Fluid Construction Grammar for Historical and Evolutionary Linguistics

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

[1]VUB AI Lab
Pleinlaan 2
1050 Brussels (Belgium)
katrien|pieter@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

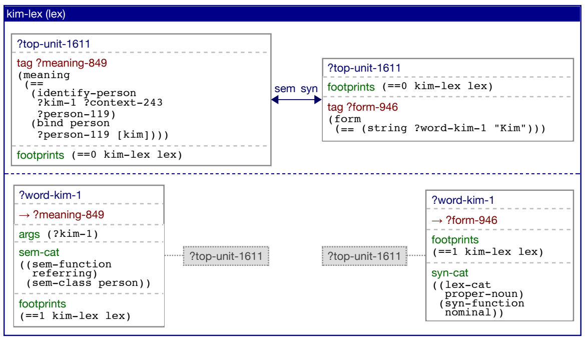
Fluid Construction Grammar is becoming increasingly more popular with historical and evolutionary linguists to answer a broader range of questions previously out of reach. Fluid Construction grammar can be used to operationalize different stages of language change and the grammaticalization processes that drive this 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). In this paper we present Fluid Construction Grammar (FCG) (Steels, 2011) as a complementary tool for historical and evolutionary linguists. FCG allows linguists to go beyond a descriptive analysis of "What has happened?" to addressing questions like "What mechanisms can explain the observed change?".

Fluid Construction Grammar (FCG) (Steels, 2011) is a fully operational unification-based language processor, capable of both parsing and producing utterances (bidirectionality)[1]. Construction grammars represent *all* linguisic 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 grammatical construction, share the same fundamental representation in FCG. As most unification-based approaches it uses feature structures to represent these constructions.

Each construction consists of two poles (one semantic/functional and one syntactic/form), each represented as a feature structure. Unlike most


```
kim-lex (lex)

?top-unit-1611
tag ?meaning-849
(meaning
 (==
  (identify-person
   ?kim-1 ?context-243
   ?person-119)
  (bind person
   ?person-119 [kim])))
footprints (==0 kim-lex lex)

sem syn

?top-unit-1611
footprints (==0 kim-lex lex)
tag ?form-946
(form
 (== (string ?word-kim-1 "Kim")))

?word-kim-1
→ ?meaning-849
args (?kim-1)
sem-cat
((sem-function
  referring)
 (sem-class person))
footprints
(==1 kim-lex lex)

?top-unit-1611

?top-unit-1611

?word-kim-1
→ ?form-946
footprints
(==1 kim-lex lex)
syn-cat
((lex-cat
  proper-noun)
 (syn-function
  nominal))
```

Figure 1: Lexical construction for the proper noun "Kim" as shown in the FCG web interface.

unification-based frameworks [e.g. HPSG (Pollard and Sag, 1994) or SBCG (Boas and Sag, 2012)], FCG uses *un*typed feature structures resulting in more liberal unification. In addition, FCG's bidirectionality uses the same constructions for production and parsing and the same processing engine, except that it is reversed. The price FCG pays for these features is that unification is split into two phases, called match and merge. The match phase is a conditional phase which checks for applicability of the construction. Only after the construction has successfully matched will the construction merge. This merge operation most closely resembles classical (yet untyped) unification.

2 Design principles of FCG

Fluid Construction Grammar is rooted in a *cognitive-functional* approach to language as opposed to a generative conception. This means that a lot of attention is paid to the processing and competence model. Instead of specifiying a grammar that can generate all valid utterances of a language (and no more), FCG was built to investigate the consequences of particular aspects of grammar with regard to representation, production, parsing, learning and propagation (in a population). For ex-

[1]See http://www.fcg-net.org for more information

ample a limited set of case markers might be easier to represent and produce but might lead to ambiguity in parsing and learning. 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 actual linguistic change over time can be modeled (van Trijp, 2010; Beuls and Steels, 2013).

A substantial and growing list of case studies now exist in FCG ranging over a rich set of grammatical phenomena and languages [for an overview see (Steels, 2011; Steels, 2012)].

