# Macro-operators for the Emergence of Construction Grammars

Luc Steels
Sony Computer Science Laboratory - Paris
VUB AI Laboratory - Brussels

November 16, 2004

**Abstract**

The paper documents a number of macro-operators that agents can use in the invention and acquisition of grammatical constructions and gives examples of their application.

## 1 Introduction

The Fluid Construction Grammar framework provides a general theory on the representation of linguistic inventories and how they can be used in the parsing and production of sentences [2, 4]. This paper focuses on how rules can be invented (as speaker) or adopted (as hearer). More specifically, we look at a number of possible high level operators that could be used by agents to expand their inventories (lexicon and grammar). Pioneering work in this area has been done by Nancy Change [1].

We use a constructivist approach, which means that agents introduce various syntactic and semantic categories, which progressively take on specific roles in the emerging language system. Parts of speech (noun, verb, etc.), grammatical relations (subject, direct object, etc.), or syntactic features (masculine, feminine, etc.) are not given in advance. Semantic frames (such as cause-transfer), semantic roles (agent, patient, etc.), or semantic features constraining roles in frames, are not given in advance either. They also have to emerge as a side effect of grammar construction.

This paper discusses the following macro-operators:

1. Lexicalisation:
   (a) Add a new word as hearer (adoption)
   (b) Add a new word as speaker (invention)
2. Grammaticalisation I. Invention:
   (a) Add a new construction as hearer (adoption)
   (b) Add a new construction as speaker (invention)

3. Grammaticalisation II. Coercion:

   (a) Fit an element within an existing construction as hearer

   (b) Fit an element within an existing construction as speaker

It turns out that the operators basically collapse to three because the mechanisms for adding a word or construction or for fitting an element in an existing construction are almost completely the same for speaker and hearer.

The paper is only a first step towards a comprehensive theory of how construction grammars may emerge in populations of agents, and the proposed operators have many limitations (discussed in the final section of this paper). Nevertheless, all the examples discussed here are operational and it is only by building in a step-wise fashion on this implementation that we can make solid progress.

The remainder of this paper assumes that the reader is familiar with the basic principles and representational conventions of Fluid Construction Grammars (see [3] and resources available through "http://arti.vub.ac.be/FCG/"). The meanings are represented as a conjunction of clauses in first order predicate calculus. The text first describes each operator and then presents an example of its use. When new rules are constructed new symbols are created which are numbered, as in ?unit-15, syn-cat-34, sem-rule-23, etc. These symbols have no specific meaning except the one that they get in the total system.

# 2 Lexicalisation

Lexicalisation means that a new word (lexical unit) is introduced in the lexical inventory of speaker or hearer. For the hearer, lexicalisation is triggered when there is a word for which there is no lexical entry yet. The hearer then reconstructs the potential meaning and associates it with the unknown word. For the speaker, lexicalisation is triggered when there are parts of the target meaning $M$ which are not yet covered by any word and so the speaker introduces a new word to cover them.

## 2.1 Lexicalisation by Hearer

The first lexicalisation-operator is used by the hearer to add a new word to his lexicon. It involves a preparatory step where the hearer detects the unknown word and reconstructs its potential meaning and then a step where the necessary new rules are constructed. These rules are then applied.

### 2.1.1 Preparation

`Detecting an unknown word (hearer)`

Given a sentence S and a target meaning M.

1. Parse S and extract the parsed meaning $P_m$.

2. Compute the uncovered part of target meaning $U_m$. This is done by taking the difference between the clauses in $M$ and those in $P_m$: $U_m = M \backslash P_m$.

3. Compute uncovered part of the sentence $U_f$. This is done by retrieving the unit in the syntactic structure that has no associated meaning specification in the semantic structure (which is the case because there was no lexical rule to contribute this meaning).

The hearer now hypothesises that $U_f$ expresses $U_m$.

Note: The implementation assumes at the moment that the whole uncovered meaning is associated with the single unknown word in the sentence.

### 2.1.2 New rules

Two new rules are constructed: (1) A new morph-rule associates the word string with a stem. Later on this rule may receive additional syntactic categories. For the time being, the stem and the word string are assumed to be identical. (2) A new lex-stem-rule associates the stem with the uncovered meaning part.

Note: The referent is by default the first argument of the first uncovered predicate.

When the rules have been constructed, the syntactic and semantic structure can be expanded in order to reflect the application of these rules:

1. The stem specification is added by the new morph-rule to the syntactic unit that contains the unknown word string.

2. The referent and meaning is added by the new lex-stem rule to the corresponding semantic unit.

### 2.1.3 Example

Suppose the hearer gets the sentence "Jack gives Jill block" which derenders into:

```
unit1
|   syn-cat: (sentence)
|   form:
|     {pattern(unit2,unit3,unit4,unit5)}
|   syn-subunits: {unit2,unit3,unit4,unit5}
unit2
|   form: {string(unit2,jack)}
unit3
|   form: {string(unit3,gives)}
unit4
|   form: {string(unit4,jill)}
unit5
|   form: {string(unit5,block)}
```

Assuming that there are already lexical entries for jack, jill and block, we get the following syntactic and semantic structure:

```
unit3
   form: {string(unit3,gives)}
unit1
   syn-cat: (sentence)
   form:
      {pattern(unit2,unit3,unit4,unit5)}
   syn-subunits: {unit2,unit3,unit4,unit5}
unit2
   form:
      {stem(unit2,jack), string(unit2,jack)}
unit4
   form:
      {stem(unit4,jill), string(unit4,jill)}
unit5
   form:
      {stem(unit5,block), string(unit5,block)}
```

```
unit1
   sem-subunits: {unit2,unit3,unit4,unit5}
unit3
unit2
   referent: ?x-77
   meaning: {jack(?x-77)}
unit4
   referent: ?x-78
   meaning: {jill(?x-78)}
unit5
   referent: ?x-79
   meaning: {block(?x-79)}
```

