LINKING IN FLUID CONSTRUCTION GRAMMARS

Luc Steels [a,b] Joachim De Beule [b] Nicolas Neubauer [b]

[a] *Sony Computer Science Laboratory, Paris*
[b] *Vrije Universiteit Brussel (VUB AI Lab), Pleinlaan 2, 1050 Brussel*

Abstract

One of the key problems in any language processing system is to establish an adequate syntax/semantics interface, and one of the major requirements of such an interface is that partial meanings contributed by individual words are properly linked with each other based on grammatical constructions. This paper reports how we deal with this problem within the context of Fluid Construction Grammars (FCG). FCG is a general unification-based inference engine which has been designed to support experiments in the self-organisation of language in a population of interacting situated embodied agents. The paper focuses on technical details pertaining to the linking problem.

1 Introduction

The work reported in this paper is part of a larger research effort to understand the origins and evolution of language by building experiments with situated embodied agents that self-organise communication systems with natural language like properties [15], [18]. Our earlier work focused on the self-organisation of lexical languages either with single words expressing perceptually grounded categories [14] or multiple word utterances expressing combinations of categories but still without grammar.

Currently we are focusing on the emergence of grammar, and obviously one of the prerequisites is that agents have ways to represent natural language like grammars and ways to invent and adopt grammars in situated embodied interactions. Modeling language emergence requires fluidity [10]: Grammatical conventions are only partly shared by members of the population and may not exist at some point. They arise by grammaticalisation phenomena as studied by historical linguists [9] and then propagate in the population based on alignment phenomena, as well documented in dialog studies [4]. The requirements of the representational formalisms we need in our experiments are therefore different from those underlying the standard formalisms currently used in computational linguistics, such as HPSG [13]. Specifically, the agent's grammars need to be dynamic because they may change after every interaction, they need to be bi-directional because we want agents to take turns being speaker and hearer, they need to support that grammatical rules will be competing with each other for dominance in the language, and they need extreme flexibility so that utterances that cannot be fully parsed or are ungrammatical from the viewpoint of the hearer can still be parsed and interpreted to some extent. We therefore call our formalism Fluid Construction Grammar (FCG), to emphasise both the fluid nature of grammatical processing and of the grammar itself, and the fact that our major inspiration comes from a recent trend in linguistics towards a construction grammar viewpoint, which is increasingly seen as a fruitful way to study human natural languages [6], [12].

Fluid Construction Grammar (FCG) has been under development since 2000 and investigations are far from finished, even though we have currently a fully operational version (see the website: http://arti.vub.ac.be/FCG/ for more details). Moreover we do not claim that FCG is the only possible way to formalise or implement construction grammars, nor that the specific view on construction grammar implied by FCG is the 'right' one. Instead our design should be seen as a contribution to ongoing cognitive linguistics research and to the developing body of implementation techniques for construction grammar. In earlier papers we already reported on factors that

could trigger the emergence of grammar [18], mechanisms for learning grammatical constructions [16], the handling of hierarchy in FCG [5]. In this paper we focus on one specific issue, which is the linking problem. It concerns the question how variables introduced by individual words can be linked together (made equal) before semantic interpretation is performed. We will show that FCG provides an elegant solution which can moreover be used both for parsing and production.

So far only two other substantial efforts towards the development of formalisms for construction grammars have been reported in the literature. One project by Kay and Filmore, known as CxG (see the detailed description in [8]), uses some of the principles and mechanisms we have used in FCG: (1) the lexico-grammar is represented as constraints on form-meaning pairings as opposed to derivational rules, (2) syntactic and semantic structures are represented as feature structures, and (3) unification is the basic engine for applying constructions in parsing or production. Many formal details in CxG have not yet been flushed out, particularly those relating to semantic and pragmatic issues, and particularly to the issue of linking which is the main topic of this paper. Consequently no computational implementations have been reported yet. Another project, known as Embodied Construction Grammar (ECG) [1], has many similar characteristics and has gone further towards a computational implementation of constructions (again using a feature-based unification style approach) and parsing mechanisms [2]. ECG addresses more extensively the issue of semantics through the mental simulation of events [7] and deals to some extent with the question of linking. However ECG is only for parsing and does not address the difficult issue of using the same grammatical representation for both parsing and production.