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

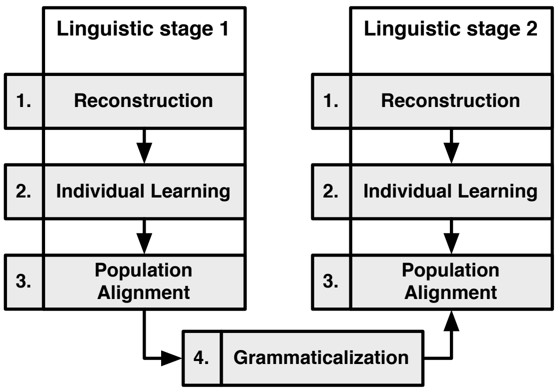


Figure 2: Schematic overview of the experimental methodology for historical and evolutionary linguists. The example here shows only two linguistic stages but there could be more.

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) which can communicatively interact with each other. Experiments in historical or evolutionary linguistics make use of this multi-agent approach.

The methodology prescribes the following steps for each stage (also see Figure 2):

Reconstruction: A full operationalization of all the constructions 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 stages.

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 can then act on these problems by altering the linguistic inventory (e.g. adding, removing or changing constructions).

Population Alignment: Addition of alignment dynamics so that a population of FCG agents can bootstrap, align and maintain a linguistic system showing key properties from the reconstructed inventory from stage 1.

Grammaticalization: The final step adds the necessary processing components so that a grammaticalization pathway as found in (corpus) data can be achieved in a population of FCG agents.

4 Features of FCG

We have already introduced some key features of FCG, like 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 5.

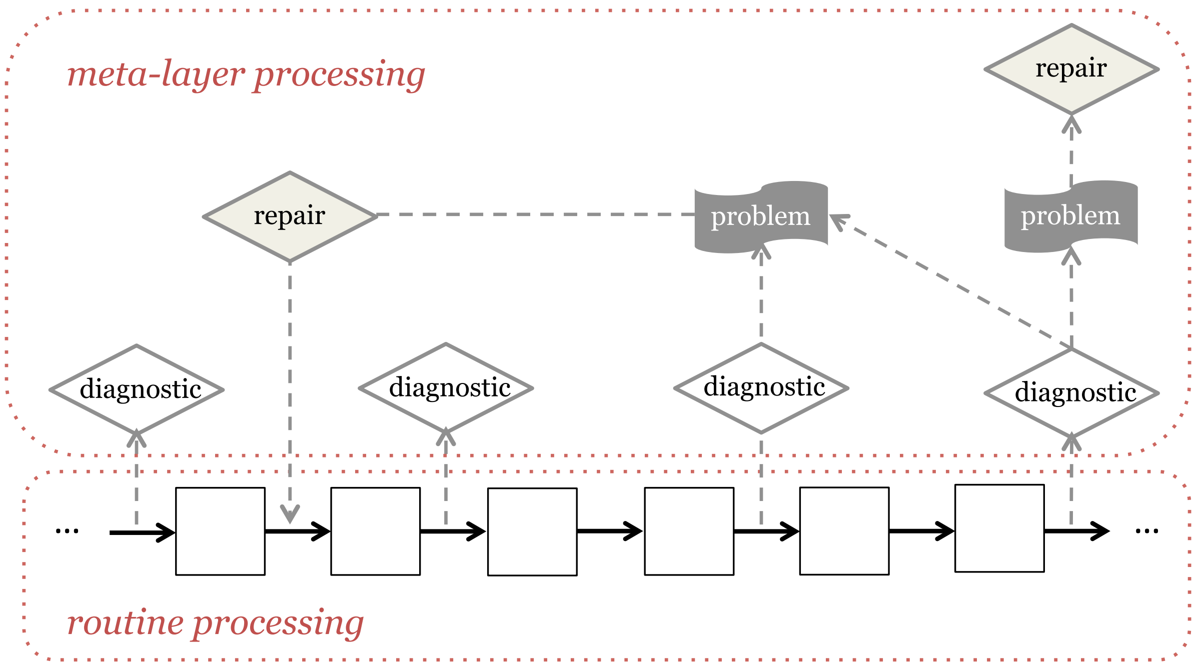


Figure 3: Reflective meta-layer architecture operating as part of an FCG agent/processor.

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 others can add new ones 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 and Beule, 2010) that can learn co-occurrence relations between constructions and "prime" constructions when they are likely to apply (see Figure 4).

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.

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, 2012).

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

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.

5 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

Left empty for review.

