The AAAI 2017 Spring Symposium on
Computational Construction Grammar and Natural Language Understanding
Technical Report SS-17-02

Requirements for Computational Construction Grammars.

Luc Steels[1,2]
(1) Institut de Biologia Evolutiva (UPF-CSIC)
Doctor Aiguader, 88. 08003 Barcelona, Spain
(2) Artificial Intelligence Laboratory, Vrije Universiteit Brussel

Abstract

Constructional approaches to language are flourishing in many branches of linguistics but have not yet been taken up in a serious way by the AI community, mainly due to the lack of mature computational formalisms and implementations to support this approach. However there is growing recent research beginning to fill this gap. This paper sketches the properties we ideally would like to see from computational construction grammars and briefly describes how these properties have been implemented in one example system, namely Fluid Construction Grammar.

Background

Language processing requires a formalism for representing lexicons and grammars as well as algorithms for semantic parsing (going from an utterance to a meaning representation), for semantic production (going from a meaning representation to an utterance), and for grammar learning. The history of research in language processing has produced a wide range of such formalisms, which can be categorized along several dimensions, such as procedural/declarative, semantic/syntactic, abstract/instance-based, symbolic/numerical, based on rewrite rules versus transition networks, emphasizing phrase structure versus dependency structure, etc.

Construction grammar is a relatively recent development in linguistic theorizing although it has very firm roots in classical grammars going back for centuries. It started to be discussed more intensely in the late nineteen-eighties, most notably by the late Charles Fillmore (Fillmore 1988). See (Goldberg 1995), (Michaelis 2012) (Boas and Sag 2012), (Croft 2011) et al.

Despite the great interest and growing success of construction grammar in empirical linguistics, diachronic linguistics, language teaching, child language research, and other fields of language studies, there is still no widely accepted formalism for construction grammar nor a robust easily accessible computational implementation that operationalises how constructions can be used in comprehending or formulating utterances, or in language learning. But the good news is that there are a number of active research programs trying to fill this gap (Dominey 2017), (Bergen and Chang 2003), (Steels 2011)(Steels and Szathmáry 2016), (Boas and Sag 2012), (Barres and Lee 2014). These formalisms differ in terms of which issues they try to capture computationally and what datastructures and algorithms they make available. Some formalisms are motivated by cognitive (as opposed to purely computational) issues, whereas others strive for relevance to neural implementation.

The goal of this paper is not to introduce a particular formalism, but to discuss the properties we ideally want these computational construction grammars to have so that we can compare and evaluate them, and that developers can more easily see learn from each other.

Tenets of construction grammars

1. **Cutting through layers** Everybody agrees that a language processing system must reconstruct meaning from form in language comprehension and find the best way to turn meaning into form in language production. It is also well accepted that this bi-directional mapping is mediated through various structures, capturing aspects of morphology, phonology, phrase structure, functional structure, argument structure, dependencies, semantic structure, information structure, discourse. Each of these structures is reflected in utterances. For example, phrase structure is signaled through grouping and word order, functional structure through positions in phrasal structures and morphology, argument structure through word ordering, case endings, and prepositions. Traditionally the different intermediary transient structures are seen as horizontal layers which each map a representation at one layer to the next one, for example, from lexemes and affixes to morphology, from parts of speech and phrases to dependencies, from phrases and dependencies to argument structure. Construction grammarians want to cut the cake in a different, vertical way.

Construction grammarians want to organize linguistic constraints into packets, usually called construction schemas or simply constructions, that may span several layers. Thus, grammatical (combinatoric) constructions may contribute or must be sensitive to information at many layers. Consider for example subject-verb agreement in Hungarian (Beuls 2011). It is poly-personal in the sense that it depends both on the subject (as in English) and on the object, but only when the subject is a 3d person singular object. A particular choice for the expression of the object is influenced by whether the ar-

gument is a definite or indefinite object, and, in some cases, whether the sentence is negated or not. The phonological structure of the verb plays a role in the selection of the suffixes that can follow and the main vowel of the verb has to belong to the same phonological class as the vowel of the suffix. So we see a variety of criteria at many different linguistic levels (meaning, argument structure, syntactic features, phonological and morphological structure) entering the decision whether and how to establish verb agreement and case marking. Linguistic theories (such as minimalism) that insist on strictly modular components and on strict hierarchical locality of information require constantly that information is moved around up and down trees and from one unit to another.

