Insight grammar learning using pro- and anti-unification
(Draft Version)

Luc Steels [1,2*] **and Paul Van Eecke** [2,3]

[1] *ICREA-Institut de Biologia Evolutiva, CSIC-UPF, Department of Experimental and Health Sciences, Universitat Pompeu Fabra, Barcelona, Spain*
[2] *Artificial Intelligence Laboratory, Computer Science Department, Vrije Universiteit Brussel, Belgium*
[3] *Sony Computer Science Laboratory, Paris*

Correspondence*:
Luc Steels, IBE (UPF-CSIC), dr. Aiguader 88, 08003 Barcelona, Spain
steels@arti.vub.ac.be

ABSTRACT

The paper proposes concrete mechanisms to achieve insight grammar learning. This form of learning attempts to surmise what kind of grammatical constructions are missing to handle an utterance that contains novel features using *abductive* inference, and is therefore complementary to corpus based statistical language learning which relies on *inductive* inference. Insight grammar learning requires the capacity of meta-level cognition in the form of diagnostics, repairs and consolidation strategies.
Anti-unification is proposed here as a powerful building block for repairing an impasse. Anti-unification is the opposite of unification. Unification is used in feature-structure based grammars to determine whether a grammatical schema fits with an utterance being processed. Unification finds the simplest substitution of variables such that two expressions, in this case a construction schema and a transient structure capturing information derived about the syntactic and semantic structure of an utterance, may become equal. Anti-unification figures out how a partially matching schema might be relaxed so that it still fits, after which the schema can to some extent be applied to extend the transient structure.
Anti-unification often overgeneralizes, and we therefore propose a second mechanism, pro-unification, as basic building for consolidating the outcome of a repair. Pro-unification takes a construction schema generalized through anti-unification and constrains it again based on the current transient structure.
We have integrated pro- and anti-unification in Fluid Construction Grammar (FCG), a fully operational computational implementation of construction grammar, and demonstrate here through a series of examples and experiments how these two mechanisms capture aspects of insight grammar learning.

Keywords: grammar learning, insight problem solving, insight learning, construction grammar, meta-cognition, anti-unification, pro-unification

1 INTRODUCTION

1.1 Language processing as problem solving

Decades of research into problem solving, starting from the seminal work of Newell and Simon in the nineteen fifties (Newell and Simon, 1972), have by now provided us with a wealth of empirical observations, models, computational implementations (Laird, 2012), (Taatgen and Anderson, 2008), and neural data (Anderson et al., 2009). Problem solving is commonly analyzed into three components:

1. A *problem state* representation, which represents the current state of knowledge of the problem solver about the problem situation, for example, a board position in a chess game.
2. A *goal* characterizing a solution state, e.g. a win in chess by check mate.
3. *Operators* to move a problem state closer to a solution state. For a game like chess, the operators consist of the possible movements of the different pieces on the chess board.

The problem solver starts from the problem state and then keeps applying operators iteratively until the solution state is reached. A chain of problem states linked by operators is called a *pathway*. Because usually, several operators can apply to a problem state, there is unavoidably a search space exploring different pathways. This space is typically combinatorially explosive and therefore cannot be searched exhaustively. Hence, effective problem solvers must also include: (i) *Macro-operators*, which allow a jump in the search space, to immediately reach the solution from an initial state, or at least go a significant way towards the solution. (ii) *priming networks*, which suggest which operators are useful to consider next once a particular operator has applied, (iii) *choice heuristics*, which help to choose which of a few possible operators is most likely to lead to a solution, and (iv) *depth heuristics*, which gauge how deep a pathway needs to be pursued before abandoning it.

Language processing can be viewed as a problem solving process. This is not very common in linguistics, perhaps because the term problem solving is associated with explicit conscious problem solving. However, there is no particular reason why the same mechanisms postulated for conscious problem solving could not operate at a level below consciousness. In fact, not much imagination is required to apply the standard model of problem solving to language processing. We need to identify the goal, the state representation, the operators and what heuristics allow search in the space of possible hypotheses (See the summary in Figure 1).

Problem solving	**Language processing**
problem state representation	transient structure
initial state	for the speaker: meaning to be formulated for the listener: utterance to be comprehended
final state	for the speaker: utterance for the listener: reconstructed meaning
problem solving operators	construction schemas
if-part	for speaker: production lock for listener: comprehension lock
then-part	for speaker: contributor and comprehension lock for listener: contributor and production lock
pathway in search space	linguistic pathway in search space
heuristics	schema score, heuristic criteria, interpretability, chunks

Figure 1. Table summarizing how the classical model of problem solving can be mapped onto language processing.

In the case of language, the speaker's initial problem state contains the meaning he wants to express and the final state contains the utterance expressing this meaning. On the way, various grammatical structures get built, such as constituent structure, functional structure, dependency structure and argument structure. The listener's initial problem state is an utterance and the final state a reconstruction of its meaning. The listener also builds the same sort of intermediary structures on the way.

A problem state representation should contain everything known about the utterance at some point in processing. In the Fluid Construction Grammar framework used here (Steels, 2011), such a representation is called a *transient structure*. It takes the form of a *feature structure*, which are representations of linguistic information commonly used in most linguistic formalisms today, such as Unification Grammar (Kay, 1984) or HPSG (Pollard and Sag, 1994). Simplifying, a feature structure contains units, features, and values for these features, which can themselves be sets of feature-value pairs. Figure 2 contains an example. This example, and all others that follow, can be inspected through a web demonstration accessible through this link: https://www.fcg-net.org/demos/frontiers-demo/. This web demonstration is an integral part of the paper because it not only illustrates the various examples discussed here, but also proves that the proposed mechanisms work. We refer to sections of the demonstration as (WD-X) where X is the index of the section within the web demo.