The target meaning is:
{give(ev1,true), give-1(ev1,obj1), give-2(ev1,obj2), give-3(ev1,obj3),
block(obj2), jill(obj3), jack(obj1)}
The sentence meaning $P_m$ extracted from the semantic structure so far is:

{block(?x-79), jill(?x-78), jack(?x-77)}

The uncovered part of the target meaning $U_m$ is therefore:

$U_m$ = {give-3(ev1,obj3), give-2(ev1,obj2), give-1(ev1,obj1), give(ev1,true)}

The uncovered part of the sentence $U_f$ is the string "gives" in unit3 because the corresponding unit in the semantic structure does not have any associated meaning after lexicon lookup:
$U_f$ = "gives"

This leads now to two new rules. A morph-rule for the word stem (which is initially hypothesised to be equal to the string itself) and a lex-stem-rule which associated the meaning with this stem:

**morph-rule** gives
  ?unit
    form: stem(?unit,"gives")
$\Longleftrightarrow$
  ?unit
    form: string(?unit,"gives")

**lex-stem-rule** lex-give-3
  ?unit
    referent: ?ev1-82
    meaning:
      give-3(?ev1-82,?x-78), give-2(?ev1-82,?x-79),
      give-1(?ev1-82,?x-77), give(?ev1-82,?true-83)
$\Longleftrightarrow$
  ?unit
    form: stem(?unit,gives)

The effect of these rules is now applied giving the following syntactic structure:

unit3
  form: {string(unit3,gives), stem(unit3,gives)}
unit1
  syn-cat: (sentence)
  form:
    {pattern(unit2,unit3,unit4,unit5)}
  syn-subunits: {unit2,unit3,unit4,unit5}
unit2
  form:
    {stem(unit2,jack), string(unit2,jack)}
unit4
  form:
    {stem(unit4,jill), string(unit4,jill)}
unit5
  form:
    {stem(unit5,block), string(unit5,block)}

The new semantic structure is now:

unit1
  sem-subunits: {unit2,unit3,unit4,unit5}
unit3
  referent: ?ev1-83
  meaning: {give-3(?ev1-83,?x78-1), give-2(?ev1-83,?x-79-1),
    give-1(?ev1-83,?x-77-1), give(?ev1-3,?true-83-1)}
unit2
  referent: ?x-77
  meaning: {jack(?x-77)}
unit4
  referent: ?x-78
  meaning: {jill(?x-78)}
unit5
  referent: ?x-79
  meaning: {block(?x-79)}

Note that the variables in the give-unit are **not** the same as those used in the other units.

## 2.2 Lexicalisation by Speaker

The second lexicalisation-operator is used by the speaker to add a new word to his lexicon. It involves a preparatory step where the speaker detects the unknown word and reconstructs its potential meaning and then a step where the necessary new rules are constructed. These rules are then applied.

### 2.2.1 Preparation

`Detecting an unknown word (hearer)`

Given a sentence S and a target meaning M, produce S and extract $U_m$ as the part of the meaning which is not covered by any word. The speaker now invents a new word $N_f$ and creates rules so that $N_f$ expresses $U_m$.

Note: The implementation assumes at the moment that the whole uncovered meaning is associated with the single unknown word in the sentence.

### 2.2.2 New Rules

Two new rules are constructed: (1) A new morph-rule associates the word string with a stem. Later on this rule may receive additional syntactic categories. For the time being, the stem and the word string are assumed to be identical. (2) A new lex-stem-rule associates the stem with the uncovered meaning part.

Note: The referent is by default the first argument of the first uncovered predicate.

When the rules have been constructed, the syntactic and semantic structure can be expanded in order to reflect the application of these rules:

1. The stem specification is added by the new morph-rule to the syntactic unit that contains the unknown word string.

2. The referent and meaning is added by the new lex-stem rule to the corresponding semantic unit.

### 2.2.3 Example

Suppose that the target meaning $M$ is equal to:

give(ev1,true), give-1(ev1,obj1),give-2(ev1,obj2), give-3(ev1,obj3), block(obj2), jill(obj3), jack(obj1)

Suppose that there are already words in the lexicon to cover the predicates block, jack and jill but not yet for the other predicates. After application of all lexical rules, the following syntactic and semantic structures are built:

```
unit-block-entry
   form:
      {string(unit-block-entry,block),
       stem(unit-block-entry,block)}
unit-jill-entry
   form:
      {string(unit-jill-entry,jill),
       stem(unit-jill-entry,jill)}
unit-jack-entry
   form:
      {string(unit-jack-entry,jack),
       stem(unit-jack-entry,jack)}
unit1
   syn-subunits: {unit-block-entry, unit-jill-entry,unit-jack-entry}
```

```
unit1
   sem-subunits: {unit-block-entry, unit-jill-entry,unit-jack-entry
   referent: ev1
   meaning:
      {give(ev1,true), give-1(ev1,obj1),
       give-2(ev1,obj2), give-3(ev1,obj3)}
unit-block-entry
   referent: obj2
   meaning: {block(obj2)}
unit-jill-entry
   referent: obj3
   meaning: {jill(obj3)}
unit-jack-entry
   referent: obj1
   meaning: {jack(obj1)}
```

The uncovered meaning is equal to

$$U_M = \text{give(ev1,true), give-1(ev1,obj1),give-2(ev1,obj2), give-3(ev1,obj3)}$$

and a new word needs to be invented to cover this expression. Let us suppose that the agent picks the word "divira".

