A (very) Brief Introduction to Fluid Construction Grammar

Luc Steels(1,2) and Joachim de Beule(1)
(1) University of Brussels (VUB AI Lab)
(2) SONY Computer Science Lab - Paris
steels@arti.vub.ac.be

Abstract

Fluid Construction Grammar (FCG) is a new linguistic formalism designed to explore in how far a construction grammar approach can be used for handling open-ended grounded dialogue, i.e. dialogue between or with autonomous embodied agents about the world as experienced through their sensory-motor apparatus. We seek scalable, open-ended language systems by giving agents both the ability to use existing conventions or ontologies, and to invent or learn new ones as the needs arise. This paper contains a brief introduction to the key ideas behind FCG and its current status.

1 Introduction

Construction grammar is receiving growing attention lately, partly because it has allowed linguists to discuss a wide range of phenomena which were difficult to handle in earlier frameworks (Goldberg, 1995; OstmanFried, 2005; Croft, 2001), and partly because it has allowed psychologists to describe in a more satisfactory way early language development (TomaselloBrooks, 1999). There were already some attempts to formalise construction grammar (KayFillmore, 1999) and build a computational implementation (BergenChang, 2003), but many open problems remain and at this early stage of fundamental research, it makes sense to explore alternative approaches. In our team, we focus on open-ended grounded dialogue, in other words how it is possible for a speaker to formulate an utterance about the world and for a hearer to understand what is meant (ClarkBrennan, 1991). The present paper briefly reports on the formalisation of construction grammar called Fluid Construction Grammar (FCG) that we have developed for this research. Although the formalism is novel in several fundamental aspects, it also builds heavily on the state of the art in formal and computational linguistics, particularly within the tradition of unification-based feature structure grammars such as HPSG (PollardSag, 1994). FCG has been under development from around 2001 and an implementation on a LISP substrate has been released through http://arti.vub.ac.be/FCG/ in 2005. The FCG core engine (for parsing and production) is fully operational and has already been used in some large-scale experiments in language grounding (SteelsLoetzsch, 2006). We do not claim to have a complete solution for all linguistic issues that arise in construction grammar, and neither do we claim that the solutions we have adopted so far are final. On the contrary, we are aware of many difficult technical issues that still remain unresolved and welcome any discussion that would bring us forward.

2 Motivations

FCG grew out of efforts to understand the creative basis of language. Language creativity is more than the application of an existing set of rules (even if the rules are recursive and thus allow an infinite set of possible sentences). Human language users often stretch and expand rules whenever the need arises,

Proceedings of the 3rd Workshop on Scalable Natural Language Understanding, pages 73–80,
New York City, June 2006.

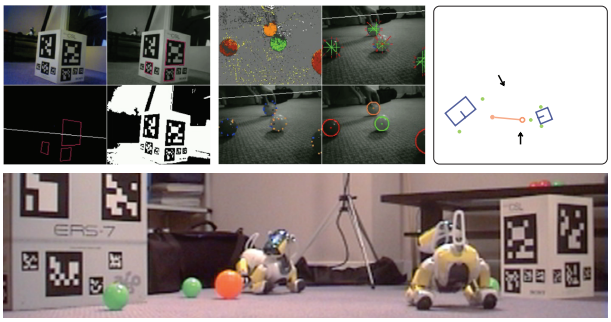

Figure 1: Typical experimental setup. The bottom shows two robots moving around in an environment that contains balls and boxes. The robots are equiped with a complex sensory-motor system, able to detect the objects and build an analog world model of their location and trajectories (as shown in the right top corner).

and occasionally invent totally new ones. So we need to understand how new aspects of language (new concepts and conceptualisations, new lexical items, new syntactic and semantic categories, new grammatical constructions, new interaction patterns) may arise and spread in a population, the same way biologists try to understand how new life forms may arise (Steels, 2003).

This motivation leads immediately to some requirements. First of all we always use multi-agent simulations so that we can investigate the spreading of conventions in a population. Agents take turns being speaker and hearer and build up competences in conceptualisation and verbalisation (for production) and parsing and interpretation (for understanding). They must be able to store an inventory of rules and apply them in either processing direction, and they must be able to expand their inventories both by inventing new constructions if necessary and by adopting those used by others. Second, the agents must have something to talk about. We are interested in grounded language, which means dialogue about objects and events in the world as perceived through a sensory-motor apparatus. We take embodiment literally. Our experiments use physical robots (Sony AIBOs) located in a real world environment (see figure 1 from (SteelsLoetzsch, 2006)) Third, the agents must be motivated to say and learn something. We achieve this by programming the robots with scripts to play language games. A language game sets up a joint attentional frame so that robots share general motives for interaction, a specific communicative goal (for example draw attention to an object), and give feedback to enable repair of miscommunication (for example through pointing). We typically perform experiments in which a population of agents starts with empty conceptual and linguistic repertoires and then builds from scratch a communication system that is adequate for a particular kind of language game. Agents seek to maximise communicative success while minimising cognitive effort.

