allows FCG constructions to reach all information in the transient structure. Returning to the example of nonlocal dependencies, an FCG grammar does not need to encode a gap in the verb's valence list and then percolate information about

the extracted element to its filler. Instead, a verb's valence list simply "points" to the units that encapsulate information about the verb's arguments (e.g. a subject- and object-unit). Other constructions, such as argument structure and information structure constructions, can then add information to the transient structure about where those arguments should be positioned in the utterance (production) or where they can be found (parsing). Differences in word order, therefore, stem from a different combination of the same constructions rather than from the use of special filler-gap constructions. Examples (14–15) thus involve the same constructions but only differ as to which of the verb's arguments was conceptualized as the topic of the utterance:[4]

(14) [The boy $_{\text{SUBJ/TOP}}$] [hit the ball $_{\text{COMMENT}}$].

(15) [The ball $_{\text{OBJ/TOP}}$] [the boy hit $_{\text{COMMENT}}$].

3.2.2 *The design level*

Constructions can easily grow very complex and making them "work" for both parsing and production can be a daunting challenge. The FCG-system therefore also features a "design level" on top of the operational level of constructions. At the design level, linguists can use *templates* that abstract away from the many necessary details that make constructions work, so they can concentrate on what is unique to a particular construction. The definitions created with these templates are then translated into the actual constructions. Such templates are mainly used for grammar engineering purposes, but no claims are made about their relevance for linguistic theory. Example (16) shows an example of how a basic (or "skeletal") lexical entry can be built using a template.

(16)
```
(def-lex-skeleton Kim-cxn
    :meaning (== (individual ?x Kim))
    :string "Kim"
    :args (?x))
```

4. How are constructions organized?

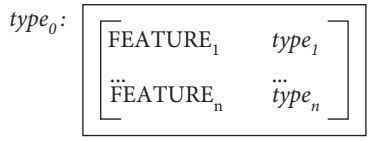
Another important feature of construction grammar is that the linguistic inventory is assumed to be structured (Croft & Cruse 2004).

4. Interested readers can inspect how the grammar works by downloading a sample grammar or checking an online demo at www.fcg-net.org/demos/ldd. The linguistic analysis of the grammar is described by van Trijp (submitted).

4.1 SBCG: A grammar signature

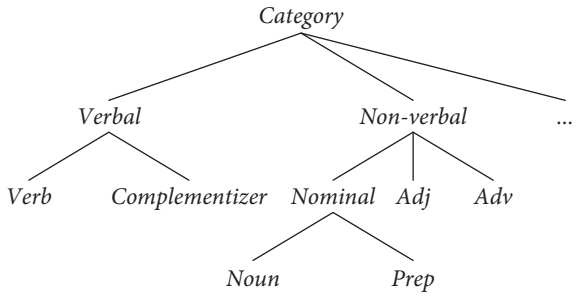
At the heart of an SBCG grammar is its *signature*, which can be considered as the grammar's ontology that specifies how grammatical descriptions need to be interpreted. First, the signature contains *type declarations* for each type of feature structure that may occur in a grammatical description. A type declaration associates a type with a set of features (i.e. its "domain"), which are typed themselves. Example (17), from Sag (2012: ex. 2), illustrates a type declaration.

(17)



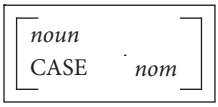
All types are organized in a multiple inheritance hierarchy. This type inheritance hierarchy replaces the earlier notion of 'constructional inheritance' in BCG. Example (18) shows the hierarchy of values for the syntactic feature CATEGORY (adopted from Sag 2012: Figure 3).

(18)



The type declarations and the type hierarchy of the signature allow to specify, for instance, that the feature CASE is only appropriate for feature structures of type *noun*, whereas the feature VERB-FORM is only appropriate for feature structures of type *verbal*. The hierarchy of types and their associated features thus support complex grammatical categories as shown in examples (19–20), while at the same time preventing ungrammatical categories as shown in example (21). All three examples are adopted from Sag (2012).

(19)



(20)

verb	
VERB-FORM	*finite*
INVERTED	+
INDEPENDENT-CLAUSE	–
AUXILIARY	+

(21)

*

noun	
VERB-FORM	*finite*
INDEPENDENT-CLAUSE	–
...	