The rest of the paper is in four parts. First we define the linking problem in some more detail. Next we introduce some elementary notions of FCG. Then we focus on a very simple example showing how FCG constructions achieve linking. We conclude with some arguments why the approach followed is adequate.

2 The Linking Problem

There is a general consensus that the lexicon of a language provides building blocks that contribute the initial meanings of a sentence, and grammatical rules or templates express constraints on aspects of syntactic and semantic structure that link these individual meanings into a larger whole. For example in the phrase "Jill walks" the word "Jill" introduces a variable ?x which will be bound to an object that is named Jill 'jill(?x)' and the word "walks" introduces two other variables, ?ev and ?y, which will be bound to a walk event and an object that is the walker in this event respectively: 'walk(?ev), walker(?ev, ?y)'. The lexicon does not specify that it is Jill which is the walker in the walk-event, however we know from the syntactic structure that Jill is the subject of the sentence which allows us to conclude that it is the walker in the walk-event introduced by the verb. Within a logic-based representation for the meaning, which will be adopted here because it is standard in computational linguistics, this kind of linking can be specified by saying that ?x and ?y are two variables that have to be bound to the same object, i.e. that they are equal. The meaning of the sentence after taking the grammar into consideration therefore becomes: 'jill(?x), walk(?ev), walker(?ev,?x)'. Here is another example. In the phrase "big block" two predicates are introduced by the lexicon: 'big(?x)' and 'block(?y)'. But it is the grammar (specifically the fact that the adjective and the noun form part of the same noun phrase) that tells us that ?x and ?y are bound to the same object, i.e. that ?x and ?y are equal.

So grammars specify how the meanings contributed by individual words are to be linked together. A second observation is that natural language grammars do not specify this linking in an ad hoc way for every combination of words (which would mean that there is a very specific pattern for linking "big" and "block" and a different one for "red" and "block"). Rather there is an intermediary layer of semantic and syntactic categorisations and the linking is specified at that level. A constraint linking a semantic frame and a syntactic pattern is called a (linking) construction in construction grammar and we will follow this terminology. An example is shown in figure 1, which relates the abstract CAUSE-MOVE semantic frame with the grammatical pattern Subject+Verb+Object+Oblique. It is applicable to sentences like "Jill slides the block to Jack". The various roles in the frame (cause, goal, theme) are mapped onto grammatical relations (subject, object, oblique). The fact that the linking now happens on a more abstract level implies that there must be cognitive machinery to categorise the specific predicate-argument structure (i.e.

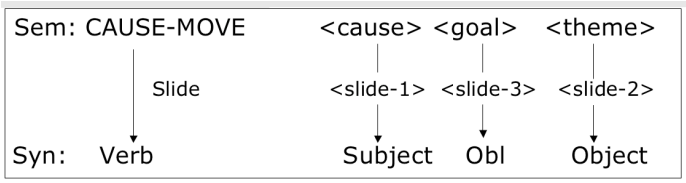

Figure 1: A linking construction relates a syntactic pattern such as Subject+Verb+Object+Oblique with a semantic frame such as CAUSE-MOVE+cause+goal+theme

```
((unit1 (sem-subunits (unit1b)))
 (unit1b
    (sem-cat (event ?x-2))
    (sem-subunits (unit3 unit2)))
 (unit2
    (referent ?x-1)
    (meaning ((jill ?x-1)))
    (sem-cat ((animate ?x-1))))
 (unit3
    (referent ?x-2)
    (meaning ((walk ?x-2) (walker ?x-2 ?x-1)))
    (sem-cat ((motion-event ?x-2) (agent ?x-2 ?x-1)))))
```

Figure 2: Example of semantic structure for the sentence "Jill walks"

the slide-event) into the CAUSE-MOVE frame and to categorise the syntactic properties of the sentence (specific lexical categories, word order, agreement, etc.) into a syntactic pattern such as SVO+Oblique.