One advantage of grounded language experiments is that we can clearly monitor whether the capacities given to the agents are adequate for bootstrapping a language system and how efficient and successful they are. By starting from scratch, we can also test whether our objective of understanding language creativity has been achieved. Of course such experiments will never spontaneously lead to the emergence of English or any other human language, but we can learn a great deal about the processes that have given rise and are still shaping such languages.

3 Meaning

The information about an utterance is organized in a semantic and a syntactic structure. The semantic structure is a decomposition of the utterance's meaning and contains language-specific semantic re-categorisations (for example a put-event is categorised as a cause-move-location with an agent, a patient and a location). The syntactic structure is a decomposition of the form of the utterance into constituents and morphemes and contains additional syntactic categorisations such as syntactic features (like number and gender), word order constraints, etc.

We follow a procedural semantics approach, in the sense that the meaning of an utterance is a program that the hearer is assumed to execute (Winograd, 1972; Johnson-Laird, 1997). Hence conceptualisation becomes a planning process (to plan the program) and interpretation becomes the execution of a program. For example, the meaning of a phrase like "the box" is taken to be a program that involves the application of an image schema to the flow of perceptual images and anchor it to a partic-

ular physical object in the scene. So we do not assume some pre-defined or pre-processed logic-style fact base containing the present status of the world (as this is extremely difficult to extract and maintain from real world perception in a noisy and fast changing world) but view language as playing an active role in how the world is perceived and categorised. It is in principle possible to use many different programming languages, but we have opted for constraint based processing and designed a new constraint programming language IRL (Incremental Recruitment Language) and implemented the necessary planning, chunking and execution mechanisms of constraint networks (SteelsBleys, 2005). A simple example of a constraint network for "the box" is as follows[1]:

```
1. (equal-to-context ?s)
2. (filter-set-prototype ?r ?s ?p)
3. (prototype ?p [box])
4. (select-element ?o ?r ?d)
5. (determiner ?d [single-unique])
```

`Equal-to-context, select-element`, etc. are primitive constraints that implement fundamental cognitive operators. `Equal-to-context` grabs the set of elements in the current context and binds it to `?s`. `Filter-set-prototype` filters this set with a prototype `?p` which is bound in (3) to `[box]`. `Select-element` selects an element `?o` from `?r` according to the determiner `?d` which is bound to `[single-unique]` in (5), meaning that `?r` should be a singleton. The constraints are powerful enough to be used both in interpretation, when semantic objects such as prototypes, determiners, categories, relations, etc. are supplied through language and values need to be found for other variables, and in conceptualisation, when these values are known but the objective is to find the semantic objects. Moreover, during conceptualization the constraints may extend the repertoire of semantic objects (e.g. introducing a new prototype) if needed, allowing the agents to progressively build up their ontologies.

[1]We use prefix notation. Order does not play a role as the constraint interpreter cycles through the network until all variables are bound or until no further progress can be made. Symbols starting with a question mark represent variables.

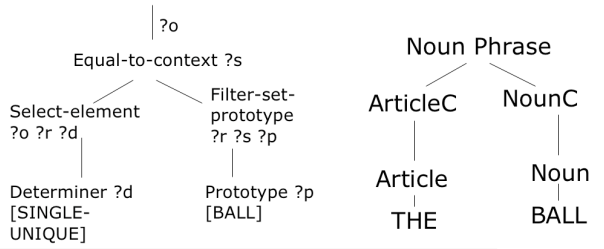

Figure 2: Left: decomposition of the constraint program for "the ball" in the semantic structure. Right: related syntactic structure. In reality both structures contain a lot more information.

4 Syntactic and Semantic Structures

As mentioned, FCG organises the information about an utterance in feature structures, similar to other feature-structure based formalisms (as first introduced by Kay (Kay, 1984)) but with some important differences. An FCG feature structure contains units which correspond (roughly) to words (more precisely morphemes) and constituents.

