Argument Structure in Fluid Construction Grammar

Remi van Trijp (remi@csl.sony.fr)
SONY Computer Science Laboratory Paris, 6, Rue Amyot, 75005 Paris, France

NOTE: This document is the English manuscript which served for translation into German. For citation, **please use the OFFICIAL REFERENCE only**:

van Trijp, Remi (2008). Argumentsstruktur in der Fluid Construction Grammar. In: Fischer, Kerstin and Anatol Stefanowitsch (eds.). *Konstruktionsgrammatik II. Von der Konstruktion zur Grammatik*. Stauffenburg Linguistik Band 47, Tübingen: Stauffenburg Verlag.

Abstract

Construction grammar theories argue that the selection and realization of a verb's arguments is largely taken care of by constructions, as opposed to lexicalist theories which constrain the argument structure of a verb in the lexicon. So far, however, no operational definition has been given which shows how this 'fusion' of a verb's arguments with the roles of a construction can be achieved. This paper reports a fully implemented solution in Fluid Construction Grammar in which the lexical entry of the verb lists its 'potential valences' from which constructions can select.

1 Introduction

Most linguists nowadays agree that there is a strong connection between the semantics of a verb and the syntactic frames or sentence types in which they can occur (also known as the question of 'argument realization'). Especially since Fillmore (1968), most linguistic theories assume that this interface between semantics and syntax is regulated by a system of 'semantic roles' such as agent, patient, goal, etc. (also called 'thematic', 'case', 'teta', or 'argument' roles). Unfortunately, the exact status of these semantic roles and their place within grammar remains a largely unresolved issue.

A widespread approach is the lexicalist account which assumes that there is a relatively small set of semantic roles. A subset of these semantic roles is listed in the lexical entry of a verb and makes up the verb's 'case frame' (Fillmore, 1968) or 'predicate frame' (Dik, 1997). For example, the verb *break* may be listed as a two-place predicate which assigns the roles of 'agent' and 'patient' to its arguments. It is assumed that the syntactic behaviour of the verb can be directly predicted from its case frame, and that the integration of lexical semantics with syntactic structures is done through a limited and universal number of mapping principles or rules. Theories on the kinds of mapping principles range from the one-to-one 'Projection Principle' in Government and Binding theory to more complex interfaces found in e.g. Lexical-Functional Grammar (Bresnan, 1982) and Pinker (1989).

Recently, however, the lexicalist account has come under serious criticism, most notably from various constructionist approaches. Goldberg (1995, p. 9-14) argues that lexicalists have to posit implausible verb senses in the lexicon. For example, a sentence like *she sneezed the napkin off the table* (example 8 in ibid. at p. 9) counts as evidence that the verb *sneeze* has a three-argument sense of 'X CAUSES Y TO MOVE Z by sneezing'. Goldberg also points to circularity in lexicalist theories: a verb like *kick* – which can occur in at least eight different argument structures – would be listed as a two-place predicate in sentences such as *he kicked the wall*, as a three-place predicate in sentences such as *he kicked him a ball*, etc. Goldberg concludes that lexicalists use these examples both for defending the claim that *kick* has *n*-ary arguments AND to explain

which syntactic complements *kick* can take (p. 11). The lexicalist approach also fails to explain coherent semantic interpretations in creative language use and coercion effects as in *A gruff 'police monk'* ***barks*** *them back to work* (Michaelis, 2003, p. 261: example 6b).

The problems with the lexicalist approach are mostly due to a view on grammar which treats semantics (fully specified in the lexicon) and syntax as separate modules. As an alternative, construction grammarians put forward the hypothesis that grammatical knowledge consists of the so-called 'syntax-lexicon continuum' (Croft, 2005) in which even argument structures are regulated through conventionalized form-meaning mappings. In other words, a speaker's choice for a certain argument structure does not solely depend on the semantics of the verb that is used, but also (and to a large extent) on the argument structure constructions, which carry meaning themselves. One well-known example of such an argument structure construction is the English ditransitive construction, as in *I gave him a book*, which maps the meaning 'X CAUSES Y TO RECEIVE Z' onto a syntactic pattern (Goldberg, 1995, p. 141-151).

In Goldbergian construction grammar, the lexical entry of a verb does not contain a list of semantic roles, but rather its specific 'participant roles'. For example, *bake* has at least two participants: a *baker* and something being *baked*. Besides occurring in transitive structures such as in example (1), *bake* can also occur in the ditransitive construction as in example (2). In the latter case, the ditransitive construction is hypothesized to add the role of 'beneficiary' to the meaning.

(1) I baked a cake.
(2) I baked him a cake.
(3) *I drove him a car.

Construction grammarians of course also have to explain why some verbs are not compatible with certain constructions. As shown in example (3), *drive* cannot occur in the same ditransitive construction meaning something like 'I drove the car (to him) with the intention of giving it to him'. Goldberg (1995, p. 50) argues that verb-specific participant roles have to be 'fused' with the semantic roles of a construction. Roughly speaking, only roles which are semantically compatible can be fused with each other (i.e. the 'semantic coherence principle'), and the profiled arguments of the verb have to be compatible with the role profile of the construction (i.e. the 'correspondence principle').

Unfortunately, the exact nature of this 'fusion' process remains highly underspecified and suffers from circularity in its definition. This paper offers a fully implemented operationalization of how lexical entries and argument structure constructions can interact with each other in Fluid Construction Grammar (FCG). The next section first introduces the examples that are used for illustrating our approach and gives a brief overview of linguistic processing in FCG. Next, the function and application of lexical entries and constructions are shown and explained.

2 Sweeping the floor in Fluid Construction Grammar

Fluid Construction Grammar (FCG) is a computational grammar formalism which has primarily been developed to support research on the emergence and evolution of grammar (Steels, 2006). This research consists of computational simulations and robotic experiments in which a population of 'artificial agents' self-organize a shared communication system. While these experiments involve the evolution of artificial languages, the linguistic relevance of FCG as a computational implementation of construction grammar has always been a point of attention.

One of the current projects of our team investigates the origins and emergence of argument structure constructions (Steels, 2004; van Trijp, 2008). In the experiment, a population of embodied artificial agents describe to each other the events that they observe in recorded scenes with puppets (Steels & Baillie, 2003). In order to increase their communicative success, to reduce the cognitive effort needed for communication, and to avoid ambiguity, these agents gradually build a shared grammatical language which features case-like markers for expressing who's doing what in the events. One of the goals of the experiment is to identify the minimal requirements that are needed to enable the agents to develop such a (grammatical) language in terms of general cognitive mechanisms, communicative interaction patterns, etc.