We get therefore the following morph-rule:

**morph-rule** morph-divira
  ?unit-28
    form: stem(?unit-28, "divira")
$\Longleftrightarrow$
  ?unit-28
    form: string(?unit-28,"divira")

And the following lex-stem-rule which associates the uncovered meaning with this stem:

**lex-stem-rule** lex-stem-divira
  ?unit-28
    referent: ?x-27
    meaning:
      give(?x-27,?true-29), give-1(?x-27,?obj1-30),
      give-2(?x-27,?obj2-31), give-3(?x-27,?obj3-32)
$\Longleftrightarrow$
  ?unit-28
    form: stem(?unit-28,"divira")

# 3 Grammaticalisation I. Invention

A new construction involves the introduction of a new semantic frame and a new syntactic pattern as well as a mapping rule linking the two. New constructions are triggered when there are variable equalities after matching the parsed meaning against the target meaning, implying that different variables are bound to the same object and hence interpretation could become more efficient if this information is known beforehand. If the variable equalities are non-nil, we say that there is a residu of semantic uncertainty that must be resolved.

We first consider the case where speaker or hearer do not have a construction yet that could be adapted for the present purpose and so they make a new one. At present it is assumed that a single construction will cover all equalities.

## 3.1 Construction added by the hearer (adoption)

### 3.1.1 Preparation

Given a sentence S and a target meaning M.

1. Parse S and extract the parsed meaning $P_m$.
2. Match $P_m$ against the target meaning $U_m$, obtaining a set of bindings $B$, and a set of equalities $E$, equal to the set of variables that are bound in $B$ to the same object.
3. if $E \neq \emptyset$ collect the set of all those units that include one of the variables in $E$ as part of their meaning. The substructure of the semantic structure made up these

units will act as the basis for the semantic frame, i.e. left pole, of the construction. The substructure of the syntactic structure made up of the same units will act as the basis for the syntactic pattern, i.e. the right pole, of the construction.

The hearer now proceeds to construct all the necessary rules for this construction and its preparation in the form syntactic and semantic categorisation rules.

### 3.1.2 New rules

**1. The Construction**

The left pole of the construction is built as follows:

1. Given the semantic substructure consisting of all the units from the semantic structure that should participate in the left pole of the construction.
2. Standardise the variables that are equal so that they become a single variable.
3. Introduce variables for the units
4. Create an overarching unit which contains all participating units as sem-subunits. It is further called the top-unit.
5. For each unit convert the meaning of that unit into an analogous set of semantic categories and construct new semantic categorisations linking each meaning to its corresponding semantic categories through a sem-rule.

Construction of the right pole involves the following steps:

1. The equivalent of all the units on the semantic pole is used as the basis for the syntactic pole and a new syntactic category for the top unit is introduced.
2. New grammatical relations are constructed for each participating unit, linking that unit with the top unit of the construction.

**2. Semantic Categorisation rules**

For each unit in the semantic structure, a number of semantic categories were introduced to build the semantic pole of the construction. Now the required sem-rules have to be built where the predicates in the meaning slot forming the left pole of the sem-rule are linked to their respective semantic categories as right pole of the sem-rule.

**3. Phrase structure rule**

Now the phrase structure rule is built. Its left pole is identical to the syntactic pattern of the construction rule. The right pole contains the constraints on the pattern. These include first of all syntactic categorisations (parts of speech) for the possible units but could include also word order, agreement, or any other syntactic feature. At this point we use only syntactic categories as restrictions. These categories are newly created.

**4. Morph rules** The morph-rules of the words involved have to be expanded, specifying that the words now belong to the syntactic categories introduced in the phrase structure rule.

## 3.2 Example

Suppose we start from the sentence "Jack gives Bill block", assuming that there are lexical entries for each individual word but no construction yet. After parsing the syntactic and semantic structures are then as follows:

```
unit5
    form:
        {stem(unit5,block), string(unit5,block)}
unit4
    form:
        {stem(unit4,jill), string(unit4,jill)}
unit2
    form:
        {stem(unit2,jack), string(unit2,jack)}
unit1
    syn-cat: (sentence)
    form:
        {pattern(unit2,unit3,unit4,unit5)}
    syn-subunits: unit2(unit3,unit4,unit5)
unit3
    form:
        {string(unit3,gives), stem(unit3,gives)}
```

```
unit5
    referent: ?x-426
    meaning: {block(?x-426)}
unit4
    referent: ?x-425
    meaning: {jill(?x-425)}
unit2
    referent: ?x-424
    meaning: {jack(?x-424)}
unit1
    sem-subunits: unit2(unit3,unit4,unit5)
unit3
    referent: ?ref-427
    meaning:
        {give(?ref-427,?x-428), give-1(?ref-427,?x-429),
         give-2(?ref-427,?x-430), give-3(?ref-427,?x-431)}
```

The parsed meaning $P_m$ is:

{block(?x-426), jill(?x-425), jack(?x-424),
give(?ref-427,?x-428), give-1(?ref-427,?x-429),
give-2(?ref-427,?x-430), give-3(?ref-427,?x-431)}

matching against this yields the set of bindings

$B$ = ((?x-424 . obj1) (?x-425 . obj3) (?x-426 . obj2) (?x-431 . obj3) (?x-430 . obj2) (?x-429 . obj1) (?x-428 . true) (?ref-427 . ev1))

and hence the equalities:

$E$ = {{?x-424, ?x-429}, {?x-426, ?x-430}, {?x-425,?x-431}}

**1. Building the construction**

The first step is to assemble all the units that will be in the syntactic and semantic pole and group them together in a superunit. Intuitively speaking, those units should be integrated that need to be connected to each other. In the present case, this means unit3 (because of ?x-429, ?x-430 and ?x-431), unit2 (because of ?x-424), unit4 (because of ?x-425), and unit5 (because of ?x-426). After standardising the variables from the equalities and introducing new variables for the units we get:

?unit-432
  referent: ?ref-427
  meaning:
    give(?ref-427,?x-428), give-1(?ref-427,?x-424),
    give-2(?ref-427,?x-426), give-3(?ref-427,?x-425)
?unit-433
  referent: ?x-424
  meaning: jack(?x-424)
?unit-434
  referent: ?x-425
  meaning: jill(?x-425)
?unit-435
  referent: ?x-426
  meaning: block(?x-426)

The next step is to introduce semantic categories for the relevant predicates, and add a new unit that groups all of the others together:

```
?unit-432
    referent: ?ref-427
    sem-cat:
        sem-cat-268(?ref-427,?x-428), sem-cat-269(?ref-427,?x-424),
        sem-cat-270(?ref-427,?x-426), sem-cat-271(?ref-427,?x-425)
?unit-433
    referent: ?x-424
    sem-cat: sem-cat-272(?x-424)
?unit-434
    referent: ?x-425
    sem-cat: sem-cat-273(?x-425)
?unit-435
    referent: ?x-426
    sem-cat: sem-cat-274(?x-426)
?unit-436
    sem-subunits:
        ?unit-432,?unit-433,?unit-434,?unit-435
```

Now we focus on constructing the right pole. For each of the units in the semantic pole, a unit is introduced in the syntactic pole and a new syntactic category (a grammatical relation) is constructed between this unit and the top unit of the construction. There is also a new overall syntactic category (syn-cat-310) for the construction as a whole. This gives the following:

```
?unit-432
    syn-cat:
        syn-cat-312(?unit-436,?unit-432)
?unit-433
    syn-cat:
        syn-cat-314(?unit-436,?unit-433)
?unit-434
    syn-cat:
        syn-cat-316(?unit-436,?unit-434)
?unit-435
    syn-cat:
        syn-cat-318(?unit-436,?unit-435)
?unit-436
    syn-cat: syn-cat-310
    syn-subunits:
        ?unit-432,?unit-433,?unit-434,
        ?unit-435
```

The left pole and right pole can then be brought together in the construction itself, which looks as follows:

**map-rule** construction-18
?unit-432
referent: ?ref-427
sem-cat:
sem-cat-268(?ref-427,?x-428), sem-cat-269(?ref-427,?x-424),
sem-cat-270(?ref-427,?x-426), sem-cat-271(?ref-427,?x-425)
?unit-433
referent: ?x-424
sem-cat: sem-cat-272(?x-424)
?unit-434
referent: ?x-425
sem-cat: sem-cat-273(?x-425)
?unit-435
referent: ?x-426
sem-cat: sem-cat-274(?x-426)
?unit-436
sem-subunits:
?unit-432,?unit-433,?unit-434,?unit-435
$\Longleftrightarrow$
?unit-432
syn-cat:
syn-cat-312(?unit-436,?unit-432)
?unit-433
syn-cat:
syn-cat-314(?unit-436,?unit-433)
?unit-434
syn-cat:
syn-cat-316(?unit-436,?unit-434)
?unit-435
syn-cat:
syn-cat-318(?unit-436,?unit-435)
?unit-436
syn-cat: syn-cat-310
syn-subunits:
?unit-432,?unit-433,?unit-434, ?unit-435

## 2. Semantic Categorisations

Each of the semantic categories that were introduced needs to be associated with the relevant predicates using a semantic categorisation rule. Here are two examples. One for give and one for jack:

**sem-rule** sem-rule-76
  ?unit-432
    referent: ?ref-427
    meaning:
      give(?ref-427,?x-428), give-1(?ref-427,?x-424),
      give-2(?ref-427,?x-426), give-3(?ref-427,?x-425)
⟹
  ?unit-432
    sem-cat:
      sem-cat-268(?ref-427,?x-428),
      sem-cat-269(?ref-427,?x-424),
      sem-cat-270(?ref-427,?x-426),
      sem-cat-271(?ref-427,?x-425)

**sem-rule** sem-rule-75
  ?unit-433
    referent: ?x-424
    meaning: jack(?x-424)
⟹
  ?unit-433
    sem-cat: sem-cat-272(?x-424)

Note that if there is already a semantic rule for the same left pole, the semantic categories are added to it.

### 3. The Phrase Structure

We now focus on introducing the necessary rules that would make the right pole of this construction active. This requires first of all a phrase structure rule whose left pole is the same as the right pole of the construction. The right pole introduces the syntactic constraints that should be satisfied to trigger this. For the time being, we assume that this is purely based on syntactic categories which each unit (associated with individual words) must satisfy as shown in the rule below.

```
ps-rule ps-syn-cat-310-rule
  ?unit-432
    syn-cat:
      syn-cat-312(?unit-436,?unit-432)
  ?unit-433
    syn-cat:
      syn-cat-314(?unit-436,?unit-433)
  ?unit-434
    syn-cat:
      syn-cat-316(?unit-436,?unit-434)
  ?unit-435
    syn-cat:
      syn-cat-318(?unit-436,?unit-435)
  ?unit-436
    syn-cat: syn-cat-310
    syn-subunits:
      ?unit-432,?unit-433,?unit-434, ?unit-435
⟺
  ?unit-436
    syn-subunits:
      ?unit-432,?unit-433,?unit-434, ?unit-435
  ?unit-432
    syn-cat: syn-cat-322
  ?unit-433
    syn-cat: syn-cat-321
  ?unit-434
    syn-cat: syn-cat-320
  ?unit-435
    syn-cat: syn-cat-319
```

## 4. Morph rules

Now all the categories must be grounded in the observed word forms. Specifically the syntactic categories must be assigned to the words, which is done by expanding the morph-rules for the words involved. Here are two examples:

```
morph-rule gives-entry
  ?unit
    form: stem(?unit,gives)
    syn-cat: syn-cat-326
⟺
  ?unit
    form: string(?unit,gives)
```

```
morph-rule block-entry
   ?unit
      form: stem(?unit,block)
      syn-cat: syn-cat-323
⟺
   ?unit
      form: string(?unit,block)
```

and so on for the other words. Note that if there is already a morph rule, the syntactic categories are added to it.

