How to Make Construction Grammars Fluid?

Luc Steels and Remi van Trijp

[draft - to appear in: Steels, L. (ed.) (2011) *Design Patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam.]

Abstract

Natural languages are fluid. New conventions may arise and there is never absolute consensus in a population. How can human language users nevertheless have such a high rate of communicative success? And how do they deal with the incomplete sentences, false starts, errors and noise that is common in normal discourse? Fluidity, ungrammaticality and error are key problems for formal descriptions of language and for computational implementations of language processing because these seem to be necessarily rigid and mechanical. This chapter discusses how these issues are approached within the framework of Fluid Construction Grammar. Fluidity is not achieved by a single mechanism but through a combination of intelligent grammar design and flexible processing principles.

1. INTRODUCTION

Human languages are inferential communication systems (Sperber & Wilson, 1986) as opposed to being coding systems, which assume that there is no intelligence in the receiver of the message. All the information to be transmitted is coded explicitly and the main issue, addressed in Shannon's information theory, is to code and decode information as efficiently and reliably as possible. Programming languages, electronic communication protocols, as used on the Internet, or formal calculi, like first order predicate logic, are examples of coding systems. In contrast, an inferential communication system assumes an intelligent receiver who is able to fill in information based on background knowledge, common sense and the shared local context. The message therefore only needs to provide enough information to actively be able to *reconstruct* the content.

The fact that human languages are inferential communication systems gives them a number of special properties. The first one is that languages can be *open-ended*. At any moment the set of available conceptualizations and linguistic conventions can be expanded by speakers if they need to express something that was not yet conventionally expressible in the language, because hearers are assumed to be intelligent enough to figure out what was meant and possibly adopt any innovations introduced by speakers. This *fluid* character of human language helps to make them adaptive to the needs of language users that keep changing as human societies evolve and become more complex. Existing conventions also tend to become so well entrenched that hearers no longer pay enough attention, which then stimulates speakers to invent novel ways of expressing the same meaning and thus increase their chance of communicative success.

The second property following from the inferential nature of human languages is that they can use the same linguistic materials in *multiple ways* (as for example illustrated by the common ambiguity and synonymy of word meanings) because an intelligent listener will nevertheless be able to figure out which meaning is intended. Multifunctionality, however, is a big headache in language processing because multiple hypotheses need to be explored both in parsing and production. The danger of a combinatorial explosion is never far off and is partly avoided by exploitation of all the relevant linguistic information and by using the context and other knowledge to constrain the meaning as fast as possible.

Third, human speakers and listeners need not be perfect in their usage of a language. The human brain cannot be expected to function without error when producing or understanding utterances at a very fast rate. Speaking involves a highly complex planning process, with speakers often starting in one direction, halting after an incomplete fragment has been produced, correcting themselves, and then possibly continuing in another direction. Speaking also involves very complex and fast articulatory movements that may easily go wrong and it requires rapid access to linguistic memory that is itself not always reliable. Listening requires first of all a reconstruction of the speech sounds, which are often hardly detectable in the input signal because they are only sloppily pronounced and influenced by noise in the environment. Listening also requires fast retrieval from lexical and grammatical memory to fetch the relevant constructions, handle combinatorial search, track multiple linguistic structures at different levels in parallel and perform the right inferences to reconstruct the meaning of the utterance. Particularly for fragmented speech and imprecise formulation, it is very hard, if not impossible, for hearers to exactly reconstruct the meaning intended by the speaker. Nevertheless, human

language communication appears to be surprisingly *robust* against this kind of fragmented and errorful input.

Finally, the conventions underlying human natural languages need not be entirely fixed and uniformly shared in a population. Even within the same dialect or social register (Labov, 1994), there is a lot of *variation*, which is unavoidable due to the open-ended nature of human languages. All language users have equal rights to expand their language, but there is no guarantee that they do this always in exactly the same way. It therefore takes time before a newly emerging convention spreads and stabilizes in the population as a whole. Moreover, different language users have a different history of interactions and different expressive needs, and they individually have to reconstruct and learn the language of the community and must keep track of changes made by others, all without a global authority to consult about the 'right way' to say something.

These various properties and the issues they raise have puzzled students of language for a long time, leading to an extensive literature proposing ways to deal with fluidity and robustness. Part of this literature comes from the field of discourse studies, which engages in observations of actual human dialog. The data shows not only the ungrammaticality and fragmented nature of almost all human spoken sentences but also that new conventions arise, conventions shift, and they are negotiated as part of the dialog (Garrod & Anderson, 1987). Historical linguistics is another source of relevant literature. This field has considered the question as to how new lexical and grammatical forms originate and how they may become adapted by a population (see e.g. Hopper, 1991; Heine, 1997). Finally, the issue has also been considered extensively in computational linguistics because deployment of real world language processing applications very quickly gets confronted with errorful and fragmented input and with communicative situations where speakers stretch conventions or invent new ones on the spot (Fouvry, 2003).

One of the explicit goals of Fluid Construction Grammar is to try and deal with fluidity and robustness. We do not propose that fluidity is achieved through a single mechanism. It is rather a general property like the safety of a car: Almost all components of a car contribute to its safety, but safety depends not only on the car itself but also on the behavior of the driver, other cars on the road, and the conditions of the environment. Similarly, fluidity concerns not only the grammars and the flexibility and versatility of linguistic processing. It depends also on how language processing is embedded within other cognitive processes and on the cooperative interactions of speaker and listener.

The remainder of this chapter discusses first how the architecture of a language system can be designed to help achieve fluidity, then how individual variation can be dealt with, and, finally, how application construction in FCG has been made more flexible. The final section shows various examples taken from implementation experiments demonstrating concretely how FCG deals with issues related to the fluidity and robustness of language.

2. SYSTEM ARCHITECTURE

For a normal language user, the lexical and grammatical parsing and production of language sentences is not a goal in itself but a subtask of a more encompassing cognitive system which takes care of many other tasks, including the build-up and maintenance of world models based on perception and action, the tracking and management of cooperative activities from which the communicative goals of a particular speech act get derived, the derivation of common sense facts that are assumed to be shared background and the articulation and recognition of speech and gesture. Each of these subtasks produces constraints and partial evidence that can be used to achieve a successful communicative interaction, but all of them are confronted with similar issues as discussed in the previous paragraphs: open-endedness, errorful and incomplete input, resource constraints, individual variation and fluid conventions. Consequently, failures in handling one subtask have to be compensated for by other subsystems so that the total is more robust than each component separately. Orchestrating a tight interaction between all subsystems involved in language is therefore a first big step towards making a language system capable of handling fluidity and robustness.

2.1. The Semiotic Cycle

Producing or comprehending sentences requires speakers and hearers to go through the *semiotic cycle* shown in Figure 1. The relevant processes take place against the background of turn-taking and attention sharing behaviors and scripts monitoring and achieving the dialog.

The processes relevant for the speaker are:

1. *Grounding:* The first set of processes carried out by both the speaker and the hearer must maintain a connection between the internal factual memory and the states and actions in the world that dialog partners want to talk about. They include segmentation, feature extraction, object recognition, event classification, object tracking, object manipulation, etc.

2. *Conceptualization:* The second set of processes must select what needs to be said and then conceptualize the world in a way that it can be translated into natural language expressions which satisfy the communicative goal that the speaker wants to achieve (Talmy, 2000). For example, if we say "the car is in front of the tree", we have conceptualized the tree as having a front which is directed towards us, and the car as being in between ourselves and this front.

3. *Production* (also known as verbalization or formulation (Levelt, 1989): This set of processes takes a semantic structure and turns it through a series of mappings into a surface form, taking into account the lexical, grammatical, morphological and phonological conventions of the language as captured by various constructions.

4. *Speech Articulation* This set of processes renders a sentence into the fast movements of the articulatory system required to produce actual speech and gestures.