A unit has a name and a set of features. Hierarchical structure is not implicitly represented by embedding one unit in another one, but explicitly by the features *syn-subunits* (for the syntactic structure) and *sem-subunits* (for the semantic structure). There is a strong correspondence between the syntactic and semantic structure built up for the same utterance (see figure 2) although there can be units which only appear in the syntactic structure (for example for grammatical function words) and vice versa. The correspondence is maintained by using the same unit names in both the semantic and syntactic structure. Units in syntactic structures have three features: (1) *syn-subunits*, (2) *syn-cat* which contains the syntactic categories, and (3) *form* containing the form associated with the unit. Units in semantic structures have four features: (1) *sem-subunits*, (2) *sem-cat* containing the semantic categories, (3) *meaning* which is the part of the utterance's meaning covered by the unit, and (4) *context* which contains variables that occur in the meaning but are 'external' in the sense that they are linked to variables occurring in the meaning of other units. An example semantic structure (in list-notation) for the left structure in figure 2 is shown in figure 3. FCG is a completely open-ended formalism in the sense that all linguistic

```
((unit-1
    (SEM-SUBUNITS (unit-3)))
 (unit-3
    (CONTEXT ((LINK ?s)))
    (MEANING ((EXTERNAL-CONTEXT ?s)))
    (SEM-SUBUNITS (unit-4 unit-5)))
 (unit-4
    (CONTEXT ((LINK ?o ?r)))
    (MEANING ((SELECT-ELEMENT ?o ?r ?d)))
    (SEM-SUBUNITS (unit-6)))
 (unit-5
    (CONTEXT ((LINK ?r ?s)))
    (MEANING ((FILTER-SET-PROTOTYPE ?r ?s ?p)))
    (SEM-SUBUNITS (unit-7)))
 (unit-6
    (CONTEXT ((LINK ?d)))
    (MEANING ((DETERMINER ?d [SINGLE-UNIQUE]))))
 (unit-7
    (CONTEXT ((LINK ?p)))
    (MEANING ((PROTOTYPE ?p [BALL])))))
```

Figure 3: Semantic structure in list-notation.

categories (syntactic or semantic) are open and in principle language-specific (as in radical construction grammar (Croft, 2001).) Thus the set of lexical categories (noun, verb, adjective, etc.), of possible semantic roles (agent, patient, etc.), of syntactic features (number, gender, politeness, etc.), and so on, are all open. The value of the *syn-cat* and *sem-cat* features consists of a conjunction of predicates (each possibly having arguments.) New categories can be introduced at any time and used as (part of) a predicate. The form of the utterance is described in a declarative manner, using predicates like *precedes* or *meets* which define linear ordering relations among the form of units or any other aspect of surface form including prosodic contour or stress.

5 Rules

A rule (also called template) typically expresses constraints on possible meaning-form mappings. Each rule has a score which reflects the success that the agent has had in using it. All else being equal, agents prefer rules with higher scores, thus reflecting frequency effects. A rule has two poles. A left pole which typically contains constraints on semantic structure formulated as a feature structure with variables, and a right pole which typically contains constraints on syntactic structure again formulated as a feature structure with variables. Rules are divided into rule subsets which help constrain the order of rule-application and design large-scale grammars. Thus we make a distinction between morph-rules, which decompose a word into a stem and pending morphemes and introduce syntactic categories; lex-stem-rules, which associate meaning with the stem as well as valence information and a role-frame; con-rules, which correspond to grammatical constructions that associate parts of semantic structure with parts of syntactic structure; and sem and syn-rules which perform inference over semantic or syntactic categories to expand semantic or syntactic structure.

All rules are bi-directional. Typically, during production, the left pole is 'unified' with the semantic structure under construction, possibly yielding a set of bindings. If successful, the right pole is 'merged' with the syntactic structure under construction. The merge operation can be understood as a partial unification, but extending the structure with those parts of the pole that were missing. During parsing, the right pole is unified with the syntactic structure and parts of the left pole are added to the semantic structure. The unification phase is thus used to see whether a rule is triggered and the merge phase represents the actual application of the rule. The FCG Unify and Merge operators are defined in great formal detail in (SteelsDeBeule, 2006). During production lex-stem-rules are applied before the con-rules and the morph-rules. During parsing the lex-stem-rules are applied right after the morph-rules. The con-rules then build higher order structure. It is enormously challenging to write rules that work in both directions but this strong constraint is very helpful to achieve a compact powerful grammar.