As a test of all these rules, we parse again the sentence "Jack gives Jill block". This results in the following syntactic and semantic structures:

```
unit3
   sem-cat:
      {sem-cat-275(?x-482,?x-483), sem-cat-276(?x-482,?x-481),
       sem-cat-277(?x-482,?x-479),
       sem-cat-278(?x-482,?x-480)}
   referent: ?x-482
   meaning:
      {give(?x-482,?x-483), give-1(?x-482,?x-481),
       give-2(?x-482,?x-479), give-3(?x-482,?x-480)}
unit2
   sem-cat: {sem-cat-279(?x-481)}
   referent: ?x-481
   meaning: {jack(?x-481)}
unit4
   sem-cat: {sem-cat-280(?x-480)}
   referent: ?x-480
   meaning: {jill(?x-480)}
unit5
   sem-cat: {sem-cat-281(?x-479)}
   referent: ?x-479
   meaning: {block(?x-479)}
unit1
   sem-subunits: {unit2,unit3,unit4,unit5}
```

unit1
  syn-subunits: {unit2,unit3,unit4,unit5}
  syn-cat: syn-cat-327(sentence)
  form:
    {pattern(unit2,unit3,unit4,unit5)}
unit5
  syn-cat:
    {syn-cat-335(unit1,unit5), syn-cat-336}
  form:
    {stem(unit5,block), string(unit5,block)}
unit4
  syn-cat:
    {syn-cat-333(unit1,unit4), syn-cat-337}
  form:
    {stem(unit4,jill), string(unit4,jill)}
unit2
  syn-cat:
    {syn-cat-331(unit1,unit2), syn-cat-338}
  form:
    {stem(unit2,jack), string(unit2,jack)}
unit3
  syn-cat:
    {syn-cat-329(unit1,unit3), syn-cat-339}
  form:
    {string(unit3,gives), stem(unit3,gives)}

The extracted meaning $P_m$ is:

{give(?x-482,?x-483), give-1(?x-482,?x-481), give-2(?x-482,?x-479), give-3(?x-482,?x-480), jack(?x-481), jill(?x-480), block(?x-479)}

when this is matched against the target meaning, we get the following set of bindings:

$B$ = ((?x-480 . obj3) (?x-479 . obj2) (?x-481 . obj1) (?x-483 . true) (?x-482 . ev1))

There are no more equalities in this binding list due to the application of the construction. Note also how all the syntactic and semantic categorisations are present in the syntactic and semantic structures.

## 3.3 Construction added by the speaker (invention)

### 3.3.1 Methods

The way that the speaker adds a new construction is totally analogous to the way that the hearer does it. Of course now the speaker is totally sure about the target meaning because he conceptualised it. After a first attempt at production, the speaker should re-enter the resulting sentence and construct a parse tree from with the parsed meaning

$P_m$ can be extracted. From this the set of bindings and equalities can be extracted and new rules can be built the same was as for the hearer.

After constructing all the rules, the speaker should produce again the desired sentence before sending it to the hearer.

### 3.3.2 Example

Suppose we continue the earlier example where a new word "divira" was constructed to cover part of the target meaning:

> $M$ = give(ev1,true), give-1(ev1,obj1),give-2(ev1,obj2), give-3(ev1,obj3), block(obj2), jill(obj3), jack(obj1)

The sentence is now: "block jill jack divira". The speaker re-enters this sentence and obtains the following semantic and syntactic structures:

```
unit-8
   referent: ?x-36
   meaning:
      {give(?x-36,?x-37), give-1(?x-36,?x-38),
       give-2(?x-36,?x-39), give-3(?x-36,?x-40)}
unit-5
   referent: ?x-35
   meaning: {block(?x-35)}
unit-6
   referent: ?x-34
   meaning: {jill(?x-34)}
unit-7
   referent: ?x-33
   meaning: {jack(?x-33)}
top-unit
   sem-subunits: {unit-8, unit-7,unit-6,unit-5)}
```

```
unit-8
   form:
      {stem(unit-8,divira), string(unit-8,divira)}
unit-5
   form:
      {stem(unit-5,block), string(unit-5,block)}
unit-6
   form:
      {stem(unit-6,jill), string(unit-6,jill)}
unit-7
   form:
      {stem(unit-7,jack), string(unit-7,jack)}
top-unit
   syn-cat: (sentence)
   syn-subunits: {unit-8, unit-7,unit-6,unit-5}
```

