Notice

This paper is the author's draft and has now been published officially as:

van Trijp Remi (2011). A Design Pattern for Argument Structure Constructions. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*, 115–145. Amsterdam: John Benjamins.

BibTeX:

```
@incollection{vantrijp2011design,
   Author = {{van Trijp}, Remi},
   Title = {A Design Pattern for Argument Structure Constructions},
   Editor = {Steels, Luc},
   Pages = {115--145},
   Booktitle = {Design Patterns in {Fluid Construction Grammar}},
   Publisher = {John Benjamins},
   Address = {Amsterdam},
   Year = {2011}}
```

A Design Pattern for Argument Structure Constructions

Remi van Trijp

Abstract

This paper presents a design pattern for handling argument structure and offers a concrete operationalization of this pattern in Fluid Construction Grammar. Argument structure concerns the mapping between 'participant structure' (who did what to whom) and instances of 'argument realization' (the linguistic expression of participant structures). This mapping is multilayered and indirect, which poses great challenges for grammar design. In the proposed design pattern, lexico-phrasal constructions introduce their semantic and syntactic potential of linkage. Argument structure constructions, then, select from this potential the values that they require and implement the actual linking.

1. Introduction

This paper proposes a design pattern for tackling the challenges of argument structure and provides a computational operationalization of this pattern in Fluid Construction Grammar. Argument structure concerns the mapping between 'participant structure' and 'argument realization'. Participant structure covers the semantic relations between events and the participants that play a role in those events. For example, a kick-event may involve a kicker and something that is being kicked. Argument realization, then, covers the morphosyntactic means that languages employ to express participant structure into a surface form (Levin & Rappaport Hovav, 2005). For instance, English speakers can express the same kick-event as *She kicked the ball* and *The ball was kicked (by her)*, depending on how they wish to profile the event. Almost every language in the world has developed some strategy to handle argument structure, ranging from word order and case to verbal marking and agreement (Palmer, 1994).

The proposed design pattern targets the main difficulty of argument structure, which is the fact that the mapping between meaning and form is multilayered and

indirect. The solution comprises an interaction between lexico-phrasal constructions and argument structure constructions, whereby the first group of constructions introduce their semantic and syntactic combinatorial potential, and in which the latter realize an actual combination by selecting and linking actual values.

This paper is structured as follows. The next section illustrates the challenges of argument structure and introduces the terminology used in this paper. Section 3 then explains the design pattern proposed in this paper and shows how the design pattern can be captured through templates in FCG. Next, more computational details are shown on linguistic processing. Section 7 finishes with a first assessment and outlook of the current proposal. The reader is expected to be familiar with the basics of FCG in order to fully grasp the technical details (Steels, 2011a,b,c).

2. Grammar Square for Argument Structure

Figure 1 offers a schematic representation of the indirect mapping from participant structure to surface form and vice versa. As can be seen, grammar mediates between meaning and form through a layer of semantic and syntactic categories.

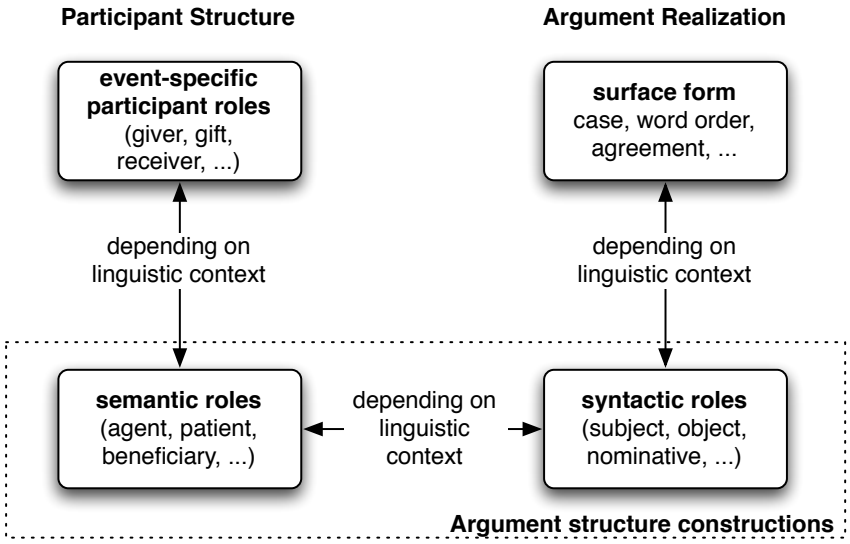

Figure 1. *The grammar square: grammar forms an abstract intermediary layer between the semantic domain of 'participant structure' and morphosyntactic 'argument realization'.*

This 'grammar square' (also see Steels, 2011a) provides us with a clearer picture of the kinds of relations that need to be captured by the grammar.

In the remainder of this paper, I will use the following terminology for highlighting various aspects of the grammar square:

- *Participant roles* are event-specific roles such as the 'kicker' and 'kicked' of a kick-event. They are part of a verb's lexical meaning.
- *Semantic roles* are more abstract event-roles such as 'Agent', 'Patient' or 'Beneficiary'. Semantic roles can be thought of as generalizations over participant roles.
- *Syntactic roles* are syntactic functions such as 'subject' and 'object', or 'nominative' and 'accusative'. Syntactic roles are not as semantically coherent as semantic roles, but they may serve a wider range of functional purposes.
- *Surface form* involves the morphosyntactic means employed by a language for indicating argument structure, such as case markers and word order.
- *Argument structure constructions* (Goldberg, 1995) are grammatical constructions that organize the mapping between semantic and syntactic roles.

The following subsections provide more detailed linguistic examples that illustrate and justify the roles played by each corner of the grammar square and the relations between them.

2.1. Mapping between Participant Roles and Semantic Roles

First of all, grammar categorizes event-specific participant roles into more abstract semantic roles such as 'Agent', 'Patient' and 'Beneficiary'. Semantic roles have claimed a central position in every major theory of grammar ever since the seminal work of Fillmore (1968). Most accounts in generative grammar assume that there is a small, finite list of universal semantic roles that are semantically unanalyzable (see Croft, 1991; Levin & Rappaport Hovav, 2005, for a discussion). However, virtually all other theories, such as lexicalist accounts (Levin & Rappaport Hovav, 2005), event decomposition approaches (Dowty, 1991; Van Valin, 2004) and cognitive-functional linguistics (Croft, 1991; Goldberg, 1995) assume a more fine-grained representation of event structure. This paper considers the list of semantic roles to be open-ended, language-specific and developed through language usage. (Also see Croft (1991); Evans & Levinson (2009) and Haspelmath (2007).)

Example 1 illustrates the widely accepted view in linguistics that the same semantic role can map onto different verb-specific participant roles. For example, the Agent in the following sentences maps onto a giver (*she*) and a seller (*he*), whereas the Patient maps onto the objects that were given (*flowers*) or sold (*his car*):

(1) a. *[She $_{SUBJ}$] gave [him $_{IND-OBJ}$] [flowers $_{OBJ}$].*
(giver-Agent) event (givee-Recipient) (given-Patient)

b. *[He $_{SUBJ}$] sold [his car $_{OBJ}$].*
(seller-Agent) event (sold-Patient)

More controversial is the hypothesis that there is also a many-to-many mapping in the other direction. Example 2 contrasts two different descriptions of the same event. In the first sentence, the floor is conceptualized as the undergoer of the sweep-action, whereas in the second sentence the floor is expressed as the location from which dust is moved away.

(2) a. *[He $_{SUBJ}$] swept [the floor $_{OBJ}$].*
(sweeper-Agent) event (swept-Patient)

b. *[He $_{SUBJ}$] swept [the dust $_{OBJ}$] [off the floor $_{OBL}$].*
(sweeper-Agent) event (swept_away-Moved) (swept-Source)

2.2. Mapping between Semantic Roles and Syntactic Roles

All linguistic theories agree that there is a difference between semantic roles (such as Agent or Patient) and syntactic roles (such as subject and object). Most textbooks take the passive construction to illustrate that the Agent of an event is not always realized as the subject of a sentence:

(3) *[The car $_{SUBJ}$] was sold.*
(sold-Patient) event

However, it suffices to look more closely at the behavior of individual verbs to see that the mapping between semantic and syntactic roles is many-to-many in active constructions as well. Example 4 shows that the verb *to receive* takes the recipient as its subject and treats the giver as an optional argument. Another example is the verb *to please* (5), which reverses the 'default' mapping whereby the most agentive-like role is expressed as the subject of active sentences (for example *I like ice cream.*). The middle construction (6) does not reverse roles but simply cuts the Agent in active sentences:

(4) *[He $_{SUBJ}$]* *received [a gift $_{OBJ}$]* *[from Jill $_{OBL}$].*
(receiver-Recipient) event (gift-Patient) (giver-Agent/Source)

(5) *[Ice cream $_{SUBJ}$]* *pleases [me $_{IND-OBJ}$].*
(liked-Experienced) event (liker-Experiencer)

(6) *[The book $_{SUBJ}$] reads [well $_{ADV}$].*
(read-Patient) event manner

2.3. Mapping between Syntactic Roles and Surface Form

Finally, the mapping between syntactic roles and their surface form is many-to-many as well. English is more sparse than heavily inflected languages in doing so, yet numerous examples can be found. Examples 7 and 8 show that the same syntactic role may appear in a different surface form depending on the linguistic context. In (7), the third person masculine pronoun is expressed as *he* if it is the subject of the main clause, but as *him* if it is the subject of the subclause. Example 8 shows how English speakers can shift word order around in order to emphasize certain parts of the utterance.

