Fluid Construction Grammar: The New Kid on the Block

Remi van Trijp[1], **Luc Steels**[1,2], **Katrien Beuls**[3], **Pieter Wellens**[3]

[1]Sony Computer Science Laboratory Paris
6 Rue Amyot
75005 Paris (France)
remi@csl.sony.fr

[2]ICREA Institute for Evolutionary Biology (UPF-CSIC)
PRBB, Dr Aiguidar 88
08003 Barcelona (Spain)
steels@ai.vub.ac.be

[3] VUB AI Lab
Pleinlaan 2
1050 Brussels (Belgium)
katrien|pieter@ai.vub.ac.be

Abstract

Cognitive linguistics has reached a stage of maturity where many researchers are looking for an explicit formal grounding of their work. Unfortunately, most current models of deep language processing incorporate assumptions from generative grammar that are at odds with the cognitive movement in linguistics. This demonstration shows how Fluid Construction Grammar (FCG), a fully operational and bidirectional unification-based grammar formalism, caters for this increasing demand. FCG features many of the tools that were pioneered in computational linguistics in the 70s-90s, but combines them in an innovative way. This demonstration highlights the main differences between FCG and related formalisms.

1 Introduction

The "cognitive linguistics enterprise" (Evans et al., 2007) is a rapidly expanding research discipline that has so far avoided rigorous formalizations. This choice was wholly justified in the 70s-90s when the foundations of this scientific movement were laid (Rosch, 1975; Lakoff, 1987; Langacker, 1987), and it remained so during the past two decades while the enterprise worked on getting its facts straight through empirical studies in various subfields such as language acquisition (Tomasello, 2003; Goldberg et al., 2004; Lieven, 2009), language change and grammaticalization (Heine et al., 1991; Barðdal and Chelliah, 2009), and corpus research (Boas, 2003; Stefanowitsch and Gries, 2003). However, with numerous textbooks on the market (Lee, 2001; Croft and Cruse, 2004; Evans and Green, 2006), cognitive linguistics has by now established itself as a serious branch in the study of language, and many cognitive linguists are looking for ways of explicitly formalizing their work through computational models (McClelland, 2009).

Unfortunately, it turns out to be very difficult to adequately formalize a cognitive linguistic approach to grammar (or "construction grammar") using the tools for precision-grammars developed in the 70s-90s such as unification (Kay, 1979; Carpenter, 1992), because these tools are typically incorporated in a generative grammar (such as HPSG; Ginzburg and Sag, 2000) whose assumptions are incompatible with the foundations of construction grammar. First, cognitive linguistics blurs the distinction between 'competence' and 'performance', which means giving up the sharp distinction between declarative and procedural representations. Next, construction grammarians argue for a usage-based approach (Langacker, 2000), so the constraints on features may change and features may emerge or disappear from a grammar at any given time.

This demonstration introduces Fluid Construction Grammar (FCG; Steels, 2011, 2012a), a novel unification-based grammar formalism that addresses these issues, and which is available as open-source software at www.fcg-net.org. After more than a decade of development, FCG is now ready to handle sophisticated linguistic issues. FCG revisits many of the technologies developed by computational linguists and introduces several key innovations that are of interest to anyone working on deep language processing. The demonstration illustrates these innovations through FCG's interactive web interface.

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 63–68,
Avignon, France, April 23 - 27 2012.