What are the computational implications of this vertical rather than horizontal organization of grammar? The first implication is that there can no longer be a strict border between rules at each layer. Construction schemas should be able to contribute directly to the build up of any kind of structure, including the meaning of the utterance, and make use of information from any other kind of structure. The datastructures that represent transient structures should use the same format for all possible perspectives, from phonetic and morphological to semantic and pragmatic. The second implication is that all information about an utterance being parsed or produced should be directly accessible, so that a construction schema can efficiently use that information, which implies strong non-locality. For example, the construction schemas for verbal agreement in Hungarian need to have access to information about person, definiteness, the phonological structure of the verb, etc.

Apart from handling better the non-modular aspects of language, this non-modular approach can in principle achieve much more efficient language processing as well, not only because all needed information is accessible directly (instead of traversing many different nodes in localized (possibly binary) trees), but also because a construction schema can group all necessary constraints and apply them right away, thus avoiding ambiguity, uncertainty, and hence search. Admittedly writing such non-modular grammars is not so easy and we therefore need novel engineering tools to see the impact of a construction schemas or the emergent interaction between different construction schemas.

2. **The nature of generalizations** The constructional perspective insists also on other ways to capture generalizations (Goldberg 1995). Construction grammarians try to unpack grammar into different construction schemas which cooperate loosely to determine the final utterance form. For example, the number of needed lexical constructions is minimized by moving some of the functional or valence information often contained in lexical entries to grammatical constructions. Thus, there would be no different lexical entries for the adjective 'singing', as in *singing canaries*, and the participle 'singing' (progressive -ing form) as in *They are singing a Bach chorale*. Instead, grammatical constructions impose appropriate functions on these forms depending on the context. Similarly, topicalization is handled by combining the *same* lexical and argument-structure constructions with a topicalization construction that determines the constituent ordering, rather than introducing complex filler-gap operations (van Trijp 2014).

3. **Construction schemas as bundles of constraints** Traditional linguistic formalisms are conceived as rule-based systems that expand in a stepwise manner transient structures in utterance comprehension and production. And so it is tempting to see construction schemas as operators as well. They have conditions (the if-part) and information they add to the transient structure (the then-part). However the nature of human language and human language processing suggests that this is not entirely satisfactory, and that we rather have to conceive of both a transient structure and a construction schema as a bundle of constraints for the following reasons:

a. Often linguistic information is not immediately available to make a decision, neither in the input utterance (in comprehension) nor in the meaning to be expressed (in production), even if we allow construction schemas to span different levels of a linguistic structure. Consider the utterance "*The sheep owned by farmer Paul and later sold to farmer Miquel used to be kept in an open field until she escaped.* It is only when the pronoun 'she' at the very end is encountered that the parser can decide that 'the sheep' is singular. Of course this underspecification can be handled by opening new branches in the search space at each point where a choice cannot yet be made. However, that leads to combinatorial explositions and is therefore only viable for simplistic grammars and (relatively) short utterances. If we view transient structures also as bundles of constraints, some of which are still to be instantiated fully, then uncertainties can be kept around in the transient structure until they are resolved. For example, we should be able to store that 'sheep' and 'used to' agree for number without having to fill in whether 'the sheep' is singular or plural.

b. Many utterances in 'real' spoken discourse are incomplete and ungrammatical. In a strict rule-based architecture, parsers (or producers) typically get stuck, unable to proceed further because the if-part of a rule did not match. What we ideally want is that a construction schema can be applied in a flexible way, which means that some of the constraints on a schema get relaxed. In many cases, the construction schema can still be applied partially and processing can continue providing enough information to make a (possibly partial) interpretation possible.