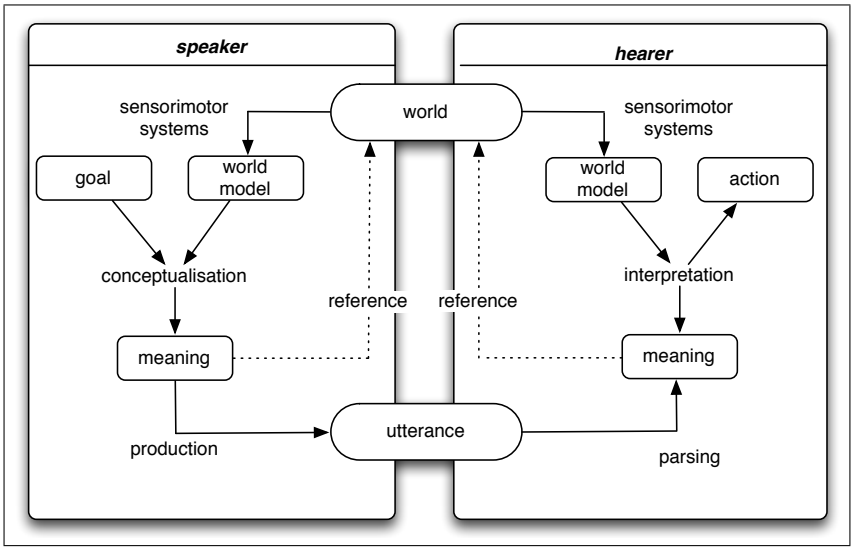

Figure 1. *The semiotic cycle summarizes the main processes that the speaker (left) and the hearer (right) go through. Sentence parsing and production is only one of the activities within this cycle.*

The processes relevant for the hearer are:

1. *Speech Recognition* The speech signal needs to be processed through a battery of signal processing and pattern recognition processes to get a reasonable set of hypotheses about the speech elements that might be present.

2. *Parsing.* The hearer uses these data to reconstruct as well as possible the meaning of the utterance that is transmitted by the speaker. Again, this process is highly elaborate due to the complexity of natural language and the presence of ambiguities.

3. *Interpretation.* The hearer must then confront the meaning resulting from the parsing process with his or her own factual memory of the world and understanding of the dialog context in order to find a correct interpretation of the utterance that fits with his or her own expectations and observations. For example, the hearer must retrieve the object in the scene that the speaker wanted him or her to pay attention to.

4. *Grounding.* The hearer must therefore also maintain a connection through perception and action between his or her internal factual memory and the states of the world, possibly including the mental states of the speaker.

At present we can construct computational models of all aspects of this cycle, including of the parsing and production processes. More specifically, the FCG system discussed in other chapters of this book can act as an embedded system for this subtask operating within a complete integrated and grounded language system that contains implementations for all the other processes as well. In fact, FCG was originally developed for this purpose in order to support experiments in which artificial agents, instantiated as physical robots, achieve embodied communication (Steels, 2003a).

How can these processes be tightly integrated? The FCG-interpreter has been designed based on the notion of a *task thread*, further called *task*, which governs the behavior of a set of *processes* (Steels & Loetzsch, 2009). Each process performs a step in the semiotic cycle and returns a certain process result. The process may be the analysis of a scene in order to come up with possible segmentations of objects and actions, the conceptualization of a scene in order to come up with a conceptualization that could be used to achieve a particular communicative goal, the application of a set of constructions (for example all morphological constructions to expand a particular transient structure), the rendering of part of the utterance into speech articulation and so on.

There are three possible outcomes of a process. (i) The result could indicate that the task can carry on with the next step in the sequence of processes associated with this task. For example, the morphological construction set came up with an appropriate transient structure which can then be further processed using another construction set (for example all categorization constructions). (ii) The result could indicate that there are several ways to continue the comprehension or production process, which means that the task has to split into several subtasks which each will explore another hypothesis. In this case, the process result will typically return some measure indicating how plausible each possible hypothesis is considered to be and, as explained in the chapter on search (Loetzsch et al., 2011), which can be used as one of the criteria for deciding which nodes in the search process will be preferentially explored. (iii) The result could indicate that a dead end was reached and that this particular task could not carry out the process given all available evidence.

This task-based architecture has the advantage that multiple hypotheses coming from different levels of processing can be explored in parallel and all information that might help to arrive at a coherent hypothesis can be integrated.

2.2. Re-entrance

The next step towards handling fluidity and robustness is to design all the processes required for language comprehension and production in such a way that they are reversible, i.e. that they can be used by processes running in both directions (from form to meaning or from meaning to form). This is certainly the case for Fluid Construction Grammar, because constructions are always bi-directional: they can be used unaltered both in parsing and production. Conceptualization and interpretation processes can be designed in a similar way, so that the process of conceptualization (planning what to say) is based on exactly the same representations as the process of interpretation. Speech systems can be designed so that they use analysis by synthesis (the so-called motor theory of speech, Liberman & Mattingly, 1985), which means that listeners segment and recognize speech units in terms of the articulatory gestures needed to produce them. Further discussion of these other subsystems is however beyond the scope of the present chapter.

If the reversibility criterion is satisfied, then a particular structure being derived by a linguistic process moving in a particular direction can be re-entered in its corresponding mirror process to see what the consequences are when processing this structure in the other direction (Steels, 2003b). For example, the process that is recognizing speech sounds and words can take an initial set of hypotheses and then

run them backwards, articulating these same words in order to predict what they should sound like and map these predictions on the incoming sound stream. Alternatively, a process performing grammatical analysis by applying constructions can take the resulting structures and then run them backwards, in order to predict what the rest of the sentence might look like or to fill in missing gaps. There is abundant evidence from the psychological and neuroscience literature that this kind of re-entrance and monitoring occurs in human subjects, not only for speech but also for other subsystems of language processing (Brown & Hagoort, 1991).

Re-entrance is possible at any stage or level. Here are some more examples for the speaker:

1. The utterance derived from the total set of processes achieving language production can be re-entered by invoking the processes achieving language comprehension in order to check whether the reconstructed meaning is equal (or sufficiently compatible) to the one that the speaker originally intended. The speaker here takes him- or herself as a model of the listener and can thus self-monitor the possible effect of his or her utterance on the listener.

2. After applying all lexical constructions to perform a first selection of which words are going to be used to build the final utterance, the speaker could already re-enter this set of words and simulate the parsing process of a listener, thus figuring out what meaning would already be derivable without any further grammar.

3. After having planned what to say by running a set of conceptualization processes, the speaker could re-enter the obtained meaning through his or her own interpretation processes in order to find out whether interpretation of this meaning would indeed achieve the communicative goal that he or she wanted to achieve within the given context him- or herself.

Re-entrance is not only useful for the speaker but is equally relevant to the listener:

1. The listener can take the meaning derived after all comprehension processes have run and then launch a language production system with this meaning, thus reconstructing how he or she would have expressed the same information him- or herself.

2. The listener can pipe transient structures, derived from applying a particular subset of constructions in parsing, back through the same construction set in

the reverse direction and thus compute how his or her own set of constructions would work out the transient structures derived from the speaker's input.

It is obvious why re-entrance is useful to achieve fluidity and robustness: It allows language users to employ their own inventories to fill in gaps, fix errors, complete partial fragments, etc. For example, if a word or a particular form constraint is missing, the listener can use re-entrance in order to reconstruct what this might have been. If the speaker re-enters the utterance he or she is about to produce, he or she can notice that certain words or phrases are ambiguous or trigger combinatorial explosions and he or she can then give preference to another path in the search space. If the listener is confronted with a phrase he or she cannot process syntactically or semantically, he or she might nevertheless be able to reconstruct the meaning from the context and other parts of the sentence and then predict which constructions are missing in his or her own inventory. All these activities are not only possible but routinely incorporated into computational experiments with Fluid Construction Grammar. They are implemented by allowing that a task thread evokes re-entrance processing at critical steps in the process.

2.3. Monitoring with diagnostics

The next step for handling fluidity and robustness is to introduce a facility that allows for monitoring the outcome of a step in a particular process, by triggering *diagnostics* operating over the outcome of a particular processing step. Diagnostics therefore introduce a meta-level running in parallel with routine processing. (See Figure 2.)

Here are some concrete examples of such diagnostics:

1. The speaker can have a diagnostic that tests at the end of all production processes whether all of the meaning he or she wanted to cover is indeed covered. If this is not the case, it suggests that his language inventory was missing certain constructions to express these meanings.

2. The speaker can re-enter the utterance to simulate how the listener would parse the utterance. He or she can then run a diagnostic to test whether a significant search space had to be explored, which signals that it might require too much cognitive effort to comprehend.

3. Another diagnostic could compare the meaning derived from these comprehension processes with the meaning that the speaker originally wanted to express.