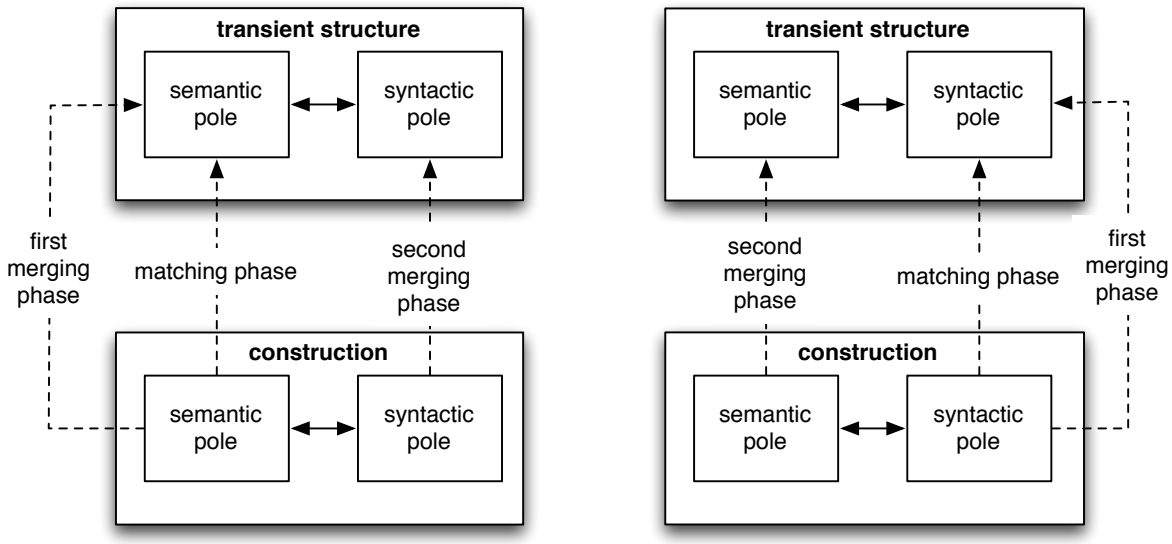

Figure 1: FCG allows the implementation of efficient and strongly reversible grammars. Left: In production, conditional units of the semantic pole of a construction are matched against a transient structure, before additional semantic constraints and the syntactic pole are merged with the structure. Right: In parsing, the same algorithm applies but in the opposite direction.

2 Strong and Efficient Reversibility

Reversible or bidirectional grammar formalisms can achieve both production and parsing (Strzalkowski, 1994). Several platforms, such as the LKB (Copestake, 2002), already achieve bidirectionality, but they do so through separate algorithms for parsing and production (mainly for efficiency reasons). One problem with this approach is that there may be a loss of coherence in grammar engineering. For instance, the LKB parser can handle a wider variety of structures than its generator.

FCG uses one core engine that handles both parsing and production with a single linguistic inventory (see Figure 1). When processing, the FCG-system builds a *transient structure* that contains all the information concerning the utterance that the system has to parse or produce, divided into a semantic and syntactic pole (both of whom are feature structures). Grammar rules or "constructions" are coupled feature structures as well and thus contain a semantic and syntactic pole.

When applying constructions, the FCG-system goes through three phases. In production, FCG first *matches* all feature-value pairs of the semantic pole of a construction with the semantic pole of the transient structure, except fv-pairs that are marked for being attributed by the construction (De Beule and Steels, 2005). Matching is a more strict form of unification that resembles a subsumption test (see Steels and De Beule, 2006). If matching is successful, all the marked fv-pairs of the semantic pole are merged with the transient structure in a first merge phase, after which the whole syntactic pole is merged in a second phase. FCG-merge is equivalent to "unification" in other formalisms. The same three-phase algorithm is applied in parsing as well, but this time in the opposite direction: if the syntactic pole of the construction matches with the transient structure, the attributable syntactic fv-pairs and the semantic pole are merged.

3 WYSIWYG Grammar Engineering

Most unification grammars use non-directional linguistic representations that are designed to be independent of any model of processing (Sag and Wasow, 2011). Whereas this may be desirable from a 'mathematical' point-of-view, it puts the burden of efficient processing on the shoulders of computational linguists, who have to find a balance between faithfulness to the hand-written theory and computational efficiency (Melnik, 2005). For instance, there is no HPSG implementation, but rather several platforms that support the implementation of 'HPSG-like' grammars: ALE (Carpenter and Penn, 1995), ALEP (Schmidt et al., 1996), CUF (Dörre and Dorna,

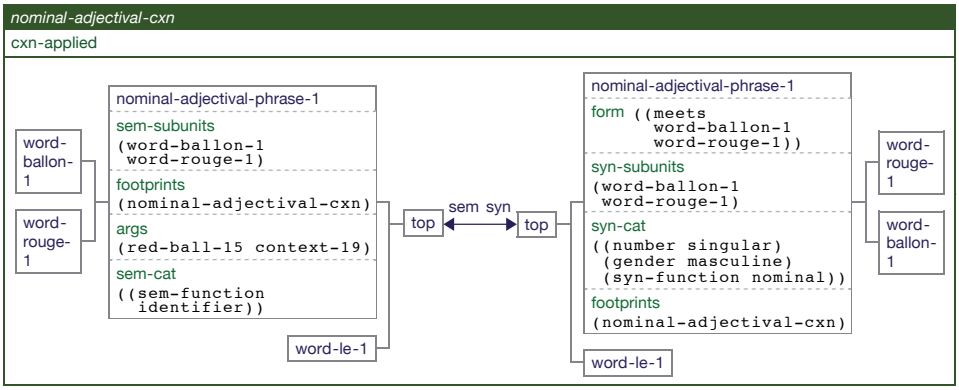

Figure 2: FCG comes equipped with an interactive web interface for inspecting the linguistic inventory, construction application and search. This Figure shows an example construction where two units are opened up for closer inspection of their feature structures.