6 Building Hierarchy

One of the innovative aspects of FCG is the way it handles hierarchy. Both the left-pole and the right-pole of a construction can introduce hierarchical structure with the J-operator (DeBeuleSteels, 2005). This way, the semantic pole of constructions (lexical or grammatical) can decompose the meaning to be expressed (which originally resides in the top node of the semantic structure) and the syntactic pole can group units together into a larger constituent. Constraints governed by the J-operator do not have to match during the unification phase. Instead they are used to build additional structure during the merge

```
(def-lex-stem-rule put-SVOL
 ((?top
   (meaning
    (==
     (event-type ?event-type
      (put (put-1 ?obj);who
           (put-2 ?obj2);puts what
           (put-3 ?obj3))))));where
  ((J ?new-unit ?top)
   (context
    (== (link ?event-type)))
   (sem-cat
    (==
     (sem-event-type ?event-type
      (cause-move-location
       (agent ?obj)
       (patient ?obj2)
       (location ?obj3)))))))
<-->
 ((?top
   (syn-subunits (== ?new-unit)))
  (?new-unit
   (form
    (== (stem ?new-unit "put"))))
  ((J ?new-unit nil)
   (syn-cat
    (== (valence SVOL))))))
```

Figure 4: Example lexical entry for “put” and illustration of the J-operator.

phase. This may include the construction of a new unit as well as pending from an existing unit and absorbing some other units.

Figure 4 shows an example which will be used further in the next section. It is a lexical rule preparing a resultative construction (GoldbergJackendoff, 2004). The semantic pole of the rule combines some stretch of meaning (the introduction of an event-type, namely a put-event) with a frame (cause-move-location with roles for agent, patient and location). These are associated with a lexical stem ”put” in the right pole which also adds a valence frame SVOL (triggering the subject-verb-object-location construction). In production, this rule triggers when a ‘put’ event-type is part of the meaning (‘==’ means ‘includes but may also contain additional expressions’). When merging the semantic pole with the semantic structure, a new unit hanging from ?top is created and the specified value of the meaning feature copied down. The new unit also receives the context and sem-cat features as specified by the J-operator. At the same time, the syntactic pole is merged with the syntactic structure and so the ?new-unit (which is already bound) is added as a subunit of ?top in the syntactic structure as well. The J-operator will then add stem and valence information. Thus the semantic structure of figure 5 will be transformed into the one of figure 6. And the corresponding syntactic structure becomes as in figure 7. In parsing, an existing syntactic unit with stem

```
((unit-2
  (meaning
   ( ..
     (event-type ev-type1
      (put (put-1 o1) (put-2 o11)
           (put-3 o22))) ... ))))
```

Figure 5: Semantic structure triggering the rule in figure 4 in production.

```
((unit-2
  (sem-subunits (... unit-3 ...)))
 (unit-3
  (meaning
   ((event-type
     ev-type1
     (put (put-1 o1) (put-2 o11)
          (put-3 o22)))))
  (context ((link ev-type1)))
  (sem-cat
   ((sem-event-type
     ev-type1
     (cause-move-location
      (agent o1) (patient o11)
      (location o22))))))
  ... )
```

Figure 6: Resulting semantic structure after applying the rule in figure 4 to the semantic structure of figure 5.

```
((unit-2
  (syn-subunits (... unit-3 ...)))
 (unit-3
  (form ((stem unit-3 "put")))
  (syn-cat ((valence SVOL))))
 ... )
```

Figure 7: Resulting syntactic structure after applying the rule in figure 4.

”put” is required to trigger the rule. If found, the rule will add the valence information to it and on

the semantic side the meaning as well as the semantic categorisation in terms of a cause-move-location frame are added.

7 Implementing Constructions

Lexical constructions provide frame and valence information for word stems and parts of meaning. Grammatical constructions bind all this together. Figure 8 shows an example of a grammatical construction. It also uses the J-operator to build hierarchy, both on the semantic side (to decompose or add meaning) and on the syntactic side (to group constituents together.) An example of a SVOL-construct is *Mary puts the milk in the refrigerator*. Before application of the construction, various units should already group together the words making up a nounphrase for the subject (which will be bound to ?subject-unit), a nounphrase for the direct object (bound to the ?object-unit) and a prepositional nounphrase (bound to ?oblique-unit). Each of these units also will bind variables to their referents, communicated as context to the others. On the semantic side the cause-move-location frame with its various roles aids to make sure that all the right variable bindings are established. On the syntactic side the construction imposes word-order constraints (expressed with the meets-predicate), the valence of the verb, and specific types of constituents (nounphrase, verbphrase, prepositional nounphrase). The SVOL construction operates again in two directions. In production it is triggered when the semantic structure built so far unifies with the semantic pole, and then the syntactic structure is expanded with the missing parts from the syntactic pole. Constraints on the syntactic pole (e.g. valence) may prevent the application of the construction. In parsing, the SVOL construction is triggered when the syntactic structure built so far unifies with the syntactic pole and the semantic structure is then expanded with the missing parts from the semantic pole. Again application may be constrained when semantic constraints in the construction prevent it.