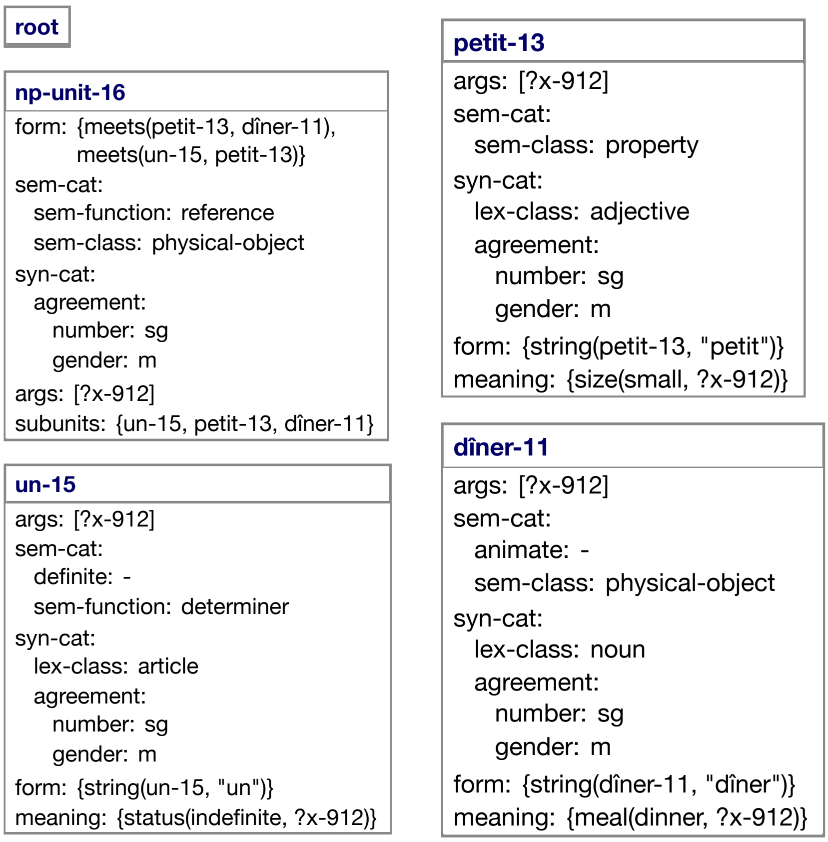

Figure 2. (See WD-1 in web demo.) This figure displays the transient structure in the form of a feature structure for the utterance "un petit dîner" (French for 'a small dinner'). It contains units for a root (which functions as an input buffer), the noun "dîner", the article "un", the adjective "petit", and the noun-phrase which groups these words. All properties and structures are represented explicitly using features and values: the syntactic and semantic categories, the meaning, the form including the string for a word and the ordering relations between constituents (represented explicitly using meets-constraints), the constituent structure (subunits), and any other information deemed relevant.

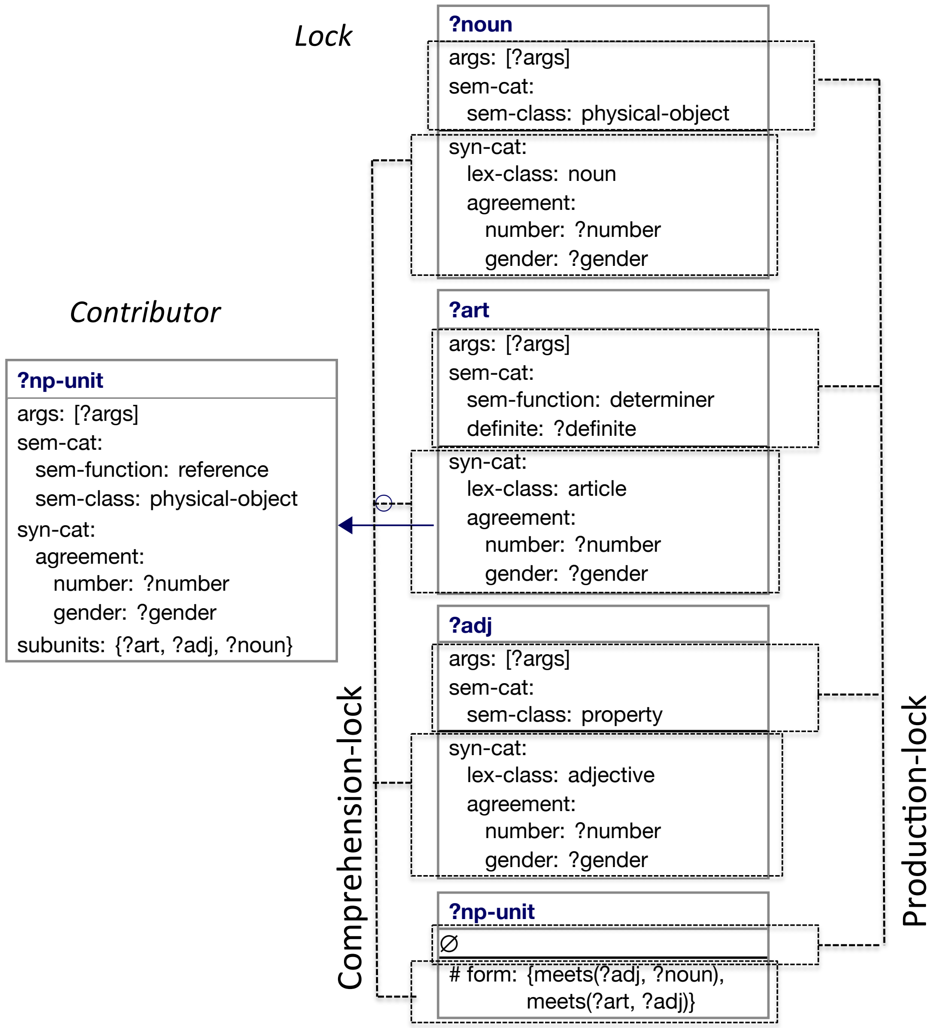

Figure 3. (See WD-2 in web demo.) A construction schema consists of a contributor (written on the left hand side) and a lock (on the right hand side). Both consist of a set of units with features and values, just like transient structures. The lock decomposes into a production-lock and a comprehension-lock. The production-lock contains the production-constraints of each unit (written first in the unit's feature structure) and the comprehension-lock the comprehension-constraints (written below the production-constraints).

To view language processing as problem solving, knowledge of the language (lexicon, morphology, syntax, semantic interpretation rules) has to be conceptualized in terms of problem solving operators that allow transitions from an initial to a final state. In FCG, an operator is equal to a construction schema (Figure 3 and WD-2). A schema specifies under what conditions additional information about the utterance can be inferred and what that information is. Its function is therefore similar to a production rule in traditional models of problem solving. The if-part of a construction schema, written on the right-hand side of the left arrow, is called the lock and the then-part, written on the left-hand side of the left arrow, is called the contributor.

Schemas are also represented using feature structures, just like transient structures. But to make them abstract, schemas contain variables that get bound in the process of matching a schema against a transient structure and applying the schema, in the sense of adding more information contained in the schema to the transient structure. A variable is written as a symbol preceded with a question mark, such as ?gender or

?NP-unit. FCG uses logic variables, familiar from logical theories of inference and logic programming languages. Logic variables are like ordinary variables in the sense that they can become bound to constants, but they can also be bound to other variables and remain unbound without leading to an error state as in normal programming languages. The list of bindings between variables and their bindings is called a *binding-list* and represented as a list of dotted-pairs, such as: ((?np-unit . np-unit-91) (?noun . fille-126) (?gender . f) (?number . sg) (?det . une-58)), where ?np-unit, ?noun, etc. are all variables and np-unit-91, fille-126, etc. are their respective bindings.

A *linguistic pathway* consists of a sequence of transient structures that are derived by the consecutive application of construction schemas (see Figure 4 and WD-3). Typically, several possible pathways have to be explored in case more than one construction schema can apply, and so we get unavoidably a search space which is combinatorially explosive. As in all cases of non-trivial problem solving, the choice for the most appropriate operator should be guided by *heuristics*, which are in the case of language partly based on how much success the construction schema has had in past language usage, stored as a score stored with the construction schema, partly on heuristic criteria such as simplicity or connectedness of syntactic structure, and partly on whether the (partial) meaning derived on the pathway so far makes sense in the current context. Other techniques such as priming networks or macro-operators based on chunking construction schemas have also been explored within the FCG grammar formalism (see Steels (2012)).