(7) *[He* **SUBJ***]* *saw* *[[him $_{SUBJ}$]* *crossing*
(seer-Experiencer) event ((crosser-Agent) event
[the street $_{OBJ}$] **OBJ***] .*
(crossed-Patient) seen-Experienced)

(8) *[A dozen roses $_{OBJ}$] [Nina $_{SUBJ}$] sent* *[her mother $_{IND-OBJ}$]!*
(sent-Patient) event (sender-Agent) (sendee-Recipient)

(Example from Goldberg, 2006, p. 21)

In the other direction, the same form can be mapped onto several functions. The following examples show how the third person pronoun *it* can play both the subject and object role:

(9) a. *[John $_{SUBJ}$]* *kicked [it $_{OBJ}$].*
(kicker-Agent) event (kicked-Patient)

b. *[It $_{SUBJ}$]* *was sent [yesterday $_{ADV}$].*
(sent-Patient) event temporal

2.4. A Constructional Approach?

Most linguists working on argument realization accept the complex mappings discussed in the previous subections. Unfortunately, they strongly disagree on how these mappings should be implemented. The most widespread approach, made popular by Pinker (1989) and adopted by theories such as LFG (Bresnan, 1982) and HPSG (Ginzburg & Sag, 2000), is the 'lexicalist account', which assumes that a verb's morphosyntactic behavior can be entirely predicted by the verb's semantics. For each different argument realization pattern, the lexicalist account needs a separate lexical item, either through homonymy or through lexical rules that derive novel lexical items from a basic lexical entry.

A particular branch of construction grammar – most outspokenly voiced by Goldberg (1995) – has challenged the traditional lexical account. The constructional analysis assumes that argument structure constructions are grammatical items that carry meaning themselves and that are even capable of imposing their semantic and syntactic properties onto verbs and their arguments. For example, in the utterance *she baked him a cake*, the ditransitive construction imposes the meaning 'X INTENDS TO CAUSE Y TO RECEIVE Z' on a verb of creation (*bake*), which does not have an inherent receiver in its meaning. The constructional account allows for a wider range of analytical possibilities than the lexicalist approach (Croft, 2003), such as coercion by construction, but also brings into question how lexical items may interact with argument structure constructions. This question is currently the topic of heavy debate, the details of which fall beyond the scope of this paper. Readers who want to get to the nitty-gritty of it are kindly referred to Boas (2003, 2005, 2008a,b); Croft (1998, 2003); Goldberg (1995, 2006); Goldberg & Jackendoff (2004); Iwata (2008); Kay (2005); Levin & Rappaport Hovav (2005); Müller (2006) and Nemoto (1998).

Unfortunately, whereas the lexicalist account can boast various computational operationalizations, such as LFG (Bresnan, 1982) and HPSG (Müller, 1996), there are only few attempts to scientifically validate the constructional voices of the debate. This paper addresses this issue and proposes a general design pattern for handling argument structure and provides a concrete operationalization of the pattern in Fluid Construction Grammar that works for both production and parsing.

3. A Design Pattern for Argument Structure

The challenge of argument structure can be reformulated as a general problem of how lexico-phrasal constructions can interact with more abstract, grammatical con-

structions in order to express different conceptualizations. The solution put forward in this paper involves two steps. First, lexical and phrasal constructions introduce their semantic and syntactic combinatorial potential. In the second step, argument structure constructions select an *actual* value from this potential and implement how semantic and syntactic categories map onto each other. The idea of connecting potential values to complex structures is firmly rooted in linguistic tradition and can at least be traced back as early as Benjamin Lee Whorf, who envisioned the linguistic inventory as a network-like structure in which "patterned 'potentials of linkage' [...] ramify from [words and morphemes] and connect them with complex patterns of linguistic formulation" (Whorf, 1973, p. 65). The remainder of this section first illustrates the design pattern through an example and then proceeds with the operationalization of the design pattern in FCG.

3.1. Example: Sent

This paper's approach can best be understood through an example. For instance, depending on the granularity of semantic representation that one chooses, the verb form *sent* contains at least three participant roles: a 'sender', a 'sendee' and a 'sent'. As illustrated in section 2, there is an indirect mapping between this participant structure, on the one hand, and which of the participant roles are overtly expressed and how they are marked, on the other. The following sentences only illustrate some of the argument realization patterns in which the verb can occur:

(10) *[Jack* $_{SUBJ}$*]* *sent* *[Jill* $_{IND-OBJ}$*]* *[a letter* $_{OBJ}$*].*
(sender-Agent) event (sendee-Recipient) (sent-Patient)

(11) *Has [the letter* $_{SUBJ}$*] been sent?*
– (sent-Patient) – event

(12) *[The letter* $_{SUBJ}$*] was sent [to Jill* $_{OBL}$*].*
(sent-Patient) event (sendee-Goal)

(13) *Sent?*
event

The linguistic facts suggest that it is impossible to implement a single definition of the verb's morphosyntactic distribution. If the context is clear and rich enough, for instance where two interlocutors have just been talking about sending an e-mail, it is even possible to cut out all of the verb's participants as shown in example 13.

Yet, patterns of argument realization are not random but are instead often conventionalized to a high degree. What is needed here, then, is some way in which the lexical construction can make predictions about how and which participant roles might be expressed without actually committing to any particular surface form realization. This effect can be achieved by giving up on the idea of a 'default' definition of a verb's grammatical behavior and let it introduce its semantic and syntactic combinatorial potential instead.

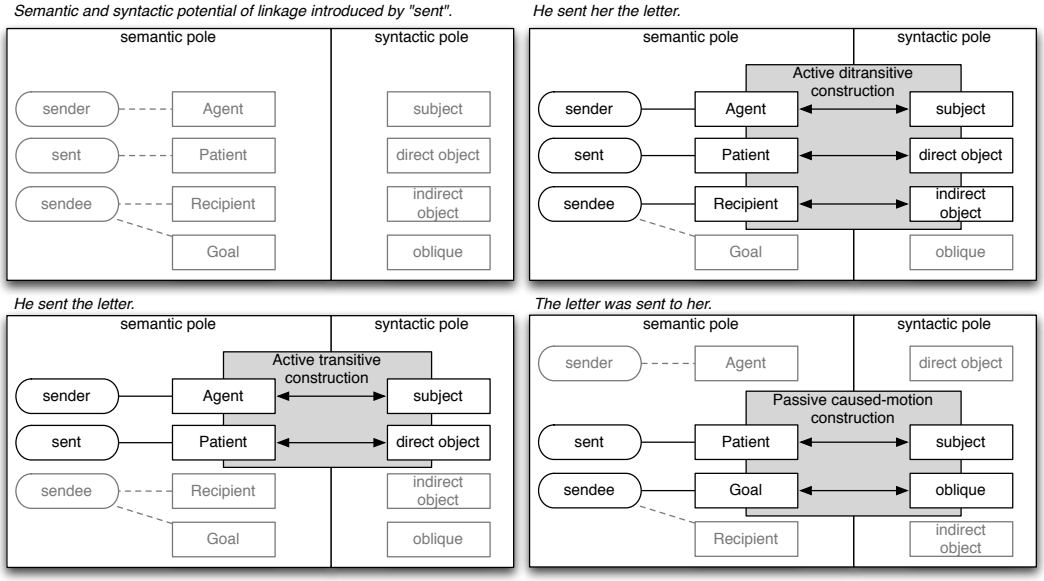

Figure 2. *This Figure illustrates how the design pattern applies for the verb form* sent. *The lexical construction for* sent *introduces its semantic and syntactic potential of linkage (top left). The other three boxes show examples of how different argument structure constructions select an actual value and implement the linking between semantics and syntax, which yields different argument realizations such as* He sent her the letter *(top right),* He sent the letter *(bottom left) and* The letter was sent to her *(bottom right). As the latter example shows, passivization does not require a derivational rule in this approach.*

Such potential is usually called *valence* in the linguistic literature. The top left of Figure 2 shows the *potential* semantic and syntactic valence of *sent*. The participant roles are listed on the left, and they are potentially linked to semantic roles such as Agent, Patient, Recipient and Goal. This 'potential' means that if the 'sender' needs to be expressed, it can be mapped onto Agent; if the 'sent' needs to be expressed, it can be mapped onto Patient; and if the 'sendee' needs to be expressed,

it can be mapped onto either the Recipient or the Goal role of an utterance. On the syntactic pole, the potential syntactic valence includes the syntactic roles subject, direct object, indirect object and oblique. As opposed to lexicalist accounts, the lexical construction does not state how semantic roles and syntactic roles should be mapped onto each other and which of them, if any, need to be overtly expressed.