3 Some brief introduction to FCG

In FCG, language processing consists in building up the semantic and syntactic aspects of a sentence structure represented as feature structures. One is not done before the other (as in strictly modular approaches to language processing) but both are built up at the same time. Following other feature-structure based formalisms, the sentence structure will consist of units with slots containing information about the unit. We use a standard prefix notation for feature-structures (similar to LISP datastructures). There is no ordering among the slots and a slot contains always a set of elements which are themselves also unordered. The elements are either atomic, or expressions in predicate calculus notation, or variables (which are atomic symbols preceded with a question mark as in ?*event*). In contrast to other feature structure formalisms [13] feature structures are not used hierarchically as fillers of slots. All units are located at the same level but are labeled (unit-1, unit-2, etc.) and hierarchical structure is explicitly represented with the slots syn-subunits (for syntactic subunits) and sem-subunits (for semantic subunits). The labels allow reference to a unit, predication over units, for example to express ordering constraints, and the use of variables that become bound to specific units. Basically there is a unit for every word in the sentence and for every group of words that forms a larger unit. But there may be units which have only a semantic role (i.e. they have no direct analog in the form) and units which have only a syntactic role (e.g. for grammatical function words like "by" which have no direct analog in meaning).

The main slots of a unit for the semantic aspects of a sentence are:

1. Referent: which is the entity in the world-model the unit is about (or a variable that will become bound to an entity)

2. Meaning: the clauses that will be used to identify the referent.

3. Sem-cat: the semantic categorisations in which the entity referred to by the unit participates.

```
((unit1
   (syn-cat (sentence))
   (syn-subunits (unit1b)))
 (unit1b
  (syn-cat (sv-sentence))
  (form ((precedes unit2 unit3)))
  (syn-subunits (unit2 unit3)))
 (unit2
  (form ((string unit2 jill) (stem unit2 jill)))
  (syn-cat (np (person 3d) (number singular))))
 (unit3
  (form ((string unit3 walks) (stem unit3 walk)))
  (syn-cat
   ((lex-cat verb) (person 3d) (number singular)))))
```

Figure 3: Example of syntactic structure for the sentence "Jill walks".

.

4. Sem-subunits: the set of subunits of this unit from the viewpoint of semantic structure.

An example for the sentence "Jill walks" is given in figure 2. From a semantic point of view the walk-event is categorised as a motion-event and the walker is the agent of the event. We will discuss this example further throughout this paper. The main slots of a unit on the syntactic side are:

1. Utterance: this is the string itself that is covered by the unit (possibly a multi-word phrase). It is not explicitly listed in the examples that follow.

2. Form: which is a declarative description of the form in terms of strings, stems, affixes, and precedence or sequencing relations between units

3. Syn-cat: the syntactic categorisations that are associated with the unit, such as number or gender.

4. Syn-subunits: the set of subunits of this unit from the viewpoint of syntax.

An example is given in figure 3.

FCG uses a logic-based representation for the syntactic and semantic categorisations and the constraints on semantic or syntactic structures that characterise a particular construction. Semantic categories are predicates over the entities in the domain of discourse which are (re-)conceptualisations of the predicates directly derived from the sensory datastreams, such as '(motion-event ?x-2) (agent ?x-2 ?x-1)'. Syntactic categories are also expressed in predicate-calculus and may predicate over units if they involve relations like precedence. '(precedes unit2 unit3) (lex-cat verb) (person 3d) (number singular)'. The final form of a sentence is described in a declarative fashion. For example, "Jill walks" is represented as '(string unit1 "jill") (string unit2 "walks") (precedes unit1 unit2)'. The explicit representation of ordering relations makes it much easier to deal with relatively free word order or parse sentences even if the word order constraints are partially violated. A logic-based representation makes it furthermore possible to invent new syntactic and semantic categories when the need arises, which is important for implementing learning and the emergence of new conventions.

In line with other unification grammar formalisms, language processing in FCG is viewed as a kind of inference process and mechanisms pioneered in logic programming and constraint propagation are employed to implement parsing and production. The sentence structure (syntactic and semantic) can be viewed as the 'fact base' against which templates are matched. The filler of a slot in a template specifies which elements have to be present in the structure being matched. In contrast to other unification-based formalisms, the templates do not use numerical indices for referring back to other parts of a structure, but variables, which can be bound not only to specific

items, such as the value for the syntactic feature person, but also to units themselves (as ?unit in figure 4).

