Fluid Construction Grammar for Historical and Evolutionary Linguistics

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

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

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

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

Abstract

Fluid Construction Grammar (FCG) is an open-source computational grammar formalism that is becoming increasingly popular for studying the history and evolution of language. This demonstration shows how FCG can be used to operationalise the cultural processes and cognitive mechanisms that underly language evolution and change.

1 Introduction

Historical linguistics has been radically transformed over the past two decades by the advent of corpus-based approaches. Ever increasing datasets, both in size and richness of annotation, are becoming available (Yuri et al., 2012; Davies, 2011), and linguists now have more powerful tools at their disposal for uncovering *which* changes have taken place. In this demonstration, we present Fluid Construction Grammar (Steels, 2011, FCG), an open-source grammar formalism that makes it possible to also address the question of *how* these changes happened by uncovering the cognitive mechanisms and cultural processes that drive language evolution.

FCG combines the expressive power of feature structures and unification with the adaptivity and robustnes of machine learners. In sum, FCG aims to be an open instrument for developing robust and open-ended models of language processing that can be used for both parsing and production. FCG can be downloaded at `http://www.fcg-net.org`.

2 Design Philosophy

Fluid Construction Grammar is rooted in a *cognitive-functional* approach to language, which is quite different from a *generative* grammar such as HPSG (Pollard and Sag, 1994). A generative grammar is a model of language competence that licenses well-formed structures and rejects ill-formed utterances. Such grammars often decide on the well- or ill-formedness of utterances by using a strong type system that defines a set of features and possible values for those features. The burden of efficient and robust language processing with a generative grammar largely rests on the shoulders of the language processor.

A cognitive-functional grammar, on the other hand, functions more like a transducer between meaning and form. In parsing, such a grammar tries to uncover as much meaning as possible from a given utterance rather than deciding on its grammaticality. In the other direction, the grammar tries to produce intelligible utterances, which are well-formed as a side-effect if the grammar adequately captures the conventions of a particular language. A cognitive-functional grammar can best be implemented without a strong type system because the set of possible features and values for them is assumed to be open-ended. Efficient and robust language processing also becomes a joint responsibility of the grammar and the linguistic processor.

3 Reversible Language Processing

As a construction grammar, FCG represents *all* linguistic knowledge as pairings of function and form (called constructions). This means that any linguistic item, be it a concrete lexical item (see Figure 1) or a schematic construction, shares the same fundamental representation in FCG.

Each construction consists of two poles (a semantic/functional one and a syntactic/form one), each represented as a feature structure. By using a separate semantic and syntactic pole, FCG allows the same construction to be efficiently parsed and produced by the same processing engine by simply changing the direction of application.

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 127–132,
Sofia, Bulgaria, August 4-9 2013.

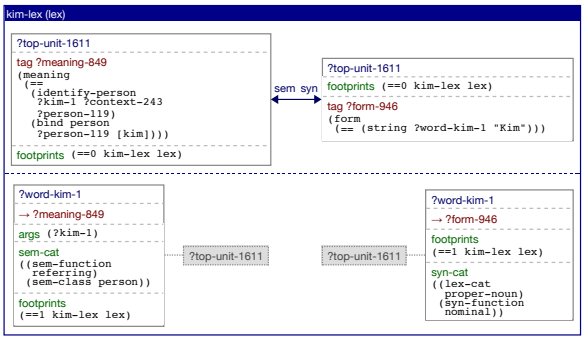

Figure 1: Lexical construction for the proper noun "Kim" as shown in the FCG web interface. All constructions are mappings between semantic (left) and syntactic feature structures (right).

FCG processing uses two different kinds of unification called *match* and *merge*. The match phase is a conditional phase which checks for applicability of the construction. The merge operation most closely resembles classical (yet untyped) unification. In production (i.e. going from meaning to form), the processor will consider a construction's semantic pole as a set of conditions that need to be satisfied, and the syntactic pole as additional information that can be contributed by the construction. In parsing (i.e. going from form to meaning), the roles of the poles are reversed.

Since FCG pays a lot of attention to the interaction between linguistic knowledge and processing, it makes it possible to investigate the consequences of particular aspects of grammar with regard to representation, production, parsing, learning and propagation (in a population of language users). For example, a small case system may be easier to represent and produce than a large system, but it might also lead to increased ambiguity in parsing and learning that the larger system would avoid. Fluid Construction Grammar can bring these differences to the surface for further computational analysis.