The other three boxes in the Figure illustrate how various argument structure constructions can then select from the combinatorial potential what they require and implement the actual linking between semantics and syntax. The top right box shows how the Active ditransitive construction selects an Agent, Patient and Recipient and maps them onto subject, direct object and indirect object, which yields utterances such as *He sent her the letter*. The Active transitive construction (bottom left) only selects an Agent and Patient on the semantic pole and subject and direct object on the syntactic pole, and thus accounts for utterances such as *He sent the letter*. In line with most construction grammar theories, the passive construction is treated as an alternative argument structure construction instead of as a derivational lexical construction. As can be seen in the bottom right of Figure 2, the Passive caused-motion construction selects a Patient and a Goal, and maps the Patient onto subject and the Goal onto oblique for utterances such as *The letter was sent to her*. When parsing utterances, the same argument structure constructions operate in the opposite direction: syntactic roles are mapped onto semantic roles, and linked to the corresponding participant roles.

3.2. Operationalization through Templates

Turning to the operationalization of the design pattern in FCG, this paper uses templates for lexical and phrasal constructions proposed by Steels (2011a,b,c) and adds its own templates for argument structure. Templates are needed for operationalizing the two steps of the design pattern for argument structure:

1. Lexical and phrasal constructions require templates for introducing their semantic and syntactic combinatorial potential. In the case of verbal constructions, a verb must introduce its potential semantic and syntactic valence. This is done with the template `def-lex-valence`. Potential valence can be percolated to phrasal units using the phrasal templates discussed by Steels (2011a). For reasons of space, this paper does not introduce templates that are devoted to the potential of linkage of nominal constructions and uses the default lexical templates instead. A more detailed approach is described by van Trijp (2011).

2. Argument structure constructions orchestrate a mapping between semantics and syntax, but they do not create additional structure. Argument structure constructions are built using a template called `def-arg-cxn`, which may encompass the following templates for argument structure:

   (a) The template `def-arg-skeleton` sets up the basic structure that is required by the argument structure.

   (b) Argument structure constructions may also introduce constructional meanings and form constraints. These constraints are defined using a template called `def-arg-require`.

   (c) The `def-arg-mapping` template is used for mapping semantic roles onto syntactic roles and for indicating participant structure through variable equalities.

The remainder of this chapter shows how these templates build constructions and how these constructions are then processed for producing or parsing utterances. It falls beyond the scope of this paper to discuss the full depth of linguistic processing. Interested readers are kindly referred to Bleys et al. (2011), De Beule & Steels (2005) and Steels & De Beule (2006) for more details on the application of constructions.

4. Representing Participant Structure

The first requirement of a satisfactory operationalization is an adequate representation of meaning, which is here achieved through first order-predicate calculus. Lexical constructions provide meaning predicates, whereas argument structure constructions connect these meanings to each other by making coreferential variables equal (Steels et al., 2005). Moreover, they can also contribute additional meanings.

4.1. Lexical meanings

Verbal lexical constructions introduce a predicate for the event itself and predicates for every participant role. For example, the verb *to send* may introduce three participant roles:

(14)
```
((send ?event)
 (sender ?event ?participant-1)
 (sendee ?event ?participant-2)
 (sent ?event ?participant-3))
```

Every symbol that starts with a question mark is a variable that can be bound to a specific referent in the world. For example, the variable `?event` can be bound to a specific send-event, `?participant-1` to the sender of that event, and so on. Other lexical items are represented in the same way. For instance, in the sentence *Jack sent Jill a letter*, the lexical entries for *Jack*, *Jill* and *a letter* introduce the following predicates:[1]

(15) `(jack ?x)`

(16) `(jill ?y)`

(17) `(letter ?z)`

Figure 3 represents these lexical meanings in the form of a network. As can be seen, the lexical meanings of *Jack*, *Jill* and *a letter* are unconnected to the verbal semantics in the network. That is, the lexical constructions already provide a lot of meaning, but they do not tell the hearer 'who did what to whom' (i.e. mark participant structure).

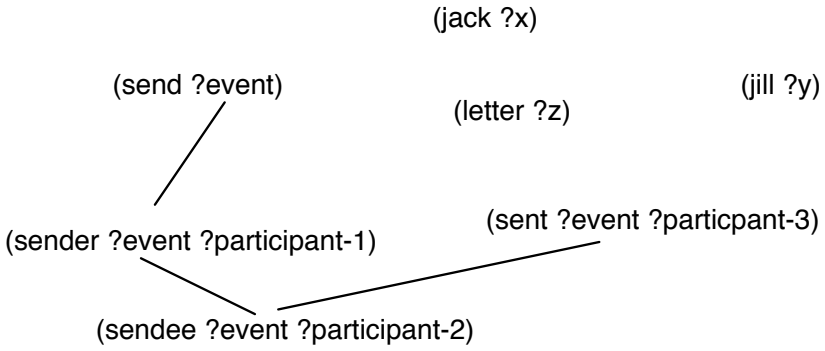

Figure 3. *Network representation of the meanings of the lexical constructions of* Jack*,* sent*,* Jill *and* a letter.

4.2. Connecting Meanings

One of the main functions of argument structure constructions is to indicate the participant structure underlying a sentence. For instance, the grammar of English

1. For the sake of convenience, this paper only focuses on argument structure and therefore ignore issues of determination, tense-aspect, etc.

makes it clear through word order that *Jack* is the sender and *Jill* the recipient in our current example. In the implementation, this function is achieved through variables, which means that variables are made equal if they are coreferential. For example, the variables for *Jack* (`?x`) and the sender of the event (`?participant-1`) are both bound to the same referent [JACK], hence their variables are made equal. Likewise, the variables for *Jill* and the sendee are made equal, and the variables for *a letter* and the object that was sent are made equal. This yields a new network in which all relevant meanings are connected to each other, as illustrated in Figure 4.

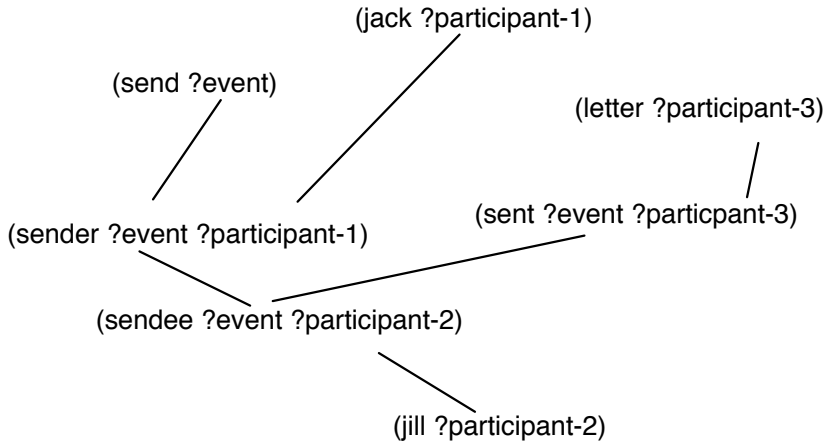

Figure 4. *Argument structure constructions connect lexical meanings to each other and thereby make the participant structure of a sentence explicit.*