c. The 'neutral' representation of lexicons and grammars, independently of their use in comprehension or production, has been one of the holy grails of research in computational linguistics for three reasons. (i) If we need only one representation of all construction schemas, then we are cutting the size of the constructicon into half. Given that a reasonable language system has probably something on the order of half a million construction schemas, this is significant. (ii) Language learning and grammar engineering can be accelerated because no different set of learning mechanisms and learning events are needed for learning a separate grammar for comprehension and another one for production. We also avoid complex bookkeeping operations to keep the two compatible. (In principle every sentence a speaker produces, he or she can also understand.) (iii) It is possible to do predictive comprehension, because the comprehension process can

simulate what is to follow by running the formulation process even on a partial transient structure, and predictive formulation, because the production process can simulate the effect on the listener by comprehending what is being produced. This predictive capacity is also crucial in learning. Viewing construction schemas as bundles of constraints is a huge step to achieve the bi-directional properties of a grammar.

It is true that language learners can often comprehend much more than they can accurately produce themselves and this is occasionally used as a counter-argument for using the same grammar representation. However we must keep in mind that language utterances use multiple cues to help the listener reconstruct the meaning of an utterance and some of these cues are redundant in order to make language processing more robust. For example, agreement in French between article, adjective and noun (as in 'un petit diner' - *a*-Masc-Sing *small*-Masc-Sing *diner*-Masc-Sing) helps to recognize a set of words as belonging to the same noun phrase but a French listener would also recognize 'une petit diner' or 'un petite diner', and simply assuming that the speaker has made a mistake.

4. **Diversity and competition** A basic tenet of the constructional approach is that different construction schemas may be vying for contribution to the analysis or synthesis of a particular utterance because often there is more than one way to express the same meaning and there is an enormous amount of ambiguity and syncretism in human language. Moreover when language is being learned, the learner may have to entertain different competing construction schemas. This competition is increased because of the tremendous variation found in human language communities. A listener is incessantly confronted with different idiolects and different dialects, and a speaker is also adjusting language based on social conditions or other contextual factors.

A computational construction grammar needs to be able to cope with this. This requires two mechanisms:

- It should be possible to store alternatives of a construction schema, together with information on how central an alternative is to the core grammar of the speaker's idiolect. For example, every construction schema can have an associated score which is manipulated by learning algorithms based on the frequency and success of a construction schema in concrete linguistic interactions. One step further is to refine this score with information about the contexts in which the particular alternative was appropriate.
- There has to be a mechanism for selecting among alternatives. This selection has to be based on a variety of criteria, one is the construction's score reflecting its entrenchment as part of the individual's constructicon. But there are other factors, for example, the probability that a particular construction schema will appear in the present context, the semantic plausibility of pursuing a particular hypothesis, the scope of the construction schema (schemas covering a larger stretch of the input utterance are preferred), etc.

5.**Networks of constructions**

Next, construction grammarians unanimously agree that construction schemas are not isolated entities but are nodes in various networks. These networks serve a variety of purposes, most importantly, they help to optimize language processing and streamline language learning. From a computational point of view, the implementation of such networks is not too complicated. It essentially requires datastructures to represent the networks, bookkeeping operations to build them up, ways in which they impact processing, and possibly visualization tools useful for grammar engineers or analysts. Here is a (non-exhaustive) list of networks that various projects have tried to accomodate in their implementations.

Family relations It is well known that for a particular type of construction (such as the noun phrase construction or the resultative construction) we can identify a fairly abstract skeleton and then many variations with increasingly more idiosyncratic details, the whole forming a family of construction schemas (Jackendoff 1997). These family relations should be represented explicitly, which is helpful in matching, because if a particular construction schema does not quite match, more abstract versions or closely related construction schemas should be tried. Family relations are also extremely valuable in learning, because new construction schemas need not be learned but can start from existing ones that are similar and then introduce variations that are appropriate for the current situation. Many theorists and computational construction grammar implementers have examined the use of inheritance hierarchies (adopted from object-oriented programming and frame-based knowledge representation systems.)