The experiment is directly relevant to construction grammar theories because it offers a fully operational implementation of how lexical entries of verbs can interact with constructions to yield these event descriptions. Indeed, one of the biggest challenges in the experiment was to find a representation which could support phenomena such as found in the following examples (4-6 were provided by Laura Michaelis, personal communication):

(4) He sweeps the floor.
(5) He sweeps the dust off the floor.
(6) *He sweeps the dust.
(7) *He sweeps the floor out of the room.

In order to account for the acceptability of examples (4) and (5), lexicalist approaches would have to posit either two distinct predicate frames in the lexical entry of *sweep*, or they would have to include two different lexical entries for the verb. In a constructionist account, only one lexical entry should be posited and two different constructions then select how the argument structure is expressed. Similarly, the unacceptability of sentences (6) and (7) are accounted for by constraints defined in the constructions rather than in the lexical entry of the verb.

This paper makes a simplification of these sentences as can be observed in examples (8) and (9). This simplification makes it possible to show their representation as they actually occur in the experiments on the emergence of argument structure constructions as reported in van Trijp (2008). It also means that they exclusively focus on the problem of argument realization, just like (most) verbal theories only look into the characteristics of the construction under investigation and do not completely spell out how to integrate them with constructions for tense, agreement, prepositional phrases, etc. As such, the representation in this paper should be seen as a **general design solution** for how argument realization can be achieved rather than describing an actual grammar of English.

(8) Jack sweep floor. (= Jack sweeps the floor)
(9) Jack sweep dust off- floor. (= Jack sweeps the dust off the floor)

2.1 Representing and linking meanings

Before going into the technical details of the implementation, it is useful to have a quick overview of the analysis that underlies the examples and how meanings are represented and linked with each other. If we consider *sweep* to be a verb of surface contact and motion (such as e.g. *wipe*, *rub*, *scrub*; see Levin & Rappaport Hovav, 1999), then sweep can take at least three participant roles: a *sweeper*, something being *swept*, and the source

from where the motion starts. One could imagine other roles as well, such as the instrument used for sweeping (a broom or a hand) or the destination of the motion, but they are not necessary for this discussion. In a logic-based representation, the meaning of *sweep* could thus be represented as follows (logic variables are here indicated with a question mark):

(10) sweep(?event-x) ∧ sweep-1(?event-x, ?object-a) ∧ sweep-2(?event-x, ?object-b) ∧ sweep-3(?event-x, ?object-c)

Note that the meaning does not only contain a logic predicate for the event, but that the participant roles themselves are explicitly represented as well. Instead of using the labels *sweeper*, *swept* and *source*, the more neutral labels *sweep-1*, *sweep-2* and *sweep-3* are used. These are in fact arbitrary labels which can be mapped by robotic or software agents onto their sensory experiences. The meanings of *jack*, *floor*, and *dust* can be represented as:

(11) jack(?object-u)
(12) floor(?object-v)
(13) dust(?object-w)

As I will show later, these meanings are introduced by the lexical entries of the words used in examples (8) and (9). However, the lexical entries do not specify how the individual meanings of these words should be combined with each other. One of the most important functions of grammatical constructions is therefore marking explicit which of these meanings are linked to each other. Since argument structure constructions express (among other things) who's doing what in an event, they indicate to the hearer which variables are equal because they refer to the same object. This can be illustrated by taking example (8). Here, the variables ?object-u and ?object-a should be made equal because they both refer to [Jack]. The equality between ?object-v and ?object-b should also be identified because they both refer to [floor]:

(14)

Jack	*sweep*	*floor*
jack(?object-u)	sweep(?event-x)	floor(?object-v)
	sweep-1(?event-x, ?object-a)	?object-v = ?object-b
?object-u = ?object-a	sweep-2(?event-x, ?object-b)	
	sweep-3(?event-x, ?object-c)	

After making the coreferential variables equal, the hearer can parse the following meanings for examples (8) and (9):

(15) {∃ ?object-a, ?object-b, ?object-c, ?event-x: jack(?object-a) ∧ floor(?object-b), sweep(?event-x) ∧ sweep-1(?event-x, ?object-a) ∧ sweep-2(?event-x, ?object-b) ∧ sweep-3(?event-x, ?object-c}
(16) {∃ ?object-a, ?object-b, ?object-c, ?event-x: jack(?object-a) ∧ dust(?object-b) ∧ floor(?object-c) ∧ sweep(?event-x) ∧ sweep-1(?event-x, ?object-a) ∧ sweep-2(?event-x, ?object-b) ∧ sweep-3(?event-x, ?object-c}

As will be discussed later, the mapping between these meanings and their forms are organized by constructions through a layer of semantic and syntactic categories. In the mini-grammar presented here, example (8) involves a construction which maps the semantic frame 'AGENT-ACTS-ON-SURFACE' onto an SVO-pattern, and example (9) involves a construction which maps the semantic frame 'AGENT-CAUSES-PATIENT-TO-MOVE' onto the syntactic pattern SVO-Oblique.

2.2 Production and parsing in Fluid Construction Grammar

Since FCG is interested in grounded language use, a lot of attention goes to the processing of utterances both in production and parsing. During processing, a speaker or hearer gradually builds up a 'reaction network' (see Figure 1) in which each node represents a stage in the build-up of the semantic and syntactic structure of the utterance. Travelling from one node to the next can be achieved by using a construction. Since there may be several hypotheses given a certain context, each link between the nodes has a 'confidence score' to guide the search. This score is based on (a) the (linguistic) context in which constructions can be applied, and (b) how successful the applied constructions have been in previous communicative situations. The language user will in the end choose the chain with the highest estimated success.