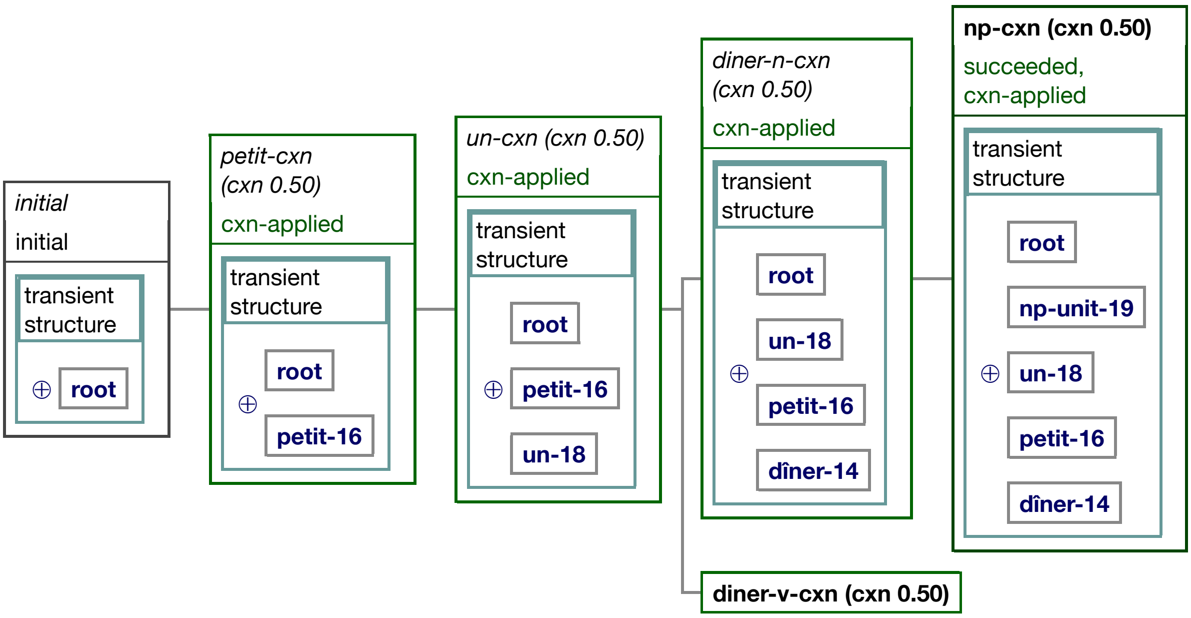

Figure 4. (See WD-3 in web demo.) Search space generated while comprehending "un petit dîner". Words are being processed with lexical constructions and then combined into larger structures through grammatical constructions. There are two pathways here because the word 'dîner' can be a noun as well as a verb. Production is possible with the same constructions and generates a similar search space (see WD-4 in web demo).

Language users are able to formulate as well as comprehend utterances and it is highly desirable that the same representation of language knowledge, and the same architecture, can be used in comprehension and formulation (Strzalkowski, 1994). This requires that language operators (construction schemas) and the engine applying them (unification) must work in a bi-directional fashion. This property is achieved in FCG by introducing two locks, a production and a comprehension lock (Figure 5). In formulation, the production lock has to match with the transient structure and if that is the case, information, both from the contributor and the comprehension lock, is added to the transient structure. In parsing, the comprehension lock has to match and if that is the case, information, both from the contributor and the production lock, is added to the transient structure. In FCG, both the match and merge operations are based on (subset) unification (Martelli and Montanari, 1982) and therefore called U-Match and U-Merge.

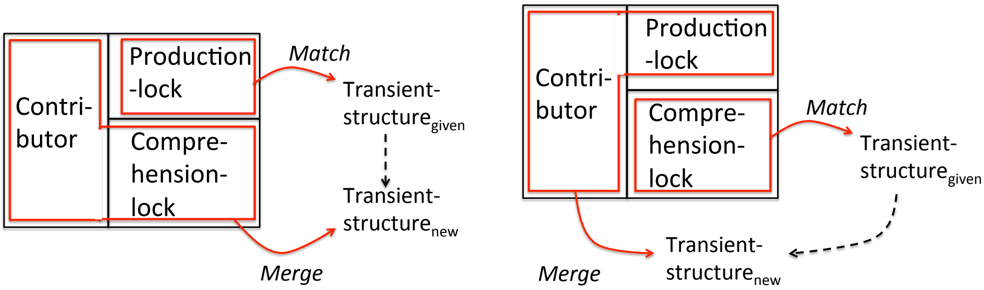

Figure 5. In production (left image), the transient structure should fit with the production-lock (Match) and then information from both the comprehension-lock and the contributor are added (Merge). In parsing (right image), the transient structure should fit with the comprehension-lock (Match) and then information from both the production-lock and the contributor are added (Merge).

Dual usage of a construction schema cuts the amount of construction schemas, and therefore the memory needed to store them, in half. It simplifies learning, because otherwise complex mechanisms need to be in place to maintain consistency between a production and a comprehension inventory. And it allows re-entrance: speakers can easily monitor their own language production by parsing what the production process is coming up with, and listeners can use the meanings they have reconstructed so far and re-produce them with their own construction inventory, predicting what is going to come next. It is true that learners typically can comprehend a lot more than they produce, but that is because comprehension does not require the same level of precision. Pragmatic inference, shared context, and common sense knowledge compensate, and schemas can partially match, ignoring some of the grammatical cues in the input (as discussed below).

1.2 Grammar learning

As widely discussed in the literature, problem solving operators can either be learned through statistical techniques or through insight learning. Insight learning consists of two steps. There is first a process of *insight problem solving*, in which routine processing reaches an impasse (Ohlsson, 1984) and is then repaired by meta-level processes (Laird, 2012). There is substantial psychological evidence that this occurs abundantly for human language, in order to cope with ungrammaticalities, errors, misunderstandings, and novel phrasings (Garrod and Anderson (1987), Dingemanse (2015)). *Insight learning* takes place when the outcome of a repair is consolidated in terms of new operators, possibly using new representations of the problem situation, so that the impasse does not occur again in the future. Consolidation does not always happen. The impasse may due to obvious errors from the side of the speaker or the repair may be shown to be unjustified after further interaction with the speaker.

Several prior computational experiments in insight grammar learning have modeled how knowledge of the context can be used for repairing a grammatical impasse (Beuls et al. (2012), Garcia-Casademont and Steels (2016), Spranger (2016)). For example, suppose that a listener L is unfamiliar with the German case system and observes a situation in which a man gives a book to a woman. The speaker S describes this situation as: ‘der Frau gibt der Mann das Buch’ (lit. the woman (dative) gives the man (nominative) the book (accusative)). Unaware of the case information, L interprets this utterance as: ‘the woman gives the man the book’. But this conflicts with L’s observation of the situation and hence L reaches an impasse. He can repair this impasse by assuming that the article “der” (in “der Frau”) is not signaling here that the woman is the subject and hence agent of the give-action, which would have required “*die* Frau”, but rather a marker of the recipient role of the give-action.

In the remainder of this paper, we model a complementary insight grammar learning strategy in which a repair is achieved *without* access to a shared context. It is appropriate in cases where no shared context is available, such as in displaced communication, or where the existing inventory of constructions cannot be applied and therefore a possible semantic interpretation, from which a repair might be possible based on the context, cannot be achieved. However, what the learner could do in such a case is relax some of the constraints on the best partially matching construction schema so that further processing becomes possible. We hypothesize that *anti-unification* (Plotkin, 1971) is a powerful general mechanism that can achieve this. Whereas unification seeks to find out how two expressions (in this case feature structures) can be made equal by finding a minimal binding-list (the most general unifier), anti-unification seeks to find out how one expression (the pattern), which is not yet unifiable with another expression (the source), can be made to unify by proposing a minimal generalization of the pattern, called the *least general generalization*.