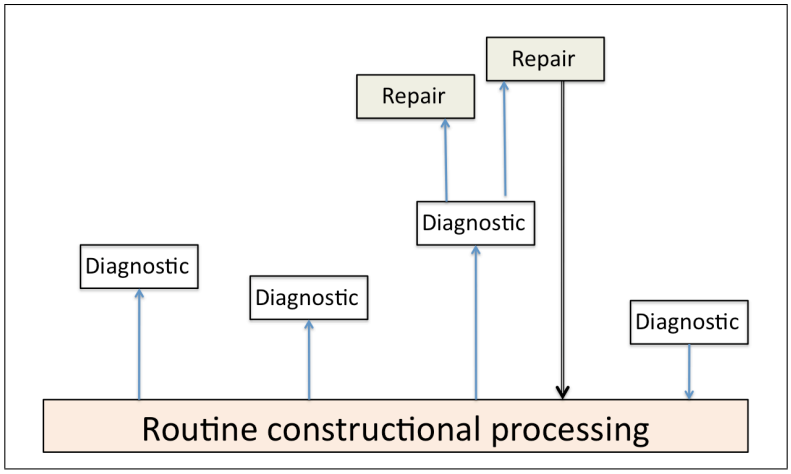

Figure 2. *The routine application of constructions during parsing and production is augmented with meta-level processes performing diagnosis and possibly repairing problems by extending the inventory of the speaker or the hearer.*

4. The hearer can have a diagnostic that tests at the end of all comprehension processes whether all the form constraints observed in the utterance (all words, intonation patterns, orderings, morphological markers, etc.) were effectively used in one way or another to construct the meaning. If that is not the case, this suggests that certain constructions are missing and that the inventory needs to be expanded or more constraints have to be added to existing constructions.

5. The hearer can have a diagnostic focusing on whether the expected agreement relations between constituents of a phrase have all been satisfied. If this is not the case, this suggests that the speaker has made an error which should be overruled, or, that the hearer should revise some of the syntactic features he or she assumed.

Hundreds of such diagnostics can be formulated. Some of them can be very general whereas others could be fine-grained, focusing on whether particular templates are correctly applied.

2.4. Repairs

A problem signaled by a diagnostic is not necessarily the end of that task thread. If other tasks run in trouble as well, it may become useful to trigger *repair strategies* that can potentially deal with the problem signaled by the diagnostic. Repairs might take various forms and depend on whether comprehension or production is the major target:

1. It is possible that problems arising in comprehension are due to errorful or fragmented input. In that case, the repair could consist in ignoring the missing elements or possibly filling them in and to continue processing as best as possible. It should not trigger any revision of the listener's inventory.

2. It is possible that problems are due to language variation. The language systems of speaker and listener are never exactly the same but the listener may still be able to process the sentence because he has also stored alternatives (see next section). In this case, renewed exploration of the search space with paths that had a lower score might still produce valid parses.

3. It is possible that problems are due to the creativity of the speaker who has stretched the usage of certain constructions in order to express novel meanings or express meanings in novel ways. For example, the speaker could have coerced an intransitive verb like "sneeze" to become a ditransitive to express cause-motion, as in the famous example sentence: "Adele sneezed the napkin off the table".

4. Finally, it is possible that the listener is missing certain constructions to adequately parse or interpret the input sentence, which would be a quite common occurrence in the early stages of language acquisition but is still required by mature language users who need to keep up with changes in their language. In this case, repairs will be much more complex because they have to expand the listener's existing inventory. For example, the listener encounters an unknown word, is able to reconstruct its possible meaning by using interpretation based on partial meaning, context and possibly additional feedback from the speaker, and can then activate a repair action by introducing a new lexical construction that associates the word form with the uncovered meaning. Once the lexicon has been extended this way, processing can continue or restart, producing a complete parse.

Many different repair strategies are useful in language production:

1. It is possible that not every aspect of the meaning coming out of conceptualization could be covered or that not all semantic and syntactic categorizations could be expressed explicitly in the surface form. In that case, repair might consist in ignoring these uncovered items and simply rendering the utterance based on the form constraints that have already been introduced in the hope that the listener is flexible and intelligent enough to reconstruct the intended meaning.

2. However, the speaker can go a step further: when meanings or categorizations cannot yet be expressed, the speaker might attempt to expand his or her existing inventory with new constructions based on recruiting existing linguistic materials in novel ways. Complex cognitive operations such as analogy, blending, or metaphor are all routinely employed by speakers to achieve this, and the novel constructions that result need to be reconstructed by hearers.

3. A lot of diagnostics operate on the outcome of re-entrance. Even though the speaker is able to express certain meanings, it could be that the chosen forms trigger combinatorial explosions, ambiguity or other cognitive difficulties for the listener. In that case, the speaker might try to repair the communication by exploring other ways to express the same meanings or by introducing novel uses of constructions or even entirely new constructions to take care of them.

Diagnostics and repairs are integrated by the FCG-interpreter within the context of task threads. A particular task will run diagnostics at critical points, for example after the application of a particular set of constructions. If problems pop up, different repair strategies might be considered and one could be run possibly allowing the relevant task to proceed further.

3. HANDLING INDIVIDUAL VARIATION

It is a well established fact that there is enormous variation in the set of constructions that members of the same language community employ, posing great challenges both to language description and to language processing. The observed variation is caused by two factors:

(i) Language reflects the social strata, age, and regional origins of speakers. Although this is most obvious for speech sounds, we also find, for example, that syntactic features, like gender in French, word order constraints, like the ordering of the auxiliaries and non-finite verbs in the main clause in Dutch, the use and expression of syntactic cases, such as the competing “laísmo-leísmo-loísmo” paradigms

in Spanish, all differ across age, style, social status and regions, even between different neigborhoods of the same city (Fernández-Ordóñez, 1999). Interestingly, language speakers are often able to recognize these differences and thus identify the social stratum and regional origins of a speaker. Sometimes they are even able to imitate them, suggesting that these social markers are part of the knowledge that speakers have of their language and that constructions must incorporate parameters for social and dialectal variation as part of their definition. This knowledge can be represented within the FCG framework in a straightforward way, for example by labeling all constructions as belonging to a particular dialectal variant.

(ii) The second factor leading to significant variation comes from the fact that speakers have to reconstruct their language systems individually based on the input available to them and driven by the communicative challenges and semantic domains that they routinely encounter. Moreover, language users all have the possibility to invent new constructions, and it is improbable that they all hit upon exactly the same forms to express similar meanings. Often the resulting variation balances out, as standard modes of expression emerge for novel meanings, but there are intermediate states in which different conventions compete with each other until one emerges as the winner. The remainder of this section focuses on how this kind of variation is incorporated within the framework of Fluid Construction Grammar.

First of all, it is assumed that each language user stores competing constructions. A construction C_1 competes with another construction C_2 if both are triggering in the same circumstances, in other words if both could potentially expand a given transient structure, but C_1 does it in another way than C_2. One example of competing constructions are synonyms. They trigger on the same meaning but use another word-stem to express that meaning. Other examples are grammatical variants, such as the use of two different morphemes to express a particular tense or the use of two different cases (such as dative or genitive) to express the same semantic role (beneficiary). Each construction has a score which is modeled as a numerical value between 0.0 and 1.0. The score plays a role in the search process. When the speaker has to choose between branches based on competing constructions, the branch created by the construction with the highest score is explored first. Similarly, the hearer prefers to explore a branch based on a construction with a higher score against competing branches. This ensures that the most preferred construction (at least according to the state of knowledge of the speaker or the hearer) will effectively play a role if it leads to a successful conclusion. Only when the choice for this construction does not work out and the search process has to backtrack, the construction with a lower score is tried.

When a new construction is built by a repair strategy, it starts with a default score δ_{init}. This score is adjusted after every interaction based on the following rule (first introduced in Steels, 1996):

1. When an interaction is successful, the score of all constructions used is increased by a factor $\delta_{success}$.
2. The score of competing constructions is decreased with a factor $\delta_{inhibit}$, leading effectively to a form of lateral inhibition.
3. When an interaction is not successful, then the score of all constructions that were used is decreased by a factor δ_{fail}.

Computer simulations (Steels & Kaplan, 2002) and mathematical proofs (De Vylder & Tuyls, 2006) have shown that this dynamics not only allows agents to be flexible in dealing with constructions that they do not prefer themselves, it also allows a population of agents to reach coherence for their language use, even if no solutions initially existed and all agents are allowed to invent new constructions from scratch. (See Figure 3.) This is due to the positive feedback loop between communicative success and the use of constructions: When a construction has been part of a successful language interaction, the dialog partners increase the score and consequently their use of the same construction in similar circumstances increases. This in turn influences other agents either to acquire this construction if they have not encountered it before or to increase its chances of later use if they already had the construction as part of their inventory.