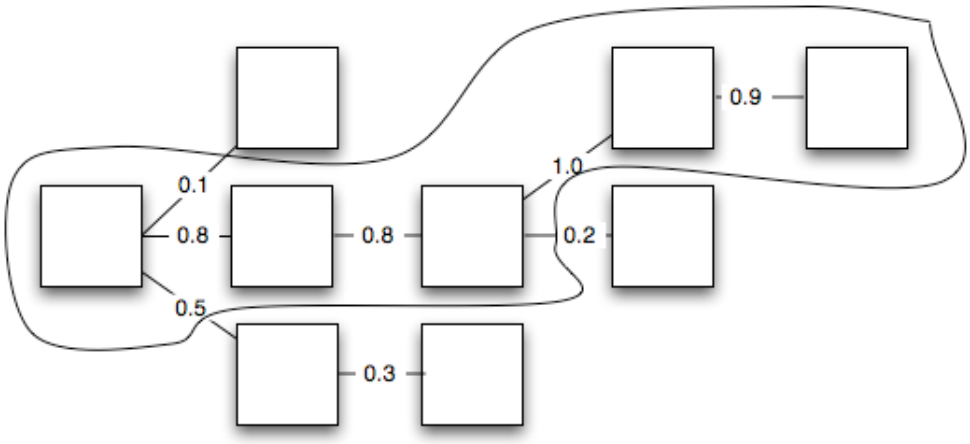


Figure 1: During processing, the language user builds up a reaction network by trying out constructions. The pathway with the highest confidence scores is selected for producing or parsing an utterance.

As is standard practice in many grammar formalisms in computational linguistics, FCG uses feature structures to represent linguistic knowledge. In fact, all linguistic knowledge (including constructions) is represented as **coupled feature structures** which couple a semantic pole to a syntactic pole. All feature structures are organized in units which are used by the basic operators of FCG for retrieving and adding new feature-value pairs. Thus, all linguistic knowledge is represented according to the following pattern:

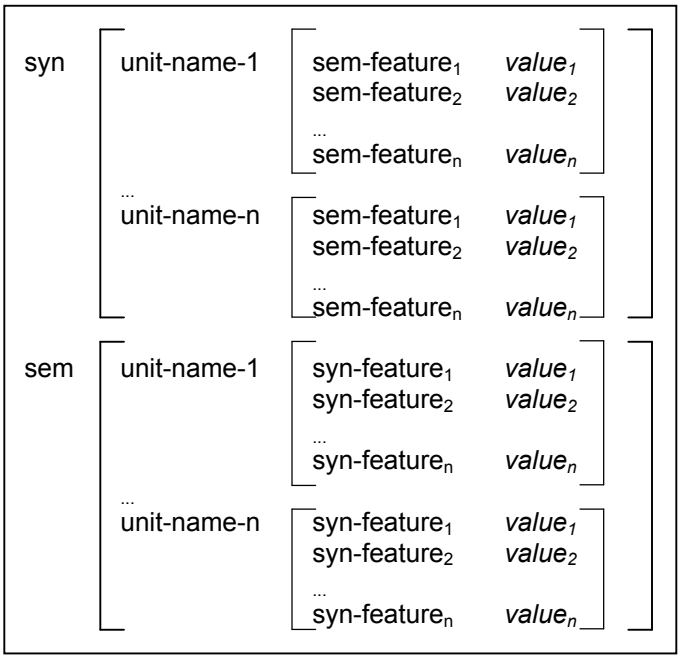
Coupled feature structure:

((**unit-name-1**
(sem-feature$_1$ *value$_1$*)
(sem-feature$_2$ *value$_2$*)
...
(sem-feature$_n$ *value$_n$*))
...
(**unit-name-n**
(sem-feature$_1$ *value$_1$*)
(sem-feature$_2$ *value$_2$*)
...
(sem-feature$_n$ *value$_n$*))) — Semantic pole

<==> — Direction marker (here: bidirectional)

((**unit-name-1**
(syn-feature$_1$ *value$_1$*)
(syn-feature$_2$ *value$_2$*)
...
(syn-feature$_n$ *value$_n$*))
...
(**unit-name-n**
(syn-feature$_1$ *value$_1$*)
(syn-feature$_2$ *value$_2$*)
...
(syn-feature$_n$ *value$_n$*))) — Syntactic pole

In the box-formalism which construction grammarians are more familiar with, this pattern roughly corresponds to (Fried & Östman, 2004, p. 29-40):



In the rest of this paper, the bracketed notation of the FCG formalism itself will be used because (a) the box notation is not very well suited for capturing the directionality of FCG constructions nor for how structure building operations should be performed (see below), and (b) the FCG notation can be directly copied and tested by people who want

to use FCG as a research tool.[1] Those who are interested in the technical details of the formalism can check De Beule & Steels (2005), Steels & De Beule (2006) and Steels, De Beule & Neubauer (2005). For others, the explanation of the lexical entries and constructions should give a clear picture of how they function in linguistic processing.

The two basic operations performed in FCG are called 'unify' and 'merge' (Steels & De Beule, 2006; not to be confused with 'merge' in Minimalism). These operators decide whether or not a construction can be used to go from one node in the reaction network to the next one. 'Unify' means that – depending on the direction of processing – the feature structure of one of the poles of a coupled feature structure acts as a set of constraints which have to be compatible with the corresponding pole of the current node in the reaction network. If all the constraints are satisfied, the other pole is 'merged' with the corresponding pole in the current node (unless merging fails because of conflicts in both poles). The combination of the unify and merge operations leads to a new coupled feature structure which is the next node in the reaction network (see Figure 2).

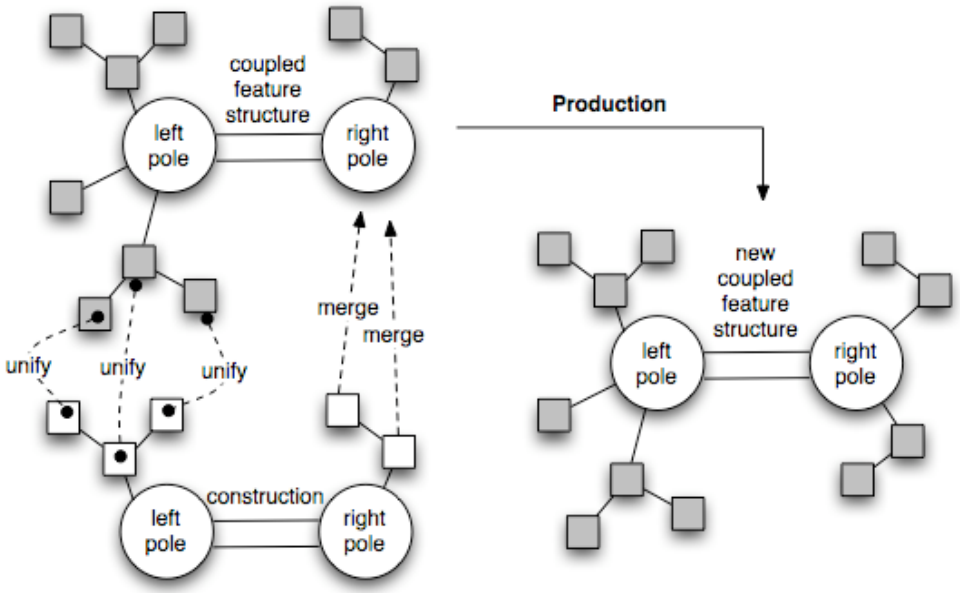


Figure 2: During production, the feature structures (squares) in the left pole of a construction (bottom left) are 'unified' with those of the current coupled feature structure (top left). If unification is successful, the right pole is merged with the coupled feature structure. This yields a new coupled feature structure which is a new node in the reaction network (right). During parsing, the same operation is performed but this time the right pole of the construction is unified and the left one is merged with the current node.