The bi-directional application of a template uses two subfunctions: Match and Merge. Match is equivalent to the standard unification operation familiar from logic programming. Merge is equivalent to subset unification, which not only binds variables but also adds all parts from the merging structure to the merged structure. It is similar to the unification operation used in most unification-based grammars.

1. PRODUCING (i.e. go from meaning to form)
   (a) Match the left pole of a template against the structure already built. If the template matches, this yields a set of bindings for the variables or a set of equalities between variables (because the sentence structure being built may also contain variables).
   (b) Merge the right pole of the template with the sentence structure. Merging means that the instantiated right pole is first matched against the sentence structure, possibly yielding additional bindings, and then the union is taken of the further instantiated right pole and the sentence structure.
2. PARSING (i.e. go from form to meaning)
   (a) Match the right pole of a template against the sentence structure. If the template matches, this yields a set of bindings for the variables or a set of equalities between variables (because the setence structure being built may also contain variables).
   (b) Merge the left pole of the template with the sentence structure. Merging means that the instantiated left pole is first matched against the sentence structure, possibly yielding additional bindings, and then the union is taken of the further instantiated left pole and the sentence structure.

So producing and parsing are totally analogous, the only thing which changes is the direction of template application. The type of the template determines which part of the sentence structure will be used in matching and merging. For example, in the case of a syn-template both the left and the right pole are syntactic, whereas in the case of a con-template the left pole is semantic and the right pole syntactic. Linking is computationally achieved as a side effect of the merge-operator in parsing.

Although technically speaking there is no strict distinction in FCG between lexicon and grammar (i.e. they are both relating aspects of semantic structure to aspects of syntactic structure and use exactly the same formal structure), we find it useful to make a distinction between different template-sets for clarity during debugging. The templates making up the lexicon are further subdivided into:

1. Morphological templates (morph-templates), which decompose words into a stem with affixes and add syntactic categorisations, such as gender or part of speech (see figure 4).
2. Lexical stem templates (lex-stem-templates), which map the stem of words to their meaning (figure 5).
3. Lexical category templates (lex-cat-templates), which map 'natural' syntactic categories (like the number of a noun, gender in the case of animate entities, tense in the case of the main verb, etc.) to bits of meaning.

There are two additional FCG-operators used in the morph-template in figure 4. The == (pronounced 'includes') operator specifies a special case of matching, specifically that it is sufficient that the elements that follow == are members of the elements of the slot. For example the syn-cat in the right pole must include '(lex-cat verb)' but may contain also other specifications. The J-operator (described in detail in [5]) specifies hierarchical units which do not have to match but are added in merging. In this particular case, when the string "walks" is encountered in parsing (application from right to left of the template), a new unit is constructed and bound to ?walk-unit and made a daugther of the unit bound to ?top. In production, ?walk-unit is already bound and no new units need to be constructed in the right pole. These operators are also used in the example of a lex-stem template (figure 5). Note that the J-operator now occurs in the left pole. This template will carve out part of the global meaning and create a new unit which has the stem "walk".

```
(def-morph "walks"
    ((?top (syn-subunits (== ?walk-unit)))
     (?walk-unit
       (syn-cat (== (lex-cat verb)(person 3d)(number singular)))
       (form (== (stem ?walk-unit "walk")))))
  <-->
  ((?top (form (== (string ?walk-unit "walks"))))
   ((J ?walk-unit ?top))))
```

Figure 4: Example of a morphological template that associates a stem and a number of syntactic categories with the string "walks".

```
(def-lex-stem walk-entry
  ((?top
    (meaning (== (walk ?ev) (walker ?ev ?obj1))))
   ((J ?walk-unit ?top) (referent ?ev)))
  <-->
  ((?top (syn-subunits (== ?walk-unit)))
   (?walk-unit (form (== (stem ?walk-unit "walk"))))))
```

Figure 5: Example of a lexical stem template that specifies the meaning associated with the stem "walk"

4 Linking with Construction Templates

Linking is licensed by templates that map semantic frames to grammatical patterns. These are called con-templates - for construction templates. Semantic categorisation templates (sem-templates) (re-)conceptualise the meaning in terms of the semantic frames that occur in constructions and syntactic categorisation templates (syn-templates) specify how a grammatical pattern is realised in terms of more specific syntactic features. For example, sem-templates map the arguments of a specific predicate, like [slide] into the semantic roles of the more abstract CAUSE-MOVE frame. Syn-templates map the grammatical relations like subject, direct-object, etc. into syntactic categories, orderings, and agreement phenomena. It could be envisioned to represent all these in a single construct (and this might also be the typical state in earlier stages of the grammar), but that would make it impossible for the same syntactic pattern or the same semantic frame to be used in more than one construction.