4.2 FCG: An open-ended linguistic inventory

FCG does not impose particular requirements on the structure of the linguistic inventory, because the inventory is assumed to be open-ended and evolving at each usage event. The FCG-system thus offers linguists various ways of 'engineering' their preferred structure in order to test their hypotheses, but it also allows the inventory to change spontaneously without intervention of the experimenter. Among the possibilities are:

1. A flat list of constructions
2. Construction sets
3. Construction networks
4. Dynamic categorial inheritance.

4.2.1 *Organization for efficient processing*

The structure of the linguistic inventory is an essential key to efficient language processing. Since all constructions in FCG share the same representation, the most elementary inventory consists of an unordered list. In such an inventory, there is almost no structure so the language processor needs to cycle through the list to figure out which constructions should be applied. However, such an inventory is not efficient from a processing point of view: the processor will need to consider on average half of the amount of constructions before it can find a solution, which would soon become intractable because a language may easily contain tens of thousands of constructions.

A slightly more complex yet much more efficient way is to group the constructions in "sets" (e.g. lexico-phrasal constructions, argument structure constructions, and so on) so the experimenter can impose the order in which each set is

considered during processing (see for example Beuls 2011). The most natural way to structure the inventory, however, is to allow the FCG-system to autonomously organize the inventory in terms of construction networks (see Figure 3, taken from Wellens 2011). There can be many different kinds of network links. One kind of link captures priming effects to make processing more efficient. For instance, the application of a determiner will prioritize the application of adjectival and nominal constructions, and the combination of a determiner and a nominal construction will prime the application of an NP-construction. These priming links emerge spontaneously as a side-effect of language usage whereby the FCG-system tracks information about which constructions frequently apply together (Wellens 2011).

4.2.2 *Organization of inheritance relations*

Most construction grammarians are, however, more concerned with how knowledge is shared (or inherited) among constructions. Constructional inheritance is only implemented in FCG's design level. Here, linguists can specify inheritance relations between constructions directly in their template definitions. For instance, Steels (2011a) shows how a *color-adjective-construction* may inherit information from an *adjective-construction*. Remember, however, that templates are translated into fully instantiated constructions, which means that at the operational level of the actual constructions, inheritance relations play no role because each construction will have its own copy of the inherited information.

Categorial inheritance, on the other hand, is supported but not imposed. The FCG-system allows its users do define a category inheritance hierarchy (similar to a type hierarchy) whose information can either be precompiled into the grammar, or which can be dynamically consulted by a construction at processing time if the construction contains explicit pointers to the hierarchy.

5. How are constructions processed?

This section discusses the processing side of language usage. Since SBCG only models language competence, however, a direct comparison to FCG is impossible. Instead, I will discuss SBCG's stance on the matter and try to evaluate which issues would arise if one would try to build a computational model of processing for an SBCG grammar. As for FCG, I will discuss its actual implementation.

5.1 The performance-independence hypothesis of SBCG

SBCG makes a strong claim to psycholinguistic plausibility, and argues that a competence grammar has to be compatible with what is known about language