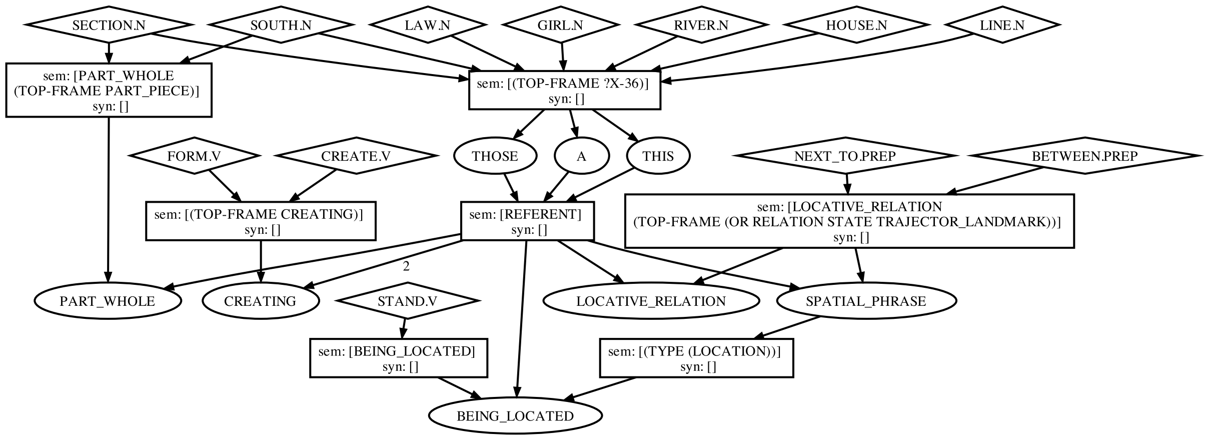

SECTION.N
SOUTH.N
LAW.N
GIRL.N
RIVER.N
HOUSE.N
LINE.N
sem: [PART_WHOLE (TOP-FRAME PART_PIECE)] syn: []
sem: [(TOP-FRAME ?X-36)] syn: []
FORM.V
CREATE.V
THOSE
A
THIS
NEXT_TO.PREP
BETWEEN.PREP
sem: [(TOP-FRAME CREATING)] syn: []
sem: [REFERENT] syn: []
sem: [LOCATIVE_RELATION (TOP-FRAME (OR RELATION STATE TRAJECTOR_LANDMARK))] syn: []
2
PART_WHOLE
CREATING
STAND.V
LOCATIVE_RELATION
SPATIAL_PHRASE
sem: [BEING_LOCATED] syn: []
sem: [(TYPE (LOCATION))] syn: []
BEING_LOCATED

Figure 4: 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].

References

[Baker et al.1998] 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.

[Beuls and Steels2013] 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.

[Beuls et al.2012] Katrien Beuls, Remi van Trijp, and Pieter Wellens. 2012. Diagnostics and repairs in Fluid Construction Grammar. In *Language Grounding in Robots*.

[Beuls2011] 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.

[Bleys et al.2011] 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.

[Boas and Sag2012] Hans C. Boas and Ivan A. Sag. 2012. *Sign-Based Construction Grammar*. The University of Chicago Press.

[Davies2011] Mark Davies. 2011. N-grams and word frequency data from the corpus of historical american english (coha).

[Loetzsch2012] Martin Loetzsch. 2012. Tools for grammar engineering. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

[Miller1995] George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November.

[Pollard and Sag1994] Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

[Steels and van Trijp2011] 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.