Priming A second network captures priming relations between construction schemas. Two schemas S1 and S2 have such a priming relation if the successful activation of S1 typically lead in the next step to the activation of S2. The relation can also specify what features added by S1 to the transient structure are the once that S2 is critically relying on to trigger. Note that the networks are different for comprehension and production. These priming networks are useful to speed up construction schema access because they prioritize which construction schema should be tried next. A priming network can be acquired automatically by simply monitoring during language processing which construction schemas get triggered. Priming networks also have a role to play in grammar engineering, because they show the processing dependencies between different construction schemas.

Construction Sets Another way to organize construction schemas is in terms of the roles they play in processing. It is a way to get back some of the benefits of organizing language processing in layers. A construction set is a set of construction schemas that form a group because they are about the same processing issues and are ideally considered before another construction set is triggered. For example, it is useful to group morphological construction schemas together and consider them before going higher up to phrasal construction schemas. This is often more efficient and allows us to deal also with default cases where no explicit information is found that overrides the most common case.(Beuls 2011) This does not imply that morphological construction schemas do not have access to other layers of linguistic

structure, nor that construction schemas in other groups cannot add additional morphological details. Organizing construction schemas in sets is also of great help in grammar engineering. It is for example exploited by the package system of ECG.

Fluid Construction Grammar

By way of example, I sketch briefly ways in which Fluid Construction Grammar (FCG) has addressed these various challenges. FCG uses feature structures to represent transient structures (See Figure 1). They represent information from any perspective in a declarative, explicit way, including hierarchical relations between units. Feature structures are also used in many other formalisms (e.g. HPSG, Unification Grammar (Kay 1984)). However, there is an important twist, designed to support non-modular direct access. Units in FCG feature structures correspond to words or word groupings (e.g. phrases). They have unique names so that any unit can be addressed directly from any other unit, and anyone of its feature values can be retrieved or filled. Units can also be addressed by a partial description, such as 'the unit which is an NP, occurs before another unit which is a VP, and has the same number and gender'. Hierarchical relations (such as phrase structure) are still representable, but as explicit features of units. They are not the vehicle through which units or information about units can be addressed and constrained.

Construction schemas are also represented with feature structures in FCG, with variables for elements that have to be bound to the transient structure. The variables are logic variables and the matching process is based on unification. There is no ordering to the units, and no constraints on which unit can be addressed. The same variable can occur at many locations, and this is a way to represent constraints such as agreement or sharing of feature values between heads and constituents.

Implementing construction schemas as bundles of constraints poses important challenges to computational construction grammars and there are several ways to realize them. FCG handles this through the use of logic variables and unification as the basis of construction application. Constraints in transient structures are represented by sharing the same logic variables and when such a variable receives a binding at one point, its value propagates to all its occurrences elsewhere in the transient structure. Anti-unification is used for flexible matching of construction schemas (Steels and Van Eecke 2016). Anti-unification means that the matcher does not seek the most general unifier by finding variable bindings that makes an expression (the triggering conditions of a construction schema) a subset of another expression (the transient structure), as in standard unification. Instead it computes the least general generalization, seeking an expression (a generalization of a construction schema) that would unify with a transient structure. Anti-unification is achieved by dropping constraints and decoupling variables, i.e. introducing two different variables in positions for which there was only one.

Although bi-directional usage of a grammar is highly desirable it cannot go at the expense of efficiency. FCG tries to balance these two somewhat conflicting challenges (bi-directionality and efficiency) by dividing the constraints in a construction schema into three bundles: There is a *comprehension lock* which contains the constraints that have to be considered in comprehension, typically constraints relating to the form of the utterance, a *production lock* which contains the constraints relevant in formulation, typically related to the meaning and other semantic aspects of the utterance, and a set of constraints to be added to the transient structure in both cases, called the *contributor*. (See Figure 1.) So the transient structure is seen as a key that opens a lock of a construction and then all constraints (from the other lock and from the contributor) are added to the transient structure.