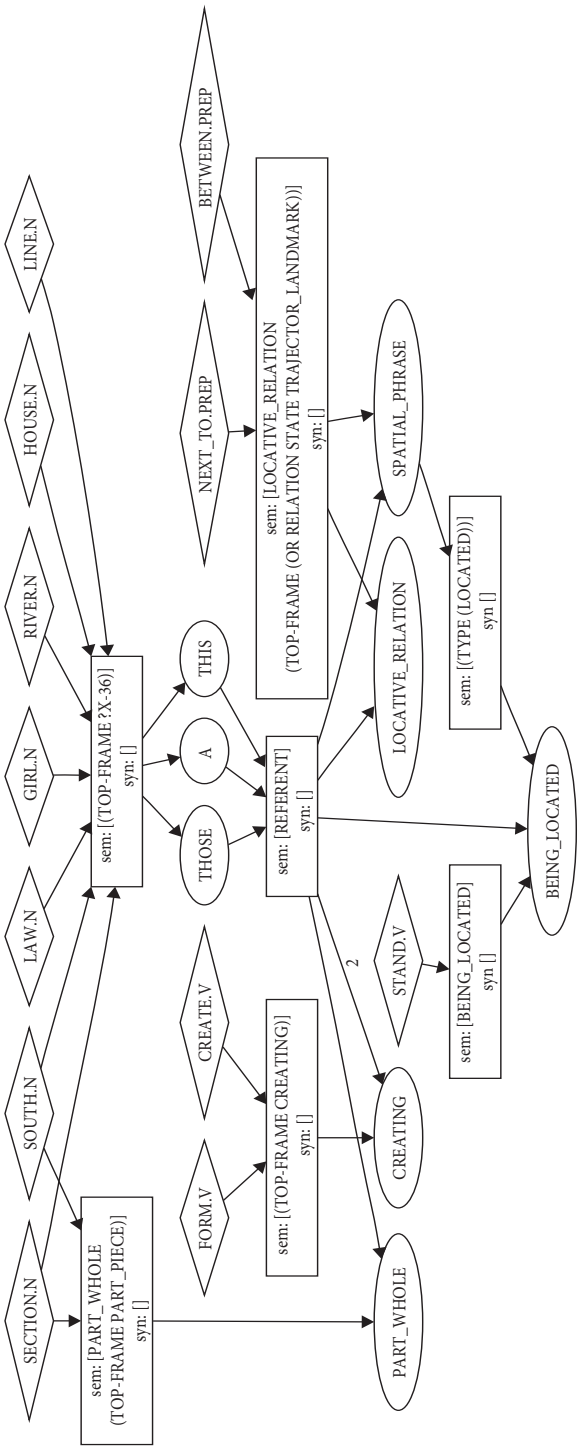


Figure 3. This partial network of constructions shows priming links for producing utterances. The FCG-system is able to build and maintain such networks based on language usage.

processing. Sag & Wasow (2011: 368) hypothesize that the best way to ensure performance-compatibility is to be performance-*independent*:

> [T]he architecture of grammar should be such that linguistic knowledge can be independently deployed in different situations. […G]iven that linguistic knowledge is process-independent, there should be no bias within a grammatical theory — whether overt or hidden, intentional or inadvertent — toward one kind of processing, rather than another.

The best way to achieve a process-neutral grammar, they argue, is to use declarative representations. However, while it is of course desirable to achieve a grammar that can be deployed in many different situations, evidence from computational complexity theory cautions against the performance-independence hypothesis.[5] Winograd (1977: 170) argues that representations are only independent of performance in the abstract case where there are no limitations on memory and processing time. The cognitive resources of real language users, however, are limited in time and space, and in cognitive science it is widely accepted that processes can only claim psychological plausibility if they remain computationally tractable (van Rooij 2008).

One way to test SBCG's performance-independence hypothesis is thus to use an SBCG grammar in a computational model of human sentence processing and demonstrate that the model is able to keep the problem of parsing and producing utterances tractable. With almost no work yet on implementing SBCG grammars (see Gromov 2010 for a first proposal), we can look at related work in HPSG to estimate how SBCG can be operationalized. Since HPSG and SBCG are constraint-based theories, a 'direct' implementation is to use a general constraint solving system such as the *Typed Feature Structure System* (Emele & Zajac 1990). As it turns out, such systems run into severe problems of efficiency and often do not even terminate their computation (Levine & Meurers 2006). Richter (2006: 126) states the problem as follows:

> [W]e cannot just write any grammar in the HPSG formalism and expect that there is a computational system that can help us effectively parse and generate sentences relative to the grammar we wrote. Closer investigation reveals that it is even difficult to determine an interesting subset of grammars for which this could always be done.

5. Manning (1995: 13) provides a straightforward counterexample from cryptography using the declarative statement "*C* is the product of the prime numbers *A* and *B*". In terms of processing, if you are given $A = 3$ and $B = 251$, it is easy to find that $C = 753$. In the other direction, when given e.g. $C = 685$, it is extremely difficult to find *A* and *B* (the answer is 5 and 137). In other words, while it is true that declarative statements abstract away from the actual processes, these statements may in fact hide computational problems that are inherently hard to solve.

Richter here points to a problem that is not unique to HPSG, but which is true for any feature structure grammar: feature structures are very expressive and flexible, but come at the cost of computational complexity (i.e. the kind of computational problem you cannot solve by simply building faster computers). Actual implementations of HPSG typically handle the problem by guiding the linguistic processor using a (rule-based) phrase structure backbone, but the disadvantage of this approach is that the "organization and formulation of the grammar is different from that of the linguistic theory" (Levine & Meurers 2006: Section 4.2.2). As a result, translating the theory "into a working computational grammar requires complex considerations" (Melnik 2007: 199) whereby the grammar engineer must strike a balance between remaining faithful to the theory and processing efficiency.

Applying all these observations to the operationalization of SBCG, we can conclude that an SBCG grammar is certainly *amenable* for computational implementation because of its formal explicitness. There are at least two computational platforms available, mostly used for implementing HPSG-based grammars, whose basic tenets are compatible with the foundations of SBCG: LKB (Copestake 2002) and TRALE (Richter 2006). However, none of these platforms support a 'direct' implementation of an SBCG grammar as a general constraint system, so SBCG's performance-independence hypothesis remains conjecture until proven otherwise.