4.3. Constructional Meanings

Making the participant structure of a sentence explicit is meaningful in itself, but one of the basic tenets of construction grammar is that grammatical constructions can also contribute meanings in the same way as lexical constructions do. Argument structure constructions are hypothesized to express 'humanly relevant scenes' in the form of more abstract event-types such as 'cause-receive' and 'cause-motion' (Goldberg, 1995, p. 39). In our example, the verb *to send* interacts with the ditransitive construction, which is associated with the more abstract constructional meaning 'X causes Y to receive Z'. In the implementation, constructional meanings can be represented as predicates as well:

(18)
```
((cause-receive ?event)
 (causer ?event ?a)
```

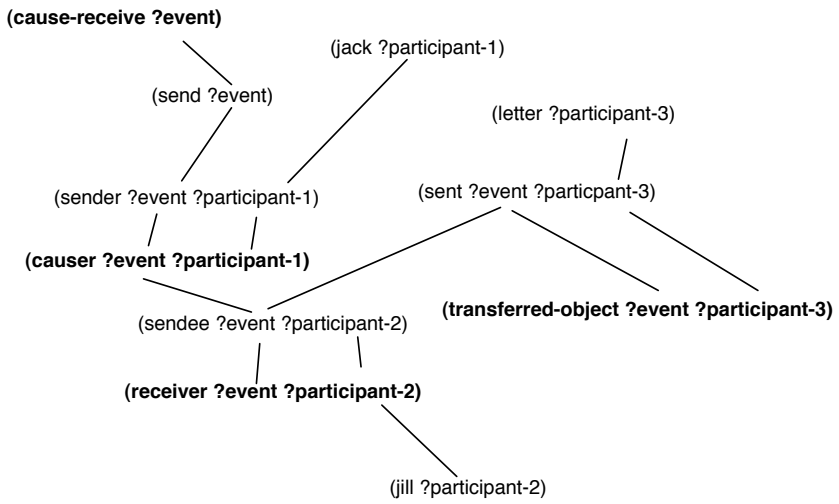

Figure 5. *Constructional meanings are represented and connected to other meanings in the same way as lexical meanings are.*

```
(transferred-object ?event ?b)
(receiver ?event ?c))
```

For clarity's sake, I will use *argument roles* for referring to the more abstract constructional roles such as `causer` and `receiver`. *Argument roles* are conceptual categories and hence part of a construction's `meaning` feature, whereas *semantic roles* are grammatical categories that are part the `sem-cat` feature. In order to avoid confusion, semantic roles are always written with a capital letter.[2] As shown in Figure 5, argument structure constructions also connect their constructional meanings to the other meanings through variable equalities.

5. Lexical Constructions

This paper follows the same approach as Steels (2011a) by building constructions through progressive design, starting with a skeleton and then adding more and more feature structures through the use of templates.

2. This only counts for the verbal explanations in this paper and not for examples from the actual implementation in which capitalization does not matter.

5.1. Verbal Lexical Constructions

The basic lexical construction for a verb is defined using the same templates as proposed by Steels (2011a). The following example illustrates the basic definition of a construction for the verb form *sent* using the `def-lex-cxn` template (including the use of the templates `def-lex-skeleton` and `def-lex-cat`):

(19)
```
(def-lex-cxn sent-lex
 (def-lex-skeleton sent-lex
  :meaning (== (send ?ev)
               (sender ?ev ?sender)
               (sendee ?ev ?sendee)
               (sent ?ev ?sent))
  :args (?ev)
  :string "sent")

 (def-lex-cat sent-lex
  :sem-cat (==1 (class event)
                (sem-function predicating))
  :syn-cat (==1 (syn-function verbal)
                (lex-cat verb))))
```

The verb's potential semantic and syntactic valence is defined using a template called `def-lex-valence`:

(20)
```
(def-lex-valence sent-lex
  :sem-roles ((agent sender)
              (patient sent)
              (recipient sendee)
              (goal sendee))
  :syn-roles (subject direct-object
              indirect-object oblique))
```

The `def-lex-valence` template contains two slots. The first slot, `:sem-roles`, takes a list of pairs as its value. Each pair consists of a semantic role and its corresponding participant role in the meaning of the verb. As can be seen, there are two potential semantic roles for the 'sendee' of the verb: Recipient (see example 10) and Goal (see example 12). The template will take this value and expand it into a feature called `sem-valence`, which itself is one of the values of the verb's `sem-cat` feature:

(21)
```
(sem-valence ((agent ?ev ?sender)
              (recipient ?ev ?sendee)
              (patient ?ev ?sent)
              (goal ?ev ?sendee)))
```

The elements in the value of `sem-valence` contain the same variable names as the ones used in the `meaning` that was defined in (19). For example, the semantic role `Agent` shares the same variable `?sender` with the participant role `sender`, which means that if the `sender` role needs to be expressed, it can be mapped onto the semantic role of `Agent`. Likewise, the participant role that takes the variable `?sent` can be mapped onto the semantic role of `Patient`.

The second slot of the `def-lex-valence` template is `:syn-roles`, which takes a list of syntactic roles as its value. In the current example, these are `subject`, `direct-object`, `indirect-object` and `oblique`. The template expands the value of this slot in a feature called `syn-valence`, which is part of the verb's `syn-cat` feature:

(22)
```
(syn-valence
    ((subject ?subj-unit)
     (object ?obj-unit)
     (indirect-object ?ind-obj-unit)
     (oblique ?obl-unit)))
```

The `syn-valence` feature does not contain any variable that corresponds to a variable in the verb's `sem-valence`, which means that there isn't a direct relation between semantic roles and syntactic roles, as illustrated in the above examples and in section 2.2. If any of these syntactic roles are actually expressed in an utterance, their variable names have to be bound to the units to which the roles are or need to be assigned.

The semantic and syntactic valence features capture the conventionalized distributional properties of verbs and therefore constrain the argument realization patterns in which they may occur. However, these are only potential values from which grammatical constructions have to select an actual valence later on.

5.2. Nominal Lexical Constructions

The same principle of combinatorial potential versus actual value can also be applied to other lexical and phrasal constructions. Just like a verbal lexical construction contains information about its semantic and syntactic valence, nominal

lexical constructions may open a stream of possibilities about which semantic and syntactic role they *might* play in a sentence. Depending on the grammatical context, other constructions may then later decide on the actual roles that are assigned to the nominal. The following examples illustrate how nominals can impose further restrictions on possible argument realization patterns in a language:

(23) ?? She gave the table a present.

(24) ?? He carried a hole to the other side of the river.

Example (23) is unacceptable to speakers of English unless the table is some kind of anthropomorphic entity with human-like qualities in a story or cartoon. The unacceptability comes from the observation that the English semantic role of Recipient is restricted to animate beings. Similarly in example 24, *a hole* is a non-tangible, non-moveable object that cannot be carried around, hence it is semantically incompatible with the English caused-motion construction or caused-motion verbs such as *to carry*. Thus, a nominal construction requires the features `sem-role` and `syn-role` that already introduce possibilities concerning the semantic and syntactic role that the nominal might play in the utterance. As the mapping between semantic and syntactic roles is based on more coarse-grained abstractions, the nominal construction also needs additional semantic properties that may block certain argument realization patterns if there is a semantic conflict with the selectional restrictions of the verb. The default lexical templates implement all these requirements in a lexical construction for *table*:

(25)
```
(def-lex-cxn table-lex

 (def-lex-skeleton table-lex
  :meaning (== (table ?referent))
  :args (?referent)
  :string "table")

 (def-lex-cat table-lex
  :sem-cat (==1 (class object)
                (sem-role ?sem-role)
                (is-animate? -)
                (is-moveable? +))
  :syn-cat (==1 (lex-cat noun)
                (syn-role ?syn-role))))
```

Due to space limitations, this paper limits the values of the `sem-role` and `syn-role` features to variables, which means that they can potentially play any role in an utterance. A more realistic and detailed account is described by van Trijp (2011). The selectional restrictions are represented as binary features such as `is-animate?` and `is-moveable?` that take either '+' or '–' as their value. They are considered to be semantic features that are grammatically relevant in a particular language. That is, they represent semantic dimensions that matter for allowing or disallowing constructions to interact with each other on a transient structure. Here, only two selectional restrictions are included for illustrative purposes. An example of a more complete treatment is discussed by Beuls (2011).