8 Fluidity, Conventionalisation and Meta-grammars

Although FCG must become adequate for dealing with the typical phenomena that we find in human natural languages, our main target is to make scientific models of the processes that underly the origins of language, in other words of the creative process by which language users adapt or invent new forms to express new meanings that unavoidably arise in an open world and negotiate tacitly the conventions that they adopt as a group. We have already carried out a number of experiments in this direction and here only a brief summary can be given (for more discussion see: (Steels, 2004; DeBeuleBergen, 2006; SteelsLoetzsch, 2006)).

In our experiments, speaker and hearer are chosen randomly from a population to play a language game as part of a situated embodied interaction that involves perception, joint attention and feedback. When the speaker conceptualizes the scene, he may construct new semantic objects (for example new categories) or recruit new constraint networks in order to achieve the communicative goal imposed by the game. Also when the speaker is trying to verbalise the constraint network that constitutes the meaning of an utterance, there may be lexical items missing or new constructions may have to be built. We use a meta-level architecture with reflection to organise this process. The speaker goes through the normal processing steps, using whatever inventory is available. Missing items may accumulate and then the speaker moves to a meta-level, trying to repair the utterance by stretching existing constructions, re-using them by analogy for new purposes, or introducing other linguistic items. The speaker also engages in self-monitoring by re-entering the utterance and comparing what he meant to say to interpretations derived by parsing his own utterance. The speaker can thus detect potential problems for the listener such as combinatorial explosions in parsing, equalities among variables which were not expressed, etc. and these problems can be repaired by the introduction of additional rules.

The hearer receives an utterance and tries to go as far as possible in the understanding process. The parser and interpreter are not geared towards checking for grammaticality but capable to handle utterances even if a large part of the rules are missing. The (partial) meaning is then used to arrive at an interpretation, aided by the fact that the context and communicative goals are restricted by the language game. If possible, the hearer gives feedback on how

he understood the utterance and whether an interpretation was found. If there is failure or miscommunication the hearer will then repair his inventory based on extra information provided by the speaker. This can imply the introduction of new concepts extending the ontology, storing new lexical items, introducing new constructions, assigning certain words to new syntactic classes, etc. Speaker and hearer also update the scores of all rules and concepts. In case of success, scores go up of the items that were used and competitors are decreased to achieve lateral inhibition and hence a positive feedback loop between success and use. In case of failure, scores go down so that the likelihood of using the failing solution diminishes. In our simulations, games are played consecutively by members of a population and we have been able to show –so far for relatively simple forms of language– that shared communication systems can emerge from scratch in populations. Much work remains to be done in researching the repair strategies needed and when they should be triggered. The repair strategies themselves should also be the subject of negotiation among the agents because they make use of a meta-grammar that describes in terms of rules (with the same syntax and processing as the FCG rules discussed here) how repairs are to be achieved.

9 Conclusions

FCG is a tool offered to the community of researchers interested in construction grammar. It allows the precise formal definition of constructions in a unification-based feature structure grammar style and contains the necessary complex machinery for building an utterance starting from meaning and reconstructing meaning starting from an utterance. FCG does not make linguistic theorising superfluous, on the contrary, the formalism is open to any framework of linguistic categories or organisation of grammatical knowledge as long as a construction grammar framework is adopted. There is obviously a lot more to say, not only about how we handle various linguistic phenomena (such as inheritance of properties by a parent phrasal unit from its head subunit) but also what learning operators can progressively build fluid construction grammars driven by the needs of communication. We refer the reader to the growing number of papers that provide more details on these various aspects.

10 Acknowledgement

This research was conducted at the Sony Computer Science Laboratory in Paris and the University of Brussels VUB Artificial Intelligence Laboratory. It is partially sponsored by the EU ECAgents project (FET IST-1940). FCG and the experiments in language evolution are team work and major contributions were made by Joris Bleys, Martin Loetzsch, Nicolas Neubauer, Wouter Van den Broeck, Remy Van Trijp, and Pieter Wellens.