5.2 The meta-architecture of FCG

In terms of psychological plausibility, FCG makes a much weaker claim than SBCG. The first concern of an FCG grammar is to achieve a working and bidirectional implementation, which may offer a relevant *characterization* of the functions that need to be supported by human cognition for achieving successful communication, without claiming that the brain implements those functions in the same way. This is common practice in computational linguistics. For instance, current probabilistic language models offer a good characterization of how the brain handles uncertainty of input without implying that the brain is performing sophisticated mathematical equations (Jurafsky 2003).

Processing in FCG is handled by two layers that are active at the same time (see Figure 4). One layer handles "routine processing", while a "meta-layer" monitors what happens in routine processing and intervenes when necessary.

5.2.1 *Routine processing*

The routine layer of FCG handles what is commonly understood as 'linguistic processing', so when the FCG-system is used for traditional grammar engineering purposes, the bulk of the implementation work is dedicated to the routine layer. It is important to repeat, however, that the goal of an FCG grammar is not

to license well-formed structures. The FCG-system will produce well-formed utterances only as a side-effect if the grammar adequately captures the conventions of a language. In parsing, the FCG grammar will thus also not reject ill-formed utterances, but instead continue processing in order to uncover as much meaning as possible.

FCG does not distinguish between its formalism and its implementation: the implementation *is* the formalism, so all work reported on FCG has so far been substantiated by a working grammar. As is the case for all feature structure grammars, however, this does not mean that anything goes: the expressive power of the FCG formalism requires careful considerations in order to achieve efficient processing models. The major difference between FCG and SBCG in this respect is that FCG does not assume that linguistic knowledge is independent from processing, so the burden of efficiency does not rest solely on the shoulders of the linguistic processor.

From the side of linguistic knowledge, more efficient processing is achieved by dividing an FCG construction between a semantic and a syntactic pole (see Section 3.2). By informing the processor about which of its feature-value pairs function as conditions that first need to be satisfied, the application of the whole construction is only considered in relevant contexts, which avoids a lot of unnecessary computation.[6] Additionally, while the feature-value pairs in the constructions are stated as declaratively as possible (with some procedural annotations), the FCG literature explicitly focuses on choosing processing-friendly representations (e.g. van Trijp 2011 presents an efficient representation for case syncretism).

From the side of linguistic processing, the FCG-system is able to build priming relations between constructions (see Section 4.2.1) and to keep track of various scores, such as a construction's frequency. All this data can be incorporated in FCG's search algorithm using well-known techniques for efficient processing from probabilistic language modeling (Bleys, Stadler, & De Beule 2011).

Besides traditional grammar engineering purposes, the FCG-system has also been explicitly designed to handle use cases that involve variation and language change. Such use cases typically involve experiments that operationalize a speech community as a population of artificial language users that engage with each other in communicative interactions (see Steels 2011c for some concrete examples). In such experiments, language is viewed as a complex adaptive system, similar to a complex ecosystem, and constructions can only 'survive' in a language insofar as they help language users to reach their communicative goals while minimizing

6. Even though in HPSG theory all constraints apply simultaneously, practical implementations often use explicit instructions for controlling the order in which constructions should be applied, such as delay statements. However, these are not part of the grammar.