5.3. Example of Parsing

After defining a number of lexical constructions, it is already possible to investigate how they are processed in either production and parsing. Here, a parsing example is provided of the sentence *Jack sent Jill a letter.* For ease of exposition, all four phrases in the utterance are treated as if they are single lexical constructions. Other papers in this volume explain in more detail how to deal with those aspects of the utterance which are scaffolded here, such as phrasal constructions (Steels, 2011a) and agreement (Beuls, 2011; van Trijp, 2011). The example also assumes that the utterance has been segmented into the following form, consisting of a string for each 'word' (or phrase) and ordering constraints (`meets`):

(26)
```
((string ?jack-unit "Jack")
 (string ?sent-unit "sent")
 (string ?jill-unit "Jill")
 (string ?letter-unit "a letter")
 (meets ?jack-unit ?sent-unit)
 (meets ?sent-unit ?jill-unit)
 (meets ?jill-unit ?letter-unit))
```

When parsing this utterance, the four lexical constructions can each apply and analyze a part of this form. The resulting transient structure is shown in Figure 6. As can be seen, each construction has created a separate unit for each phrase on both the semantic and syntactic poles. When the meanings of each unit are inspected, it is clear that each meaning predicate still has its own unique variable, which indicates that the meanings of the utterance are not connected to each other yet. If production were undertaken, the constructions would have created a similar transient structure.

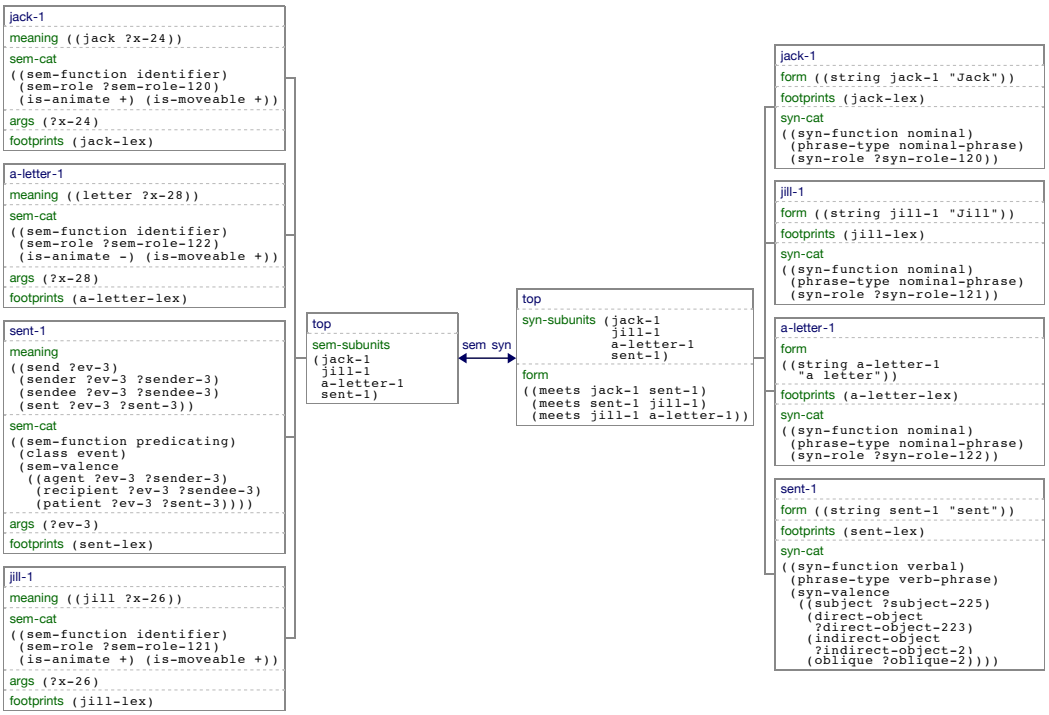

Figure 6. *Transient structure after applying the lexical constructions.*

6. Argument Structure Constructions

As was illustrated in Figure 2, argument structure constructions implement a mapping between semantic and syntactic categories, and they decide on the *actual* valence and roles of lexico-phrasal units from a unit's combinatorial potential. Under certain conditions, it is also possible for constructions to *impose* their syntactic and semantic constraints rather than select them. This operation can be observed in coercion effects, as in the well-known example *Pat sneezed the napkin off the table* (Goldberg, 1995, p. 3), where the caused-motion construction adds a caused-motion reading to the verb *to sneeze* (i.e. Pat caused the napkin to move off the table by sneezing), which usually behaves as an intransitive verb. This paper only covers routine processing of argument structure; issues concerning flexibility and robustness in language processing are discussed by Steels & van Trijp (2011).

All argument structure templates are grouped together with the template `def-arg-cxn`, which takes the following form:

(27)
```
(def-arg-cxn  cxn-name
 (def-arg-skeleton  cxn-name
     ...)
  ...)
```

6.1. Setting up the Argument Structure

Instantiating an argument structure construction always starts with setting up its basic structure using the `def-arg-skeleton` template. This template lists a unit for the main event of the utterance and all the units for the participants of that event that need to be overtly expressed. Each unit takes two slots (`:sem-cat` and `:syn-cat`) which are used for constraining the type of unit that the argument structure construction requires. Here is the definition of the basic skeleton of an active ditransitive construction using the template:

(28)
```
(def-arg-skeleton ditransitive-cxn
 ((?event-unit
   :sem-cat (==1 (sem-function predicating))
   :syn-cat (==1 (syn-function verbal)))
  (?agent-unit
   :sem-cat (==1 (sem-function identifier))
   :syn-cat (==1 (syn-function nominal)))
  (?recipient-unit
   :sem-cat (==1 (sem-function identifier))
   :syn-cat (==1 (syn-function nominal)))
  (?patient-unit
   :sem-cat (==1 (sem-function identifier))
   :syn-cat (==1 (syn-function nominal)))))
```

The above template creates a construction with one verbal unit and three nominal units. Since our current example treats phrases as if they were lexical constructions, the template only specifies the required syntactic and semantic function of each unit. In a more realistic approach, it would also identify a unit's phrase type.

6.2. Adding Constructional Meaning and Form

Just like lexical and other types of constructions, argument structure constructions are able to handle or impose form and meaning. This information is specified through the `def-arg-require` template, which states that a certain form or meaning is 'required' by the construction when it is used for matching, or 'imposed' by the construction when it is used in merging. (See Bleys et al., 2011, for more on the matching and merging phases of constructional application.)

The current example assumes a fixed word order for ditransitive constructions, which is represented in the slot `:cxn-form`. In more realistic grammars, however, the word order of a declarative construction may shift depending on considerations of the information structure of a sentence. Interested readers can check Micelli (2012) to see how such cases can be handled as well. The constructional meanings (see section 4) fill the `:cxn-meaning` slot. The template uses the names of the units in which it is going to store the constructional forms and meanings:

(29)
```
(def-arg-require ditransitive-cxn
 ((?event-unit
   :cxn-meaning
    (==
     (cause-receive ?ev)
     (causer ?ev ?causer)
     (receiver ?ev ?receiver)
     (transferred-object
        ?ev ?transferred-object))
   :cxn-form
    (==
     (meets ?agent-unit ?event-unit)
     (meets ?event-unit ?recipient-unit)
     (meets
        ?recipient-unit ?patient-unit)))))
```

6.3. Participant Structure and Mapping between Semantics and Syntax

The most important function of argument structure constructions – mapping semantics onto syntax and thereby indicating participant structure – is captured through a template called `def-arg-mapping`. This template has two main slots: `:event` for specifying the actual valence of the event-unit, and `participants` for specifying the actual semantic and syntactic roles of the participants.

The value of the slot `:event` is a list that starts with the unit-name of the event-unit, which is here `?event-unit`. Next, three slots have to be filled: `:args`, `:sem-valence` and `:syn-valence`. In both valence slots it is crucial to use the correct variable names. For example, the variable name `?causer` for the Agent role is the same one as the variable name that was used for the argument role `causer` in the `def-arg-require` template, which represents the fact that they are linked to each other. Similarly, the Recipient role shares a variable with the `receiver`, the Patient shares a variable with the `transferred-object`, and so on. For each syntactic role in the `:syn-valence` slot, the variable of the corresponding unit-name is used.

The `:participants` slot lists the units of the participants. For each unit, there are three slots: `:sem-role`, `:syn-role` and `:args`. The first two slots require the name of the semantic or syntactic role that is assigned to the unit (e.g. Agent and subject). The `:args` slot is used for indicating participant structure by linking the meaning of the participant units to the meaning of the verbal unit. The value of this slot therefore always shares a variable with one of the variables in the semantic valence of the verb unit. The use of the `:args` slot is also discussed in more detail by Steels (2011a).