1993), LIGHT (Ciortuz, 2002), LKB (Copestake, 2002), ProFIT (Erbach, 1995), TDL (Krieger and Schäfer, 1994), TFS (Emele, 1994), and others (see Bolc et al., 1996, for a survey). Unfortunately, the optimizations and technologies developed within these platforms are often considered by theoretical linguists as engineering solutions rather than scientific contributions.

FCG, on the other hand, adheres to the cognitive linguistics assumption that linguistic performance is equally important as linguistic competence, hence processing becomes a central notion in the formalism. FCG representations therefore offer a 'what you see is what you get' approach to grammar engineering where the representations have a direct impact on processing and vice versa. For instance, a construction's division between a semantic and syntactic pole is informative with respect to how the construction is applied.

Some grammarians may object that this design choice forces linguists to worry about processing, but that is entirely the point. It has already been demonstrated in other unification-based formalisms that different grammar representations have a significant impact on processing efficiency (Flickinger, 2000). Moreover, FCG-style representations can be directly implemented and tested without having to compromise on either faithfulness to a theory or computational efficiency.

Since writing grammars is highly complex, however, FCG also features a 'design level' on top of its operational level (Steels, 2012b). On this level, grammar engineers can use templates that build detailed constructions. The demonstration shows how to write a grammar in FCG, switching between its design level, its operational level and its interactive web interface (see Figure 2). The web interface allows FCG-users to inspect the linguistic inventory, the search tree in processing, and so on.

4 Robustness and Learning

Unification-based grammars have the reputation of being brittle when it comes to processing novelty or ungrammatical utterances (Tomuro, 1999). Since cognitive linguistics adheres to a usage-based view on language (Langacker, 2000), however, an adequate formalization must be robust and open-ended.

A first requirement is that there can be different degrees of 'entrenchment' in the grammar: while some features might still be emergent, others are already part of well-conventionalized linguistic patterns. Moreover, new features and constructions may appear (or disappear) from a grammar at any given time. These requirements are hard to reconcile with the *type hierarchy* approach of other formalisms, so FCG does not implement *typed* feature structures. The demonstration shows how FCG can nevertheless prevent over-licensing of linguistic structures through its matching phase and how it captures generalizations through its templates – two benefits typically associated with type hierarchies.

Secondly, FCG renders linguistic processing fluid and robust through a meta-level architecture, which consists of two layers of processing, as shown in Figure 3 (Beuls et al., 2012). There is a routine layer in which constructional processing takes place. At the same time, a meta-layer

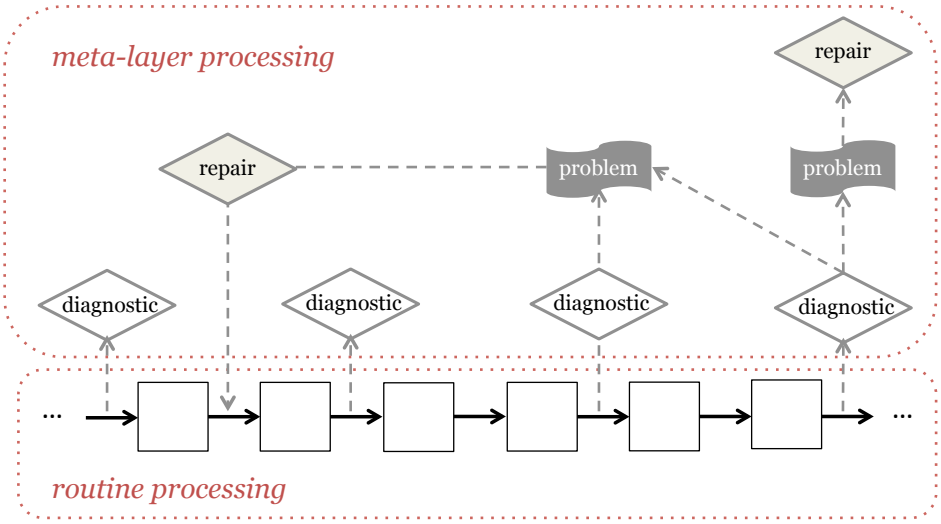

Figure 3: There are two layers of processing in FCG. On the routine level, constructional processing takes place. At the same time, a meta-layer of diagnostics and repairs try to detect and solve problems that occur in the routine layer.

is active that runs *diagnostics* for detecting problems in routine processing, and *repairs* for solving those problems. The demonstration shows how the meta-layer is used for solving common problems such as missing lexical entries and coercion (Steels and van Trijp, 2011), and how its architecture offers a uniform way of implementing the various solutions for robustness already pioneered in the aforementioned grammar platforms.