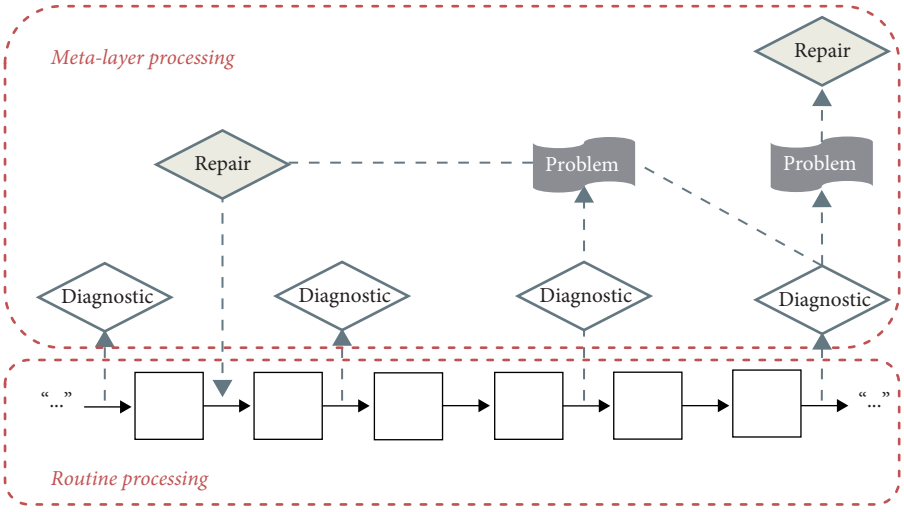


Figure 4. FCG features a meta-layer for robust and open-ended language processing. One layer takes care of routine processing, while a meta-layer monitors the routine layer and may intervene with repairs if a problem is diagnosed.

the cognitive resources required for doing so. This means that constructions are constantly competing with each other for becoming (or remaining) the dominant convention in the language. Whether or not a construction is able to propagate in the population depends on several "linguistic selection criteria", such as communicative success, cognitive effort, ease of pronunciation, acoustic distinctiveness, and so on (Steels 2011c). For example, a word form whose ending is phonologically reduced in rapid speech may become more popular than the original form (despite its strong entrenchment) if the new form requires less articulatory effort without harming communicative success (Beuls & Steels 2013).

5.2.2 *Open-Ended Language Processing*

A cognitive-functional grammar does not try to account for well-formed utterances, but rather how language users are able to express new conceptualizations in an infinite number of ways, and how they arrive at meaningful interpretations even when utterances color outside the lines of what is considered to be grammatical. Language processing is remarkably robust against ill-formed utterances, creative language use and novelty.

In order to make language processing robust and open-ended, FCG features a meta-architecture for not only representing conventional constructional knowledge, but also the strategies that language users employ for tackling communicative challenges, such as how to categorize a new word, how to express a novel meaning, and so on. These strategies are defined in FCG's meta-layer that

monitors the application of constructions and intervenes when a problem arises. More specifically, the meta-layer contains *diagnostics* for detecting problems in routine processing and *repairs* for solving those problems. Operational examples of diagnostics and repairs, including coercion and constructional learning, can be found in Steels and van Trijp (2011), Beuls, van Trijp, & Wellens (2012), and van Trijp (2012).

6. Conclusion: Two scientific models of language

The nature-nurture debate has long been considered to be the major rift between linguists, but that is actually untrue: virtually all linguists agree that humans have language-ready brains but they disagree on how much innate knowledge should be assumed. The real divide in linguistics is which scientific model is considered to be the most suited one for the study of language, and the comparison of this paper demonstrates quite clearly that SBCG and FCG adopt two different models.

6.1 Theoretical physics vs. Darwinian evolutionary theory

SBCG follows the scientific model of theoretical physics, a viewpoint that was popularized by Chomsky (1965) and adopted well-beyond generative linguistics. In theoretical physics, mathematical models and abstractions are used for explaining and predicting natural phenomena. For instance, a theoretical physicist will first study the mechanics of an ideal frictionless system before looking at how an actual ball rolls down a hill. The same equations of mechanics can then be applied to all moving objects by adding the correct parameters and quantities of friction. Likewise, a mathematical conception of grammar focuses on the abstract competence or knowledge that an idealized speaker has of his or her language, and assumes that there is a set of principles that apply to all instances. One example is the locality of SBCG constructions, which entails that all languages can best be described as constraints on local trees.

FCG, on the other hand, adopts a Darwinian model of science as found in evolutionary biology and generalizes it to cultural evolution (Croft 2000; Steels 2011c). The epistemology of a Darwinian model is quite different from that of theoretical physics: an evolutionary biologist does not assume a set of equations that can be applied to all phenomena, but operates in a general framework that helps to find explanations. For each case, the specific details need to be worked out. The FCG-formalism therefore makes as little assumptions as possible about how the grammar of a language should be operationalized: it is agnostic as to what kind of structural relations exist (e.g. phrase structures, dependency relations, or others),

Table 1. This Table summarizes this paper's comparison between SBCG and FCG.

	SBCG	**FCG**
Scientific model	Theoretical physics (abstract calculus)	Evolutionary theory (complex adaptive system)
Linguistic approach	Generative (competence model)	Cognitive-functional (parsing and production)
Formalization	Mathematical (amenable for implementation)	Computational (implemented)
Constructions	Static type constraints	Dynamic mappings
Constructicon	Signature and grammar	Open-ended inventory
Processing	Assumption of processing-independence	Bidirectional processing model

how the linguistic inventory should be organized or which learning mechanisms are most appropriate.