It is exactly this ability to monitor the impact of grammatical choices, that has sparked the interest of an increasingly wide audience of historical and evolutionary linguists. With FCG, different historical stages can be implemented (which addresses questions about representation and processing) but FCG also comes bundled with a reflective learning framework (Beuls et al., 2012) for learning the key constructions of each stage. That same architecture has proven to be adequately powerful to implement processes of grammaticalization so that

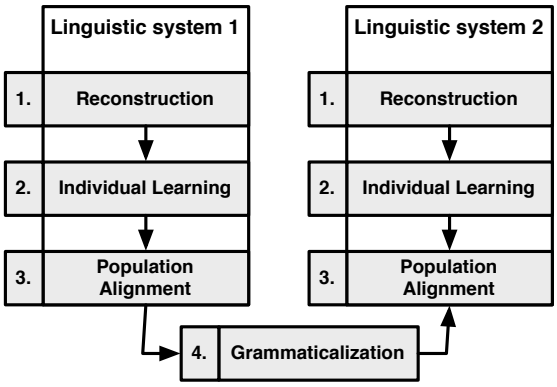

Figure 2: Schematic overview of the experimental methodology for historical and evolutionary linguists. The example here shows only two linguistic stages but there could be more.

actual linguistic change over time can be modeled (van Trijp, 2010; Beuls and Steels, 2013; Wellens and Loetzsch, 2012).

4 How to set up an evolutionary linguistics experiment in FCG?

As the FCG processor can both produce and parse utterances it is possible to instantiate not one but a set or population of FCG processors (or FCG agents) that can communicatively interact with each other. Experiments in historical or evolutionary linguistics make use of this multi-agent approach where all agents engage in situated pairwise interactions (language games) (Steels, 2012b).

In this systems demo we will focus on a recent experiment in the emergence of grammatical agreement (Beuls and Steels, 2013). The language game consists of two agents in which one agent (the speaker) has to describe one or more (max three) objects in a scene to the other agent (the hearer). Each object can be described by one or more words. It follows that without any grammatical marking it would be difficult (often impossible) for the hearer to figure out which words describe the same object and thus to arrive at a successful interpretation. The hypothesis is that the introduction of agreement markers helps solve this ambiguity.

Next to setting up a language game script the methodology consists of operationalizing the *linguistic strategies* required for a population to bootstrap and maintain a particular *linguistic system* (in this case nominal agreement). Examples of lin-

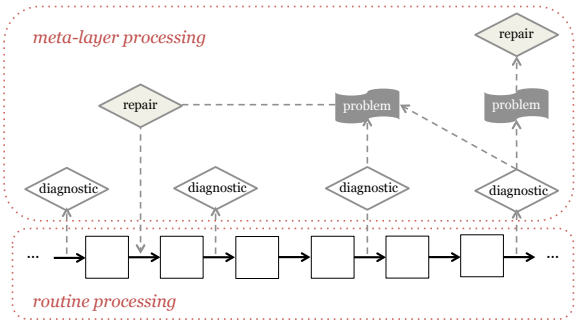

Figure 3: Reflective meta-layer architecture operating as part of an FCG agent/processor.

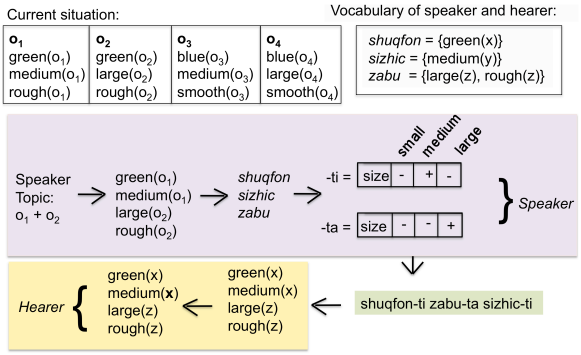

Figure 4: Meaningful marker strategy.

guistic systems already investigated include German case (van Trijp, 2012a; van Trijp, 2013), the grammatical expression of space (Spranger and Steels, 2012), the emergence of quantifiers (Pauw and Hilferty, 2012) and the expression of aspect in Russian (Gerasymova et al., 2012) [for an overview see (Steels, 2011; Steels, 2012a)].

An experiment generally investigates multiple linguistic systems of increasing complexity where each system can, but need not, map to a stage along an attested grammaticalization pathway. Most often a stage is introduced in order to gradually increase the complexity of the emergent dynamics. In this demo we posit four systems/strategies, (1) a baseline purely lexical strategy, (2) a strategy to bootstrap and align formal (meaningless) agreement markers, (3) a strategy to bootstrap and align meaningful agreement markers, and finally (4) a strategy that allows re-use of existing lexical constructions as markers (grammaticalization).