[1] There is an open-source software implementation of Fluid Construction Grammar available for free at www.emergent-languages.org.

When in production mode, the speaker unifies the left pole of a construction (typically the semantic pole) with that of the current node; and if successful he merges the right pole (typically the syntactic pole) with that of the current coupled feature structure. During parsing, exactly the same linguistic items are used, but this time in the opposite direction: here, the right pole has to unify with the current node after which the left pole is merged with it.

Finally, FCG also features structure building operators, which for example can be used to specify hierarchical relations between units. In this paper, the 'J-operator' is used in the lexical entries and constructions. Roughly speaking, all units marked with the J-operators are ignored during the unification phase, but are added or merged during the merge operation. The exact use of the J-operator is not relevant for this discussion. For a more technical specification, see De Beule & Steels (2005).

4 Parsing "jack sweep dust off-floor"

The previous section introduced a couple of examples involving a sweep-event, discussed how the meaning of these examples can be represented and linked, and gave a brief sketch of how linguistic processing is handled by Fluid Construction Grammar. This section goes into more details on how they are actually parsed. We will start with the sentence *Jack sweep dust off- floor* ('Jack sweeps the dust off the floor').

At the beginning of the parsing process, the hearer creates a first node in his reaction network, which is a coupled feature structure with an empty semantic and syntactic pole. Then he segments the sentence into strings and lumps together this information (along with the observed word order) into one unit on the syntactic pole:

```
<Node-1: coupled-feature-structure
((top-unit))
<==>
((top-unit
    (form    ((string jack-unit "jack") (string sweep-unit "sweep") (string dust-unit "dust")
              (string off-unit "off-") (string floor-unit "floor") (meets jack-unit sweep-unit)
              (meets sweep-unit dust-unit) (meets dust-unit off-unit) (meets off-unit floor-unit)))))>
```

4.1 Unifying and merging lexical entries

In the next step, the hearer will perform a lexical look-up for all the words that he has put together in the syntactic/right pole of node-1. For this, we need a lexical entry for each of the words. The lexical entry for *jack* looks as follows:

```
<Lexical entry: jack
((?top-unit                                          ⎫
      (meaning (== (jack ?object-1))))               ⎪
 ((J ?new-unit ?top)                                 ⎬  semantic pole
      (referent ?object-1)                           ⎪
      (sem-cat animate-object)))                     ⎭
<==>                                                 ⎫
((?top-unit                                          ⎪
      (form (== (string ?new-unit "floor"))))        ⎬  syntactic pole
 ((J ?new-unit ?top)                                 ⎪
      (syn-cat (== (pos noun)))))>                   ⎭
```

Since the hearer acts in parsing mode, the right pole of the entry needs to be unified with the right-pole of node-1, and the left-pole of the entry has to be merged with the left-pole

of node-1. The right pole of the entry here specifies that there must be some unit which contains in its form-feature the string "jack". This is indeed the case: the variable *?top-unit* (indicated by a question-mark) unifies with the symbol 'top-unit' in node-1 and there is a corresponding string. Since the unification is successful, the meaning for the word *jack* can be added to the semantic (left) pole.

Note that there is also a J-operator on both sides. Without going into technical details, the J-operator here specifies that there needs to be a new unit which contains all the information for the word *jack*. It also specifies that this new unit is a sub-unit of the original top-unit. Finally, the J-operator also adds a syntactic and semantic categorization to the unit: *jack* is specified as a noun in the right (syntactic) pole, and as an animate-object in the left (semantic) pole. One could devise many other categorizations and features for *jack*, but they are not necessary for understanding the example here so they are left out for convenience's sake.

Since both unification and merge were successful, a new node is created in the reaction network. Here we see that the other words are still in the top-unit, but that there is a new unit for *jack* both in the semantic and in the syntactic pole:

```
<Node-2: coupled-feature-structure
((top-unit
    (sem-subunits (jack-unit)))
 (jack-unit
    (meaning ((jack?object-1)))
    (sem-cat animate-object)
    (referent ?object-1)))
<==>
((top-unit
    (form    ((string floor-unit "floor") (string sweep-unit "sweep") (string dust-unit "dust")
              (string off-unit "off-") (meets jack-unit sweep-unit) (meets sweep-unit dust-unit)
             (meets dust-unit off-unit) (meets off-unit floor-unit)))
    (syn-subunits (jack-unit)))
 (jack-unit
    (form    ((string jack-unit "jack")))
    (syn-cat((pos noun)))))>
```

The lexical entries for *floor* and *dust* look almost exactly the same, apart from their semantic categorization and meaning:

```
<Lexical entry: floor
      (?top-unit
          (meaning (== (floor ?object-3))))
       ((J ?new-unit ?top)
        (referent ?object-3)
        (sem-cat surface-object)))
      <==>
      ((?top-unit
          (form (== (string ?new-unit "floor"))))
       ((J ?new-unit ?top)
        (syn-cat (== (pos noun)))))>
```

```
<Lexical entry: dust
      ((?top-unit
          (meaning (== (dust ?object-2))))
       ((J ?new-unit ?top)
        (referent ?object-2)
        (sem-cat moveable-object)))
      <==>
      ((?top-unit
          (form (== (string ?new-unit "dust"))))
       ((J ?new-unit ?top)
        (syn-cat (== (pos noun)))))>
```

The lexical entry for *sweep*, however, is a bit more complicated. Its main function is the same as for the other lexical entries: given the string "sweep", it will merge the meaning of this word with the semantic pole of the current node in the reaction network. The main difference with the other words lies in the semantic and syntactic categorization of *sweep*.