4. FLEXIBILITY AND COERCION

Tight integration of all processes involved in the semiotic cycle, re-entrance, meta-level diagnostics and repairs, as well as scoring and lateral inhibition dynamics all help to make language processing more robust and flexible. Yet, there is still more that can be done, specifically by the non-standard application of constructions through various repair strategies. By non-standard application we mean that

1. Matching can be made more flexible by allowing that some units or some syntactic and semantic constraints required by a construction are absent or incomplete in the transient structure.
2. Merging can be made more powerful by applying it to transient structures which have not passed the matching phase, thus coercing a transient structure to have certain properties which are not initially present.

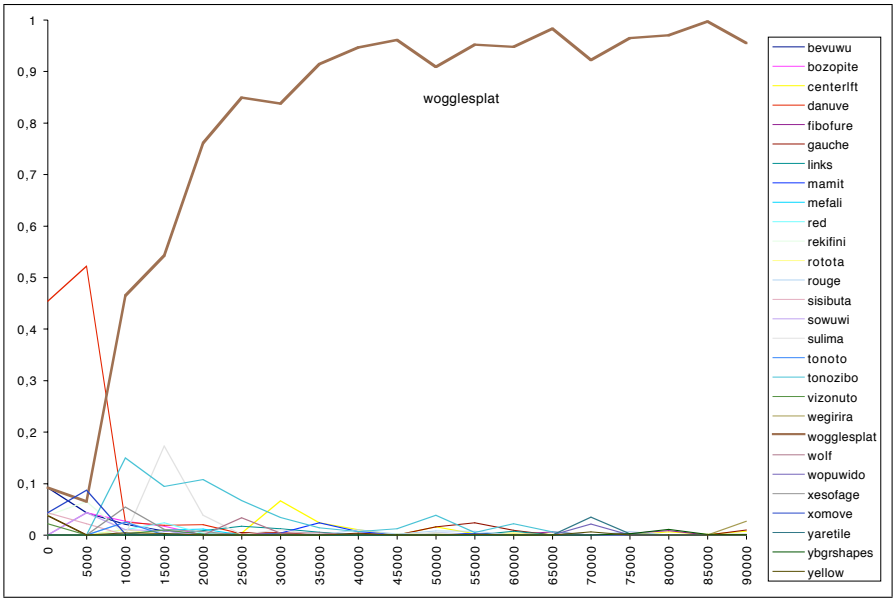

Figure 3. *This graph shows the results of experiments with artificial agents examining the impact of lateral inhibition dynamics on construction alignment. The x-axis shows the number of interactions by a population of 500 agents over time. The y-axis shows the frequency with which competing words are used. After about 5000 games there is a phase transition making one word ("wogglesplat") dominant.*

These operations are powerful mechanisms for dealing with missing elements in the input (for example, a word form not recognized by the speech recognition subsystem), with unknown elements (for example a word is encountered that is not yet part of the lexical inventory of the speaker), with elements used in ways that violate the hearer's language system or with novel uses of existing constructions to express meanings that could otherwise not be expressed. The remainder of this section gives some concrete examples taken from working experiments.

4.1. Processing Unknown Words and Meanings

Once language users have built up a repertoire of form-function conventions of their language, they can exploit this knowledge to make top-down predictions when they encounter unknown words or when they need to express novel mean-

ings. Through a learning process that is called *bootstrapping*, they can exploit the results of these predictions for learning new form-meaning mappings very rapidly. This section demonstrates how an unknown word can be processed and how this process forms the basis for *syntactic* bootstrapping, and briefly discusses how the same approach can be used for processing new meanings and achieving *semantic* bootstrapping.

The starting point of the example is a grammar for lexical and phrasal constructions that is described by Steels (2011a), which is capable of parsing and producing phrasal utterances such as *the mouse* or *the very green mouse*. Next, the language user is confronted with the word *Jabberwock* (from Lewis Caroll's poem *Jaberwocky*) while parsing the phrasal utterance *the very green Jabberwock*. The processing example is initialized with the following configurations:

- Construction-set: Lexical and phrasal constructions are collected in one set. There is no predefined order of application.
- Goal-tests: There are two goal-tests that determine whether processing is finished and successful: `no-applicable-cxns` and `no-form-in-top-unit`. The first one simply implies that if no constructions can be applied, processing ends. The second goal-test checks whether all the form elements of the observed utterance (i.e. every word form and the observed word order) have been handled during processing. This check considers a form element to be processed as soon as it is removed from the top-unit and relocated in any other unit in the transient structure, so the goal-test simply needs to verify whether there is any form element left in the top-unit of the transient structure.

Figure 4 shows the search tree during routine parsing of the utterance *the very green Jabberwock* and the expansion of the final node in that search tree. As can be seen in the Figure, routine processing is capable of applying the lexical constructions for *the*, *very* and *green*, of categorizing them as determiners, adverbs or adjectives, and of grouping *very green* in an adverbial-adjectival phrase. However, the search gets stuck after the application of the *determiner-cxn* because the goal-test `no-form-in-top-unit` fails. As shown in the expanded node, this goal-test reports that the string *Jabberwock* and some word order constraints are left unanalyzed in the top-unit.

The goal-test `no-form-in-top-unit` can be coupled to a diagnostic that detects forms that remain unanalyzed during parsing. In this instance, the diagnostic reports to the meta-layer that the unanalyzed form contains a string and two word

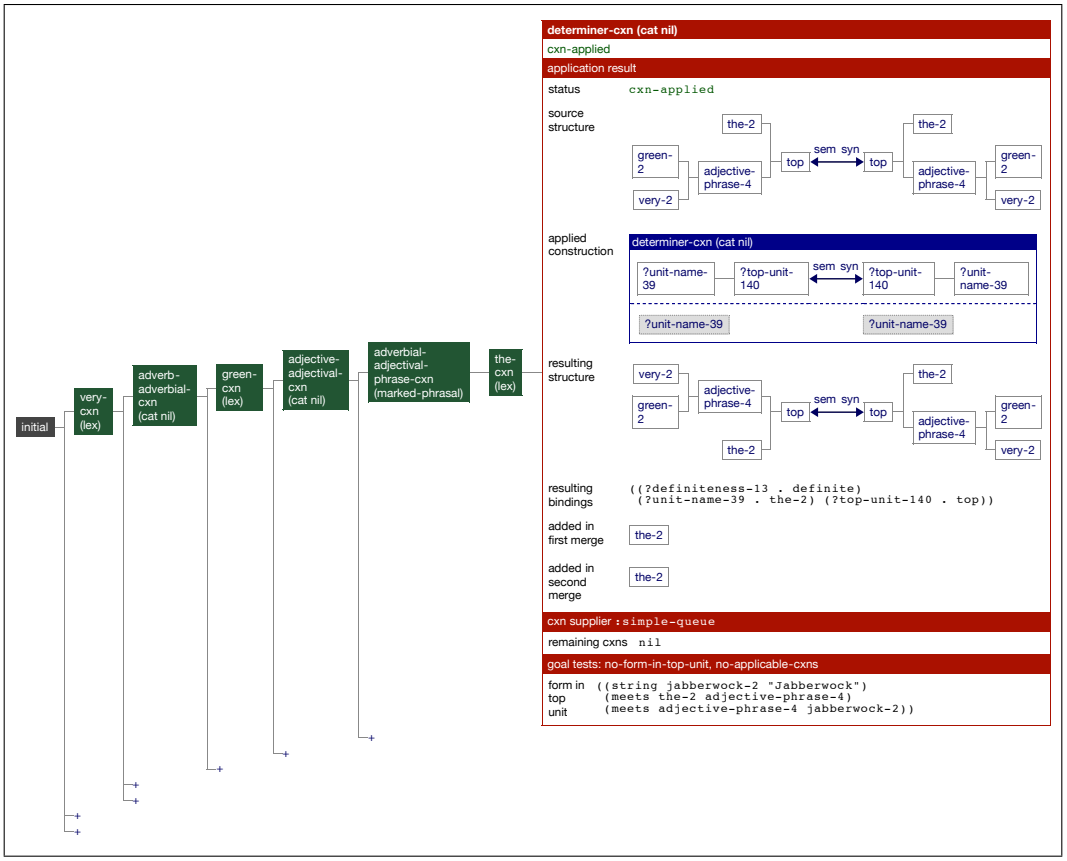

Figure 4. *Routine processing of* the very green Jabberwock *reaches a dead end because the goal-test* `no-form-in-top-unit` *fails. Alternative branches in search are not shown due to space limitations.*

order constraints. Since there is an uncovered string, one possible repair is to treat that string as if it were a lexical item.