Figure 6 shows an example of a construction template. The J-operator uses now a third argument which pulls additional units into subunits of the newly created unit (in this case ?event-unit and ?agent-unit are pulled in as subnodes of the new ?SV-unit which is itself a subnode of the ?top-unit). Production starts from the meaning '((walk ev1) (walker ev1 obj1) (Jill obj1) (status obj1 single-object) (discourse-role obj1 external))' which is decomposed into several units with lex-stem templates and reconceptualised with the sem templates as '((motion-event ev1) (agent ev1 obj1))', yielding the semantic structure similar to that in figure 2 except that all variables have instantiated values. This matches with the left pole of the construction, which then merges the information in the right pole with the syntactic structure built so far, yielding something similar to that in figure 3 but still without specific strings yet (they are added by the morph rules). Note that a new SV-unit is created both on the semantic side and on the syntactic side and that precedence relations and agreement in terms of person and number constrain the right pole.

Linking is done as part of the parsing process. Initially the lexicon contributes the meanings of individual words, and these are expanded with sem templates to add semantic categories, yielding:

```
((unit1 (sem-subunits (unit2 unit3)))
 (unit2
  (referent ?x-1)
  (meaning ((jill ?x-1)))
  (sem-cat ((animate ?x-1))))
```

```
(def-con Motion-Event
    ((?top-unit
      (sem-subunits (== ?event-unit ?agent-unit)))
     (?event-unit
      (referent ?event)
      (sem-cat (== (motion-event ?event)
   (agent ?event ?agent))))
     (?agent-unit
      (referent ?agent)
      (sem-cat (== (animate ?agent))))
     ((J ?SV-unit ?top-unit (?event-unit ?agent-unit))
      (sem-cat (event ?event))))
  <-->
  ((?top-unit
    (syn-subunits (== ?agent-unit ?event-unit))
    (form (== (precedes ?agent-unit ?event-unit))))
   (?event-unit
    (syn-cat (== (lex-cat verb) (number ?number) (person ?person))))
   (?agent-unit
    (syn-cat (== NP (number ?number) (person ?person))))
   ((J ?SV ?top-unit (?event-unit ?agent-unit))
    (syn-cat (== SV-sentence)))))
```

Figure 6: Example of a construction template as usable for "Jill walks". On the semantic side, it assumes a motion-event with an animate agent. On the syntactic side, a noun-phrase and a verb grouped into an SV-sentence.

```
 (unit3
  (referent ?x-2)
  (meaning ((walk ?x-2) (walker ?x-2 ?x-3)))
  (sem-cat ((move-event ?x-2) (agent ?x-2 ?x-3)))))
```

Note that the variables ?x-1 and ?x-3 are not yet equalised. The syntactic structure matches completely with the right pole of the Motion-Event construction and the semantic structure is then merged (unified) with the left pole to arrive at the semantic structure in figure 2. As part of the merging process, the bindings: ?agent/?x-1, ?event/?x-2, and ?agent/?x-3 were obtained and after merging ?x-1 and ?x-3 are equalised to yield the semantic structure shown in figure 2.

5 Conclusions

This paper reported progress in the development of Fluid Construction Grammar, a formalism designed for supporting experiments in emergent grammar. We focused in particular on the issue of linking, i.e. the combining of the meanings of individual words to construct the meaning of the whole. It turns out that focusing on how variables become equal is a very good approach for framing the linking process and unification (as part of the merging process) is a straightforward way to implement it. We have seen that the constructions and other templates used by the grammar can be used both in parsing and production. In [18] we discuss in more detail how speakers can themselves discover (by re-entrance) that certain variables need to be equalised and how constructions with associated semantic and syntactic categorisation templates can be invented and learned.