FCG has facilities addressing the other challenges listed earlier. It is possible to have alternative construction schemas competing with each other, partly based on a score reflecting their past success and hence entrenchement in the population. Construction schemas are organized in networks, including a priming network which is acquired through language use (see Figure 2 based on (Wellens 2011)), and construction sets.

Tools

There are still many other challenges for computational construction grammars that are worth discussing, particularly with respect to learning or gaining greater control over construction schemas. Due to space limitations, I focus now only on engineering tools that would be desirable for developing broad-coverage grammars. Developing computational construction grammars is similar to the writing of complex computer programs by a team of developers and similar engineering tools are needed, such as wiki's documenting the code, repositories keeping track of changes, etc. In addition, grammar development can profit greatly from a set of additional tools:

1. Inspecting constructional processing. Grammar developers absolutely require a powerful interface with which they can inspect the construction schemas in a constructicon, they can trigger the comprehension or production of an utterance, inspect the search space, examine specific nodes in the search space by seeing the state before and after the application of a construction schema, and inspect the criteria by which a particular path was preferred. Ideally the interface should allow the selective testing of alternative paths, for example, by injecting a construction schema variant at a particular point in the search space and see the effect.

Several such browsers are being developed by the different teams building computational construction grammars. For example, Fluid Construction Grammar (FCG) features a browser in the form of a web interface (see Figure 3). It uses the facilities of web browsers so that it is machine independent.

2. Web demonstrations It is virtually impossible to describe in a single paper all the information that is needed to understand how a particular grammar has been set up or how certain technical issues (e.g. long-distance dependencies) have been handled. One way out is to create demonstrations through the web in the form of a set of web pages that mix text and interaction. However the interaction has

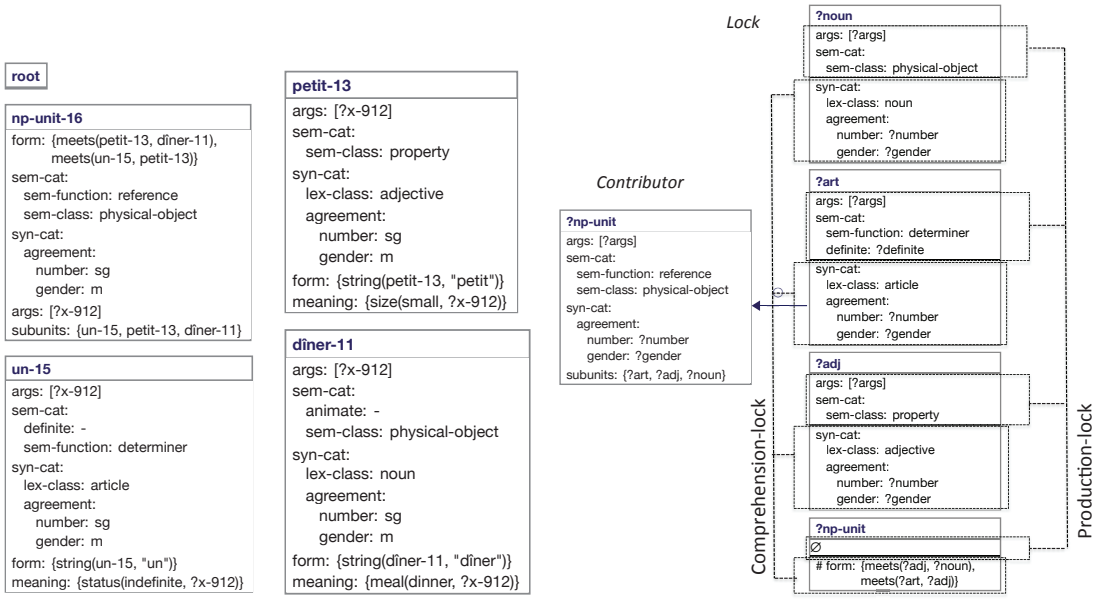

Figure 1: Left: Example of a transient structure in FCG. It contains a set of units with features and values. These feature structures represent the constraints known so far about an utterance being parsed or produced. Right: Example of a construction schema with a comprehension lock, production lock and contributor. Many of the elements in this feature structure are variables (denoted with a question-mark in front) that will get bound in the unification process.

been pre-compiled so that the viewer can click on any node as if interacting live with the constructional processor, but this interaction is simulated by displaying ready-made files. The advantage of this approach is that viewers use the familiar environment of web browsing to inspect grammars and grammatical processing and they can do that through any device, including tablets. It is a very effective way to share research results, is very useful in live presentations, and allows the easy construction of web tutorials.

