

FCGlight: A system for studying the evolution of natural language

Vlad Saveluc, Liviu Ciortuz
Faculty of Computer Science Iasi, Romania
vlad.saveluc@gmail.com , ciortuz@info.uaic.ro

Abstract

We defined FCGlight, a refined version of the Fluid Construction Grammar (FCG), which is a formalism for studying the evolution of the natural language. We picked a core subset of FCG, and expressed it in the semantic framework of the Order-Sorted Features (OSF) logic. This allows for efficient processing, and also gives FCG a solid formal background for further analysis and improvement.

Inspired from the conception of LIGHT, a system for natural language processing with large scale unification grammars, we developed a prototype system which implements FCGlight and can conduct language evolution experiments in a multi-agent population. We proved the functionalities of this system by running an experiment which models the evolution of the Russian verb aspect.

1. Introduction

The emergence and development of the natural language and the human cognition processes are two difficult to study phenomena because scientists cannot test their theories in an empirical way. The objective of the EU FP7 “ALEAR” project is to build an intelligent environment for conducting experiments in this research area. These experiments, called language games [1] are “embodied” in the sense that they use autonomous humanoid robots, equipped with the ability to move, perceive the reality and communicate with each other. Each agent has its own grammar which can change during the course of an experiment, according to a set of learning strategies. These strategies consist of language hypothesis to be tested. If these hypotheses are correct, after a number of interactions the agents will converge to similar grammars, allowing them to successfully communicate.

Because of the nature of the language experiments, with different grammar versions changing all the time, the mainstream grammars used for natural language processing cannot be really used. Their main drawback is that they assume that grammar does not change.

Among different approaches aiming to overcome this limitation, a new formalism has recently been proposed, namely Fluid Construction Grammar (FCG) [2]. FCG is a flavor of construction grammars [9] and has two distinctive features. The first one is that it allows for the grammar to change. This property is called fluidity and gives the name of the formalism. The second property is the bi-directionality of the rules, which means that the same construction can be used for both parsing and production. This is needed because during a language game, one agent can be in turn hearer and listener.

Construction grammars are based on the mechanism of feature structure (FS) unification. FCG started to work with such a formalism, but several operators and extensions were added, which caused it to diverge from this family of grammars. Compared to the mainstream unification grammars, FCG lacks a clearly defined declarative semantics, while its procedural semantics is quite complicated. Also it lacks a solid logic framework in which one could conduct a formal analysis of its properties.

We rewrote a core subset of FCG into a well-established feature constraint formalism, and then we showed that this subset is sufficient for conducting advanced language games.

The plan of this paper is as follows: section 2 will informally describe the FCG formalism; section 3 will introduce OSF, the logical framework on which we build FCGlight; section 4 will present the core subset of FCG that we have delimited and show how it can be expressed in the OSF-logic framework; section 5 will provide some technical details regarding the implementation of the FCGlight system, and section 6 will describe a language game for learning the Russian verb aspects. We describe the main steps of the initial experiment and also the significant changes we have made when reimplementing it in FCGlight.

2. FCG: A brief overview

FCG has two types of structures: coupled featured structures (CFSs) and constructions (or rules), which apply on CFSs. A CFS has two poles, containing

semantic and respectively syntactic information. Each pole is composed of several units, further divided into feature-value pairs. These values consist of constraints expressed as predicates. Usually there are variable coreferenced inside a pole.

The rules are similar to the CFSs but they may contain also some special operators: the J operator, the tag operator and certain operators imposing restrictions of the feature values.

The application of a rule to a CFS consists of two stages: unification, and merging. Depending on the type of processing (parsing or production) the poles will play a different role. In the parsing process, the system will first try to unify the syntactic pole of the rule with the syntactic pole of the CFS. If this is successful it will merge the semantic poles. The production process goes the other way, with the semantic poles being unified and the syntactic poles being merged. The operators mentioned earlier have different behavior depending on the role played by the pole they are in.

In FCG, a pole is considered to unify with another if its restrictions are a subset of the later. In the unification process, the parser will check if the pole of the CFS contains all the restrictions from the corresponding pole of the rule. If this happens, the information from the other pole of the rule will be added in the CFS through the merge operation.