Implementing and linking together all the components involved in a single system is a highly non-trivial undertaking and our methodology prescribes the following four steps to undertake for each system (see also Figure 2).

Reconstruction: A full operationalization of all the constructions (lexical and grammatical) involved in the chosen linguistic phenomena. When multiple agents are initialized with these constructions they should be able to communicate successfully with each other. This stage serves primarily to test and verify intuitions about the different linguistic systems.

Individual Learning: Implementation of learning algorithms (or re-use of existing ones) so that one agent can learn the constructions based on the input of another agent. These learning operations are generally divided into *diagnostics* and *repair strategies* (see Figure 3). Diagnostics continually monitor FCG processing for errors or inefficiencies and generate problems if they are found. Repair strategies then act on these problems by altering the linguistic inventory (e.g. adding, removing or changing constructions).

Population Alignment: There exists a large gap between the cognitive machinary needed for learning an existing linguistic system (step 2) and bootstrapping, aligning and maintaining a complete linguistic system from scratch. In this step individual learning operators are extended with alignment strategies.

Grammaticalization: Moving from one linguistic system to another is the final step of the experiment. The challenge is to find and implement the mechanisms that drive grammaticalization (Heine and Kuteva, 2007) in line with observed grammaticalization pathways.

As an example we'll give a short sketch of one possible game as played in the meaningful marker strategy as schematically shown in Figure 4. The sketch shows a context of four objects (O_1 to O_4), each described by three features. The speaker chooses topic $O_1 + O_2$ which, given his vocabulary (shown top right), results in uttering "shuqfon sizhic zabu". Words "shuqfon" and "sizhic" both describe parts of O_1 and "zabu" of O_2. In order to explicitly communicate this linking the speaker attaches the markers "-ti" and "-ta" so that their meaning is compatible with the objects they are linking as shown in the Figure. This allows

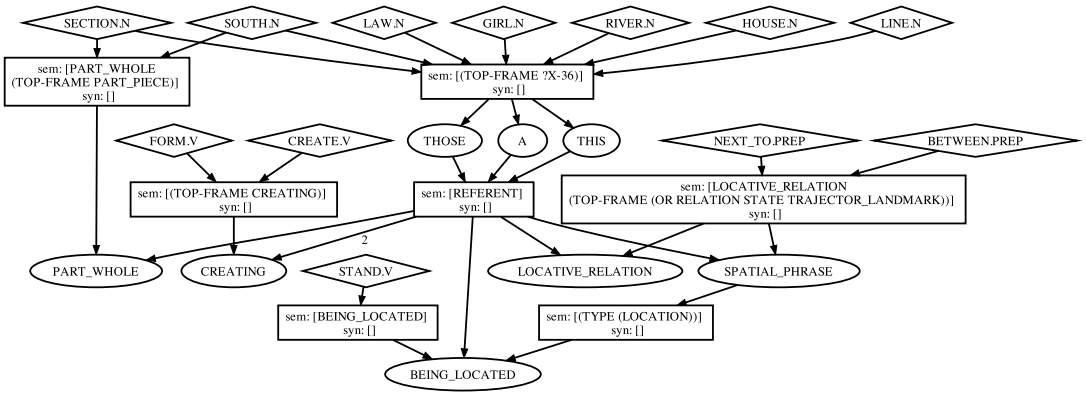

Figure 5: A network of constructions. Diamond shaped nodes represent lexical constructions, egg shaped nodes represent grammatical constructions and rectangular nodes represent semantic categories. Arrows can be read as "primes". For example the preposition between [BETWEEN.PREP] primes the category LOCATIVE RELATION which in turn primes both the [LOCATIVE RELATION] and [SPATIAL PHRASE] constructions. Both of these constructions also require a semantic category [REFERENT].

the hearer to arrive at a single non-ambiguous interpretation. For more details we refer the reader to (Beuls and Steels, 2013) and the web demo at http://ai.vub.ac.be/materials/plos-agreement/.

5 Features of FCG

A number of key features of FCG have already been introduced. Reversible bidirectional processing, a single data representation for all linguistic knowledge, a reflective meta-layer architecture for learning and a multi-agent component for managing multiple interacting FCG instances. Other features, some of which are unique to FCG, include, but are not limited to:

Web interface: FCG comes with a rich HTML/AJAX based web interface (Loetzsch, 2012) where it can show fine-grained information to the user in a user-friendly manner through the use of expandable elements. See Figure 6.