Take a look at the features ‘syn-frame’ in the syntactic pole and ‘sem-frame’ in the semantic pole:

```
<Lexical entry: sweep
      ((?top-unit
            (meaning (==    (sweep ?event-x)
                            (sweep-1 ?event-x ?obj-x)
                            (sweep-2 ?event-x ?obj-y)
                            (sweep-3 ?event-x ?obj-z))))
       ((J ?new-unit ?top)
            (sem-frame (== (sem-role-agent ?unit-a ?obj-x)
                            (sem-role-surface ?unit-b ?obj-y)
                            (sem-role-moveable ?unit-c ?obj-y)
                            (sem-role-source ?unit-d ?obj-z)))))
      <==>
      ((?top-unit
            (form (== (string ?new-unit “sweep”))))
       ((J ?new-unit ?top)
            (syn-frame (==  (syn-role-subject ?unit-1)
                            (syn-role-object ?unit-2)
                            (syn-role-oblique ?unit-3)))))>
```

At first glance, it seems that the lexical entry contains a list of semantic and syntactic roles similar to lexicalist approaches. The big difference is however that these frames do not list the argument structure of the event but its **potential categorizations or valences**. In the syntactic pole, the syn-frame merely states that the verb *sweep* can (but does not have to) occur in a syntactic structure together with some unit playing the role of subject, some unit playing the role of object, some unit playing the role of oblique, or some combination of the aforementioned syntactic roles. Similarly, the semantic frame states that the arguments of *sweep* can play the roles of agent, surface, moveable patient or source, but it does not specify which combinations of argument realization are possible or even obligatory. As will be discussed later, it is the construction which selects the correct combination and determines how the semantic and syntactic roles are related to each other (also see Figure 3).

Note that the lexical entry of the verb is also agnostic as to which unit will potentially play e.g. the subject-role and to which semantic role this syntactic role should be mapped. This is captured through the variable names: the variable ‘?unit-1’ will later be bound to some unit by the construction. The variable name is also different from for example the variable name ‘?unit-a’ in the semantic pole, which later may be bound to the unit that plays the semantic role of agent. The fact that ‘?unit-1’ and ‘?unit-a’ are different variable names makes it possible that they receive a different binding. This is necessary when the speakers for example want to distinguish between an active (mapping between agent and subject) and a passive construction (mapping between patient and subject).

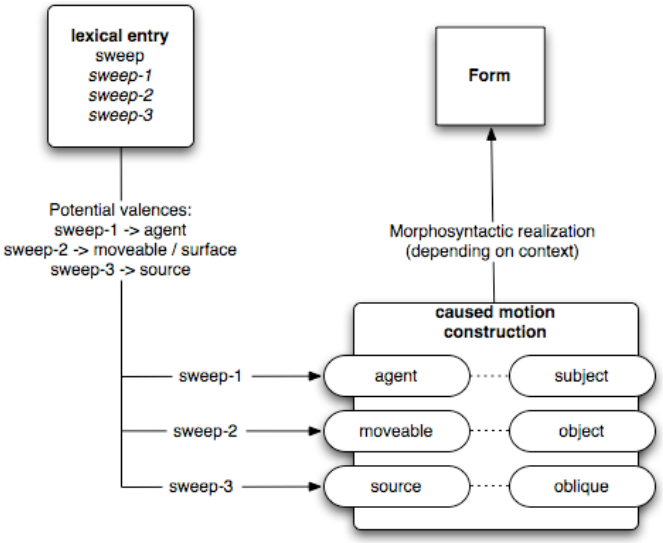


Figure 3: The relation between the meaning of an event and its morphosyntactic realization is indirect and multilayered. This figure shows how a lexical entry introduces its 'potential valences' from which the construction selects the right combination. The construction then maps this meaning onto a syntactic pattern which in itself is realized in a certain form (depending on linguistic and extra-linguistic context).

Also note that in the semantic pole, the semantic roles are linked to the event-specific participants *sweep-1*, *sweep-2* and *sweep-3* through their variable names. For example, the variable '?obj-x' is shared by *sweep-1* and the semantic role of agent. The semantic frame of *sweep* here states that the role *sweep-2* can *potentially* be categorized as a surface role or as a moveable patient. Again, which categorization actually applies depends on the construction that is used. The semantic and syntactic frames are gradually acquired through language use and they should not be seen as a rigid set of possibilities. Instead, the potential uses of a verb can be extended if needed for communicative purposes and each possibility can become conventionalized or become obsolete in the speech community (see van Trijp, 2008, for more details).

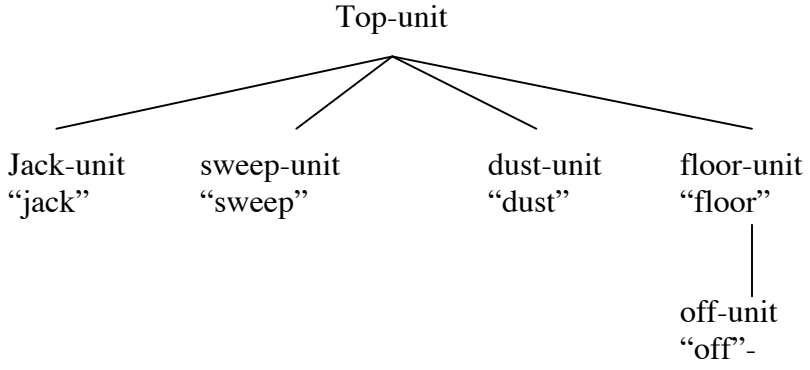
Unifying and merging the lexical entries for *sweep*, *floor* and *dust* (the order does not matter) will lead the hearer to a fifth node in the reaction network:

```
<Node-5: coupled-feature-structure
((top-unit
     (sem-subunits (jack-unit sweep-unit dust-unit floor-unit)))
 (jack-unit
     (meaning ((jack ?object-1)))
     (sem-cat animate-object)
     (referent ?object-1))
 (sweep-unit
     (meaning ((sweep ?event-x) (sweep-1 ?event-x ?obj-x)
                  (sweep-2 ?event-x ?obj-y) (sweep-3 ?event-x ?obj-z)))
     (sem-frame ((sem-role-agent ?unit-a ?obj-x)
                    (sem-role-surface ?unit-b ?obj-y)
                    (sem-role-moveable ?unit-c ?obj-y)
                    (sem-role-source ?unit-d ?obj-z))))
```

```
 (dust-unit
     (meaning ((dust ?object-2)))
     (sem-cat moveable-object)
     (referent ?object-2))
 (floor-unit
     (meaning ((floor ?object-3)))
     (sem-cat surface-object)
     (referent ?object-3)))
<==>
((top-unit
     (form     ((string off-unit "off-") (meets jack-unit sweep-unit) (meets sweep-unit dust-unit)
                (meets dust-unit off-unit) (meets off-unit floor-unit)))
     (syn-subunits (jack-unit sweep-unit dust-unit floor-unit)))
 (jack-unit
     (form ((string jack-unit "jack")))
     (syn-cat ((pos noun))))
 (sweep-unit
     (form ((string sweep-unit "sweep")))
     (syn-frame   ((syn-role-subject ?unit-x)
                   (syn-role-object ?unit-y)
                   (syn-role-oblique ?unit-z))))
 (dust-unit
     (form ((string dust-unit "dust")))
     (syn-cat ((pos noun))))
 (floor-unit
     (form     ((string floor-unit "floor")))
     (syn-cat((pos noun)))))>
```