Such a facility has been built for FCG, and web demonstrations are now a standard feature of publications about this system. An example of an FCG grammar for Portuguese clitics, as reported in (Marques and Beuls 2016), can be accessed through this link: https://www.fcg-net.org/demos/propor-2016/. An example demonstrating how flexible construction schema matching works and plays a role in constructional learning, as discussed in (Steels and Van Eecke 2016) is given in https://www.fcg-net.org/demos/frontiers-demo/.

3. External services Another development made possible thanks to the availability of cloud services is to set up servers that interactively perform grammar processing through remote access. Such a facility has been set up for FCG, and is running live as an interactive web service. It is possible to type in (or select) sentences, see the whole process unfold, and interactively inspect what happened, using the same web interface as in normal browsing. Such a service makes it easier for potential users to experiment with the system because they do not need to install the underlying processing (which are LISP-based), and they do not need to bother with updates as system development proceeds.

Conclusions

Constructing a computational formalism for any kind of approach to grammar is a difficult and complex matter, and this is no different in the case of constructional formalization and processing. Several alternatives are being pursued at the moment and this is welcome because in this way the possible design space for computational construction grammars gets explored faster. We urgently need common benchmarks to compare implementations not just among computational construction grammars but also across other grammar formalisms that are tackling the same phenomena.

Developing a formalism is one thing, but building a user community is equally hard. It is difficult and time consuming to master a grammar formalism and be sufficiently proficient to develop real grammars for it. There is indeed a very steep learning curve, particularly if one is not familiar with AI programming techniques in general. This explains why only a few formalisms have ever reached a significant user community. For computational construction grammar we are still in the earliest phases of user community formation but there are clear signs that this is beginning to happen.

Acknowledgement

The author has conducted this research as an ICREA research professor, located at the Institut de Biologia Evolutiva (UPF-CSIC) in Barcelona. Additional funding has come from the Atlantis project, an ERA-net project coordinated

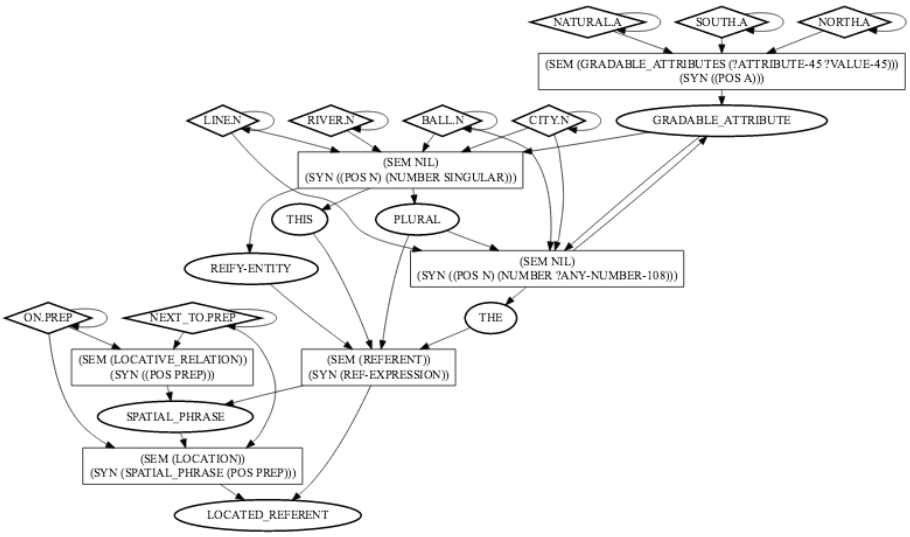

Figure 2: Priming Networks capture which construction schemas successfully triggered after a particular construction schema. And they therefore help to prioritize which schema to try first. This is one of the many different networks in which constructicons can get organized.

by the VUB Artificial Intelligence Laboratory. I am very indebted to the team that has been involved in the development of FCG. The project has been going on for almost 15 years and we keep finding ways to improve the formalism: to make it more efficient and able to handle more challenging linguistic phenomena. I am particularly indebted to team members who made important recent contributions and gave feedback on this paper: Katrien Beuls, Remi van Trijp, and Paul van Eecke.