5 Efficiency

Unification is computationally expensive, and many technical solutions have been proposed for efficient processing of rich and expressive feature structures (Tomuro, 1999; Flickinger, 2000; Callmeier, 2001). In FCG, however, research on efficiency takes a different dimension because performance is considered to be an integral part of the linguistic theory that needs to be operationalized. The demonstration allows conference participants to inspect the following research results on the interplay between grammar and efficiency:

- In line with construction grammar, there is no distinction between the lexicon and the grammar. Based on language usage, the linguistic inventory can nevertheless organize itself in the form of *dependency networks* that regulate which construction should be considered when in processing (Wellens and De Beule, 2010; Wellens, 2011).
- There is abundant psycholinguistic evidence that language usage contains many ready-made language structures. FCG incorporates a chunking mechanism that is able to create such canned phrases for faster processing (Stadler, 2012).
- Morphological paradigms, such as the German case system, can be represented in the form of 'feature matrices', which reduce syntactic and semantic ambiguity and hence speed up processing efficiency and reliability (van Trijp, 2011).
- Many linguistic domains, such as spatial language, are known for their high degree of polysemy. By distinguishing between actual and potential values, such polysemous structures can be processed smoothly (Spranger and Loetzsch, 2011).

6 Conclusion

With many well-developed unification-based grammar formalisms available to the community, one might wonder whether any 'new kid on the block' can still claim relevance today. With this demonstration, we hope to show that Fluid Construction Grammar allows grammar engineers to unchart new territory, most notably in the relation between linguistic competence and performance, and in modeling usage-based approaches to language.

References

Johanna Barðdal and Shobhana Chelliah, editors. *The Role of Semantic, Pragmatic and Discourse Factors in the Development of Case*. John Benjamins, Amsterdam, 2009.

Katrien Beuls, Remi van Trijp, and Pieter Wellens. Diagnostics and repairs in Fluid Construction Grammar. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin, 2012.

Hans C. Boas. *A Constructional Approach to Resultatives*. Stanford Monograph in Linguistics. CSLI, Stanford, 2003.

Leonard Bolc, Krzysztof Czuba, Anna Kupść, Malgorzata Marciniak, Agnieszka Mykowiecka, and Adam Przepiórkowski. A survey of systems for implementing HPSG grammars. Research Report 814 of IPI PAN (Institute of Computer Science, Polish Academy of Sciences), 1996.

Ulrich Callmeier. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, 2001.

Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge UP, Cambridge, 1992.

Bob Carpenter and Gerald Penn. *The Attribute Logic Engine (Version 2.0.1)*. Pittsburgh, 1995.

Liviu Ciortuz. LIGHT – a constraint language and compiler system for typed-unification grammars. In *Proceedings of The 25th German Conferences on Artificial Intelligence (KI 2002)*, volume 2479 of *LNAI*, pages 3–17, Berlin, 2002. Springer-Verlag.

Ann Copestake. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, 2002.

William Croft and D. Alan Cruse. *Cognitive Linguistics*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, 2004.

J. De Beule and L. Steels. Hierarchy in fluid construction grammar. In U. Furbach, editor, *Proceedings of the 28th Annual German Conference on Artificial Intelligence*, volume 3698 of *Lecture Notes in Artificial Intelligence*, pages 1–15, Berlin, Germany, 2005. Springer Verlag.

Jochen Dörre and Michael Dorna. CUF – a formalism for linguistic knowledge representation. In Jochen Dörre, editor, *Computational Aspects of Constraint Based Linguistic Descriptions*, volume I, pages 1–22. DYANA-2 Project, Amsterdam, 1993.

Martin C. Emele. The typed feature structure representation formalism. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, Ikoma, Nara, 1994.

Gregor Erbach. ProFIT: Prolog with features, inheritance and templates. In *Proceedings of EACL-95*, 1995.

Vyvyan Evans and Melanie Green. *Cognitive Linguistics: An Introduction*. Lawrence Erlbaum Associates / Edinburgh University Press, Hillsdale, NJ/Edinburgh, 2006.