Here is an intuitive example: The listener gets the utterance “he facebooked me this morning”. The word “facebooked” violates the application of the past tense formation construction that requires a verb as root. However, by relaxing this constraint, the listener can go ahead, parse the utterance further using the transitive construction and possibly come to some sort of interpretation. If this interpretation makes sense, he can extend the lexical construction of “facebook” with the information that this word has also the potential to be a verb, so that, next time, this usage of the original noun “facebook” can be handled by routine processing.

Although anti-unification is a powerful repair strategy there is also a risk. The generalization of a construction is often so broad that it would also allow many other cases to be processed which should not. Here is an intuitive example: Suppose the listener has already a construction schema that handles noun phrases consisting of an article, an adjective, and a noun, as in “the surprising goal”, but now encounters a noun phrase without adjective, namely “the goal”. The art-adj-noun construction schema fails to match this situation but it can be generalized by relaxing the constraint that an adjective unit has to be present. This generalized construction schema is then applicable and the listener can derive enough information to make semantic interpretation possible. But if this generalized construction schema would be stored in consolidation, it would also accept noun phrases with another word order, such as “goal the”. We hypothesize that a novel operation, which we call *pro-unification*, can avoid overgeneralization. Pro-unification takes a generalized construction and makes it more specific again, namely by introducing constraints from the case that provoked the repair (see Figure 6). For the noun-phrase example, it means that constraints on word order between article and noun are reintroduced.

2 MATERIALS AND METHODS

We have worked out these hypotheses about the role of pro- and anti-unification, designed and implemented algorithms for them, and integrated implementations of these algorithms within the Fluid Construction Grammar framework. We have also conducted experiments to understand the strength and limitations

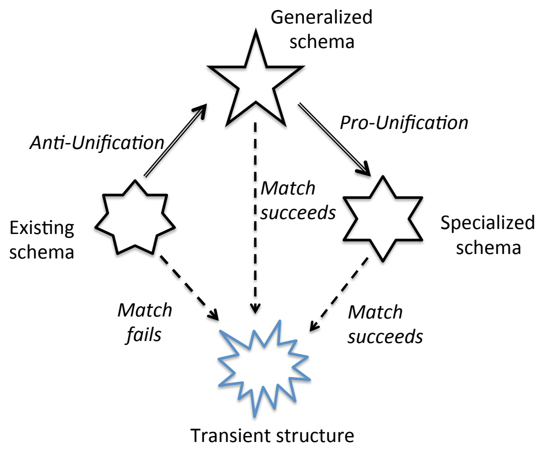

Figure 6. Anti-unification generalizes a construction schema so that it becomes applicable and pro-unification specializes it again to take into account properties present in the transient structure. Anti-unification is proposed here as a mechanism for repairing impasses when no construction schemas are applicable anymore and pro-unification is proposed as a mechanism to avoid overgeneralization.

of these mechanisms, reported later in the results section. This section first provides more detail on the mechanisms themselves, illustrated with examples in the web demonstration.

2.1 Diagnostics

Part of the power of Fluid Construction Grammar comes from the fact that it is able to process an utterance as far as possible, even though there are unknown words, ungrammaticalities, or disconnected fragments on the way. Moreover FCG performs semantic parsing. It not only derives various syntactic structures but calculates a semantic network which is then interpreted against the listener's world model of the situation. For example, even if there is a lexical construction missing for a word, other words, occurring later in the utterance, still get processed and may lead to partial syntactic structures, partial semantic networks and partial semantic interpretations. So, processing never gets totally stuck because of a missing or mismatching construction schema. Instead the key impasse, to be used in the experiments reported here, is that the semantic networks supplied by different words cannot be integrated into a single fully connected semantic network.

Concretely, we use a variant of typed second-order predicate calculus for representing the semantics of utterances. It is represented graphically in terms of a network where the nodes represent predications, always in the form of a triple ⟨type, predicate, value⟩, and the links represent co-reference relations between the variables or constants occuring in these predications. For example, the semantic network for the French utterance "le petit garçon mange un bon repas" (the small boy eats a good meal) is shown as in Figure 7 a. For an ungrammatical utterance, or an utterance for which there are missing constructions, the semantic network obtained after application of all grammatical constructions is not fully connected For example, "la petit garçon mange un bon repas" (the small boy eat a good meal) violates the number-agreement between article and noun ("la" is feminine of "le"), hence the noun-phrase construction fails to become active and the transitive clause construction as well. But note that "un bon repas" could still be parsed and contribute to a partially connected network (see Figure 7b.).

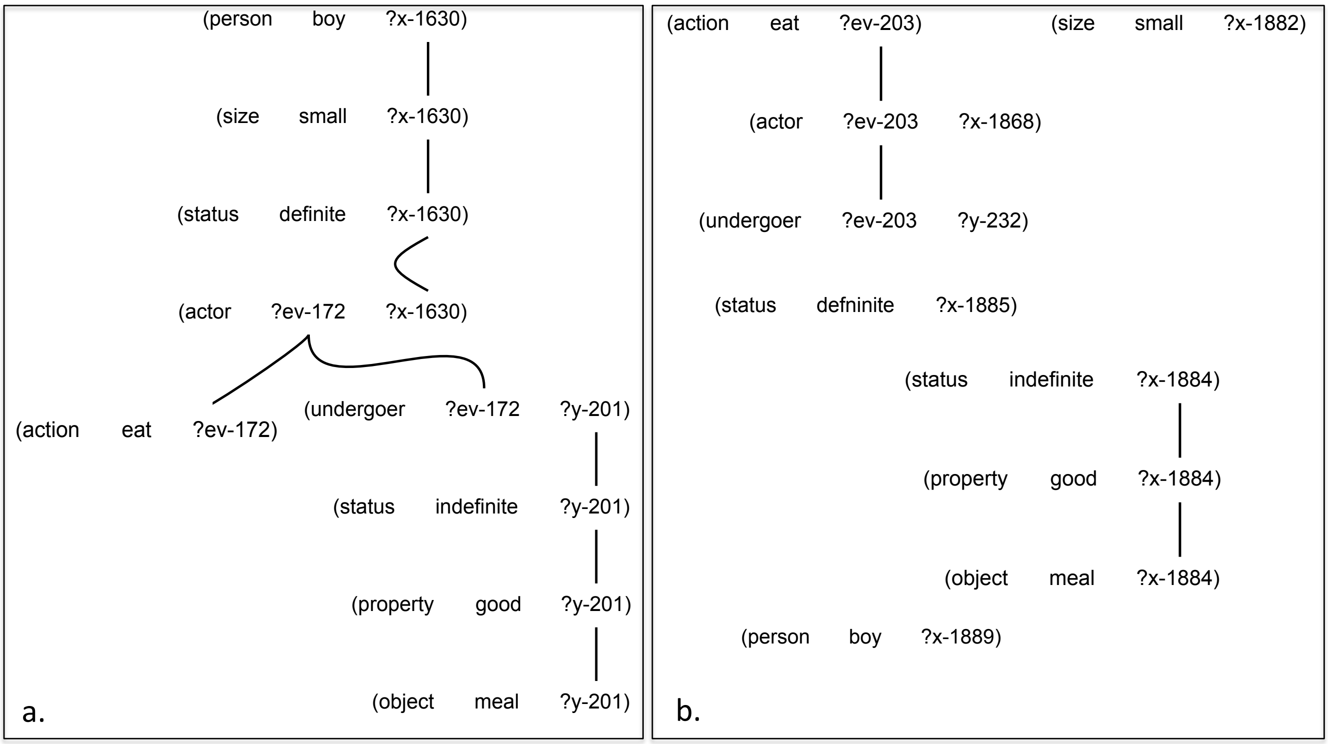

Figure 7. Figure a. (on the left) shows the semantic network for “le petit garçon mange un bon repas” (the small boy eats a good meal). Predications (nodes in the network) are introduced by individual words and co-referential links by grammatical constructions. The variable names ?x-1630, ?x-1882, etc. are all generated by the FCG system itself. Figure b. (on the right) shows an example of a network which is not fully connected. It is only partially interpretable against the world model and hence constitutes an impasse.