References

Barres , V., and Lee, J. 2014. Template construction grammar: from visual scene description to language comprehension and agrammatism. *Neuroinformatics*, 12(1):181–208.

Bergen, B. K. and Chang, N.C. 2003. Embodied construction grammar in simulation-based language understanding. In J.O. Ostman and M. Fried, editors, *Construction Grammar(s): Cognitive and Cross-Language Dimensions*. John Benjamin Publ Cy., Amsterdam.

Beuls, K. 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.

Boas, H. and Sag, I. editors. 2012. *Sign-based Construction Grammar*. CSLI Publications, Palo Alto Ca.

Chang, J. De Beule, and Micelli, V. 2011. Computational construction grammar: Comparing ecg and fcg. In L. Steels, editor, *Design Patterns in Fluid Construction Grammar*, pages 259–288. John Benjamin Publ Cy., Amsterdam.

Croft, W. 2001. Radical Construction Grammar: syntactic theory in typological perspective. Oxford University Press, Oxford.

Dominey, P., A-L. Mealier, G. Pointeau, S. Mirliaz, M. Finlayson. 2017. Dynamic Construction Grammar and steps towards the narrative construction of meaning. AAAI-SSS17 Symposium on Computational Construction Grammar. Stanford Ca.

Fillmore, C. J. 1988. The mechanisms of "Construction Grammar". In *Proceedings of the Fourteenth Annual Meeting of the Berkeley Linguistics Society*, pages 35–55, Berkeley CA, 1988. Berkeley Linguistics Society.

Goldberg, A. 2006. *Constructions At Work: The Nature of Generalization in Language.* Oxford: Oxford University Press..

Goldberg, A. 2015. Fitting a slim dime between the verb template and argument structure construction approaches. *Theoretical Linguistics*, to appear.

Goldberg, A. E. 1995. *A Construction Grammar Approach to Argument Structure*. Chicago UP, Chicago.

Jackendoff, R. editor. 1997. *The architecture of the language faculty*. The MIT Press, Cambridge Ma.

Kay, M. 1984. Functional unification grammar: A formalism for machine translation. In *Proceedings of the International Conference of Computational Linguistics*.

Marques, T., and Beuls, K. 2016. A Computational Construction Grammar Approach to Pronominal Clitics in European Portuguese. 12th International conference on Computational Processing of the Portuguese Language. Springer-Verlag, Berlin.