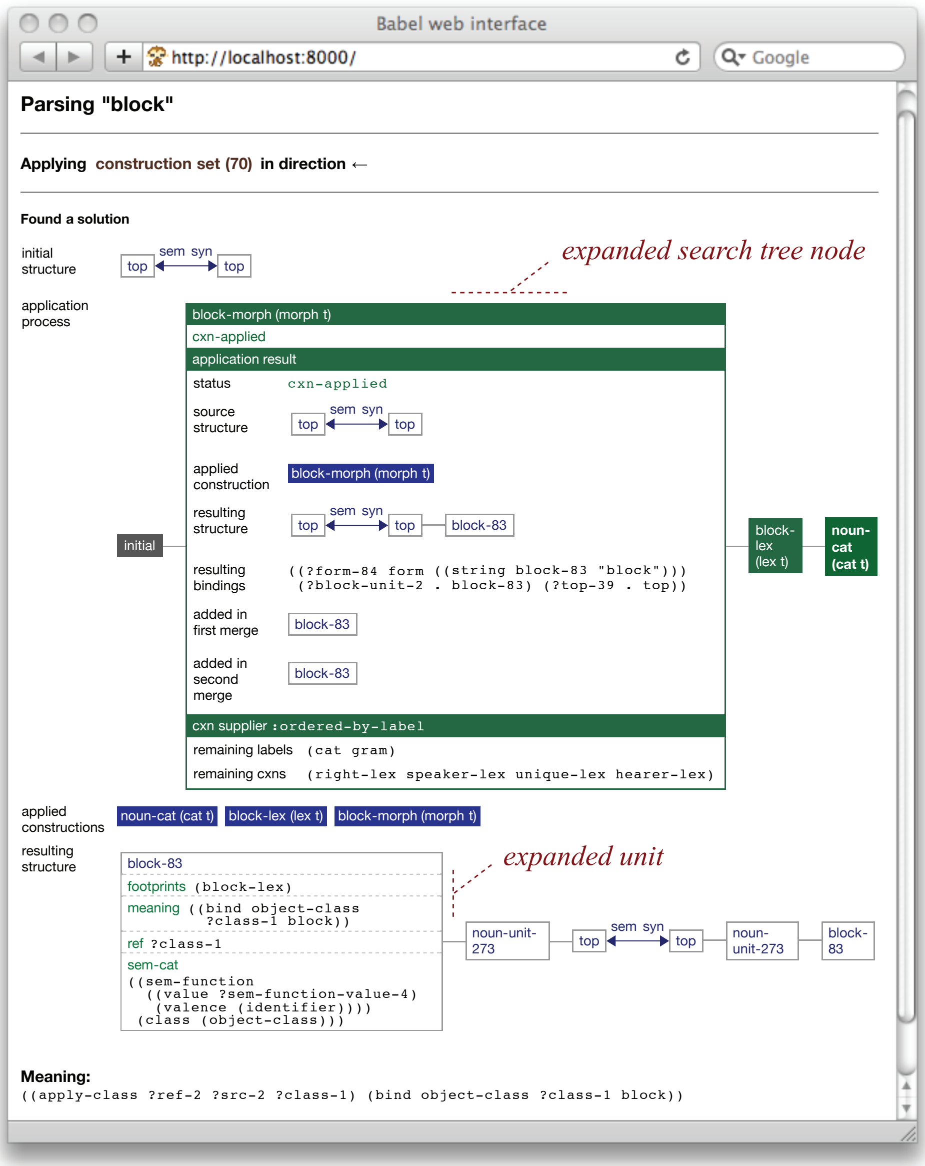
[Steels2011] Luc Steels, editor. 2011. *Design Patterns in Fluid Construction Grammar*. John Benjamins.

[Steels2012] Luc Steels, editor. 2012. volume 7249 of *Lecture Notes in Computer Science*. Springer, Berlin.

[van Trijp2010] Remi van Trijp. 2010. Grammaticalization and semantic maps: Evidence from artificial language evolution. *Linguistic Discovery*, 8:310–326.

[van Trijp2012] Remi van Trijp. 2012. A reflective architecture for language processing and learning. In Luc Steels, editor, *Computational Issues in Fluid Construction Grammar*. Springer Verlag, Berlin.

[Wellens and Beule2010] 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.



Parsing "block"

Applying construction set (70) in direction ←

Found a solution

initial structure: top ← sem syn → top

application process: initial — block-morph (morph t) — block-lex (lex t) — noun-cat (cat t)

block-morph (morph t)
cxn-applied
application result

status	cxn-applied
source structure	top ← sem syn → top
applied construction	block-morph (morph t)
resulting structure	top ← sem syn → top — block-83
resulting bindings	((?form-84 form ((string block-83 "block"))) (?block-unit-2 . block-83) (?top-39 . top))
added in first merge	block-83
added in second merge	block-83

cxn supplier :ordered-by-label

remaining labels	(cat gram)
remaining cxns	(right-lex speaker-lex unique-lex hearer-lex)

applied constructions: noun-cat (cat t) block-lex (lex t) block-morph (morph t)

resulting structure:

```
block-83
footprints (block-lex)
meaning ((bind object-class
          ?class-1 block))
ref ?class-1
sem-cat
((sem-function
   ((value ?sem-function-value-4)
    (valence (identifier))))
 (class (object-class)))
```

noun-unit-273 — top ← sem syn → top — noun-unit-273 — block-83

Meaning:
```
((apply-class ?ref-2 ?src-2 ?class-1) (bind object-class ?class-1 block))
```

Figure 5: 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.

[Yuri et al.2012] 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.