Vyvyan Evans, Benjamin K. Bergen, and Jörg Zinken. The cognitive linguistics enterprise: An overview. In V. Evans, B.K. Bergen, and J. Zinken, editors, *The Cognitive Linguistics Reader*. Equinox Publishing, London, 2007.

Daniel P. Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000.

Jonathan Ginzburg and Ivan A. Sag. *Interrogative Investigations: the Form, the Meaning, and Use of English Interrogatives*. CSLI Publications, Stanford, 2000.

Adele E. Goldberg, Devin M. Casenhiser, and Nitya Sethuraman. Learning argument structure generalizations. *Cognitive Linguistics*, 15 (3):289–316, 2004.

Bernd Heine, Ulrike Claudi, and Friederike Hünnemeyer. *Grammaticalization: A Conceptual Framework*. University of Chicago Press, Chicago, 1991.

Martin Kay. Functional grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society*, pages 142–158. Berkeley Linguistics Society, 1979.

Hans-Ulrich Krieger and Ulrich Schäfer. TDL – a type description language for HPSG. part 1: Overview. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 893–899, Kyoto, 1994.

George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press, Chicago, 1987.

Ronald W. Langacker. *Foundations of Cognitive Grammar: Theoretical Prerequisites*. Stanford University Press, Stanford, 1987.

Ronald W. Langacker. A dynamic usage-based model. In Michael Barlow and Suzanne Kemmer, editors, *Usage-Based Models of Language*, pages 1–63. Chicago University Press, Chicago, 2000.

David Lee. *Cognitive Linguistics: An Introduction*. Oxford University Press, Oxford, 2001.

Elena Lieven. Developing constructions. *Cognitive Linguistics*, 20(1):191–199, 2009.

James L. McClelland. The place of modeling in cognitive science. *Topics in Cognitive Science*, 1:11–38, 2009.

Nurit Melnik. From "hand-written" to computationally implemented HPSG theories. In Stefan Müller, editor, *Proceedings of the HPSG05 Conference*, Stanford, 2005. CSLI Publications.

Eleanor Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104:192–233, 1975.

Ivan A. Sag and Thomas Wasow. Performance-compatible competence grammar. In Robert D. Borsley and Kersti Börjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 359–377. Wiley-Blackwell, 2011.

Paul Schmidt, Sibylle Rieder, Axel Theofilidis, and Thierry Declerck. Lean formalisms, linguistic theory, and applications. grammar development in ALEP. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 286–291, Copenhagen, 1996.

Michael Spranger and Martin Loetzsch. Syntactic indeterminacy and semantic ambiguity: A case study for German spatial phrases. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam, 2011.

Kevin Stadler. Chunking constructions. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin, 2012.

Luc Steels, editor. *Design Patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam, 2011.

Luc Steels, editor. *Computational Issues in Fluid Construction Grammar*. Springer, Berlin, 2012a.

Luc Steels. Design methods for Fluid Construction Grammar. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin, 2012b.

Luc Steels and Joachim De Beule. Unify and merge in Fluid Construction Grammar. In P. Vogt, Y. Sugita, E. Tuci, and C. Nehaniv, editors, *Symbol Grounding and Beyond.*, LNAI 4211, pages 197–223, Berlin, 2006. Springer.

Luc Steels and Remi van Trijp. How to make construction grammars fluid and robust. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 301–330. John Benjamins, Amsterdam, 2011.

Anatol Stefanowitsch and Stefan Th. Gries. Collostructions: Investigating the interaction of words and constructions. *International Journal of Corpus Linguistics*, 2(8):209–243, 2003.

Tomek Strzalkowski, editor. *Reversible Grammar in Natural Language Processing*. Kluwer Academic Publishers, Boston, 1994.

Michael Tomasello. *Constructing a Language. A Usage Based Theory of Language Acquisition*. Harvard University Press, 2003.

Noriko Tomuro. *Left-Corner Parsing Algorithm for Unification Grammars*. PhD thesis, DePaul University, Chicago, 1999.

Remi van Trijp. Feature matrices and agreement: A case study for German case. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 205–236. John Benjamins, Amsterdam, 2011.

Pieter Wellens. Organizing constructions in networks. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam, 2011.

Pieter Wellens and Joachim De Beule. Priming through constructional dependencies: A case study in Fluid Construction Grammar. In A. Smith, M. Schouwstra, Bart de Boer, and K. Smith, editors, *The Evolution of Language (EVOLANG8)*, pages 344–351, Singapore, 2010. World Scientific.