Because of the fluidity of the language, the parsing process is different in FCG than in mainstream unification grammars. Unlike for instance LIGHT, the parsing process can be partial: not all the words need to be covered by the parsing tree. Also in FCG all the rules are uni-directional and there is no start symbol. Its place is somehow taken by a so-called top-unit, to which all the other units (morphological, lexical, syntactic or semantic) get linked in a way or another. The parsing process consists of several stages. The rules are partitioned in 4 subsets: lexical, morphological, mapping and semantic rules. The morphological and semantic rules introduce syntactic and semantic categories, while mapping rules map between them. These sets of rules are applied subsequently, depending on the type of processing. In parsing the order is lexical, morphological, mapping and semantic rules; while in production the semantic and morphological steps exchange their place.

The learning strategy used in the FCG experiments is based on reinforcement learning. Each rule has a confidence score, based on the success it had on previous interactions. This mechanism creates a competition between the rules, where there more used ones will win, while the less used rules tend to be eliminated.

3. OSF: A logical framework for FCGLight

We chose the Order-Sorted Feature (OSF) logic formalism proposed by Ait-Kaci [4]. This choice was natural because the coupled feature structures, the main elements from FCG have a structure similar to the OSF feature structures. Also, due to its use of sorts, OSF will add a new possibility to the learning experiments, namely learning in hierarchies (lattices) of concepts, which exploits a partial order relation (generalization/specialization) between grammars [3]. This mechanism could enhance the current learning mechanism used in FCG experiments, which is reinforcement-based learning.

OSF proved in the past to be a good background for natural language processing. It was used for instance by the LIGHT system [5]. LIGHT was developed for efficient processing of HPSG-like unification grammars, but it is not restricted to HPSG. LIGHT comprises two main levels: feature structure unification, and the control level (actually performing active bottom-up chart-based parsing) upon the unification level. An important number of optimizations were included into the LIGHT system; they were fine tuned so as to achieve efficient parsing with LinGO, the large scale HPSG grammar for English developed at CSLI, Stanford University [10].

OSF is a feature constraint logics. Each OSF-term (or feature structure) can be expressed as a conjunction of elementary constraints. There are three types of elementary constraints: sort constraints, feature constraints and equality constraints. For example, the OSF term

$$\text{term1:sort1}(\text{FEATURE1} \rightarrow X:\text{string}, \\ \text{FEATURE2} \rightarrow X)$$

where X marks a coreference of the two features, is equivalent to the following set of basic constraints $\{(\text{term1:sort1}), (\text{term1.FEATURE1} = XZ), (X:\text{string}), (\text{term1.FEATURE2} = X)\}$.

There are two basic feature structure operations that are used in the language processing: subsumption and unification. A FS subsumes another if its elementary restrictions are a sub-set of the elementary restrictions of the later. The unification of two FSs consists of another FS whose elementary constraint set is equal to the union of the elementary constraint sets of the two initially given FSs. The algorithms for computing both subsumption and unification are efficient and they can unambiguously determine the term substitutions. If we view the terms as hierarchical structures, these algorithms are equivalent to a basic tree search algorithm, resulting in a linear time complexity and constant memory consumption.

As a matter of terminology, we have to make the remark, the FCG unification corresponds to OSF FS subsumption, while the FCG merge operation

corresponds to OSF FS unification. In the sequel we will implicitly use the OSF terminology.

4. FCG goes light

We chose a subset of the essential features from FCG and expressed them into the OSF formalism.

Although the CFSs terms have a tree-like structure similar to that of OSF-terms, there are proprieties that do not have a straightforward representation in OSF. However it is easy to see that for designating the two poles of a CFS we can simply add two features called SYN and SEM to each unit. Also, the structure of a regular pole is very similar with the one of a OSF-term, except for the slots and their values. These values are lists of constraints instead of feature structures. These lists of constraints be restricted to rooted clauses (conjunctions) of elementary constraints. Therefore we transformed slots into features and their values into OSF-terms, building them around the root variable in the list of constraints. Here is a example of a list of predicates, describing in FCG the event “john gives a book to marry”.

```
((give-event ?ev)
(give-agent ?ev john)
(give-object ?ev book)
(give-recipient ?ev mary))
```

We can transform this into a feature structure having the variable ?ev as root. Here is the equivalent OSF-term:

```
#ev:give(
  agent → john ,
  object → book ,
  recipient → mary
)
```

FCG represents hierarchies of units by using a special feature named SEM-SUBUNITS, for the units int the semantic pole and the feature SYN-SUBUNITS for the syntactic one. The value associated with these features is a list of variables coreferenced with other units, as in the following example

```
((?root-unit
  (SEM-SUBUNITS (?child1 ?child2)))
(?child1
  ... restrictions ... )
(?child2))
```

This structure can be partially expressed in terms of OSF basic constraints, but unfortunately it will not respect the rule that a term cannot have multi-valued

features. Multi-valued features affects the efficiency of the unification algorithms.