2.2 Anti-unification

Anti-unification needs to find the least general generalization that unifies a pattern (which is a lock or contributor in a construction schema) and a source (which is a transient structure). It decomposes into two steps: unit pairing and unit adjustment.

Unit pairing tries to pair the units of the pattern and the source. This is a non-trivial problem because the names of the units in the pattern are variables. The standard (subset) unification algorithm has been extended to yield a graded rather than yes/no answer (i.e. match or no match). The graded answer specifies in how far the two units are matching and what conflicts appear, for example, which variables from the pattern are bound to different values in the source.

Then there are three cases: (i) Some units from the pattern Match with a unit in the source. They can therefore be paired and incorporated as such in the generalized pattern. (ii) Some units from the pattern find no equivalent in the source. These pattern-units cannot be paired, and are therefore left out in the generalized pattern (unit-deletion). (iii) Some units match only partly and then some adjustment of the pattern is needed before including it in the generalized pattern. Usually there are several possibilities, and they are ranked based on a cost-function (explained later), so that the most plausible unit-pairing can be considered first in acting out the repair and in consolidation.

Unit adjustment. If a unit pattern does not completely match with the source, there are four operations that are performed to generalize the pattern so that it can nevertheless match: variable-decoupling, value-relaxation, predicate-relaxation, and feature-relaxation.

A. Variable-decoupling means that the same variable occurs more than once in the pattern, but there are different values in the positions of these variables in the source. For example, the English subject-verb construction requires agreement between the subject and the verb for number, which is implemented by having the same variable for the number features of the subject and verb unit. Suppose however, that a sentence has to be parsed that violates this, such as “she play in the garden”. Language users are effortlessly able to cope with this - maybe not even noticing the error. Anti-unification handles this by assuming that the variables defining number for the subject and the verb are different, i.e. the original single variable, for example ?number, gets decoupled into two variables, for example ?number-1 and ?number-2, which can then each individually bind to a different value. Match now works and the rest of the construction can be applied.

Here are two other intuitive examples illustrating this powerful mechanism. They can be inspected through the web demonstration by clicking on boxes and structures to see more detail.

Example 1. Ignoring agreement failure (shown in WD-5). Consider a French nominal phrase which requires agreement for number and gender between article, adjective, and noun, as in “une petite fille” where “une” translates as ‘a’ (feminine singular), “petite” as ‘small’ (feminine singular), and “fille” as ‘girl’ (feminine singular). Number and gender also percolate from these constituents to the noun-phrase unit as a whole. But suppose now that the phrase “un petit fille” has to be parsed. Because “un” and “petit” are masculine, they do not agree with “fille” which is feminine. Variable-decoupling solves this problem by assuming different variables in the generalized construction. The noun-phrase can be built and the semantic pole of the construction applied so that processing can continue with the rest of the utterance.

Example 2. Handling word order deviation (see Figure 8 and WD-6). Suppose the language learner already knows a construction for the French nominal phrase where the ordering of constituents is a sequence of article, adjective and noun, as in “un beau dîner” (a beautiful dinner). But now the phrase “un dîner formidable” (lit. ‘a dinner splendid’) has to be parsed. The constituent ordering constraint is violated and Match fails. However, anti-unification can solve this by decoupling the variables in the meets-constraints that define this ordering, thus neutralizing them. Concretely, the original specification in the NP-unit requires: meets(?adj, ?noun) and meets(?art, ?adj), whereby the units for ?art, ?adj, and ?noun are defined as part of the construction schema. By changing the variables in the ordering specification to meets(?art, ?x) and meets(?y, ?z) the variables ?x, ?y, and ?z get bound to whatever unit satisfies these meets-constraints in the utterance but they no longer have to be the same as the units ?adj and ?noun defined in the construction schema.

B. Value-Relaxation means that there is a different value for a particular feature in the pattern and in the source. This can be resolved by assuming that the value is a variable in the more general pattern and then it can bind to the value in the source. A value can either be an atomic constant or a set of feature-value pairs. Here is an intuitive example:

Example 3. Mismatch in feature values (shown in WD-7). Suppose that there is a noun-phrase construction that includes a determiner and a noun, and that the determiner unit has to have the lex-class ‘article’. This construction could handle an utterance such as “the children”. But now, the utterance “many children” has to be processed, whereby the lex-class of “many” is ‘quantifier’. Match blocks the noun-phrase construction because article and quantifier are different values of the lex-class feature. Anti-unification resolves this by