(30)
```
(def-arg-mapping ditransitive-cxn
 :event
  (?event-unit
    :args (?ev)
    :sem-valence
       (==1
        (agent ?ev ?causer)
        (recipient ?ev ?receiver)
        (patient ?ev ?transferred-object))
    :syn-valence
     (==1 (subject ?agent-unit)
          (indirect-object ?recipient-unit)
          (direct-object ?patient-unit)))
 :participants
  ((?agent-unit
     :sem-role agent
     :syn-role subject
     :args (?causer))
   (?recipient-unit
    :sem-role recipient
    :syn-role indirect-object
    :args (?receiver))
  (?patient-unit
    :sem-role patient
    :syn-role direct-object
    :args (?transferred-object))))))
```

6.4. Example of Parsing

Let's illustrate how argument structure constructions are processed starting from the transient structure as depicted in Figure 6, which was obtained after applying four lexical constructions for *Jack*, *sent*, *Jill* and *a letter*. During parsing, a successful application of an argument structure construction involves the following steps: (a) it identifies which units play which syntactic roles, (b) it maps the syntactic roles onto semantic roles, (c) it indicates the participant structure and (d) it adds constructional meanings.

The first step is the identification of which units play which syntactic roles. Recall that the `def-arg-require` template specified that the ditransitive construction

expects a particular word order. Using this information, the construction can bind the variables for its unit names to their corresponding units in the transient structure: `?agent-unit` is bound to `jack-unit`, `?recipient-unit` is bound to `jill-unit` and `?patient-unit` is bound to the `letter-unit`. Since the `def-arg-mapping` template repeats the construction's unit-names in the event's `:syn-valence` slot, the corresponding syntactic roles can be unambiguously assigned to the correct units. This means that `jack-unit` plays the subject role, `jill-unit` the indirect object role, and `letter-unit` the direct object role.

Next, the construction maps syntactic roles onto semantic roles. The `def-arg-mapping` template specified that subject maps onto Agent, indirect object onto Recipient and direct object onto Patient. Having identified which units play which semantic roles, the construction can also make the utterance's participant structure explicit by making coreferential variables equal. This is achieved through the equalities between the variables in the `:args` slots of the nominal units and the variables in the event-unit's `:sem-valence` slot. Since the verbal lexical construction had already specified how its semantic roles have to be linked to its participant roles, the meanings of the participant units are automatically linked to the meanings of the event unit. FInally, the construction adds the constructional meaning to the transient structure that was specified in the `def-arg-require` template. The resulting transient structure is shown in Figure 7.

6.5. Applying Argument Structure Constructions in Production

In production, argument structure constructions assign semantic roles and map them onto syntactic roles. Since the speaker knows what he or she wants to say, the participant structure is already clear from the start so there are no variables in the meanings that need to be expressed:

(31)
```
((send ev-1)
 (sender ev-1 [JACK])
 (sendee ev-1 [JILL])
 (sent ev-1 [LETTER])
 (cause-receive ev-1)
 (causer ev-1 [JACK])
 (receiver ev-1 [JILL])
 (transferred-object ev-1 [LETTER])
 (jack [JACK])
 (jill [JILL])
 (letter [LETTER]))
```

The construction exploits the participant structure for figuring out which units play which semantic roles. For example, through the equality of `?causer` in the `args` feature of `?agent-unit` and in the verb's semantic valence, the construction is able to identify *Jack* as the Agent of the utterance. Analogously to parsing, the construction then exploits unit-names for mapping semantic roles onto syntactic roles. Finally, the construction can add its constructional form constraints to the transient structure.

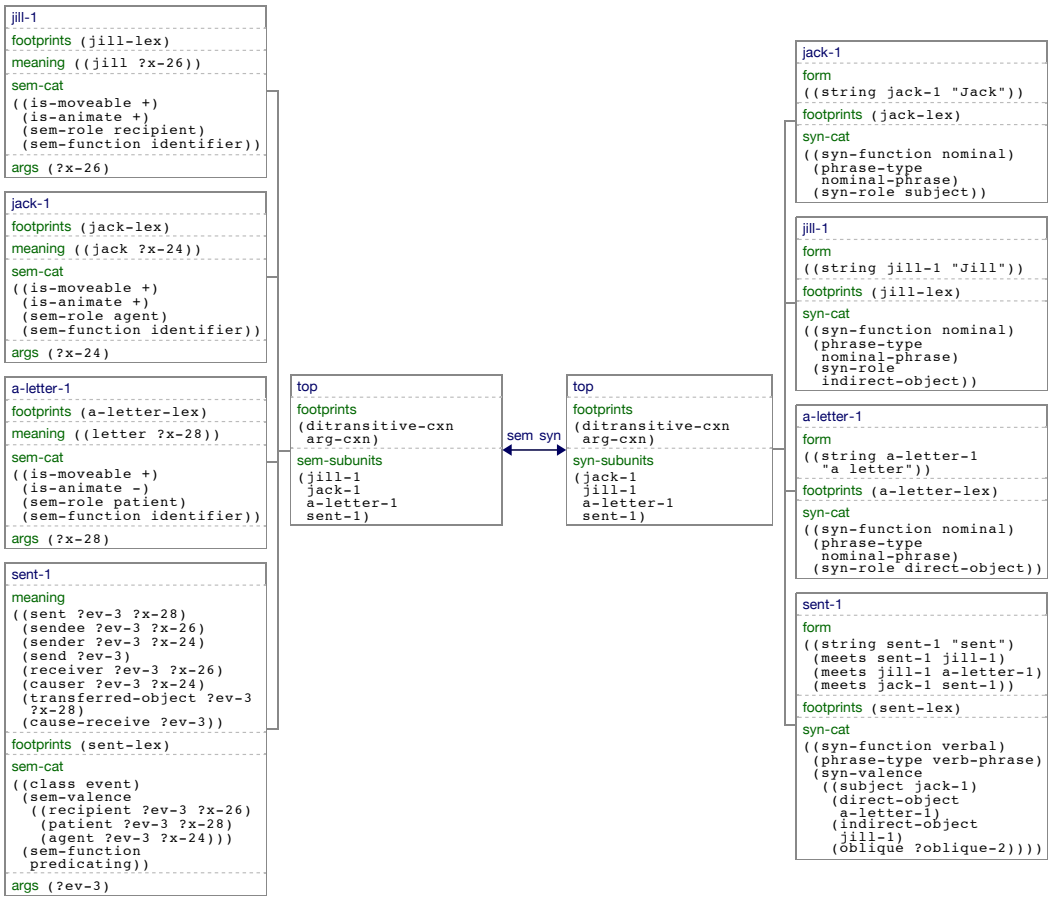

Figure 7. *The resulting transient structure after applying the ditransitive construction.*

7. Assessment and Outlook

The previous sections proposed a fully operational implementation of argument structure in Fluid Construction Grammar that works for both production and comprehension. At the heart of this operationalization lies the design pattern that allows some constructions to introduce their semantic and syntactic potential from which other constructions may select an actual value. What follows is the assessment of this design pattern with respect to the following two questions:

1. Does the design pattern contribute to a better formalization of the chosen domain (in this case: argument structure)?
2. What are the consequences of using the design pattern for grammar engineering and language processing?

7.1. A Usage-Based Approach

In order to answer the first question, it is necessary to clearly state the objectives of the formalization. Within the family of construction grammar theories, there are roughly two different views on what linguistics should be concerned with, leading to two different scientific objectives. The first view treats construction grammar as a 'generative theory' in the sense that the grammar should account for all the possible sentences of a particular language. Example representatives are Berkeley Construction Grammar (Kay & Fillmore, 1999; Kay, 2005) and Sign-Based Construction Grammar (Michaelis, 2009). The other view is a usage-based approach to language (Langacker, 2000) that takes the effects of communication into account in its grammatical descriptions. The usage-based approach accepts various degrees of entrenchment of linguistic conventions, and assumes that the linguistic inventory of a speaker is dynamically updated after each communicative interaction. Examples of this approach are Cognitive Grammar (Langacker, 1987), Lakovian/Goldbergian construction grammar (Lakoff, 1987; Goldberg, 1995, 2006) and Radical Construction Grammar (Croft, 2001). This paper subscribes to the usage-based view on language as well.