The repair proposed here works in a two-legged operation: first, it introduces a new lexical construction for kick-starting the parsing process, and next, it consolidates the results obtained from processing. The new lexical construction is created using a 'generic' lexical construction, which is a lexical construction that remains underspecified for meaning, form or categories through the use of variables. The

generic lexical construction is defined using the same templates as for lexical constructions (see Steels, 2011a):

```
(def-lex-cxn generic-lexical-cxn

 (def-lex-skeleton generic-lexical-cxn
  :meaning (?unknown-meaning ?set ?context)
  :args (?set ?context)
  :string ?unknown-string)

 (def-lex-cat generic-lexical-cxn
  :sem-cat ?sem-cat
  :syn-cat ?syn-cat))
```

The generic lexical construction is never considered during routine processing because it is so general that it would cause a combinatorial explosion in search, as it triggers on any string in parsing or any meaning in production. Moreover, the use of the variables `?sem-cat` and `?syn-cat` allows the generic construction to interact with any other construction that requires a `sem-cat` or `syn-cat` feature, whatever their values might be.

In principle, the repair can just insert the generic lexical construction in the construction-set and continue processing. However, since the current objective also involves word learning through syntactic bootstrapping, it is better to create a new lexical construction that forms the first basis for acquiring the new word. The new lexical construction is created by copying the generic lexical construction, which is achieved through the `:inherits-from` keyword in the `def-lex-cxn` template (see REF JORIS). The construction's underspecified string is then replaced by `"Jabberwock"` through the `def-lex-require` template, which works analogously to the `def-phrasal-require` template as defined by Steels (2011a):

```
(def-lex-cxn jabberwock-cxn
 :inherits-from generic-lexical-cxn

 (def-lex-require jabberwock-cxn
  :cxn-string "Jabberwock"))
```

Figure 5 shows what happens when the new construction is inserted in the construction-set: the `jabberwock-cxn` can immediately be applied, which subsequently triggers the application of the `adjectival-nominal-cxn` and the

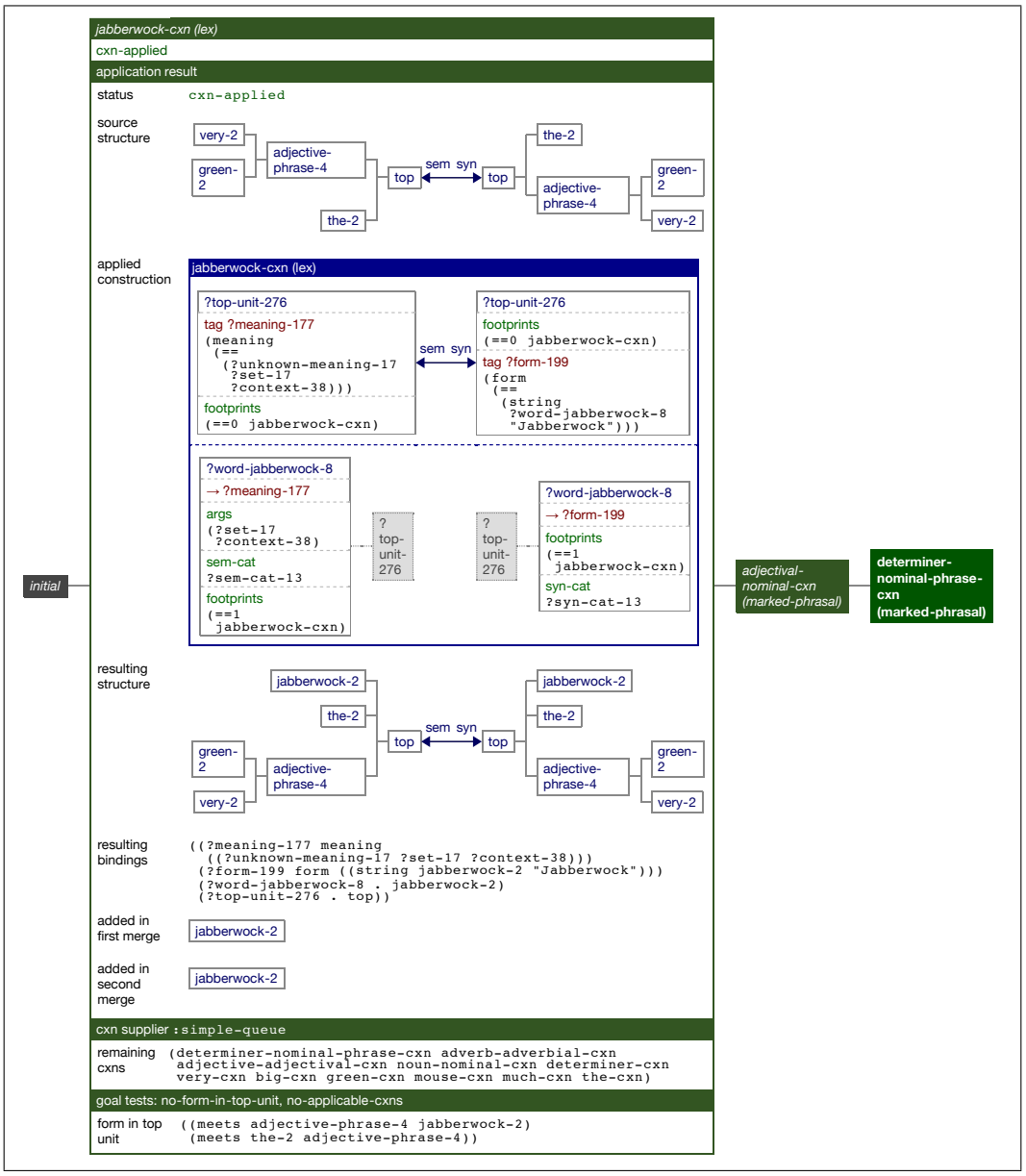

Figure 5. *After the meta-layer has introduced a new lexical construction in the construction-set, processing continues from where the routine layer failed. The application of the new construction subsequently triggers the application of other constructions.*

determiner-nominal-phrase-cxn. The expanded node shows the application of jabberwock-cxn. As can be seen, the transient structure before application is the same one as where routine processing was blocked before. Now, however, the new lexical construction creates a new unit for the string "Jabberwock" and fills this unit with underspecified feature-value pairs.

How can the application of the new construction trigger the application of other constructions? The first one in line is the adjectival-nominal-cxn, which requires an adjectival and a nominal unit. The first requirement is satisfied by *very green*, which was already identified as an adjectival phrase during routine processing. As for the nominal unit, the construction expects the following syn-cat:

```
(syn-cat
 (==1 (number ?number)
      (syn-function nominal)))
```

Since the syn-cat feature of the Jabberwock-unit is underspecified through a variable, the adjectival-nominal-cxn can simply unify and therefore impose its own syn-cat onto the Jabberwock-unit. On the semantic pole, the construction can then impose its semantic constraints as well, including the unit's sem-function (identifier) and how its meaning should be linked to the meaning of the adjectival phrase. The adjectival-nominal-cxn in turn provides all the requirements for triggering the determiner-nominal-phrase-cxn, which groups the whole utterance into a nominal phrase, taking care of agreement and linking of meanings. At this point, no other constructions apply and all the goal-tests are satisfied, so parsing was successful despite encountering an unknown word.