After parsing the syntactic and semantic structures are then as follows:

unit5
  form:
    {stem(unit5,block), string(unit5,block)}
unit4
  form:
    {stem(unit4,jill), string(unit4,jill)}
unit2
  form:
    {stem(unit2,jack), string(unit2,jack)}
unit1
  syn-cat: (sentence)
  form:
    {pattern(unit2,unit3,unit4,unit5)}
  syn-subunits: unit2(unit3,unit4,unit5)
unit3
  form:
    {string(unit3,gives), stem(unit3,gives)}

unit5
  referent: ?x-426
  meaning: {block(?x-426)}
unit4
  referent: ?x-425
  meaning: {jill(?x-425)}
unit2
  referent: ?x-424
  meaning: {jack(?x-424)}
unit1
  sem-subunits: unit2(unit3,unit4,unit5)
unit3
  referent: ?ref-427
  meaning:
    {give(?ref-427,?x-428), give-1(?ref-427,?x-429),
     give-2(?ref-427,?x-430), give-3(?ref-427,?x-431)}

There are no unknown words obviously because the speaker produced this sentence himself. The parsed meaning $P_m$ is:

give(?x-36,?x-37), give-1(?x-36,?x-38), give-2(?x-36,?x-39), give-3(?x-36,?x-40), block(?x-35), jill(?x-34), jack(?x-33)

matching this against $M$ yields the set of bindings

((?x-33 . obj1) (?x-34 . obj3) (?x-35 . obj2) (?x-40 . obj3) (?x-39 . obj2) (?x-38 . obj1) (?x-37 . true) (?x-36 . ev1))

and hence the equalities:

((?x-33 ?x-38) (?x-35 ?x-39) (?x-34 ?x-40))

The speaker now constructs rules in the same way as the hearer earlier on. Here is the construction itself:

**map-rule** construction-2
  ?unit-42
    referent: ?x-33
    sem-cat: sem-cat-8(?x-33)
  ?unit-43
    referent: ?x-34
    sem-cat: sem-cat-9(?x-34)
  ?unit-44
    referent: ?x-35
    sem-cat: sem-cat-10(?x-35)
  ?unit-45
    referent: ?x-36
    sem-cat:
      sem-cat-11(?x-36,?x-37), sem-cat-12(?x-36,?x-33),
      sem-cat-13(?x-36,?x-35), sem-cat-14(?x-36,?x-34)
  ?unit-41
    sem-subunits:
      ?unit-42,?unit-43,?unit-44 ?unit-45
$\Longleftrightarrow$
  ?unit-42
    syn-cat:
      syn-cat-16(?unit-41,?unit-42)
  ?unit-43
    syn-cat:
      syn-cat-18(?unit-41,?unit-43)
  ?unit-44
    syn-cat:
      syn-cat-20(?unit-41,?unit-44)
  ?unit-45
    syn-cat:
      syn-cat-22(?unit-41,?unit-45)
  ?unit-41
    syn-cat: syn-cat-14
    syn-subunits:
      ?unit-42,?unit-43,?unit-44,
      ?unit-45

The phrase structure rule that expands the right hand side of the construction:

**ps-rule** ps-syn-cat-14-rule
?unit-42
syn-cat:
syn-cat-16(?unit-41,?unit-42)
?unit-43
syn-cat:
syn-cat-18(?unit-41,?unit-43)
?unit-44
syn-cat:
syn-cat-20(?unit-41,?unit-44)
?unit-45
syn-cat:
syn-cat-22(?unit-41,?unit-45)
?unit-41
syn-cat: syn-cat-14
syn-subunits:
?unit-42,?unit-43,?unit-44, ?unit-45
$\Longleftrightarrow$
?unit-41
syn-subunits:
?unit-42,?unit-43,?unit-44, ?unit-45
?unit-42
syn-cat: syn-cat-26
?unit-43
syn-cat: syn-cat-25
?unit-44
syn-cat: syn-cat-24
?unit-45
syn-cat: syn-cat-23

Then there are various rules which assign syntactic categories to individual words. Here is the example of "divira":

**morph-rule** morph-1
?unit-28
form: stem(?unit-28,"divira")
syn-cat: syn-cat-23
$\Longleftrightarrow$
?unit-28
form: string(?unit-28,"divira")

Finally there are various semantic categorisation rules, such as the one here for jack:

**sem-rule** sem-cat-8
?unit-42
referent: ?x-33
meaning: jack(?x-33)
$\Longleftrightarrow$
?unit-42
sem-cat: sem-cat-8(?x-33)

Or the semantic categorisation rule for the give event:

**sem-rule** sem-cat-5
  ?unit-45
    referent: ?x-36
    meaning:
      give(?x-36,?x-37), give-1(?x-36,?x-33),
      give-2(?x-36,?x-35), give-3(?x-36,?x-34)
$\Longleftrightarrow$
  ?unit-45
    sem-cat:
      sem-cat-11(?x-36,?x-37), sem-cat-12(?x-36,?x-33),
      sem-cat-13(?x-36,?x-35), sem-cat-14(?x-36,?x-34)

# 4 Grammaticalisation II. Coercion

The building of new constructions from scratch is rare. Usually a construction already exists and if a situation arises that requires additional grammar, agents first try to re-use existing constructions by coercing certain predicates and words to fit in, which can be done by introducing additional syntactic and semantic categorisation rules. We consider at this point the simplest example of this type of coercion, and much more research is needed to identify the 'grammar-making operations' that are allowed in a language.

## 4.1 Fit into existing construction (as hearer)

First we consider the case of the hearer. Assume that the hearer gets the utterance "Jack gives Frank block ", and has the give-construction as built in the previous section, which covers most of the sentence, except "Frank". If the inventory is extended in such a way that the semantic and syntactic properties associated with the unit for "Frank" trigger this construction, then a straightforward resolution of semantic uncertainty is achieved. There are two questions: (1) how to find which construction fits best with the incoming sentence, and (2) how to coerce the alien element to fit.