The meaning in the above coupled feature structure is still unlinked, as was explained in section 2.1. In other words, the hearer knows the meaning of the individual words, but does not know yet who is doing what in the sweep-event. The next step is therefore to unify and merge the correct construction that can solve this problem.

However, in this example there is still the string "off-" left in the top-unit which first needs to be parsed. In this simplified example, "off-" acts as some kind of case marker which assigns the oblique role to the argument that immediately follows it. This is implemented in a morphological rule (in which both poles are syntactic) which uses the structure building operator J for first introducing a new unit for the marker itself and then specifying that it should become a sub-unit of the unit which is marked by it. Without repeating the entire coupled feature structure, the syntactic structure now looks as follows:



The morphological rule itself looks as follows:

```
<Morphological rule: off-
((?top-unit
        (syn-subunits (== ?some-unit)))
 (?some-unit
        (syn-role syn-role-oblique)))
<==>
((?top-unit
        (form (== (string ?marker-unit "off-")
                  (meets ?marker-unit ?some-unit))))
 ((J ?marker-unit ?top))
 ((J ?some-unit ?top (?marker-unit)))>
```

4.2 The caused motion construction

The utterance *Jack sweeps the dust off the floor* is a typical example of the caused motion construction (Goldberg, 1995, chapter 7), which here carries the meaning of 'X causes Y to move Z by sweeping'. In the semantic pole, the construction selects from the verb the semantic roles of agent, moveable patient and (in this simplified grammar) source. On the syntactic pole it assigns the syntactic roles of subject, object and oblique to the arguments. The construction looks as follows:

```
<Construction: caused-motion
((?top-unit
        (sem-subunits (== ?unit-a ?unit-b ?unit-c ?unit-d)))
 (?unit-a
        (sem-frame (==    (sem-role-agent ?unit-b ?obj-x)
                          (sem-role-moveable ?unit-c ?obj-y)
                          (sem-role-source ?unit-d ?obj-z))))
 (?unit-b
        (referent ?obj-x)
        (sem-cat animate-object))
 (?unit-c
        (referent ?obj-y)
        (sem-cat moveable-object))
 (?unit-c
        (referent ?obj-z)
        (sem-cat ?sem-cat))
 ((J ?unit-b NIL) (sem-role sem-role-agent))
 ((J ?unit-c NIL) (sem-role sem-role-moveable))
 ((J ?unit-d NIL) (sem-role sem-role-source)))
<==>
((?top-unit
        (form (== (meets ?unit-b ?unit-a) (meets ?unit-a ?unit-c) (meets ?unit-c ?unit-d)))
        (syn-subunits (== ?unit-a ?unit-b ?unit-c ?unit-d)))
 (?unit-a
        (syn-frame (==  (syn-role-subject ?unit-b)
                        (syn-role-object ?unit-c)
                        (syn-role-oblique ?unit-d))))
 (?unit-d
        (syn-role syn-role-oblique))
 ((J ?unit-b NIL) (syn-role syn-role-subject))
```

```
((J ?unit-c NIL) (syn-role syn-role-object)))>
```

Since the hearer is parsing, the right pole needs to be unified with the current node in the reaction network. The right pole here demands there to be some unit with at least four sub-units and with a certain word order among them (i.e. the 'meets' constraints). The last argument also has to have the syntactic role of oblique. The right pole of the current node satisfies all the constraints: the unit for *jack* receives subject status (which is here taken care of by the J-operator) and *dust* becomes the object. *Floor* was already marked as oblique by the morphological rule that was shown in the previous section.

The construction binds the variables '?unit-b' to 'jack-unit', '?unit-c' to 'dust-unit' and '?unit-d' to 'floor-unit'. The construction repeats the same variables in the semantic pole so that the role of agent can be assigned to *jack*, the role of patient to *dust*, and the role of source to *floor*. This means that the construction can now make all the variables equal that refer to the same object: '?obj-x' (of which the lexical entry stated that it was shared with the filler of *sweep-1*), '?obj-y' (with the filler of *sweep-2*) and '?obj-z' (with the filler of *sweep-3*). This leads to the following node in the reaction network:

```
<Node-7: coupled-feature-structure
((top-unit
    (sem-subunits (jack-unit sweep-unit dust-unit floor-unit)))
 (jack-unit
    (meaning ((jack ?object-1)))
    (sem-role sem-role-agent)
    (sem-cat animate-object)
    (referent ?object-1))
 (sweep-unit
    (meaning ((sweep ?event-x) (sweep-1 ?event-x ?object-1)
              (sweep-2 ?event-x ?object-2) (sweep-3 ?event-x ?object-3)))
    (sem-frame ((sem-role-agent jack-unit ?object-1)
                (sem-role-surface dust-unit ?object-2)
                (sem-role-moveable dust-unit ?object-2)
                (sem-role-source floor-unit ?object-3))))
 (dust-unit
    (meaning ((dust ?object-2)))
    (sem-role sem-role-moveable)
    (sem-cat moveable-object)
    (referent ?object-2))
 (floor-unit
    (meaning ((floor ?object-3)))
    (sem-role sem-role-source)
    (sem-cat surface-object)
    (referent ?object-3)))
<==>
((top-unit
    (form    ((meets jack-unit sweep-unit) (meets sweep-unit dust-unit)
              (meets dust-unit off-unit) (meets off-unit floor-unit)))
    (syn-subunits (jack-unit sweep-unit dust-unit floor-unit)))
 (jack-unit
    (form ((string jack-unit "jack")))
    (syn-role syn-role-subject)
```

```
    (syn-cat ((pos noun))))
(sweep-unit
    (form ((string sweep-unit "sweep")))
    (syn-frame  ((syn-role-subject jack-unit)
                 (syn-role-object dust-unit)
                 (syn-role-oblique floor-unit))))
 (dust-unit
    (form ((string dust-unit "dust")))
    (syn-role syn-role-object)
    (syn-cat ((pos noun))))
 (floor-unit
    (form    ((string floor-unit "floor") (meets off-unit floor-unit)))
    (syn-subunits (off-unit))
    (syn-role syn-role-oblique)
    (syn-cat((pos noun)))
 (off-unit))>
```