Because of the special procedural semantics introduced in FCG by the tag and J operators, we could not express a construction with a single OSF-term. A construction (or rule) will be translated in two groups of constraints, one for parsing and one for production. Such a group is split into subgroups of constraints, according to 3 stages of rule application. The first stage, the precondition, checks whether the first subgroup of constraints subsumes a corresponding set of constraints in the target CFS. The second stage performs constraint reduction, and the third step is based on the unification operator.

In order to translate the FCG constructions/rules into the OSF formalism we first had to find an equivalent to the FCG special operators mentioned earlier.

The J operator is used for creating the parsing/production trees. In FCG unification it acts as a checker for a given set of constraints, while in the merging step it will add its constraints into the specified unit, eventually creating it as a new sub-unit linked to an already existing unit. For implementing this behavior we modified the original algorithms for subsumption and unification operations, described by Ait-Kaci [4].

The tag operator indicates a substructure (set of elementary) constraints to be deleted and eventually moved elsewhere. In LIGHT, the parsing process is monotonic, which means that if we successfully apply a rule to a term, the old term will subsume the new one. In FCG, if we apply a construction to a CFS the resulting CFS will contain more information than the old one, but usually in a different arrangement. In FCGlight we chose to express the tag operator in a monotonic way. The tag operator is replaced by REDUCE operations in the second stage of rule application, i.e. parser/producer actions. These operations are implemented using the OSF unification We duplicate the tagged information, but also mark it as reduced in the original place.

To better illustrate the above principles of translation from FCG into the OSF formalism we will give an example of a FCG rule and its equivalent in FCGlight format.

```
((?top-unit
  (tag ?meaning (meaning (== (read ?event)
    (reader ?event ?agent) )))
  ((J ?verb-unit ?top-unit)
  ?meaning
  (referent ?event)
  (sem-cat (==1 (base-type ?event event))))
  <-->
  ((?top-unit
    (tag ?form (form (== (string ?verb-unit "cita")))))
```

```

((J ?verb-unit ?top-unit)
 ?form
 (syn-cat (=1 (pos verb)
 (gender ?agent ?agent-gender)
 (case ?object accusative))))))

```

This is a lexical entry for the Russian verb “cita” (to read). In parsing mode it will try to subsume the syntactic pole. This process checks if it exists a “cita” string inside the form feature. If so, it will be tagged with the variable ?form. In the unification phase, it will create a new unit (?verb-unit) in the syntactic pole through the J operator which will hold the information tagged in the subsumption step and also the information about the syntactic category: part-of-speech (pos), gender and case. In the semantic pole, the unification operation will add the semantic information into a similar unit. This information consists of the event type (read), the agent involved, the base type of the action. Although the read and reader predicates are inside the tag operator they will not act as restrictions, because in the FCG unification mode the tag operator acts only as an assignment to a variable. This construction is the equivalent in FCGLight to the following groups of constraints:

Production

precond	#top-unit(MEANING → << #event:read(READER #agent)
producer actions	REDUCE: #top-unit(MEANING → << #event:read(READER #agent)
main	#top-unit(SEM-SUBUNITS → << #verb-unit(MEANING → << #event:read(READER → #agent) >> REFERENT → #event, SEM-CAT → top(BASE-TYPE(#event) -> event)))

Parsing

precond	#top-unit(FORM → #verb-unit(STRING → "cita"))
parser actions	REDUCE: #top-unit(FORM → #verb-unit(STRING → "cita"))
main	#top-unit(SYN-SUBUNITS → << #verb-unit(STRING → "cita", SYN-CAT → verb:pos(GENDER(#agent) → #agent-gender, CASE(#object) → accusative)) >>)

In the parsing process the “precond” term from the Parsing table will be subsumed, then the reduction actions will be applied and finally the main term will be submitted to unification. In production a similar process happens. These two tables from above can be easily computed with the following algorithm:

Input: FCG rule
Output: syn, sem //tables

Initialize the fields from the syn and sem tables
with empty term of sort top (most general sort)

for pole in {syn, sem}
tagged ← empty //holds (key, value) associations

if pole = syn then subunits ← SYN-SUBUNITS
else subunits ← SEM-SUBUNITS

for-each unit in pole of FCG rule, except J-units
for-each feature in unit

if feature is (tag ?variable value)
add (?variable, value) to tagged
add "REDUCE: value"
to pole[parser/producer actions]
restrictions ← value

else //normal feature
restrictions ← feature

unify pole[precond] with restrictions

for-each J-unit in pole of FCG rule
(child, parent) ← J-unit
if parent = nil
unit = child
else
unit = new empty unit
unify pole[main] with
parent(subunits → << unit >>)

for-each item in J-unit
if item is variable
value ← tagged[item]
else
value ← item
unify pole[main] with value

5. Implementation of FCGLight

We implemented a proof-of-concept system for showing that mechanisms previously described make

us able to conduct FCG language games in the new framework.