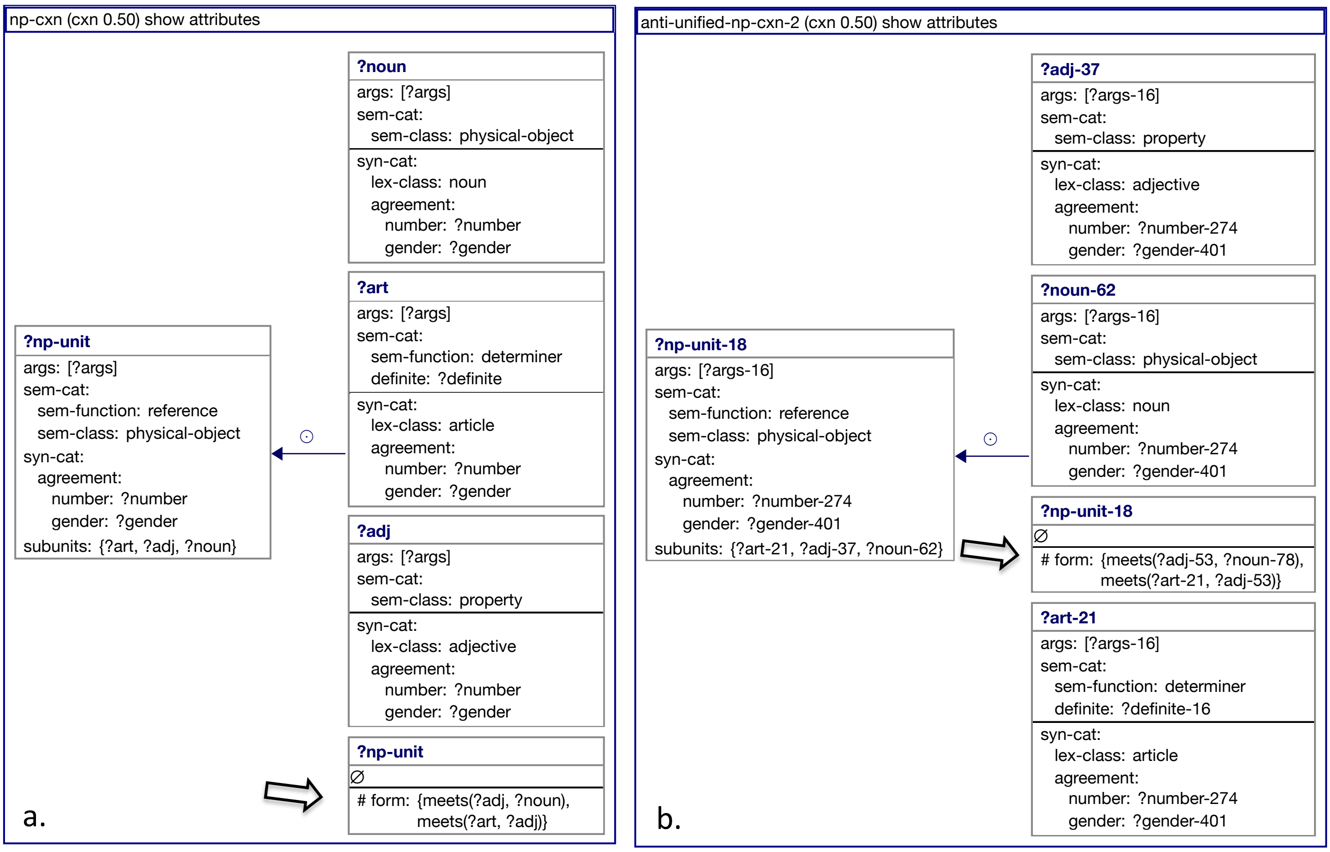

Figure 8. (WD-6) Figure a. (on the left) shows the original noun-phrase construction. Figure b. (on the right) shows the generalized noun-phrase construction based on anti-unification. Note that the order in which units appear in a construction schema is random and the constituent ordering expected in the utterance is described explicitly using meets constraints. Unit-names are renamed in the generalized construction schema: ?np-unit in a. maps to ?np-unit-18 in b., ?art to ?art-21 , ?adj to ?adj-37, and ?noun to ?noun-62. (All these indices are generated automatically.) Compare now the form-feature in ?np with that in ?np-unit-18 (both indicated with a black arrow). The form-feature of ?np-unit-18 in the generalized construction (b.) includes now meets(?adj-53, ?noun-78) and meets(?art-21, ?adj-53). So the ?adj variable in the original construction has been decoupled into ?adj-21 and ?adj-53 and the ?noun variable into ?noun-62 and ?noun-78, thus neutralizing the ordering constraint. This generalized construction schema now handles the novel word order "un dîner formidable" (see demo WD-6 in web demo).

assuming that the value of the lex-class feature is a variable. Processing then continues and the noun-phrase construction can build the noun-phrase and add semantic constraints to the transient structure.

C. Unit deletion (shown in WD-8). There are possibly units in the pattern that do not have correspondents in the source. For example, the inventory of the learner could contain the noun-phrase construction shown in Figure 8 on the left, with an article, an adjective, and a noun, covering utterances such as 'a nice dinner', but then be confronted with the utterance 'a dinner' which contains fewer units. Resolution in this case consists in leaving out the adjective unit in the generalized construction (as in Figure 9a.).

D. Feature or predicate deletion (shown in WD-8). Not only units but also features can appear in the pattern but not in the source. This blocks match but gets resolved by anti-unification, when the feature is deleted from the generalized pattern. Features can also include predicates such as meets constraints. Feature deletion is illustrated by the example shown in Figure 9. Because the adjective unit is no longer there, the meets constraint between the article and the adjective has become irrelevant and it is therefore eliminated.

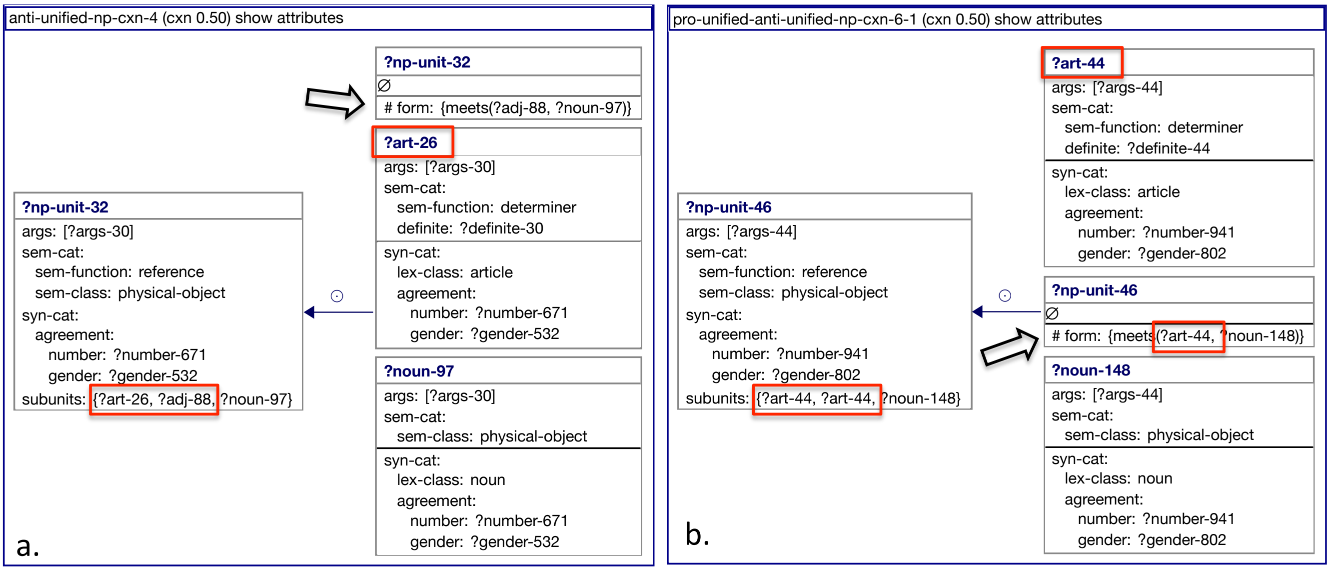

Figure 9. Figure a. (on the left - WD-8 in the web demo) shows the anti-unification of the np-construction shown earlier in Figure 8 a. It is now generalized to cope with a situation where a unit (namely the adjective unit) is absent. The mapping from the original construction schema to this generalization is ?np-unit (in Fig 8 a.) to ?np-unit-32 in Fig **??** a., ?noun to ?noun-97, ?art to ?art-26, and ?adj to a dummy unit ?adj-88. We see that the adjective unit has been deleted. Note that one of the meets-constraints, namely meets(?art,?adj) has been deleted as well. Figure b. (on the right - WD-10 in the web demo) shows the result of pro-unification of the construction schema on the left (Fig a.) (as discussed below). The mapping from the anti-unified to the pro-unified construction schema is: ?np-unit-32 to ?np-unit-46, ?noun-97 to ?noun-148, ?art-26 to ?art-44. Occurrences of ?adj-88 have been replaced by ?adj-44. The inverse order noun / article is no longer possible.

It was mentioned earlier that there are usually different possible pairings between a partially matching construction and the transient structure and that a cost-function computes which alternative might heuristically be the best one to pursue first. In descending order of penalty, there is a cost for deletion of a unit, mismatch of a unit, deletion of a feature, deletion of a negated feature, decoupling of a variable, and variable relaxation.