The hearer has now used all the matching linguistic items that he knows, so now he can extract the meanings from the semantic pole of the coupled feature structure. This yields the following meaning:

(17) {∃ ?object-1, ?object-2, ?object-3, ?event-x: jack(?object-1) ∧ dust(?object-2) ∧ floor(?object-3) ∧ sweep(?event-x) ∧ sweep-1(?event-x, ?object-1) ∧ sweep-2(?event-x, ?object-2) ∧ sweep-3(?event-x, ?object-3}

As can be seen in the meaning, all variables that have the same referent have been made equal by the construction. The hearer thus knows that *jack* was the sweeper, that *dust* was the patient of the event, and that the *floor* was the source of the motion. In the aforementioned experiments on the emergence of argument structure, the robotic or software agents then check whether this parsed meaning fits with their world model.

5 Producing "jack sweep floor"

This section gives an overview of how an utterance such as *jack sweep floor* can be produced in Fluid Construction Grammar. In this case, the caused motion construction is not used, but a construction which maps the semantic frame 'X acts on surface Y' onto the syntactic frame 'Subject-Verb-Object'. Suppose that the speaker starts with the following meaning:

(18) ((jack object-1) (floor object-2) (sweep event-1) (sweep-1 event-1 object-1) (sweep-2 event-1 object-2) (sweep-3 event-1 NIL))

NIL stands for 'empty', which means that the speaker did not conceptualize any source of motion, or that he wishes to only express the arguments of *sweep-1* and *sweep-2*. In order to verbalize this meaning, the speaker constructs a reaction network. The first node in the reaction network is just a coupled feature structure in which the above meaning is placed in one unit in the semantic pole, and an empty syntactic pole:

```
<Node-1: coupled-feature-structure
((top-unit
     (meaning    ((jack object-1) (floor object-2)
                  (sweep event-1) (sweep-1 event-1 object-1)
                  (sweep-2 event-1 object-2) (sweep-3 event-1 NIL)))))
<==>
((top-unit))>
```

Next, the speaker builds new nodes in the network by unifying and merging the lexical entries that cover these meanings. Suppose that the speaker has the same lexical entries as given in section 4 and that all three of them (for *jack*, *sweep* and *floor*) are a successful match, then the coupled feature structure looks as follows:

```
<Node-4: coupled-feature-structure
((top-unit
     (sem-subunits (jack-unit sweep-unit floor-unit)))
 (jack-unit
     (meaning ((jack object-1)))
     (sem-cat animate-object)
     (referent object-1))
 (sweep-unit
     (meaning ((sweep event-1) (sweep-1 event-1 object-1)
               (sweep-2 event-1 object-2) (sweep-3 event-1 object-3)))
     (sem-frame ((sem-role-agent ?unit-a object-1)
                 (sem-role-surface ?unit-b object-2)
                 (sem-role-moveable ?unit-c object-2)
                 (sem-role-source ?unit-d NIL))))
(floor-unit
     (meaning ((floor object-2)))
     (sem-cat surface-object)
     (referent object-2)))
<==>
((top-unit
     (syn-subunits (jack-unit sweep-unit floor-unit)))
 (jack-unit
     (form ((string jack-unit "jack")))
     (syn-cat ((pos noun))))
 (sweep-unit
     (form ((string sweep-unit "sweep")))
     (syn-frame  ((syn-role-subject ?unit-x)
                  (syn-role-object ?unit-y)
                  (syn-role-oblique ?unit-z))))
 (floor-unit
     (form ((string floor-unit "floor")))
     (syn-cat ((pos noun)))))>
```

Since the speaker knows which meaning he wants to express, there are no variable equalities left in the semantic pole. Note that the lexical entry for *sweep* has added its *potential* semantic roles to the current structure: *jack* can map onto agent, and *floor* can either map onto surface or moveable object. There is however no specification yet about which semantic roles will be selected. On the syntactic pole, it is also still undecided which argument will become the subject, object or oblique of the sentence. As with the

semantic frame, the syntactic frame just lists all possible syntactic roles that may be selected by the construction.

Trying to unify the caused motion construction leads to a failure: first of all, *floor* is categorized as a static surface-object and thus violates the construction's constraint that the patient has to be moveable, and second, the source argument is missing. So a different construction needs to be unified and merged to move to the next node in the reaction network. In this example, the following 'agent-acts-on-surface' construction would do the trick:

```
<Construction: agent-acts-on-surface
((?top-unit
        (sem-subunits (== ?unit-a ?unit-b ?unit-c)))
 (?unit-a
        (sem-frame (==    (sem-role-agent ?unit-b ?obj-x)
                          (sem-role-surface ?unit-c ?obj-y))))
 (?unit-b
        (referent ?obj-x)
        (sem-cat animate-object))
 (?unit-c
        (referent ?obj-y)
        (sem-cat surface-object))
 ((J ?unit-b NIL) (sem-role sem-role-agent))
 ((J ?unit-c NIL) (sem-role sem-role-surface)))
<==>
((?top-unit
          (form (== (meets ?unit-b ?unit-a) (meets ?unit-a ?unit-c)))
          (syn-subunits (== ?unit-a ?unit-b ?unit-c)))
 (?unit-a
        (syn-frame (==    (syn-role-subject?unit-b)
                          (syn-role-object ?unit-c))))
 ((J ?unit-b NIL) (syn-role syn-role-subject))
 ((J ?unit-c NIL) (syn-role syn-role-object)))>
```