Diagnostics and repairs are not only useful for achieving robustness in processing, they also form the basis for learning novel form-function mappings. The second part of the repair is to consolidate the outcome of the repaired parsing process through syntactic bootstrapping. The idea behind syntactic bootstrapping is that "if the syntactic structures [of a language] are truly correlated with the meanings, the range of structures will be informative for deducing which words goes with which concept" (Gleitman, 1990, p. 30). Indeed, more information about Jabberwock can be inferred by looking at the final transient structure and retrieve all feature-value pairs that were added to the Jabberwock-unit by other constructions. In the current example, the following sem- and syn-cat feature values can be retrieved for the word Jabberwock and stored in the newly created jabberwock-cxn:

- (sem-cat ((is-countable +)
  (sem-function identifier)))

- `(syn-cat ((number ?number)`
  `(syn-function nominal)))`

The same approach can also be used for achieving *semantic bootstrapping*, i.e. predicting a word's syntactic categorization and actualization based on its meaning (Grimshaw, 1981). If a diagnostic reports an unexpressed meaning during routine processing, a repair strategy may create a new lexical construction using the generic lexical construction, but already filling in the meaning that needs to be expressed. Just like in the above example, processing continues from there and other constructions provide additional grammatical information of how such a meaning is typically expressed in the language.

4.2. Coercion

A lot of language creativity does not involve any new words or constructions, but rather the *coercion* of existing linguistic items in novel usage patterns. This section illustrates how FCG handles coercion through the sentence *Adele sneezed the napkin off the table*. The problem of this utterance is well-known: *sneeze* is the canonical example of an intransitive verb, but here it is realized as a caused-motion event, meaning 'Adele CAUSED the napkin to MOVE off the table by sneezing' (Goldberg, 1995). Goldberg (2006, p. 22) suggests that processes of accommodation or coercion allow constructions to "be construed as not being in conflict" so they can be combined with each other. In other words, coercion can be considered as a solution to a *mismatch* between the semantic and/or syntactic information of the constructions that need to interact with each other.

As it turns out, FCG's standard techniques for applying constructions (see Bleys, 2011) provide the technical requirements for operationalizing coercion. A *mismatch* occurs when the matching phase fails during the application of a construction. If necessary, FCG can then skip the matching phase and immediately turn to its more flexible merging operation, which always succeeds as long as there are no conflicts. However, doing so is a costly operation and easily leads to a combinatorial explosion in search (e.g. allowing utterances such as *she sneezed her boyfriend*), so it is better to use a meta-layer than blindly allowing constructions to perform coercion. A second reason for using a meta-layer is that utterances that require coercion provide good learning opportunities for acquiring emergent patterns in a language. So the key to achieving coercion is to decide *when* a construction is allowed to skip its matching phase.

The example in this section illustrates these issues in more detail. It requires the reader to be familiar with the approach to argument structure as discussed by van Trijp (2011). In order to focus on the operationalization of coercion, it scaffolds some of the complexity involved, for example by treating phrases such as *off the table* as single lexical constructions and by ignoring issues concerning Tense, Aspect and so on.

4.2.1. *Defining a Grammar Fragment*

The example uses four lexical constructions: *Adele*, *sneezed*, *the napkin* and *off the table*. Each of these constructions introduces its semantic and syntactic combinatorial potential that allow it to interact with other constructions, which need to select an actual value from this potential during processing (van Trijp, 2011). In the case of verbal constructions, this involves a potential semantic valence and a potential syntactic valence. The following definition of *sneezed* allows the verb to occur in intransitive patterns:

```
(def-lex-cxn sneezed-cxn

 (def-lex-skeleton sneezed-cxn
  :meaning (== (sneeze ?ev)
               (sneezer ?ev ?sneezer))
  :args (?ev)
  :string "sneezed")

 (def-lex-cat sneezed-cxn
  :sem-cat (==1 (sem-function predicating))
  :syn-cat (==1 (syn-function verbal)))

 (def-valence sneezed-cxn
  :sem-roles ((agent sneezer))
  :syn-roles (subject)))
```

Meanings in this example are represented using first-order logic. The meaning of *sneezed* includes a predicate for the event itself and one for the 'sneezer', which is here assumed to be the only participant role involved in a sneeze-event. Besides `sneezed-cxn`, there are three nominal constructions, which have underspecified `sem-role` and `syn-role` features, which captures the fact that nominal units can play any semantic and syntactic role in a sentence. The only exception in the example is *off the table*, whose `syn-role` is `oblique` as its preposition excludes it from being the subject or object of an utterance:

```
(def-lex-cxn Adele-cxn
 (def-lex-skeleton Adele-cxn
  :meaning (== (adele ?x))
  :args (?x)
  :string "Adele")

 (def-lex-cat Adele-cxn
  :sem-cat (==1 (sem-role ?sem-role)
                (sem-function referring))
  :syn-cat (==1 (function nominal)
                (syn-role ?syn-role))))

(def-lex-cxn napkin-cxn
 (def-lex-skeleton napkin-cxn
  :meaning (== (napkin ?x))
  :args (?x)
  :string "the napkin")

 (def-lex-cat napkin-cxn
  :sem-cat (==1 (sem-role ?sem-role)
                (sem-function referring))
  :syn-cat (==1 (function nominal)
                (syn-role ?syn-role))))

(def-lex-cxn table-cxn
 (def-lex-skeleton table-cxn
  :meaning (== (table ?x))
  :args (?x)
  :string "off the table")

 (def-lex-cat table-cxn
  :sem-cat (==1 (sem-role ?sem-role)
                (sem-function referring))
  :syn-cat (==1 (function nominal)
                (syn-role oblique))))
```

Besides ‘lexical’ constructions, the example also involves some basic argument structure constructions, of which the intransitive and caused-motion constructions are the most relevant ones. Here is the intransitive construction:

```
(def-arg-cxn intransitive-cxn
 (def-arg-skeleton intransitive-cxn
  ((?event-unit
    :sem-cat (==1 (sem-function predicating))
    :syn-cat (==1 (syn-function verbal)))
   (?agent-unit
    :sem-cat (==1 (sem-function referring))
    :syn-cat (==1 (syn-function nominal)))))

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

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

The above templates first set up a skeletal construction that contains an event-unit and an agent-unit. The `def-arg-require` template defines an SV word order (the agent-unit comes before the event-unit), and finally the `def-arg-mapping` template states that the semantic role Agent maps onto the syntactic role subject. The templates do not define a constructional meaning for the intransitive construction, as the construction is too abstract to be associated with a coherent meaning. Next, the caused-motion construction is defined using the same templates:

```
(def-arg-cxn caused-motion-cxn
 (def-arg-skeleton caused-motion-cxn
  ((?event-unit
    :sem-cat (==1 (sem-function predicating))
    :syn-cat (==1 (syn-function verbal)))
   (?agent-unit
    :sem-cat (==1 (sem-function referring))
```

```
   :syn-cat (==1 (syn-function nominal)))
  (?patient-unit
   :sem-cat (==1 (sem-function referring))
   :syn-cat (==1 (syn-function nominal)))
  (?locative-unit
   :sem-cat (==1 (sem-function referring))
   :syn-cat (==1 (syn-function nominal)))))

(def-arg-require caused-motion-cxn
 ((?event-unit
   :cxn-meaning (== (cause-move ?ev)
                    (causer ?ev ?causer)
                    (moved ?ev ?moved)
                    (source ?ev ?source))
   :cxn-form (== (meets ?agent-unit ?event-unit)))))

(def-arg-mapping caused-motion-cxn
 :event
 (?event-unit
  :args (?ev)
  :sem-valence (==1 (agent ?ev ?agent)
                    (patient ?ev ?patient)
                    (locative ?ev ?locative))
  :syn-valence (==1 (subject ?agent-unit)
                    (object ?patient-unit)
                    (oblique ?locative-unit))
  :fusion ((?agent ?causer)
           (?patient ?moved)
           (?locative ?source)))
 :participants
 ((?agent-unit
   :sem-role agent
   :syn-role subject
   :args (?agent))
  (?patient-unit
   :sem-role patient
   :syn-role object
   :args (?patient))
  (?locative-unit
```

```
    :sem-role locative
    :syn-role oblique
    :args (?locative))))))
```

As can be seen, the caused-motion construction as defined in this example shows some degree of overlap with the intransitive construction: both require an event and an agent. On top of that, the caused-motion construction requires a patient-unit and a locative unit. It also carries its own constructional meaning ('x causes y to move from location z'), in which the locative participant is restricted to the source location of the motion-event for convenience's sake.

4.2.2. *Detecting a Problem in Routine Processing*

Suppose that the hearer observes the utterance *Adele sneezed the napkin off the table* and starts a parsing task. First, the four lexical constructions apply and add meaning, semantic and syntactic categories to the transient structure. Next, the collection of argument structure constructions is considered. First, the caused-motion construction fails to apply even though it finds all of the units that it requires (three nominal and one verbal units). The failure is due to a mismatch between the construction's required syntactic valence (a subject, object and oblique) and the potential syntactic valence provided by the verb (which only contains a subject). The construction is therefore put aside for the time being.

The intransitive construction, on the other hand, can successfully match and apply. It finds all the units that it requires (one nominal and one verbal unit), its word order constraints are satisfied, and its required syntactic valence matches with the potential syntactic valence provided by *sneezed*. The intransitive construction is in fact the only argument structure construction that can be applied during routine processing, and it is also the last one. At this moment in processing, parsing yields the following meanings:

```
((sneeze ?ev) (sneezer ?ev ?sneezer) (adele ?sneezer)
 (napkin ?obj-x) (table ?obj-y))
```

In the parsed meaning, the predicates `sneezer` and `adele` are connected through each other through the shared variable `?sneezer`. The predicates `napkin` and `table`, however, are unconnected from the rest of the meaning, which may cause problems for interpretation. Clearly, this parsed meaning does not indicate the proper participant structure of the sentence, nor does it convey the caused-motion meaning that the speaker implied. It is therefore necessary to implement a diagnostic that autonomously detects whether the grammar has dealt with the argument structure of the sentence adequately.

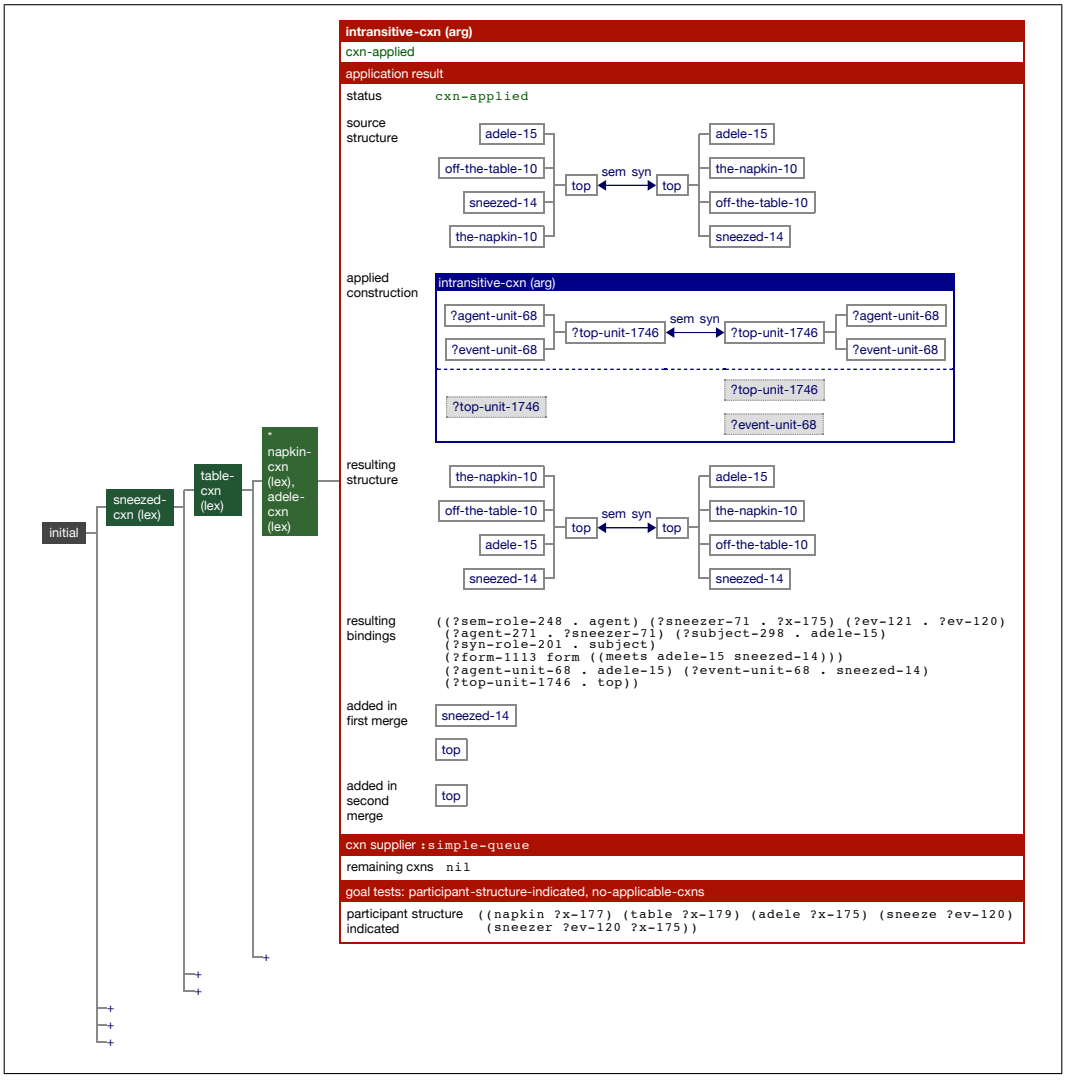

Figure 6. *Given the right diagnostics, the FCG-system can detect that the grammar did not handle the argument structure of a sentence adequately.*

In the case of English speakers, it is plausible to assume that they have developed the necessary meta-knowledge about their language. Part of that knowledge is that the participant structure of a sentence is always marked on a clausal level. A diagnostic for English argument structure can thus simply test whether the meanings of every unit within a clause are linked to the meanings of other units through variable equalities. Figure 6 shows the current example of routine processing. The final node in the search tree is expanded and shows the application of the intransitive construction. The goal-test `participant-structure-indicated` implement the diagnostic just described and returns a failure. The diagnostic then reports a problem and FCG's meta-layer is activated to try and repair the problem.

4.2.3. *Repair through Coercion*

As said before, a possible repair is coercion: instead of first having a matching phase, FCG tests whether a construction can impose its feature structure through the merge operation. As the above diagnostic is specific to argument structure constructions, the repair also only considers those kinds of constructions for merging.

If coercion is allowed, not only the intransitive construction can apply, but the caused-motion construction as well. The resulting transient structure of applying the caused-motion construction is shown in Figure 7. Since only the caused-motion construction satisfies the goal-test of indicating the complete participant structure, it is chosen as the best branch in the search process. In the case of multiple constructions satisfying the goal-test, the hearer must use other contextual knowledge and cognitive processes to figure out which is the most plausible one. Coercing the caused-motion construction leads to the following parsed meaning, in which all coreferential variables are made equal and to which the caused-motion construction has added a caused-motion sense:

```
((Adele ?x) (napkin ?y) (table ?z)
 (sneeze ?ev) (sneezer ?ev ?x)
 (cause-move ?ev) (causer ?ev ?x)
 (moved-object ?ev ?y) (source ?ev ?z))
```

4.2.4. *Consolidating: Learning from Coercion*

If coercion has helped the hearer in achieving a successful communicative interaction, s/he can try to learn from coercion in order to be able to use *sneeze* as a caused-motion verb in future utterances (as in *She sneezed the tissue off the table* or *she sneezed the foam off the cappuccino*). In other words, the language user will try to consolidate this particular usage experience. This is possible by performing the following steps:

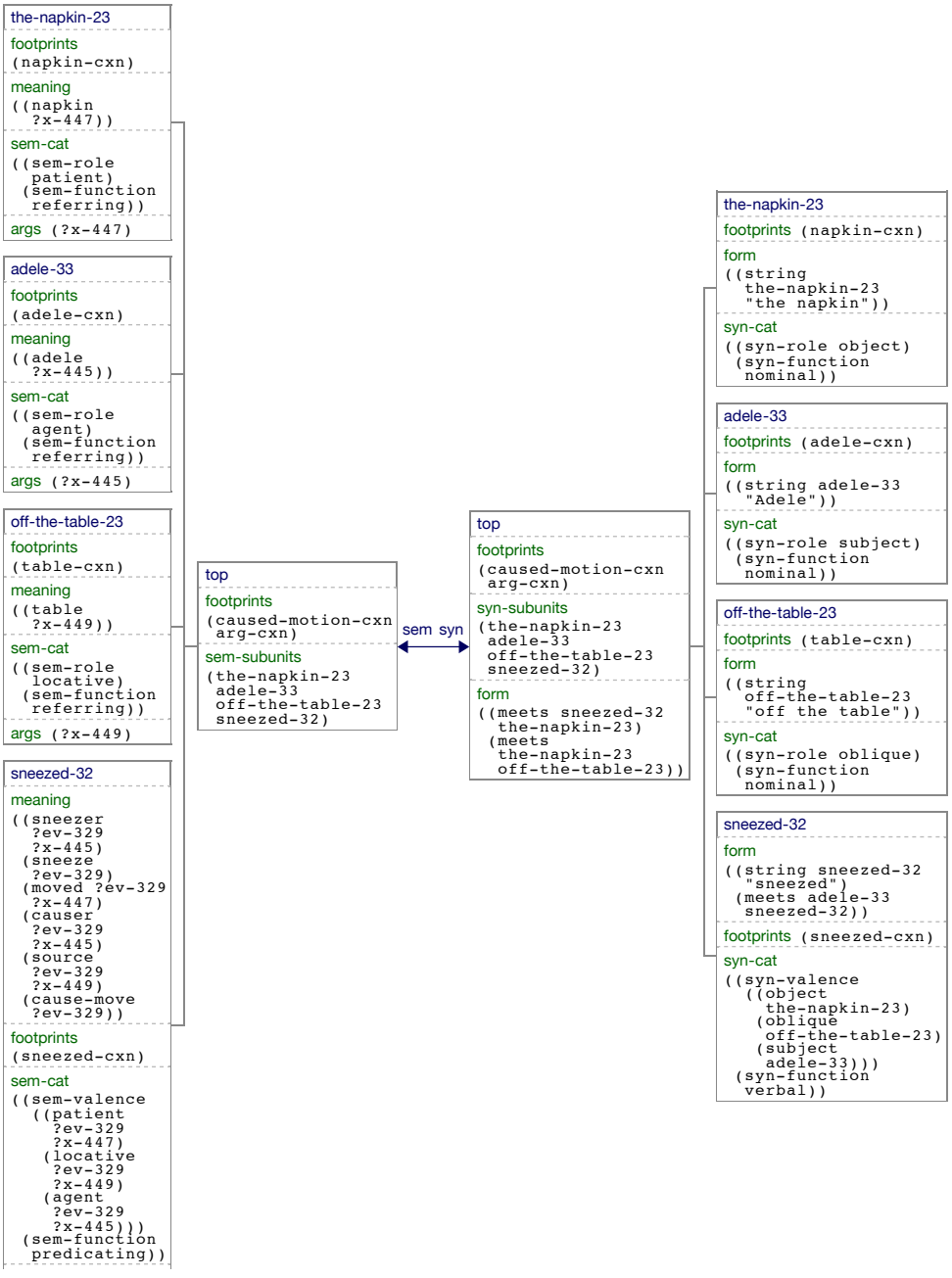

Figure 7. *Resulting transient structure after coercing the caused-motion construction.*

1. Check which feature-value pairs are added by the coerced construction;
2. Check which units these pairs belong to and retrieve the constructions that were responsible for creating the units by checking their `footprints` feature or by inspecting the search history.
3. Update the constructions and, if necessary, create additional links between constructions in the linguistic inventory.

FCG provides all the necessary information for the first step, so we can easily find that the construction has added the following syntactic roles to the verbal unit on the syntactic pole:

```
((object ?napkin) (oblique ?table))
```

And the following semantic roles on the semantic pole:

```
((patient ?ev ?y) (location ?ev ?z))
```

Next, the `footprints` feature tells us that the verbal unit was created by the `sneezed-lex` construction, whose categories `sem-valence` and `syn-valence` can now be updated using the same `def-valence` template as before. Finally, the language user creates a link between the verb *sneeze* and the caused-motion construction. As there is only one usage experience so far, this link will have a low confidence score because the language user doesn't know yet whether a caused-motion usage of *sneeze* is a valid convention of his/her language. The degree of entrenchment of this distribution may increase if the language user observes or performs more successful instances, or the link may be forgotten if there is no reinforcement in later interactions.

5. CONCLUSIONS

There is a long tradition in linguistics that sees a grammar as prescribing how a language should be spoken, and consequently parsing is seen as the process of deciding whether a sentence is grammatical. Parsers therefore block as soon as situations are encountered that are ungrammatical. However, grammaticality is not what is most significant to human listeners. They need to understand what the speaker is saying and do not pay attention to whether the utterances they hear adhere strictly to the well established syntactic conventions of their language. Indeed, listeners may

not even notice that there are ungrammaticalities (just as readers may not notice spelling mistakes).

Similarly, speakers really only worry about strict grammaticality when they are writing texts, since it is only then that they have adequate time to systematically revise their sentences in order to satisfy the conventions that the authors believe to be the standard. In normal discourse, utterances have to be produced so fast that speakers cannot spend enough time to ensure grammaticality, which typically means that sentence fragments are produced as fast as they are ready and subsequently may not fit within the larger plan of a sentence.

Fluid Construction Grammar drops the linguistic obsession with grammaticality judgements and suggests that grammar and grammatical processing must be designed to be flexible, so that incomplete fragments, unconventional expressions and errors do not entirely block language processing. Parsing and production should be able to proceed as far as possible, including up to the point of interpreting available fragments or articulating partial phrases, so that feedback can be obtained and the dialog can be repaired.

References

Bleys, Joris (2011). Search in linguistic processing. In Steels (2011b).

Brown, C.M., P. Hagoort (Eds.) (1991). *The Neurocognition of Language*. Oxford: Oxford University Press.

De Vylder, B., K. Tuyls (2006). How to reach linguistic consensus: A proof of convergence for the naming game. *Journal of Theoretical Biology*, 242(4), 818–831.

Fernández-Ordóñez, Ines (1999). Leísmo, laísmo, loísmo: Estado de la cuestión. In I. Bosque, V. Demonte (Eds.), *Gramática Descriptiva de la Lengua Española*, vol. I, 1319–1390. Madrid: RAE – Espasa Calpe.

Fouvry, Frederik (2003). *Robust Processing for Constraint-Based Grammar Formalisms*. Ph.D. thesis, University of Essex, Colchester.

Garrod, S., A. Anderson (1987). Saying what you mean in dialogue: A study in conceptual and semantic coordination. *Cognition*, 27, 181–218.

Gleitman, Lila (1990). The structural sources of verb meanings. *Language Acquisition*, 1(1), 3–55.

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

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

Grimshaw, J. (1981). Form, function, and the language acquisition device. In C.L. Baker, J.J. McCarthy (Eds.), *The Logical Problem of Language Acquisition*, 183–210. Cambridge MA: MIT Press.

Heine, Bernd (1997). *The Cognitive Foundations of Grammar*. Oxford: Oxford University Press.

Hopper, Paul (1991). Emergent grammar. In E. Traugott, B. Heine (Eds.), *Approaches to Grammaticalization*. Amsterdam: John Benjamins.

Labov, William (1994). *Principles of Linguistic Change. Volume 1: Internal Factors*. Oxford: Basil Blackwell.

Levelt, W.J.M. (1989). *Speaking*. Cambridge MA: MIT Press.

Liberman, A.M., I.G. Mattingly (1985). The motor theory of speech perception revised. *Cognition*, 21, 1–36.

Loetzsch, Martin, Joris Bleys, Joachim De Beule (2011). Linguistic processing in FCG. In Steels (2011b).

Sperber, D., D. Wilson (1986). *Relevance: Communication and Cognition*. Cambridge, MA: Harvard University Press.

Steels, Luc (1996). A self-organizing spatial vocabulary. *Artificial Life*, 2(3), 319–332.

Steels, Luc (2003a). Evolving grounded communication for robots. *Trends in Cognitive Sciences*, 7(7), 308–312.

Steels, Luc (2003b). Language re-entrance and the 'inner voice'. *Journal of Consciousness Studies*, 10(4-5), 173–185.

Steels, Luc (2011a). A design pattern for phrasal constructions FCG. In Steels (2011b).

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

Steels, Luc, Frédéric Kaplan (2002). Bootstrapping grounded word semantics. In T. Briscoe (Ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, 53–73. Cambridge: Cambridge University Press.

Steels, Luc, Martin Loetzsch (2009). Babel: A tool for running experiments on the evolution of language. In S. Nolfi, M. Mirolli (Eds.), *Evolution of Communication and Language in Embodied Agents*, 307–313. Berlin: Springer.

Talmy, Leonard (2000). *Toward a Cognitive Semantics, Concept Structuring Systems*, vol. 1. Cambridge, Mass: MIT Press.

van Trijp, Remi (2011). A design pattern for argument structure constructions. In Steels (2011b).