References

Bergen, B.K. and N.C. Chang. (2003) *Embodied Construction Grammar in Simulation-Based Language Understanding*. Technical Report 02-004, International Computer Science Institute, Berkeley.

Clark, H. and S. Brennan (1991) *Grounding in communication*. In: Resnick, L. J. Levine and S. Teasley (eds.) Perspectives on Socially Shared Cognition. APA Books, Washington. p. 127-149.

Croft, William A. (2001). *Radical Construction Grammar; Syntactic Theory in Typological Perspective*. Oxford: Oxford University Press.

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.

De Beule, J. and B. Bergen (2006) *On the Emergence of Compositionality*. Proceedings of the Evolution of Language Conference VI, Rome.

Goldberg, A.E. (1995) *Constructions.: A Construction Grammar Approach to Argument Structure*. University of Chicago Press, Chicago

Goldberg, A. and R. Jackendoff (2004) *The English Resultative as a Family of Constructions*. Language 80 532-568.

Johnson-Laird, P.N. (1997) *Procedural Semantics*. Cognition, 5 (1977) 189-214.

Goldberg, A. (2003) *Constructions: A new theoretical approach to language* Trends in Cognitive Science. Volume 7, Issue 5, May 2003 , pp. 219-224.

Kay, P. and C. Fillmore (1999) *Grammatical constructions and linguistic generalizations: the Whats X doing Y? construction*. Language 75(1), 133.

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

Pollard, C.and I. Sag (1994) *Head-driven Phrase Structure Grammar.* CSLI Stanford Univ, Calif.

Ostman, Jan-Ola and Mirjam Fried (eds.) (2005) *Construction Grammars: Cognitive grounding and theoretical extensions*. W. Benjamins, Amsterdam.

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

Steels, L. (2004) *Constructivist Development of Grounded Construction Grammars*. In D. Scott, W. Daelemans and M. Walker (Eds.), Proceedings Annual Meeting of Association for Computational Linguistics Conference. Barcelona: ACL , (pp. 9-16).

Steels, L. and J. De Beule (2006) *Unify and Merge in FCG*. In: Vogt, P. et.al. (eds.) Proceedings of EELC III. Lecture Notes in Computer Science. Springer-Verlag, Berlin.

Steels, L. and J. Bleys (2005) *Planning What To Say: Second Order Semantics for Fluid Construction Grammars*. In: Proceedings of CAEPIA 2005. Santiago de Compostella.

Steels, L. and M. Loetzsch (2006) *Perspective Alignment in Spatial Language*. In: Coventry, K., J. Bateman and T. Tenbrink (2006) Spatial Language in Dialogue. Oxford University Press. Oxford.

Tomasello, M. and P.J. Brooks (1999) *Early syntactic development: A Construction Grammar approach* In: Barrett, M. (ed.) (1999) *The Development of Language* Psychology Press, London. pp. 161-190.

Winograd, T. (1972) *Understanding natural language*. New York, Academic Press.

```
(def-con-rule SVOL-Phrase
  ((?top
    (sem-subunits
      (== ?subject-unit ?verb-unit
       ?object-unit ?oblique-unit)))
   (?subject-unit
    (context (== (link ?subject))))
   (?verb-unit
    (context (== (link ?event ?event-type)))
    (sem-cat
      (== (sem-event-type ?event-type
           (cause-move-location
            (agent ?subject)
            (patient ?object)
            (location ?oblique))))))
   (?object-unit
    (context (== (link ?object))))
   (?oblique-unit
    (context (== (link ?oblique))))
   ((J ?new-unit ?top
       (?subject-unit ?verb-unit
        ?object-unit ?oblique-unit))
    (context (== (link ?event)))))
 <-->
  ((?top
    (form
      (==
       (meets ?subject-unit ?verb-unit)
       (meets ?verb-unit ?object-unit)
       (meets ?object-unit
              ?oblique-unit)))
    (syn-subunits
      (== ?subject-unit ?verb-unit
       ?object-unit ?oblique-unit)))
   (?subject-unit
    (syn-cat
     (== (constituent NounPhrase))))
   (?verb-unit
    (syn-cat
      (== (constituent VerbPhrase)
          (valence SVOL))))
   (?object-unit
    (syn-cat
     (== (constituent NounPhrase))))
   (?oblique-unit
    (syn-cat
     (== (constituent PrepNounPhrase))))
   ((J ?new-unit ?top
       (?subject-unit ?verb-unit
        ?object-unit ?oblique-unit))
    (syn-cat
     (== (constituent sentence))))))
```

Figure 8: A resultative construction.