Customizable processing: Linguistic processing is implemented as a search process (Bleys et al., 2011). The user has easy access to the most important parameters influencing this process. Examples of these are the heuristics and the tests that determine whether a node represents an acceptable solution. FCG comes bundled with a library of heuristics and goal tests and with a bit of programming skills users can add new primitives easily.

Customizable construction inventory: By default, FCG stores all constructions in one large set. FCG however supplies a number of different taxonomies, both for conceptual and efficiency reasons. One popular option is to organize constructions in smaller subsets (Beuls, 2011) like lexical, morphological, functional, etc. Another option is to use networks (Wellens, 2011) that can learn co-occurrence relations between constructions and "prime" constructions when they are likely to apply (see Figure 5).

Interfaces to external repositories: FCG can connect to external repositories like Framenet (Baker et al., 1998) and Wordnet (Miller, 1995) to load thousands of lexical entries (Micelli et al., 2009; Wellens and Beule, 2010).

Robustness: FCG continues operation as far as it can get even if some constructions do not apply (Steels and van Trijp, 2011). Supplied with appropriate diagnostics and repair strategies FCG can even recover from errors (van Trijp, 2012b).

Open source: Best of all, FCG is freely downloadable and open source (http://www.fcg-net.org). It is written in Common Lisp (CLOS) and compatible with most popular lisp implementations (SBCL, CCL, Lispworks, ...).

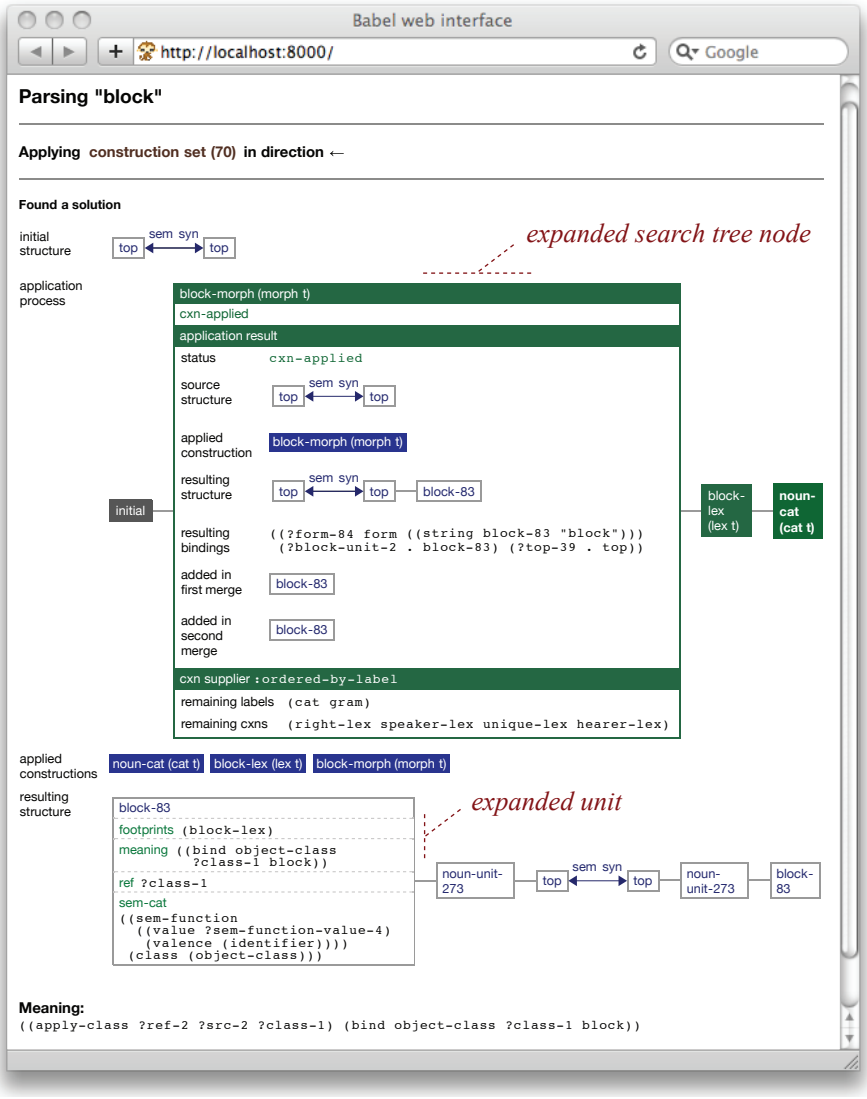

Figure 6: An example of parsing the noun "Block" as shown in the FCG web interface. Users can click on nearly every element to show an expanded version.