2.3 Pro-unification

Anti-unification works by neutralizing aspects of a construction schema that are violated by a transient structure: missing units, incompatibilities in feature values of units, violations of ordering constraints, etc. It returns a new generalized construction schema (the least general generalization). When the generalization is applied, it can still deduce many properties of the utterance that are often enough to push the parsing process forward. However, the generalized construction schema is often too general to be included as such in the learner's construction inventory. For example, an ordering constraint may be neutralized through anti-unification but the constituent ordering present in the transient structure should be preserved in the new construction. Pro-unification constrains the generalized construction schema to avoid over-generalization.

The pro-unification algorithm implemented here is quite straightforward: it matches the generalized construction schema against the transient structure to obtain a set of bindings. It then looks whether there are variables bound to the same constants in the transient structure and makes these variables equal by replacing them with a new variable, thus obtaining a new pro-unified construction schema. Here are two clarifying examples (also shown in the web demonstration). They build further on examples given earlier.

Example 6. Consolidating a new constituent order. (shown in Figure 10 and WD-9). Recall Example 2, which illustrates how a new constituent ordering, namely "un dîner formidable" (lit. "a dinner splendid"), can be handled by relaxing the meets-constraints of the noun-phrase unit in a noun-phrase construction. As shown in Figure 8 b., this was done by decoupling variables. The ?adj variable in the original construction has been decoupled into ?adj-324 and ?adj-308 and the ?noun variable into ?noun-302 and ?noun-318. Pro-unification couples them based on which constituent ordering occurs in the transient structure.

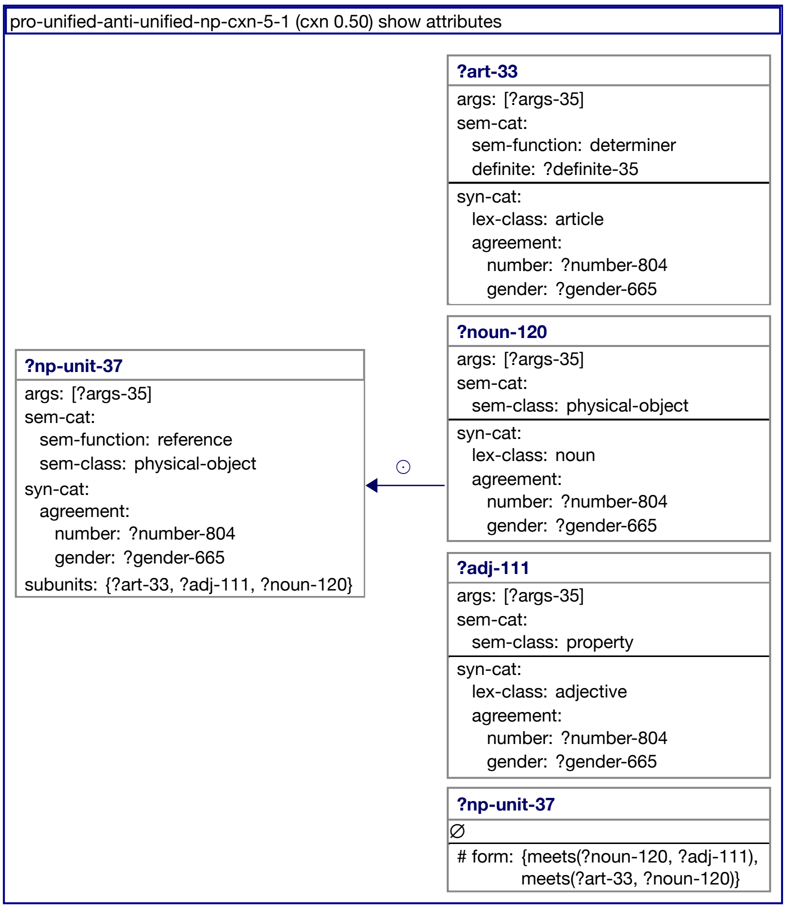

Figure 10. (WD-9) The outcome of using pro-unification with the generalized construction schema shown earlier in Figure 8b. We note that the variables in the meets-constraints, namely ?noun-120, ?adj-111, and ?art-33 are now equal to the units defined for article, adjective and noun, thus reinforcing the article-noun-adjective ordering observed in the transient structure.

What is particularly interesting and powerful is that the construction schema learned through pro- and anti-unification is not only usable in parsing but also in production, due to the bi-directional usability of FCG construction schemas. This is illustrated in WD-10. The variables in the semantic network derived from "un dîner formidable" are instantiated with (Skolem) constants yielding the semantic network 'quality(splendid, o-1), status(indefinite, o-1), meal(dinner,o-1)'. When the available grammar is applied to this network, two utterances are produced: "un dîner formidable" and "un formidable dîner", which are both permissible in French.

Example 7. Leaving out a unit. (See webdemo WD-10). Recall Example 4, which illustrates how anti-unification may remove a unit that was obligatory in the original construction schema, but in the process also eliminates other constraints, such as ordering. Figure 9b. shows the result of pro-unification, which re-instates the ordering constraints present in the transient structure by the same mechanism as in the previous example, i.e. substitution of variables bound to the same constant with a single variable. WD-10 also shows that the acquired construction schema can immediately be used in utterance production as well.

We stress that the current proposal is not the last word on operationalizing the consolidation phase of insight learning. On the one hand, insight learning is also possible with information gleaned from the context or from general world knowledge. On the other hand, we have already identified further extensions of the pro-unification algorithm which we will report in forthcoming papers.

3 RESULTS

The web demonstration shows that the pro- and anti-unification operators are computationally viable and a powerful mechanism to implement insight problem solving and insight learning. We have conducted larger scale experiments that exercise this implementation. Here, we just look at one illustrative example that uses a small French grammar fragment with 21 verbs, 74 nouns, 95 adjectives, 4 articles, and construction schemas for a masculine noun-phrase with an article, adjective and noun, and a transitive clause, with a subject, transitive verb, and direct object, so that utterances such as: “le petit garçon mange un bon repas” (the small boy eats a good meal) can be parsed.

Then we supply three data sets:

- Set-1: a set of grammatical utterances that can be parsed with the initial construction inventory without requiring additional learning.
- Set-2: a set of utterances that is grammatical (in French) but new to the learner and hence requires learning. It contains utterances such like “la petite fille regarde la vieille dame” (lit: ‘the little girl looks (at) the old woman’), which requires an extension of the NP construction which initially can only handle masculine, or ”le film a une fin apocalyptique” (lit: ‘the movie has an ending apocalyptic’), which requires learning adjectives in postposition and learning noun phrases consisting of only an article and a noun.
- Set-3: a set of ungrammatical utterances (for French) that should not be parsable. It contains utterances such as ”le joueur marque une but formidable” (lit: ‘the player marks a goal splendid’), which has a violation of grammatical agreement between the article “une” and the noun “but” (goal).

We have carried out an experiment with three conditions, namely using unification, unification + anti-unification and using unification + anti-unification + pro-unification. We report the percentage of correct sentences for each of the conditions on each of the test sets in Figure 11 and more details are shown as WD-11 in the web demo.

We can see that using unification, only the correct utterances (set-1) can be parsed. Combining unification and anti-unification, the novel grammar in the utterances in set-2 can be acquired through insight learning up to 80 %, but there is massive overgeneralization, so that 60 % of ungrammatical sentences get parsed as well. Using unification, anti-unification and pro-unification, we see that the percentages are the same for set-1 and set-2, but there is no overgeneralization anymore.