We wrote a system in C++, inspired by LIGHT. The LIGHT system has its own format for expressing grammars. It contains sections for describing the sort hierarchy, the lexical entries and the constructions. Each lexical entry and construction will be compiled in a different C function, in a flat format described by Ait-Kaci [4]. Below, we have an example of the compiled form for the term which in the previous section expressed the event “john gives mary a book”:

```
void term1() {
    push_cell(0);
    set_sort(0, “give”);
    push_cell(1);
    set_sort(1, “john”);
    set_feature(0, “agent”, 1);
    push_cell(2);
    set_sort(2, “book”);
    set_feature(0, “object”, 2);
    push_cell(3);
    set_sort(3, “mary”);
    set_feature(0, “recipient”, 3);
}
```

For efficiency reasons the strings that represent the sort types and feature names are implemented using a symbol dictionary, but here we represented them for readability purposes explicitly. As the reader can see, this flatten format has many similarities with the basic constraints. Also is a very convenient form of representing hierarchical structures, because the system will not need a parser, but only four simple functions that will be linked together with the above code.

FCGlight implements the OSF unification algorithm with the modifications described in the previous section. Instead of using bidirectional rule that are processed in a different way in parsing and production we have two unidirectional rules consisting of two terms each. The first one represents the precondition and will be checked for subsumption wrt the target CFS, and the second is equivalent with the body of rule and the reduction actions. In order to ensure the assure the fluidity of the language, we synchronized the 4 terms using a system of pattern matching.

FCGlight has also all the functionalities needed for conducting multi-agent language games. Each agent has its own grammar and a meta-layer consisting of function for diagnosis of problems and repair strategies. When a problem is encountered (for example before trying to communicate the utterance one agent re-enters it into its own system and finds it ambiguous) a repair strategy is triggered, and it can alter the existing rules, or create new ones.

The output of a experiment is a HTML presentation that traces the interactions, the problems that eventually occurred, the solution found by the agents and the evolution of their grammars.

6. A case study: Learning the Russian verb aspects

We tested our system on the language game described in Gerasymova's MSc thesis [6] for learning the aspect of Russian verbs. This grammatical category is a very complex system, scientists trying to understand how it is actually acquired. There are many rules and exceptions for expressing the aspect but the experiment focuses on the aspect of simple verbs, which cover most of the cases. A temporal semantic is assigned to a Russian verb by prefixation (for example, “po-cita-l” means read for-a-while, “na-cita-l” implies that the reading is complete). Also a semantic of an ongoing action is assigned for a non-prefixed verb. The experiment develops a grammar that will mark prefixed verbs with the perfective aspect and non-prefixed verbs with the imperfective aspect.

This experiment shows how a productive aspectual grammar can be learned by artificial agents. It is inspired by the learning stages of a child learning his/her mother tongue. This is more relevant for autonomous robots, because they face similar challenges having to derive rules on their own, using the observed language examples.

According to recent findings in psycholinguistics [8], the linguistic units used by people feature different structures and levels of abstractness. First, children start talking in holophrastic units which they learn by imitating adults (for example gimme-that, i-wanna-do-it). Children use these holophrases over and over again until they are able to create novel grammatical rules with an increased level of abstractness. These are called item-based constructions because there have a slot which can be filled by a word or a phrase, that will determine the function of the whole utterance (for example “X is broken” or “put X here”). The learning of item-based constructions represents the second stage in the learning process. The third step, and the most difficult one is the acquisition of the abstract constructions. These are created by further generalization of the available constructions, but usually without any concrete linguistic material.

This learning strategy is embodied into an FCG language game, played within a population of robotic agents. There are two categories of agents: one with a fully developed aspectual system (teachers) and one with only lexical entries (learners) and/or fewer rules.