First the speaker tries to unify the semantic pole with that of the current coupled feature structure in the reaction network. This leads to a success because all the constraints are satisfied: the construction expects some unit with three sub-units, one of which is an animate-object and one of which is a surface-object. The construction also selects the semantic roles of 'agent' and 'surface' from the verb's potential valences and assigns them to the correct arguments. Next, the right pole of the construction is merged with the right pole of the current node. Since it is specified that the agent maps onto subject and the surface maps onto object in this construction, the right syntactic roles are assigned to the arguments, including their word order. The new node looks as follows:

```
<Node-5: coupled-feature-structure
((top-unit
    (sem-subunits (jack-unit sweep-unit floor-unit)))
 (jack-unit
    (meaning ((jack object-1)))
    (sem-role sem-role-agent)
    (sem-cat animate-object)
    (referent object-1))
 (sweep-unit
```

```
     (meaning ((sweep event-1) (sweep-1 event-1 object-1)
                    (sweep-2 event-1 object-2) (sweep-3 event-1 object-3)))
     (sem-frame ((sem-role-agent jack-unit object-1)
                      (sem-role-surface floor-unit object-2)
                      (sem-role-moveable floor-unit ?object-2)
                      (sem-role-source ?unit-d NIL))))
(floor-unit
     (meaning ((floor object-2)))
     (sem-role sem-role-surface)
     (sem-cat surface-object)
     (referent object-2)))
<==>
((top-unit
     (form     ((meets jack-unit sweep-unit) (meets sweep-unit floor-unit)))
     (syn-subunits (jack-unit sweep-unit floor-unit)))
 (jack-unit
     (form ((string jack-unit "jack")))
     (syn-role syn-role-subject)
     (syn-cat ((pos noun))))
(sweep-unit
     (form ((string sweep-unit "sweep")))
     (syn-frame   ((syn-role-subject jack-unit)
                      (syn-role-object floor-unit)
                      (syn-role-oblique ?unit-z))))
 (floor-unit
     (form ((string floor-unit "floor")))
     (syn-role syn-role-object)
     (syn-cat((pos noun))))>
```

The speaker has no other constructions or linguistic items that could unify and merge, so he is almost done processing. In the final phase, he takes all the formal constraints specified in the syntactic pole of the last node and renders them into the following utterance:

(19) "jack sweep floor"

6 Conclusion

This paper presented a fully working implementation in Fluid Construction Grammar of how a verb's semantics can interact with argument structure constructions in order to yield different patterns of argument realization. After reviewing some problems with the definition of this 'fusion process' offered by Goldberg (1995), the paper proposes that the lexical entry of a verb introduces its potential semantic and syntactic roles to the structure which is being produced or parsed by a language user. This 'potential' – which is gradually acquired and possibly even extended through language use – does not provide a priori combinations, nor does it specify how the meanings of an utterance should be combined with each other. In the next step it is the argument structure construction that selects how this potential should be used for mapping meaning to form and vice versa.

This proposal was illustrated with the production and parsing of two example utterances: *jack sweep dust off-floor* and *jack sweep floor*. Both examples are simplified representations of English-like sentences and are directly related to experiments on the

emergence of argument structure constructions in a population of embodied artificial agents. As such, they can be regarded as 'design solutions' for the problem of argument realization which have to be complemented in linguistic theories with constructions for tense, aspect, etc. The traces of the examples show how the licensing of a construct depends on a combination of all constructions and lexical items involved rather than specifying all constraints in the lexical entry of the verb (as is common practice in lexical approaches).

Acknowledgements

This paper grew out of a workshop of the Scientific Network on Construction Grammar on 'Construction Grammar Architectures', held at 1-3 February 2008 in Bremen, Germany. The author wishes to thank the organisers Kerstin Fischer and Anatol Stefanowitsch, as well as all the participants, for their insightful comments and feedback. He is also greatly indebted to Luc Steels and his colleagues at the Vrije Universiteit Brussel and the SONY Computer Science Laboratory in Paris.

References

Bresnan, J. (1982), *The Mental Representation of Grammatical Relations*, Cambridge MA: MIT Press.

De Beule, J. / L. Steels (2005), "Hierarchy in Fluid Construction Grammar", in: Furbach, U. (Ed.), *Proceedings of the 28th Annual German Conference on AI (KI 2005)*, Berlin-Heidelberg: Springer, 1-15.

Dik, S. (1997), *The Theory of Functional Grammar. Part 1: The Structure of the Clause*, Berlin: Mouton De Gruyter.

Fillmore, C. (1968), "The case for case", in: Bach, E. / R. Harms (Eds.), *Universals in Linguistic Theory*. New York: Holt, Reinhart & Winston, 1-88.

Fried, M. / J.-O. Östman (2004), "Construction Grammar: a thumbnail sketch", in: Fried, M. / J.-O. Östman (Eds.), *Construction Grammar in a Cross-Language Perspective*, Amsterdam: John Benjamins, 11-86.

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

Levin, B. / Rappaport Hovav, M. (1999), "Two structures for compositionally derived events", in: *Proceedings of SALT 9*, Ithaca NY: Cornell Linguistics Circle Publications, 127-144.

Michaelis, L. (2003), "Headless constructions and coercion by construction", in:Francis, E.J. / L. Michaelis (Eds.), *Mismatch: Form-Function Incongruity and the Architecture of Grammar*, Stanford: CSLI Publications, 259-310.

Pinker, S. (1989), *Learnability and Cognition: The Acquisition of Argument Structure*. Cambridge MA: MIT Press.

Steels, L. (2004). "Constructivist development of grounded construction grammars", in: Scott, D. / W. Daelemans / M. Walker (Eds.), *Proceedings of the Annual Meeting of Association for Computational Linguistics (ACL)*, Barcelona: ACL, 9-16.

Steels, L. / J.-C. Baillie (2003), "Shared grounding of event descriptions by autonomous robots", *Robotics and Autonomous Systems* 43(2-3), 163-173.

Steels, L. / J. De Beule (2006), "Unify and merge in Fluid Construction Grammar", in: Vogt, P. / Y. Sugita / E. Tuci / C. Nehaniv (Eds.), *Symbol Grounding and Beyond*, Berlin: Springer, 197-223.

Steels, L. / J. De Beule / N. Neubauer (2005), "Linking in Fluid Construction Grammar", in: *Proceedings of BNAIC 2005*, Brussels: Belgian Royal Society of Arts and Sciences, 11-18.

van Trijp, R. (2008), "Semantic roles in Fluid Construction Grammar", in: Smith, A.D.M. / K. Smith / R. Ferrer i Cancho (Eds.), *The Evolution of Language (EVOLANG 7)*, London: World Scientific Publishing.