### 4.1.1 Steps in rule construction

1. The first step is similar to the creation of a new construction: Based on the equalities, all those units in the semantic structure are assembled that have variables participating in one of the equalities.
   item Then all rules are examined to find whether there are any construction rules whose left pole matches partially with this substructure. The best matching rule is chosen.

2. Suppose such a rule is found, then this rule tells the agent what the semantic category will be of the unit that did not match and a new sem-rule can be introduced to fix this problem.

These processing steps also help the agent to compute which grammatical relation in the right pole of the rule is missing, and from there the phrase-structure rule can be found

that tells the agent which syntactic conditions (such as syntactic categorisation) has to be satisfied, so that this syntactic category can be added to the morph-rule of the string attached to the same unit.

### 4.1.2 Example

Suppose the hearer has already a construction for "give" as shown earlier, based on the sentence "Jack gives Jill block ". The next sentence is "Jack gives Frank block " covering the target meaning:

> give(ev1,true), give-1(ev1,obj1), give-2(ev1,obj2), give-3(ev1,obj3), block(obj2), frank(obj3), jack(obj1)

The hearer first needs to add a new lexical rule (using the lexicalisation operator) to cover the predicate 'frank(?x)' with the word "frank". Once this is done the syntactic and semantic structure are:

unit3
- syn-cat: (syn-cat-39)
- form:
  - {stem(unit3,gives), string(unit3,gives)}

unit5
- syn-cat: (syn-cat-36)
- form:
  - {stem(unit5,block), string(unit5,block )}

unit2
- syn-cat: (syn-cat-38)
- form:
  - {stem(unit2,jack), string(unit2,jack)}

unit1
- syn-cat: (sentence)
- form:
  - {pattern(unit2,unit3,unit4,unit5)}
- syn-subunits: {unit2, unit3,unit4,unit5}

unit4
- form:
  - {string(unit4,frank), stem(unit4,frank)}

unit3
  sem-cat:
    {sem-cat-15(?x-74,?x-75), sem-cat-16(?x-74,?x-76),
     sem-cat-17(?x-74,?x-77), sem-cat-18(?x-74,?x-78)}
  referent: ?x-74
  meaning:
    {give(?x-74,?x-75), give-1(?x-74,?x-76),
     give-2(?x-74,?x-77), give-3(?x-74,?x-78)}
unit2
  sem-cat: {sem-cat-19(?x-72)}
  referent: ?x-72
  meaning: {jack(?x-72)}
unit5
  sem-cat: {sem-cat-21(?x-73)}
  referent: ?x-73
  meaning: {block(?x-73)}
unit4
  referent: ?ref-79
  meaning: {frank(?ref-79)}
unit1
  sem-subunits: {unit2,unit3,unit4,unit5}

The set of bindings $B$ is equal to:

((?x-72 . obj1) (?ref-79 . obj3) (?x-73 . obj2) (?x-78 . obj3) (?x-77 . obj2) (?x-76 . obj1) (?x-75 . true) (?x-74 . ev1))

and we therefore have the following equalities:

((?x-72 ?x-76) (?x-73 ?x-77) (?ref-79 ?x-78))

The following construction is available to the hearer:

**map-rule** construction-3
  ?unit-55
    referent: ?ref-49
    sem-cat:
      sem-cat-15(?ref-49,?x-50), sem-cat-16(?ref-49,?x-46),
      sem-cat-17(?ref-49,?x-48), sem-cat-18(?ref-49,?x-47)
  ?unit-56
    referent: ?x-46
    sem-cat: sem-cat-19(?x-46)
  ?unit-57
    referent: ?x-47
    sem-cat: sem-cat-20(?x-47)
  ?unit-58
    referent: ?x-48
    sem-cat: sem-cat-21(?x-48)
  ?unit-54
    sem-subunits:
      ?unit-55,?unit-56,?unit-57, ?unit-58
$\Longleftrightarrow$
  ?unit-55
    syn-cat:
      syn-cat-29(?unit-54,?unit-55)
  ?unit-56
    syn-cat:
      syn-cat-31(?unit-54,?unit-56)
  ?unit-57
    syn-cat:
      syn-cat-33(?unit-54,?unit-57)
  ?unit-58
    syn-cat:
      syn-cat-35(?unit-54,?unit-58)
  ?unit-54
    syn-cat: syn-cat-27
    syn-subunits:
      ?unit-55,?unit-56,?unit-57, ?unit-58

With the following ps-rule:

**ps-rule** ps-syn-cat-27-rule
?unit-55
syn-cat:
syn-cat-29(?unit-54,?unit-55)
?unit-56
syn-cat:
syn-cat-31(?unit-54,?unit-56)
?unit-57
syn-cat:
syn-cat-33(?unit-54,?unit-57)
?unit-58
syn-cat:
syn-cat-35(?unit-54,?unit-58)
?unit-54
syn-cat: syn-cat-27
syn-subunits:
?unit-55,?unit-56,?unit-57, ?unit-58
⟺
?unit-54
syn-subunits:
?unit-55,?unit-56,?unit-57,?unit-58
?unit-55
syn-cat: syn-cat-39
?unit-56
syn-cat: syn-cat-38
?unit-57
syn-cat: syn-cat-37
?unit-58
syn-cat: syn-cat-36

However the construction cannnot trigger because the Frank-unit (unit4) does not have the required syntactic category (syn-cat-37) nor the required semantic category (sem-cat-20). The problem is solved with the following two rules:

A sem-rule that categorises Frank as belonging to semantic category sem-cat-20:

**sem-rule** sem-cat-13
?unit
referent: ?ref-79
meaning: frank(?ref-79)
⟺
?unit
sem-cat: sem-cat-20(?ref-79)

The morph-rule for "Frank" is expanded so that it now belongs to syntactic category syn-cat-37:

**morph-rule** frank-morph
  ?unit
    form: stem(?unit,"frank")
    syn-cat: syn-cat-37
⟺
  ?unit
    form: string(?unit,"frank")

## 4.2 Fit into existing construction (as speaker)

### 4.2.1 Methods

The way that the speaker fits a new element into an existing construction is totally analogous to the way that the hearer does it, except for the preparatory work. The speaker re-parses the syntactic structure that was constructed (the structure typically does not reach strings but only stems if constructions cannot apply) and then extracts the set of bindings and the equalities. If there are any equalities, the speaker tries to find the construction that was the closest and uses that as a basis for creating an extension.