The reader is encouraged to take a look at http://www.fcg-net.org/projects/design-patterns-in-fluid-construction-grammar for a selection of demonstrations of Fluid Construction Grammar.

6 Conclusion

Fluid Construction Grammar is a mature technology that can be used by computational linguists to complement more traditional corpus-based approaches. FCG builds on many existing and proven technologies and adds new innovations to the mix resulting in a user friendly, yet powerful and extensible framework for in-depth investigations in natural language phenomena.

Acknowledgments

The FCG formalism is being developed at the Artificial Intelligence Laboratory of the Vrije Universiteit Brussel and the Sony Computer Science Laboratory in Paris. Pieter Wellens has been supported by the ESF EuroUnderstanding project DRUST funded by FWO and by the Vrije Universiteit Brussel. Katrien Beuls received funding from a strategic basic research grant from the agency for Innovation by Science and Technology (IWT). Remi van Trijp is funded by the Sony Computer Science Laboratory Paris. We would also like to thank Michael Spranger for his contributions to the FCG formalism.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics*, Morristown, NJ, USA. Association for Computational Linguistics.

Katrien Beuls and Luc Steels. 2013. Agent-based models of strategies for the emergence and evolution of grammatical agreement. *PLoS ONE*, 8(3):e58960, 03.

Katrien Beuls, Remi van Trijp, and Pieter Wellens. 2012. Diagnostics and repairs in Fluid Construction Grammar. In Luc Steels and Manfred Hild, editors, *Language Grounding in Robots*. Springer Verlag, Berlin.

Katrien Beuls. 2011. Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 237–264. John Benjamins, Amsterdam.

Joris Bleys, Kevin Stadler, and Joachim De Beule. 2011. Search in linguistic processing. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 149–179. John Benjamins, Amsterdam.

Mark Davies. 2011. N-grams and word frequency data from the corpus of historical american english (coha).

Kateryna Gerasymova, Michael Spranger, and Katrien Beuls. 2012. A language strategy for aspect: Encoding aktionsarten through morphology. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 257 – 276. John Benjamins.

Bernd Heine and Tania Kuteva. 2007. *The Genesis of Grammar: A Reconstruction*. Oxford University Press, October.

Martin Loetzsch. 2012. Tools for grammar engineering. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

V. Micelli, R. van Trijp, and J. De Beule. 2009. Framing fluid construction grammar. In N.A. Taatgen and H. van Rijn, editors, *the 31th Annual Conference of the Cognitive Science Society*, pages 3023–3027. Cognitive Science Society.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November.

Simon Pauw and Joseph Hilferty. 2012. The emergence of quantifiers. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 277 – 304. John Benjamins.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Michael Spranger and Luc Steels. 2012. Emergent functional grammar for space. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 207 – 232. John Benjamins, Amsterdam.

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

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

Luc Steels, editor. 2012a. *Computational Issues in Fluid Construction Grammar*, volume 7249 of *Lecture Notes in Computer Science*. Springer, Berlin.

Luc Steels, editor. 2012b. *Experiments in Cultural Language Evolution*. John Benjamins, Amsterdam.

Remi van Trijp. 2010. Grammaticalization and semantic maps: Evidence from artificial language evolution. *Linguistic Discovery*, 8:310–326.

Remi van Trijp. 2012a. Not as awful as it seems : Explaining german case through computational experiments in fluid construction grammar. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 829–839.

Remi van Trijp. 2012b. A reflective architecture for language processing and learning. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

Remi van Trijp. 2013. Linguistic assessment criteria for explaining language change: A case study on syncretism in German definite articles. *Language Dynamics and Change*, 3(1).

Pieter Wellens and Joachim De Beule. 2010. Priming through constructional dependencies: a case study in fluid construction grammar. In *The Evolution of Language (EVOLANG8)*, pages 344–351. World Scientific.

Pieter Wellens and Martin Loetzsch. 2012. Multi-dimensional meanings in lexicon formation. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 143–166. John Benjamins, Amsterdam.

Pieter Wellens. 2011. Organizing constructions in networks. In Luc Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 181–201. John Benjamins, Amsterdam.

Lin Yuri, Michel Jean-Baptiste, Lieberman Aiden Erez, Orwant Jon, Brockman Will, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *ACL (System Demonstrations)*. The Association for Computer Linguistics.