The difference between both approaches becomes more clear through an example. According to Goldberg (1995, p. 53), the verb *to hand* takes three obligatory participant roles (as in *Jack handed Jill a letter*), hence it would be ungrammatical to say **Jack handed a letter*. Since the generative approach is mainly concerned with grammaticality judgments, it would simply dismiss the latter example as a valid English utterance. The usage-based approach, however, argues that even though the

verb *to hand* is not conventionally associated with the transitive construction, the sentence is still intelligible to native speakers of English given the right contextual information. So one important assessment criteria for the FCG implementation is to see whether it can still come up with a good parse. First we define the lexical entry for the verb form *handed* using the `def-lex-cxn` template:

(32)
```
(def-lex-cxn handed-lex
 (def-lex-skeleton handed-lex
  :meaning (== (hand ?ev)
               (hander ?ev ?hander)
               (handee ?ev ?handee)
               (handed ?ev ?handed))
  :args (?ev)
  :string "handed")
 (def-lex-cat handed-lex
  :sem-cat (==1 (sem-function predicating)
                (class event))
  :syn-cat (==1 (syn-function verbal)
                (phrase-type verbal-phrase)))
 (def-lex-valence handed-lex
  :sem-roles ((agent hander)
              (recipient handee)
              (patient handed))
  :syn-roles (subject direct-object
              indirect-object oblique)))
```

Next, the argument structure templates are used for defining the transitive construction. In the following definition, the `def-arg-require` template does not specify any constructional meaning for the transitive constructions. This is a deliberate choice because many linguists might argue that very abstract constructions (such as the transitive construction) cannot be associated with any specific argument frame and hence only express grammatical functions. The remainder of the definition looks similar to that of the ditransitive construction, with the difference that there is no recipient-unit:

(33)

```
(def-arg-cxn transitive-cxn

 (def-arg-skeleton transitive-cxn
  ((?event-unit
    :sem-cat (==1 (sem-function predicating))
    :syn-cat (==1 (syn-function verbal)))
   (?agent-unit
    :sem-cat (==1 (sem-function identifier))
    :syn-cat (==1 (syn-function nominal)))
   (?patient-unit
    :sem-cat (==1 (sem-function identifier))
    :syn-cat (==1 (syn-function nominal)))))

 (def-arg-require transitive-cxn
  ((?event-unit
    :cxn-form
     (==
      (meets ?agent-unit ?event-unit)
      (meets ?event-unit ?patient-unit)))))

 (def-arg-mapping transitive-cxn
  :event
   (?event-unit
    :args (?ev)
    :sem-valence (==1 (agent ?ev ?agent)
                      (patient ?ev ?patient))
    :syn-valence
     (==1 (subject ?agent-unit)
          (direct-object ?patient-unit)))
  :participants ((?agent-unit
                  :sem-role agent
                  :syn-role subject
                  :args (?agent))
                 (?patient-unit
                  :sem-role patient
                  :syn-role direct-object
                  :args (?patient)))))
```

If the FCG-system now parses the utterance *Jack handed a letter*, the transitive construction can successfully apply, as shown in Figure 8. Application is possible because the construction finds its required semantic and syntactic roles in the verb's potential valence and it finds the right number of participant units. Parsing the utterance yields the following meanings:

(34)
```
((hand ?ev)
 (hander ?ev ?jack-ref)
 (handee ?ev ?handee)
 (handed ?ev ?letter-ref)
 (jack ?jack-ref)
 (letter ?letter-ref))
```

The transitive construction successfully indicates that *Jack* is the hander of the utterance and that *a letter* is the object handed over. The variable for the handee role (`?handee`) is unconnected to the rest of the network, hence it remains implicit who is the recipient. In other words, the FCG implementation doesn't break down but comes up with a parse that corresponds to how native speakers of English would comprehend the utterance. This fact suggests that the design pattern proposed in this paper, which rests on an interplay between constructions, is better suited for usage-based accounts of language than traditional implementations in which morphosyntactic behavior is defined in a single and fixed position (e.g. defining a verb's behavior entirely in the lexicon).

7.2. Consequences for Grammar Design

Every linguist agrees that language is full of subregularities and pockets of exceptions, hence it doesn't take much effort to find attested examples in corpora or on the web in which for example *to hand* is actually used as a transitive verb. As argued above, the design pattern proposed in this paper can handle such infrequent cases without resorting to additional operations or formal tools. However, it doesn't make a distinction between strongly entrenched and less acceptable cases. In terms of grammar design, the technique therefore needs to be complemented with ways to dynamically steer the search process in which FCG looks for the best verbalization or parse of an utterance.

One particular consequence is that the language user needs to keep track of 'coapplication links' in his or her linguistic inventory. Coapplication links are links between constructions that have applied together to verbalize or analyze an

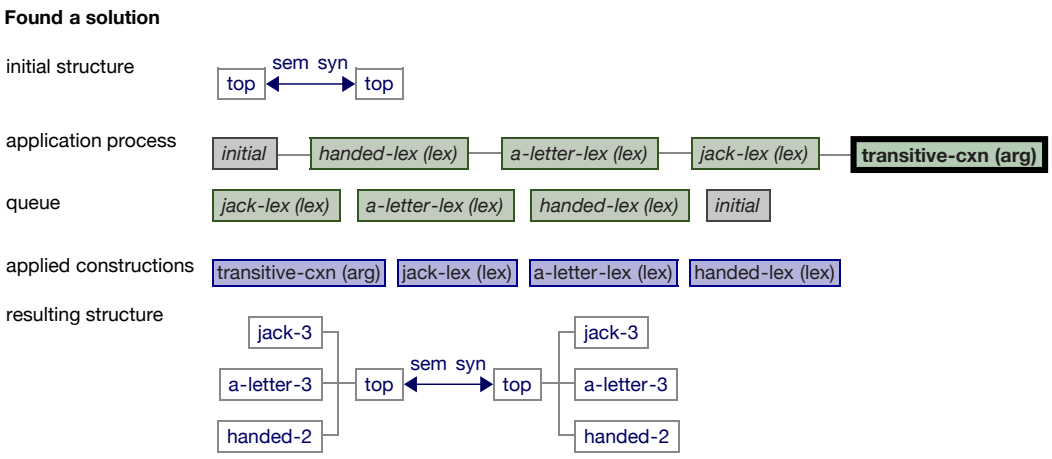

Figure 8. *The FCG implementation allows the verb* to hand *to occur in a transitive argument realization pattern.*

utterance. Each link has a score that reflects the frequency of coapplication and hence the degree of acceptability for two or more constructions to interact with each other. Figure 9 illustrates such links for the verb *to hand*. As can be seen, the verb has strong coapplication links with, for instance, the ditransitive and prepositional ditransitive constructions, but a weak link with the transitive construction. The scores of these coapplication links are dynamically updated after each linguistic interaction. Besides coapplication links, other network links may exist between constructions. These issues are explored in-depth by Wellens & De Beule (2010) and Wellens (2011).

The combination of coapplication links with the design pattern of potential versus actual values arguably also allows linguists to make better predictions about possible changes in a language. For example, the coapplication link between *to hand* and the transitive construction may be infrequent in present-day English, but at some point become a perfectly conventionalized usage in the language. It is a widely accepted phenomenon that semantic and syntactic overlap between constructions may trigger novel distributional patterns.

The implementation proposed in this paper of course also has its limits and it cannot account for all cases of novelty or unconventional language usage: it only accommodates for unconventional utterances in which the argument structure con-

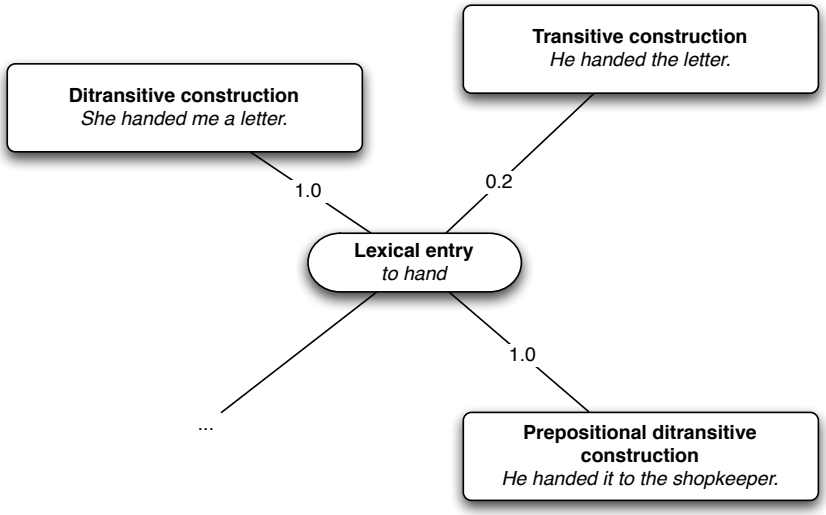

Figure 9. *The linguistic inventory keeps coapplication links between constructions. The coapplication scores in this network are dynamically updated through language usage and reflect the degree of entrenchment of two (or more) constructions interacting with each other. In the most simple case, a coapplication link just counts raw frequency and therefore its score has no upper bound limit.*

structions still find their required valence in the verb's potential. In case of coercion by construction, however, an argument structure construction needs to impose additional semantic and syntactic roles, for which additional solutions are necessary. Such cases fall beyond the scope of this paper and are dealt with in a later chapter in this book (Steels & van Trijp, 2011).