The consequences of the different scientific models of SBCG and FCG are summarized in Table 1. SBCG is a generative linguistic theory that aims to account for all well-formed utterances of a language. It makes an abstraction of a language user's linguistic knowledge in the form of a competence model, which is assumed to be completely independent of a processing model. SBCG works towards a mathematical formalism with the same precision as the HPSG formalism, which is amenable for computational implementation. SBCG constructions are defined as static constraints on the local tree configurations of a modern CFG.

FCG takes a cognitive-functional or usage-based approach that models how speakers express conceptualizations through language and how listeners parse utterances into interpretable meanings. In FCG, competence and performance are two sides of the same coin, and the two cannot be sharply separated from each other. All work in FCG therefore operationalizes computational processing models capable of parsing and producing constructional knowledge. Constructions are defined as dynamic mappings between meaning and form.

6.2 Conclusion

Construction grammar comprises a host of researchers who operate under many different assumptions. Many of those researchers wish to be more explicit and precise in their analyses through rigorous formalization. This paper thus offers a broad comparison between SBCG and FCG, two formal approaches to construction grammar that are good candidates for this demand.

This comparison has shown that SBCG and FCG are almost at opposite ends of the construction grammar spectrum, which is reflected in the particular design choices of their formalisms. Given that language can be studied from many

different perspectives, it is highly beneficial that different formalizations are currently being explored in a spirit of tolerance for those different design choices. The goal of this paper was, therefore, not to argue for which approach is 'best' but, rather, to make the far-reaching theoretical divergences between SBCG and FCG explicit in order to help other linguists to decide which formalization is most suited to their needs.

References

Beuls, K. (2011). Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 237–264). Amsterdam: John Benjamins.

Beuls, K., & Steels, L. (2013). Agent-based models of strategies for the emergence and evolution of grammatical agreement. *PLoS ONE*, *8*(3): e58960.

Beuls, K., van Trijp, R., & Wellens, P. (2012). Diagnostics and repairs in Fluid Construction Grammar. In L. Steels & M. Hild (Eds.), *Language grounding in robots* (pp. 215–234). New York, NY: Springer Verlag.

Blevins, J., & Sag, I. A. (2013). Phrase structure grammar. In M. den Dikken (Ed.), *Cambridge handbook of generative syntax* (pp. 202–225). Cambridge: Cambridge University Press.

Bleys, J., Stadler, K., & De Beule, J. (2011). Search in linguistic processing. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 149–179). Amsterdam: John Benjamins.

Boas, H. C., & Sag, I. A. (Eds.). (2012). *Sign-based construction grammar.* Stanford, CA: CSLI Publications.

Chang, N., De Beule, J., & Micelli, V. (2012). Computational construction grammar: Comparing ECG and FCG. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar* (pp. 259–288). Heidelberg: Springer Verlag.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*, 113–124.

Chomsky, N. (1957). *Syntactic structures*. The Hague/Paris: Mouton.

Chomsky, N. (1958). A review of B. F. Skinner's *Verbal behavior*. *Language*, *35*(1), 26–58.

Chomsky, N. (1965). *Aspects of the theory of syntax.* Cambridge, MA: MIT Press.

Copestake, A. (2002). *Implementing typed feature structure grammars.* Stanford, CA: CSLI Publications.

Croft, W. (2000). *Explaining language change: An evolutionary approach*. Harlow, Essex: Longman.

Croft, W. (2005). Logical and typological arguments for radical construction grammar. In J.-O. Östman & M. Fried (Eds.), *Construction grammars: Cognitive grounding and theoretical extensions* (pp. 273–314). Amsterdam: John Benjamins.

Croft, W., & Cruse, D. A. (2004). *Cognitive linguistics*. Cambridge: Cambridge University Press.

Dik, S. C. (1978). *Functional grammar*. Amsterdam: North-Holland.

Dresher, F., & Hornstein, N. (1976). On some supposed contributions of artificial intelligence to the scientific study of language. *Cognition*, *4*(4), 321–398.

Emele, M. C., & Zajac, R. (1990). Typed unification grammars. In H. Karlgren (Ed.), *Proceedings of the 13th international conference on computational linguistics* (pp. 293–298). Helsinki: Association for Computational Linguistics.

Fillmore, C. J. (1968). The case for case. In E. Bach & R. Harms (Eds.), *Universals in linguistic theory* (pp. 1–88). New York, NY: Holt, Rhinehart and Winston.