The language game consists of a series of interactions. In these interactions both a learner and teacher will perceive the world in a similar way. The

world consists of two actors: a girl (“Masha”) and a boy (“Misha”) performing the same action but with a different time semantic (for instance Misha finished reading “Misha nacital”, or Masha is reading “Masha pocitala”). The teacher will utter a question that will discriminate between the actors. If the learner understands the question it will provide an answer and the interaction is considered a success, otherwise it is a failure. In the later case the learner will try to repair its grammar using its learning capabilities. These consists of several diagnosis functions and grammar repair strategies.

The simplest repair strategy is “internalize-utterance”. When the learner perceives an unknown utterance it will memorize it as a whole and will be able give the correct answer to a further question. This is equivalent to learning a holophrase.

From two holophrase constructions the gender can be generalized. This strategy was not described by Gerasymova. In the Russian language the gender of a verb is expressed via the suffix: “Misha po-cita-l” vs. “Masha po-cita-la”. However the questions are asked in masculine: “Kto po-cita-l?”. Even if the answer is Masha, the learner will internalize the masculine version because the linguistic material provided to him is the question. In order to solve this problem we added another step before learning a holophrase: the learner will try to express the answer through a complete sentence. If the holophrase that he wants to learn can't be used, we will remove the restrictions that block its application. For example he will not be able to use the holophrase “po-cita-l” for expressing Masha is reading because the unit for the suffix “-l” needs an agreement with masculine gender of the verb. We observe the the only untagged string is the suffix and also that the unification failed because of the gender agreement and we remove these two restrictions from the rule. It will result a rule covering “po-cita”.

The next step is the generalization of the prefix. After he learns a new holophrase, the learner will look into his inventory and check if there are other similar entries with the same prefix but a different verb. If this is the case, it will generalize a new “item-based construction” expressing the meaning of a prefix (for example “po-X” for actions in progress).

In the experiment described by Gerasymova there are two additional steps. In the first step a new syntactic category is derived marking the existence or not of a prefix. This corresponds to the syntactic category o aspect. Next a mapping rule is learned that maps this syntactic category into a semantic one. We proposed a alternative solution which combines these

two steps into a single one. This step benefits from the functionality of the sort hierarchy: the semantic category is expressed by generalization of the events into ongoing and non-ongoing events.

7. Further work

Right now our system is a proof of concept. There are several places where it can be improved and optimized. One goal is to develop it into a fully-fledged extensible system that allows a rapid development of new language games.

Another direction is the development of strategies for new language games. One possibility is the learning of clitic pronouns in the Romanian language, a rather difficult issue for non-native speakers.

Acknowledgements: This work was funded by the “ALEAR” EU FP7 research project.

9. References

- [1] Steels, L. (1996). A self-organizing spatial vocabulary. *Artificial Life*, 2(3):319–332.
- [2] Steels, L. and De Beule, J. Unify and merge in fluid construction grammar. In Paul Vogt, Yuuga Sugita, Elio Tuci, and Chrystopher L. Nehaniv, editors, *EELC*, volume 4211 of *Lecture Notes in Computer Science*, pages 197–223. Springer, 2006.
- [3] Ciortuz, L. *FCGlight: Making the bridge between Fluid Construction Grammars and main-stream unification grammars by using feature constraint logics*. Technical Report, Faculty of Computer Science Iasi 2010
- [4] Ait-Kaci, H. and Di Cosmo, R. Compiling order-sorted feature term unification. H. A' Technical report, Digital Paris Research Laboratory, 1993.
- [5] Ciortuz, L. *LIGHT — a constraint language and compiler system for typed- unification grammars*. In *KI-2002: Advances in Artificial Intelligence*, volume 2479, pages 3–17, Berlin, Germany, 2002. Springer-Verlag.
- [6] Gerasymova, K. *Acquisition of aspectual grammar in artificial systems through language games*, 2009. Humboldt Universitaet zu Berlin, Germany, MS thesis.
- [7] Steels, L. (1997b). Self-organizing vocabularies. In Langton, C. G., editor, *Proceeding of Alife V*.
- [8] Tomasello, M. (2000). First steps toward a usage-based theory of language acquisition. *Cognitive Linguistics*, 11-1/2:61–82.
- [9] Goldberg, A. (2003). Constructions: A new theoretical approach to language. *Trends in Cognitive Science*. Volume 7, issue 5, 2003, pp. 219-224.
- [10] Flickinger, D. and Copestake, A. and Sag, I. *HPSG analysis of English*. In Wolfgang Whalster, editor, *Verbmobil: Foundations of speech-to-speech processing*, *Artificial Intelligence*, pp. 254-263, Springer Verlag, 2000.