8. Conclusions

This paper has illustrated how argument structure can be handled in Fluid Construction Grammar. It first presented the challenges of argument structure by showing examples of the indirect and multilayered mapping between meaning and form. Next, it proposed a design pattern that relies on the interplay between constructions, in which some constructions introduce their semantic and syntactic combinatorial

potential from which others select an actual value and implement a mapping between semantics and syntax.

The paper offered several templates that operationalize this design pattern. More specifically, these templates introduce the features `sem-valence` and `syn-valence` for verbal constructions, and `sem-role` and `syn-role` for nominal constructions. Argument structure constructions then select the valence that they require and organize the mapping between semantic and syntactic roles. They also indicate participant structure through variable equalities and may contribute additional constructional meanings to the utterance.

Finally, the paper argued that this approach to argument structure answers better to the requirements of usage-based accounts of language than techniques that have been designed for making grammaticality judgments. In order to handle different degrees of acceptability, however, it needs to be complemented with techniques for steering linguistic processing, for example through coapplication links between constructions or other network relations.

Acknowledgements

The research described in this paper was funded by the Sony Computer Science Laboratory Paris, the EU FP 6 ECAgents project and the EU FP7 Alear project. I wish to thank Luc Steels for his invaluable feedback on this work, as well as my colleagues from Sony CSL Paris and the VUB AI-Lab at the University of Brussels, particularly Katrien Beuls, Joachim De Beule and Vanessa Micelli for their useful comments. I also thank the anonymous reviewers for their constructive feedback, which helped to improve this paper. All remaining errors are of course my own.

References

Beuls, Katrien (2011). Construction sets and unmarked forms: A case study for Hungarian verbal agreement. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Bleys, Joris, Kevin Stadler, Joachim De Beule (2011). Search in linguistic processing. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Boas, Hans (2003). *A Constructional Approach to Resultatives*. Stanford Monograph in Linguistics. Stanford: CSLI.

Boas, Hans (2005). Determining the productivity of resultative constructions: A reply to Goldberg & Jackendoff. *Language*, 81(2), 448–464.

Boas, Hans (2008a). Determining the structure of lexical entries and grammatical constructions in construction grammar. *Annual Review of Cognitive Linguistics*, 6, 113–144.

Boas, Hans (2008b). Resolving form-meaning discrepancies in construction grammar. In Jaako Leino (Ed.), *Constructional Reorganization*, 11–36. Amsterdam: John Benjamins.

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

Croft, William (1991). *Syntactic Categories and Grammatical Relations. The Cognitive Organization of Information*. Chicago: Chicago UP.

Croft, William (1998). Event structure in argument linking. In Miriam Butt, Wilhelm Geuder (Eds.), *The Projection of Arguments: Lexical and Compositional Factors*, 21–63. Stanford: CSLI Publications.

Croft, William (2001). *Radical Construction Grammar: Syntactic Theory in Typological Perspective*. Oxford: Oxford UP.

Croft, William (2003). Lexical rules vs. constructions: A false dichotomy. In Hubert Cuyckens, Thomas Berg, René Dirven, Klaus-Uwe Panther (Eds.), *Motivation in Language Studies: Studies in Honour of Günter Radden*, 49–68. Amsterdam: John Benjamins.

De Beule, Joachim, Luc Steels (2005). Hierarchy in Fluid Construction Grammar. In Ulrich Furbach (Ed.), *KI 2005: Advances In Artificial Intelligence. Proceedings of the 28th German Conference on AI, Lecture Notes in Artificial Intelligence*, vol. 3698, 1–15. Berlin: Springer.

Dowty, David (1991). Thematic proto-roles and argument selection. *Language*, 67, 547–619.

Evans, Nicholas, Stephen Levinson (2009). The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32(5), 472–484.

Fillmore, Charles (1968). The case for case. In Emmon Bach, Robert Harms (Eds.), *Universals in Linguistic Theory*, 1–88. New York: Holt, Rhinehart and Winston.

Ginzburg, Jonathan, Ivan A. Sag (2000). *Interrogative Investigations: the Form, the Meaning, and Use of English Interrogatives*. Stanford: CSLI Publications.

Goldberg, Adele (1995). *A Construction Grammar Approach to Argument Structure*. Chicago: Chicago UP.

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

Goldberg, Adele, Ray Jackendoff (2004). The english resultative as a family of constructions. *Language*, 80(3), 532–568.

Haspelmath, Martin (2007). Pre-established categories don't exist. *Linguistic Typology*, 11(1), 119–132.

Iwata, Seizi (2008). *Locative Alternation: A Lexical-Constructional Approach*, *Constructional Approaches to Language*, vol. 6. Amsterdam: John Benjamins.

Kay, Paul (2005). Argument structure constructions and the argument-adjunt distinction. In Miriam Fried, Hans Boas (Eds.), *Grammatical Constructions: Back to the Roots*, 71–98. Amsterdam: John Benjamins.

Kay, Paul, Charles J. Fillmore (1999). Grammatical constructions and linguistic generalizations: The what's x doing y? construction. *Language*, 75, 1–33.

Lakoff, George (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: The University of Chicago Press.

Langacker, Ronald (1987). *Foundations of Cognitive Grammar. Volume 1*. Stanford: Stanford University Press.

Langacker, Ronald (2000). A dynamic usage-based model. In Michael Barlow, Suzanne Kemmer (Eds.), *Usage-Based Models of Language*, 1–63. Chicago: Chicago University Press.

Levin, Beth, Malka Rappaport Hovav (2005). *Argument Realization*. Research Surveys in Linguistics. Cambridge: Cambridge University Press.

Micelli, Vanessa (2012). Field topology and information structure - a case study for German constituent order. In Luc Steels (Ed.), *Computational Issues in Fluid Construction Grammar*. Berlin: Springer.

Michaelis, Laura A. (2009). Sign-based construction grammar. In B. Heine, H. Narrog (Eds.), *The Oxford Handbook of Linguistic Analysis*, 155–176. Oxford: Oxford University Press.

Müller, Stefan (1996). The babel-system – an HPSG prolog implementation. In *Proceedings of the Fourth International Conference on the Practical Application of Prolog*, 263–277. London.

Müller, Stefan (2006). Phrasal or lexical constructions? *Language*, 82(4), 850–883.

Nemoto, Noriko (1998). On the polysemy of ditransitive *save*: The role of frame semantics in construction grammar. *English Linguistics*, 15, 219–242.

Palmer, Frank (1994). *Grammatical Roles and Relations*. Cambridge: Cambridge UP.

Pinker, Steven (1989). *Learnability and Cognition: The Acquisition of Argument Structure*. Cambridge: Cambridge UP.

Steels, Luc (2011a). A design pattern for phrasal constructions. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc (2011b). A first encounter with Fluid Construction Grammar. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc (2011c). Introducing Fluid Construction Grammar. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Steels, Luc, Joachim De Beule (2006). Unify and merge in Fluid Construction Grammar. In P. Vogt, Y. Sugita, E. Tuci, C. Nehaniv (Eds.), *Symbol Grounding and Beyond.*, LNAI 4211, 197–223. Berlin: Springer.

Steels, Luc, Joachim De Beule, Nicolas Neubauer (2005). Linking in Fluid Construction Grammar. In *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC '05)*, 11–18. Brussels, Belgium.

Steels, Luc, Remi van Trijp (2011). How to make Construction Grammars fluid and robust. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

van Trijp, Remi (2011). Feature matrices and agreement: A case study for German case. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Van Valin, Robert (2004). Semantic macroroles in Role and Reference grammar. In Rolf Kailuweit, Martin Hummel (Eds.), *Semantische Rollen*, 62–82. Tübingen: Narr.

Wellens, Pieter (2011). Organizing constructions in networks. In Luc Steels (Ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.

Wellens, Pieter, Joachim De Beule (2010). Priming through constructional dependencies: a case study in Fluid Construction Grammar. In A. Smith, M. Schouwstra, B. de Boer, K. Smith (Eds.), *The Evolution of Language (EVOLANG8)*, 344–351. Singapore: World Scientific.

Whorf, Benjamin (1973). *Language, Thought and Reality. Selected Writings of Benjamin Lee Whorf.* Cambridge, MA: MIT Press. Orig. published 1956.