The 20 % failure in learning sentences from set-2 shows that the mechanisms discussed in this paper cannot handle all possible cases, but this is as expected. Anti-unification, as operationalized here, can only handle repairs due to a matching conflict in a single construction schema. It does not handle cases

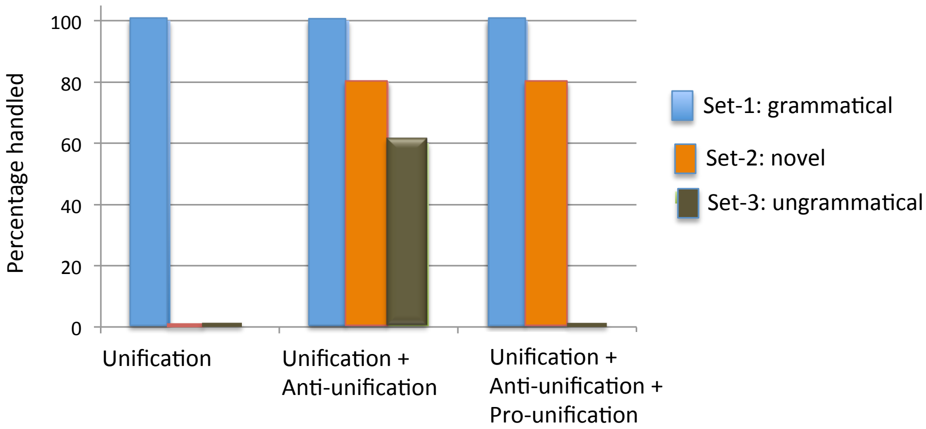

Figure 11. (WD-11) Experimental result with a set of grammatical utterances (set-1), a set of utterances with novel grammar for the learner (set-2), and a set of ungrammatical utterances (set-3). Three configurations have been tested. (i) *Unification.* No learning takes place. All utterances in set-1 can be parsed, those in set-2 and set-3 cannot be handled, i.e. an impasse is reached. (ii) *Unification and anti-unification.* All utterances in set-1 can be parsed, those in set-2 can be acquired through insight learning up to 80 %, but there is massive overgeneralization so that 60 % of ungrammatical sentences get parsed as well. (iii) *Unification, anti-unification and pro-unification* are all used. We see that the percentages are the same for set-1 and set-2 but overgeneralization has dropped because parsing of the utterances in set-3 drops to 0 %.

where several construction schemas, when merged in a single construction schema, might be able to parse the utterance, as for example, "un beau dîner formidable", which combines an NP-construction with an adjective in preposition and one with an adjective in postposition.

4 CONCLUSION

This paper studied mechanisms for insight grammar learning. Insight learning requires first the capacity for insight problem solving, which is to be triggered when a routine solution to a problem is not available. In the case of grammar, this means that parsing is halted because there is no construction schema that matches completely with the transient structure, or when there is some other impasse, such as incompatibility between the semantic network extracted so far from the utterance and the context. Insight problem solving requires a meta-cognitive layer which runs diagnostics to detect the nature of the impasse and repair strategies to try and resolve the problem. Insight problem solving is often needed in normal language usage because of ungrammaticalities, incomplete fragments, and speaker innovations.

We have shown that anti-unification is a very general operator that is useful in repairing an impasse. Anti-unification weakens the constraints of a construction schema so that it becomes applicable to a transient structure. We do not argue that this is the only mechanism needed for repairing an impasse. If a world model or abundant common sense or task knowledge is available, then this is usually a better approach. However, anti-unification is useful when these sources of knowledge are *not* available, or cannot be accessed because not enough of a connected semantic network could be drawn from the utterance to attempt interpretation within the current context.

Insight learning happens when the learner consolidates the result of insight problem solving, which means that a new grammatical construction is built and added to the learner's inventory. Usually the outcome of anti-unification is too general for this purpose. Therefore, we proposed here a novel mechanism, called pro-unification, that specializes a construction generalized through anti-unification so that it re-integrates properties of the current case and hence avoids that the new construction is too general.

5 CONFLICT OF INTEREST

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

LS and PVE jointly developed the theoretical basis of the paper. PVE took the lead for the implementation of pro- and anti-unification and integration within the Fluid Construction Grammar framework. LS took the lead for the writing the paper.

FUNDING

This paper was funded by the EU Project Insight (for LS) and the Marie Curie Initial Training Network Essence (for PVE).

ACKNOWLEDGMENTS

The authors gratefully acknowledge institutional support from ICREA and the Institute for Evolutionary Biology (UPF-CSIC) in Barcelona and the Wissenschaftskolleg in Berlin for LS and from the Sony Computer Science Laboratory in Paris and the VUB Artificial Intelligence Laboratory in Brussels for PVE. We are also strongly indebted and grateful to the many researchers who contributed to the design and implementation of Fluid Construction Grammar, and for technical discussions about insight grammar learning with Katrien Beuls, Emilia Garcia-Casademont, Michael Spranger and Remi van Trijp in particular.

SUPPLEMENTAL DATA

A complete web demonstration of all mechanisms discussed in the paper is provided through: https://www.fcg-net.org/demos/frontiers-demo/.

REFERENCES

Anderson, J. R., Anderson, J. F., Ferris, J. L., Fincham, J. M., and Jung, K. J. (2009). The lateral inferior prefrontal cortex and anterior cingulate cortex are engaged at different stages in the solution of insight problems. *PNAS* 106 (26), 10799–10804

Beuls, K., van Trijp, R., and Wellens, P. (2012). Diagnostics and repairs in fluid construction grammar. In *Language Grounding in Robots*, eds. L. Steels and M. Hild (New York: Springer Verlag). 215–234

Dingemanse, M. e. a. (2015). Universal principles in the repair of communication problems. *Plos One* 10(9, e0136100

Garcia-Casademont, E. and Steels, L. (2016). Grammar learning as insight problem solving. *The Journal of Cognitive Science* 5(7), 27–62

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

Kay, M. (1984). Functional unification grammar: A formalism for machine translation. In *Proceedings of the International Conference of Computational Linguistics* (Association for Computational Linguistics), 75–78

Laird, J. (2012). *The SOAR cognitive architecture* (Cambridge Ma: The MIT Press)

Martelli, A. and Montanari, U. (1982). An efficient unification algorithm. *Transactions on Programming Languages and Systems (TOPLAS)* 4(2), 258–282

Newell, A. and Simon, H. (1972). *Human problem solving* (Englewood Cliffs, NJ: Prentice Hall)

Ohlsson, S. (1984). Restructuring revisited: Ii. an information processing theory of restructuring and insight. *Scandinavian Journal of Psychology* 25, 117–129

Plotkin, G. (1971). A further note on inductive generalization. In *Machine Intelligence 6*, eds. B. Melzer and D. Michie (Edinburgh: Edinburgh University Press). 101–124

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar* (Chicago: University of Chicago Press)

Spranger, M. (2016). *The evolution of grounded spatial language.* (Berlin: Language Science Press)

Steels, L. (ed.) (2011). *Design Patterns in Fluid Construction Grammar* (Amsterdam: John Benjamins)

Steels, L. (ed.) (2012). *Computational Issues in Fluid Construction Grammar*, vol. Lecture Notes in AI, 7249 (New York: Springer Verlag)

Strzalkowski, T. (1994). *Reversible Grammar in Natural Language Processing* (Amsterdam: Kluwer Academic Publishers)

Taatgen, N. and Anderson, J. (2008). Act-r. In *In: Constraints in Cognitive Architectures* (Cambridge University Press)