Research funded by Sony CSL with additional funding from the EU FET-ECAgents project. Joachim De Beule is a VUB research assistent.

References

[1] Bergen, B.K. and N.C. Chang. (2003) *Embodied Construction Grammar in Simulation-Based Language Understanding.* In J-O Ostman and M. Fried (Eds.) Construction Grammar(s): Cognitive and Cross-Language Dimensions John Benjamin Publ Cy. Amsterdam.

[2] Bryant, J. (2004) Scalable Construction-Based Parsing and Semantic Analysis. In Proceedings of the Second International Workshop on Scalable Natural Language Understanding. Boston 2004

[3] Chang, N.C. and T.V. Maia. 2003. *Learning Grammatical Constructions* In: Cohen, P. and T. Oates (2001) p. 10-16.

[4] Clark, H. H., and Brennan, S. A. (1991). Grounding in communication. In: Resnick, L.B., J.M. Levine, and S.D. Teasley (Eds.). Perspectives on socially shared cognition. Washington: APA Books.

[5] De Beule, J. and L. Steels (2005) Hierarchy in Fluid Construction Grammar. In: Furbach, U. (eds) (2005) Proceedings of KI-2005. Lecture Notes in AI 3698. Springer-Verlag, Berlin. p.1-15.

[6] Goldberg, A.E. 1995. *Constructions.: A Construction Grammar Approach to Argument Structure.* University of Chicago Press, Chicago

[7] Feldman, J. and S. Narayanan (2003). Embodied Meaning in a neural theory of language . Brain and Language. to appear.

[8] Fried, M. and J-O Ostman (2004) Construction Grammar: A thumbnail sketch. Chapter 2. In: Fried, M. and J-O Ostman (2004) Construction Grammar in a Cross-Language Perspective. John Benjamins Pub. Cy. Amsterdam. pp. 11-87.

[9] Traugott, E. and Heine, B. (1991) Approaches to Grammaticalization. Volume I and II. John Benjamins Publishing Company, Amsterdam.

[10] Hopper, P. (1987) Emergent grammar. Berkeley Linguistics Conference (BLS), 13:139–157

[11] Kay, P. and C. Fillmore (1999) Grammatical Constructions and Linguistic Generalizations: the What's X doing Y? Construction Language 75, pp. 1-33.

[12] Michaelis, L. 2004. *Entity and Event Coercion in a Symbolic Theory of Syntax* In J.-O. Oestman and M. Fried, (eds.),Construction Grammar(s): Cognitive Grounding and Theoretical Extensions. Constructional Approaches to Language series, volume 2. Amsterdam: Benjamins.

[13] Pollard, C. and I. Sag 1994. *Head-driven phrase structure grammar.* Center for the Study of Language and Information of Stanford University. University of Chicago Press, Chicago.

[14] Steels, L., F. Kaplan, A McIntyre and J. Van Looveren (2001) Crucial factors in the origins of word-meaning. In Wray, A., editor, The Transition to Language, Oxford University Press. Oxford, UK, 2002.

[15] Steels, L. (2003) Evolving grounded communication for robots. Trends in Cognitive Science. Volume 7, Issue 7, July 2003 , pp. 308-312.

[16] Steels, L. (2004) Constructivist Development of Grounded Construction Grammars Scott, D., Daelemans, W. and Walker M. (eds) (2004) Proceedings Annual Meeting Association for Computational Linguistic Conference. Barcelona. p. 9-19.

[17] Steels, L. (2005) What triggers the emergence of grammar?. AISB'05: Proceedings of the Second International Symposium on the Emergence and Evolution of Linguistic Communication (EELC'05), pages 143-150, Hatfield, 2005. University of Hertfordshire

[18] Steels, L. (2005) The Emergence and Evolution of Linguistic Structure: From Lexical to Grammatical Communication Systems. Connection Science. Vol. 17(3)

[19] Talmy, L. (2000) Toward a Cognitive Semantics: Concept Structuring Systems (Symbol, Speech, and Communication) The MIT Press, Cambridge Ma.