### 4.2.2 Example

Here is an example. Suppose that the speaker has already a give construction (as in the previous example) and now wants to express the target meaning, in other words "Jack gives Anita block":

> give(ev1,true), give-1(ev1,obj1), give-2(ev1,obj2), give-3(ev1,obj3), block(obj2), anita(obj3), jack(obj1)

Let us assume that the speaker has lexicalised 'give' as "moruque" and 'anita' as "tyzemy". The existing construction is:

**rule** construction-4
  ?unit-104
    referent: ?x-95
    sem-cat: sem-cat-22(?x-95)
  ?unit-105
    referent: ?x-96
    sem-cat: sem-cat-23(?x-96)
  ?unit-106
    referent: ?x-97
    sem-cat: sem-cat-24(?x-97)
  ?unit-107
    referent: ?x-98
    sem-cat:
      sem-cat-25(?x-98,?x-99), sem-cat-26(?x-98,?x-95),
      sem-cat-27(?x-98,?x-97), sem-cat-28(?x-98,?x-96)
  ?unit-103
    sem-subunits:
      ?unit-104,?unit-105,?unit-106, ?unit-107
$\Longleftrightarrow$
  ?unit-104
    syn-cat:
      syn-cat-42(?unit-103,?unit-104)
  ?unit-105
    syn-cat:
      syn-cat-44(?unit-103,?unit-105)
  ?unit-106
    syn-cat:
      syn-cat-46(?unit-103,?unit-106)
  ?unit-107
    syn-cat:
      syn-cat-48(?unit-103,?unit-107)
  ?unit-103
    syn-cat: syn-cat-40
    syn-subunits:
      ?unit-104,?unit-105,?unit-106, ?unit-107

After producing the sentence, the speaker obtains the following syntactic structure:

```
unit1
|  syn-subunits: {unit-lex-stem-3, unit-lex-stem-2,unit-block-entry, unit-jack-entry}
unit-block-entry
|  form: {stem(unit-block-entry,block)}
unit-jack-entry
|  form: {stem(unit-jack-entry,jack)}
unit-lex-stem-2
|  form: {stem(unit-lex-stem-2,moruqe)}
unit-lex-stem-3
|  form:
|     {string(unit-lex-stem-3,tyzemy),
|      stem(unit-lex-stem-3,tyzemy)}
```

Because the construction did not trigger, there are no syntactic categories for its components, and hence the morph-rules are not triggered, except for "tyzemy" which does not have any syntactic categories yet. This syntactic structure is then re-entered to yield the following meaning:

jack(?x-110), block(?x-111), give(?x-112,?x-113), give-1(?x-112,?x-114), give-2(?x-112,?x-115), give-3(?x-112,?x-116), anita(?x-117)

and hence the set of bindings:

$B$ = ((?x-110 . obj1) (?x-117 . obj3) (?x-111 . obj2) (?x-116 . obj3) (?x-115 . obj2) (?x-114 . obj1) (?x-113 . true) (?x-112 . ev1))

and equalities:

((?x-110 ?x-114) (?x-111 ?x-115) (?x-117 ?x-116))

Using the same process as the hearer in the previous example, the speaker discovers that the construction becomes applicable on the semantic side when the following sem-rule is added:

```
sem-rule anita-sem-cat-23
  ?unit
    referent: ?x-117
    meaning: anita(?x-117)
⟺
  ?unit
    sem-cat: sem-cat-23(?x-117)
```

And the morph-rule for "tyzemy" is expanded:

```
morph-rule morph-3
  ?unit-109
    form: stem(?unit-109,tyzemy)
    syn-cat: syn-cat-51
⟺
  ?unit-109
    form: string(?unit-109,tyzemy)
```

# 5 Conclusions

We discussed a number of macro-operators for learning construction grammars. These macro-operators are only the very first illustration of this process and much remains to be done to make them more powerful. More 'syntactic sugar' needs to be introduced by the grammaticalisation operators, specifically in the phrase structure rules. Also the criteria for re-using a construction should become much more sophisticated. Ideally all this should be done on the basis of meta-rules which specify how - in specific circumstances - the grammar needs to be expanded.

**Acknowledgement**

This work was funded by the Sony Computer Science Laboratories in Paris, the FET ECAgents Projects through Sony CSL, and the ESF OMLL project through the VUB AI Laboratory. Many technical contributions by Joachim de Beule and Nicolas Neubauer are gratefully acknowledged.

# References

[1] Chang, N. and T. Maia (2001) Learning Grammatical Constructions. AAAI Spring Symposium on Learning Grounded Representations. Stanford University.

[2] Steels, L. (2004a) Constructivist Development of Grounded Construction Grammars Scott, D., Daelemans, W. and Walker M. (eds) (2004) Proceedings Annual Meeting Association for Computational Linguistic Conference. Barcelona. p. 9-1

[3] Steels, L (2004b) Fluid Construction Grammar Tutorial. Tutorial. http://arti.vub.ac.be/FCG/

[4]