Fillmore, C. J. (1988). The mechanisms of "construction grammar". In *Proceedings of the fourteenth annual meeting of the Berkeley Linguistics Society* (pp. 35–55). Berkeley, CA: Berkeley Linguistics Society.

Fried, M. (2009). Construction grammar as a tool for diachronic analysis. *Constructions and Frames*, *1*(2), 261–290.

Gazdar, G. (1981). Unbounded dependencies and coordinate structure. *Linguistic Inquiry*, *12*, 155–184.

Gazdar, G., Klein, G., Pullum, G., & Sag, I. A. (1985). *Generalized phrase structure grammar*. Oxford/Cambridge, Ma: Blackwell Publishing/Harvard University Press.

Geeraerts, D. (2008). Cognitive linguistics. In H. Momma & M. Matto (Eds.), *Blackwell companion to the history of the English language* (pp. 618–629). Oxford: Wiley-Blackwell.

Goldberg, A. E. (1995). *Constructions: A construction grammar approach to argument structure.* Chicago, IL: Chicago University Press.

Gromov, P. (2010). *Implementing sign-based construction grammar with TRALE*. Master's thesis, University of Essex.

Halliday, M.A. (1973). *Explorations in the functions of language.* London: Edward Arnold.

Jurafsky, D. (2003). Probabilistic modeling in psycholinguistics: linguistic comprehension and production. In R. Bod, J. Hay, & S. Jannedy (Eds.), *Probabilistic linguistics* (pp. 39–96). Cambridge, MA: MIT Press.

Kaplan, R., & Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan (Ed.), *The mental representation of grammatical relations* (pp. 173–281). Cambridge, MA: MIT Press.

Kay, M. (1979). Functional grammar. In *Proceedings of the fifth annual meeting of the Berkeley Linguistics Society* (pp. 142–158). Berkeley, CA: Berkeley Linguistics Society.

Kay, P., & Fillmore, C. J. (1999). Grammatical constructions and linguistics generalizations: The *what's x doing y?* construction. *Language*, *75*(1), 1–33.

Lakoff, G., & Thompson, H. (1975). Introducing cognitive grammar. In *Proceedings of the first annual meeting of the Berkeley Linguistics Society* (pp. 295–313). Berkeley, CA: Berkeley Linguistics Society.

Langacker, R. W. (1987). *Foundations of cognitive grammar. Volum 1: Theoretical prerequisites.* Stanford, CA: Stanford University Press.

Levine, R. D., & Meurers, W. D. (2006). Head-driven phrase structure grammar: Linguistic approach, formal foundations, and computational realization. In K. Brown (Ed.), *Encyclopedia of language and linguistics* (2nd ed.) (pp. 237–252). Oxford: Elsevier.

Manning, C. D. (1995). Dissociating functor-argument structure from surface phrase structure: The relationship of HPSG Order Domains to LFG. Ms., Carnegie Mellon University. Retrieved April 19, 2013 from http://nlp.stanford.edu/~manning/papers/hpsglfg1.pdf

Melnik, N. (2007). From "hand-written" to computationally implemented HPSG theories. *Language and Computation*, *5*(2), 199–236.

Michaelis, L. (2013). Sign-based construction grammar. In T. Hoffman & G. Trousdale (Eds.), *The Oxford handbook of construction grammar* (pp. 133–152). Oxford: Oxford University Press.