Michaelis, L. 2012. Making the Case for Construction

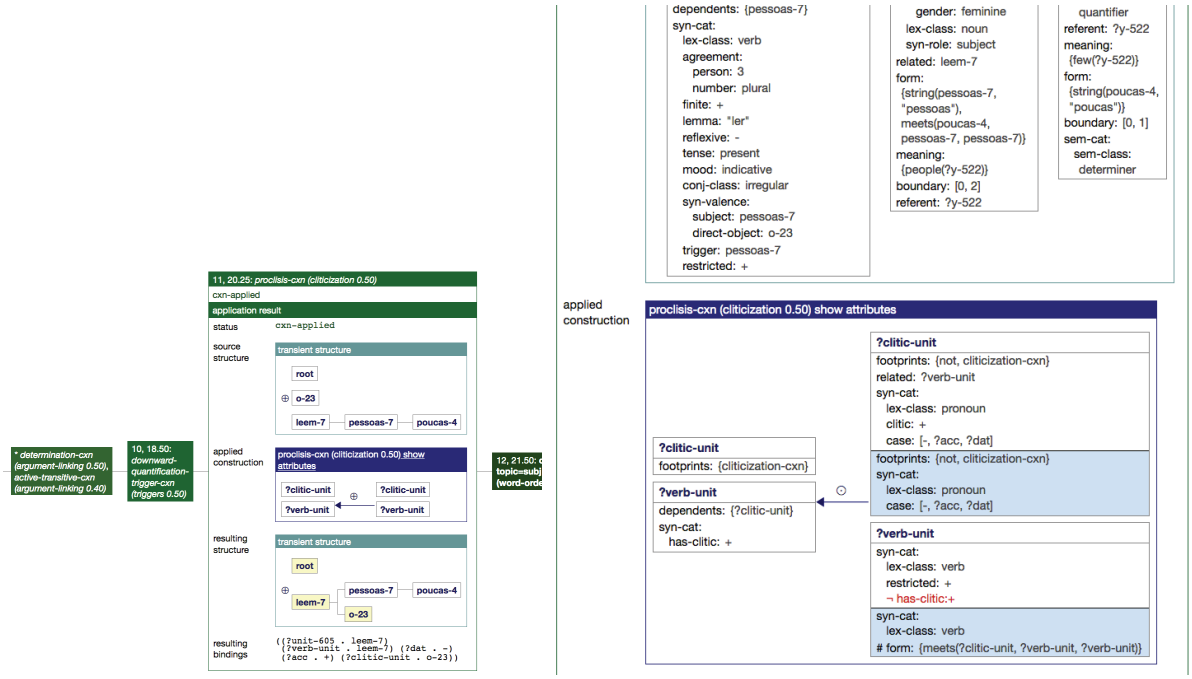

Figure 3: A grammar development environment requires above all a sophisticated interface to inspect constructicons and the comprehension and production of utterances. This Figure shows a snapshot of a Portuguese grammar in action through the FCG web interface. Left: The boxes show steps (or groups of steps) in the search space. One box is opened and shows the initial state, final state and construction schema, as well as variable bindings and additional information relevant for controlling the search process. Each of these can be expanded further to see more details. Here the box in the middle (application of the proclisis construction) is opened. Right: part of the expansion of transient structure at this node in the search space (shown only partially) and at the bottom expansion of the construction schema that was used in this step.

Grammar. In H. Boas and I. Sag, (eds.), Sign-Based Construction Grammar. Stanford: CSLI Publications. 31-69.

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

Steels, L. 2012. *Experiments in cultural language evolution*. John Benjamins Pub., Amsterdam, 2012.

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

Steels, L. and De Beule, J. 2006. Unify and merge in fluid construction grammar. In P. Vogt, Y. Sugita, E. Tuci, and C. Nehaniv, editors, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, LNAI 4211, pages 197–223. Springer-Verlag, Berlin.

Steels, L., and Szathm'ary, E. 2016. Fluid Construction Grammar as a Biological System. *Linguisics Vanguard*, 5 (2):

Steels, L. and Van Eecke, P. 2016. *Insight language learning with Pro- and Anti-Unification.* Frontiers in Psychology, submitted.

van Trijp, R. 2013. A comparison between Fluid Construction Grammar and sign-based construction grammar. *Constructions and Frames*, 5(1):88–116.

Garcia-Casademont, E., and L. Steels. 2016. Grammar learning as insight problem solving. *The Journal of Cognitive Science*, 5(17):27–62

van Trijp, R. 2010. Argument realization in Fluid Construction Grammar. In Hans C. Boas, editor, *Computational Approaches to Construction Grammar and Frame Semantics*. John Benjamins, Amsterdam.

van Trijp, R. 2014. Long-distance dependencies without filler-gaps: A cognitive-functional alternative in Fluid Construction Grammar. Language and Cognition 6(2): 242-270.

Remi van Trijp. 2011. Feature matrices and agreement: A case study for German case. In Steels, L., editor, *Design Patterns in Fluid Construction Grammar*, pages 205–235. John Benjamins, Amsterdam.

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

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