Müller, S. (2010). *Grammatiktheorie* (2nd rev. ed.). Tübingen: Stauffenburg Verlag.
Ohori, T. (2005). Construction grammar as a conceptual framework for linguistic typology: A case from reference tracking. In M. Fried & H. C. Boas (Eds.), *Grammatical constructions: Back to the roots* (pp. 215–237). Amsterdam: John Benjamins.
Östman, J.-O., & Fried, M. (2004). Historical and intellectual background of construction grammar. In M. Fried & J.-O. Östman (Eds.), *Construction grammar in a cross-language perspective* (pp. 1–10). Amsterdam: John Benjamins.
Pollard, C. (1984). *Generalized phrase structure grammars, head grammars, and natural language*. Unpublished Doctoral Dissertation, Stanford University, Stanford, CA.
Pollard, C., & Sag, I. A. (1994). *Head-driven phrase structure grammar*. Chicago, IL/Stanford, CA: University of Chicago Press/CSLI Publications.
Pullum, G. K., & Scholz, B. C. (2001). On the distinction between model-theoretic and generative-enumerative frameworks. In P. de Groote, G. Morrill, & C. Retoré (Eds.), *Logical aspects of computational linguistics: 4th international conference* (pp. 17–43). Berlin: Springer Verlag.
Reape, M. (1994). Domain union and word order variation in German. In J. Nerbonne, K. Netter, & C. Pollard (Eds.), *German in head-driven phrase structure grammar* (pp. 151–197). Stanford, CA: CSLI Publications.
Richter, F. (2006). *A web-based course in grammar formalisms and parsing.* Retrieved November 24, 2011 from http://milca.sfs.uni-tuebingen.de/A4/Course/PDF/gramandpars.pdf
Russel, S. J., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (2nd ed.). Upper Saddle River, NJ: Prentice Hall/Pearson Education.
Sag, I. A. (2010). English filler-gap constructions. *Language*, *86*(3), 486–545.
Sag, I. A. (2012). Sign-based construction grammar: An informal synopsis. In H. C. Boas & I. A. Sag (Eds.), *Sign-based construction grammar* (pp. 69–202). Stanford, CA: CSLI Publications.
Sag, I. A., Kaplan, R., Karttunen, L., Kay, M., Pollard, C., Shieber, S., & Zaenen, A. (1986). Unification and grammatical theory. In *Proceedings of the fifth annual meeting of the West Coast conference on formal linguistics* (pp. 238–254). Stanford, CA: CSLI Publications.
Sag, I. A., & Wasow, T. (2011). Performance-compatible competence grammar. In R. D. Borsley & K. Börjars (Eds.), *Non-transformational syntax: Formal and explicit models of grammar* (pp. 359–377). Wiley-Blackwell.
Shieber, S. (1986). *An introduction to unification-based approaches to grammar* (Vol. 4). Stanford, CA: CSLI Publications.
Steels, L. (1978a). *Introducing conceptual grammar* (Tech. Rep. No. WP-176). MIT.
Steels, L. (1978b). *Some examples of conceptual grammar* (Tech. Rep. WP-177). MIT.
Steels, L. (1990). Components of expertise. *AI Magazine*, *11*(2), 28–49.
Steels, L. (1995). A self-organizing spatial vocabulary. *Artificial Life Journal*, *2*(3), 319–332.
Steels, L. (1998). The origins of syntax in visually grounded robotic agents. *Artificial Intelligence*, *103*(1–2), 133–156.
Steels, L. (2000). Language as a complex adaptive system. In M. Schoenauer et al. (Eds.), *Proceedings of the 6th international conference on parallel problem solving from nature* (pp. 17–26). Berlin: Springer Verlag.
Steels, L. (2011a). A design pattern for phrasal constructions. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 71–114). Amsterdam: John Benjamins.
Steels, L. (Ed.). (2011b). *Design patterns in Fluid Construction Grammar.* Amsterdam: John Benjamins.

Steels, L. (2011c). Modeling the cultural evolution of language. *Physics of Life Reviews*, *8*(4), 339–356.
Steels, L. (Ed.). (2012). *Computational issues in Fluid Construction Grammar*. Heidelberg: Springer Verlag.
Steels, L., & Brooks, R. (Eds.). (1995). *The artificial life route to artificial intelligence: Building embodied, situated agents*. New Jersey: Lawrence Erlbaum.
Tesnière, L. (1959). *Éléments de syntaxe structurale.* Paris: Klincksieck.
van Rooij, I. (2008). The tractable cognition thesis. *Cognitive Science*, *32*, 939–984.
van Trijp, R. (2011). Feature matrices and agreement: A case study for German case. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 205–235). Amsterdam: John Benjamins.
van Trijp, R. (2012). A reflective architecture for robust language processing and learning. In L. Steels (Ed.), *Computational issues in Fluid Construction Grammar* (pp. 51–74). Heidelberg: Springer Verlag.
van Trijp, R. (submitted). Long-distance dependencies without filler-gaps: A cognitive-functional alternative in Fluid Construction Grammar.
Wellens, P. (2011). Organizing constructions in networks. In L. Steels (Ed.), *Design patterns in Fluid Construction Grammar* (pp. 181–201). Amsterdam: John Benjamins.
Winograd, T. (1977). On some supposed suppositions of generative linguistics about the scientific study of language. A response to Dresher and Hornstein's *On some supposed contributions of artificial intelligence to the scientific study of language*. *Cognition*, *5*, 151–179.

Author's address

Sony Computer Science Laboratory Paris
6 Rue Amyot
75005 Paris (France)

remi@csl.sony.fr