8.2. Interfacing the lexicon, the ontology and the grammaticon: a programmatic approach

Now that the linguistic-conceptual linkage has been fully specified, we may take this discussion one step further, i.e. towards the computational treatment of grammar. Recall that the LCM distinguishes between 4 levels of meaning construction: level 1 or the argument structure layer, which has been the focus of our attention in the analytical section; level 2 or implicational layer; level 3 or illocutionary layer; and level 4, which deals with discourse structure. In the Grammaticon, each of these constructional realms is computationally implemented in a so-called 'Constructicon' (cf. Periñán and Arcas 2010a). Quoting Mairal, Ruiz de Mendoza and Periñán (2011), while the former "is the name given to that grammatical module that is employed to describe the inventory of constructional schemata in a given language", the latter "is one of the repositories in the four representational layers" of the LCM. As these scholars point out, both terms, however, are not interchangeable.

Throughout some of the previous sections, we have dealt with two co-existing interlinguas, namely CLSs and COREL, each of which serves a different function: the former works on the lexical-syntactic algorithm, while the latter serves as input to an automated reasoner. Hence, if basic CLSs emerge from the combined information of both the lexical and conceptual levels, various derived CLSs will stem from this previous structure to represent each of the argument structure constructions belonging to Level 1 or the core-grammar layer. At this level, a basic CLS may be transformed or expanded in order to incorporate those arguments that cannot be derived from the event structure of the verb (e.g. *John ate the plate clean*). In turn, COREL intervenes in the rest of the levels proposed by the LCM (i.e. implicational, illocutionary or discursive), in which case, the reasoning engine will work by making explicit whatever is implicit (e.g. the idea of complaint in the oft-quoted example *What's the child doing playing* *with that knife?*). Figure 5.26 below, an extended version of Figure 5.19, shows how a derived CLS is automatically generated by means of combining the core-grammar of the verb (i.e. basic CLS) with the grammatical information located in the L1-Constructicon (i.e. the caused-motion construction):

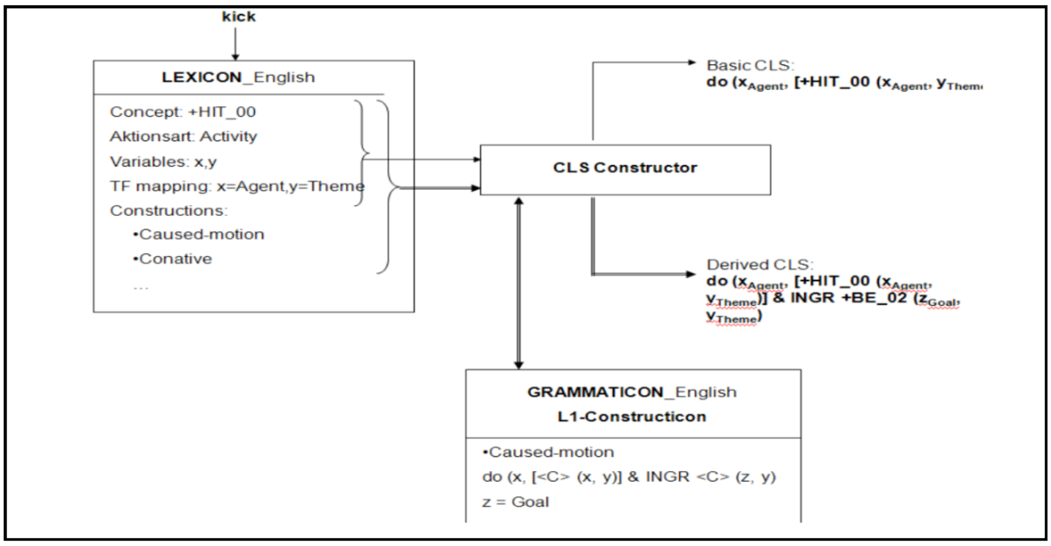
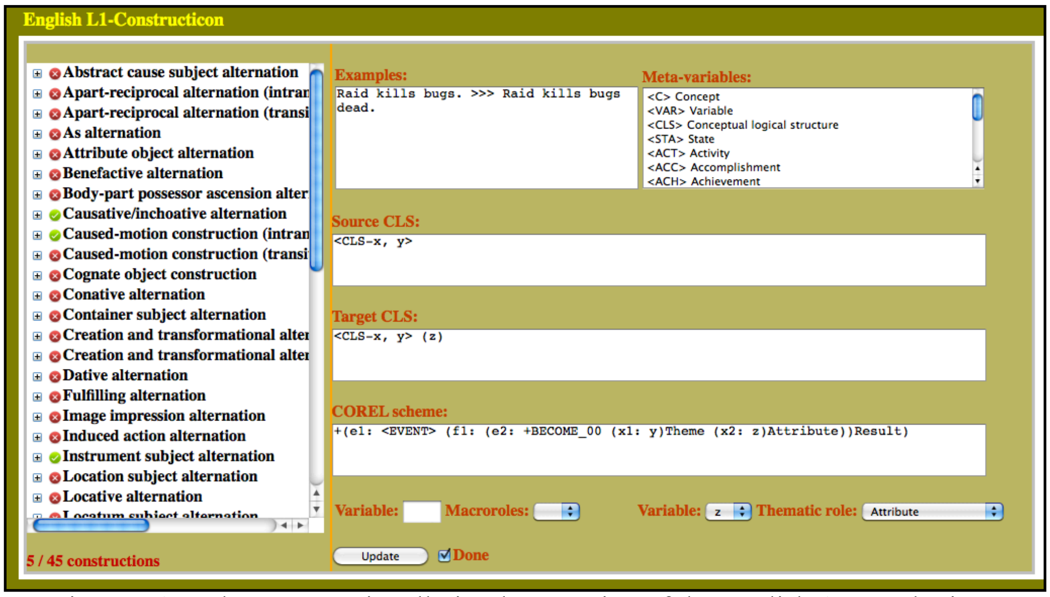

kick
LEXICON_English
Concept: +HIT_00
Aktionsart: Activity
Variables: x,y
TF mapping: x=Agent,y=Theme
Constructions:
•Caused-motion
•Conative
...
CLS Constructor
Basic CLS:
do (x_{Agent}, [+HIT_00 (x_{Agent}, y_{Them}
Derived CLS:
do (x_{Agent}, [+HIT_00 (x_{Agent}, y_{Theme})] & INGR +BE_02 (z_{Goal}, y_{Theme})
GRAMMATICON_English
L1-Constructicon
•Caused-motion
do (x, [<C> (x, y)] & INGR <C> (z, y)
z = Goal

Figure 5.26: The lexical-grammatical-conceptual linkage (borrowed from Mairal, Ruiz de Mendoza and Periñán 2011).

Figure 5.26 features the intricate process carried out by the machine when linking the three domains discussed thus far. In this representation, the CLS Constructor gathers the linguistic information of the predicate *kick* (which recall, would not exist without the previous creation of the conceptual unit +HIT_00) and delivers a basic CLS (i.e. do (X_{Agent} [+HIT_00 (X_{Agent} Y_{Theme})]. In turn, the CLS Constructor takes this basic CLS as input, which together with the grammatical information of the caused-motion construction, results in an expanded or modified conceptual structure or derived CLS. Further note that not only is the semantic representation of the basic CLS made out of conceptual data, but also, in keeping with the *Equipollence Hypothesis*, the semantic representation of the different argument structure construction of the L1-Constructicon will also be done *via* CLSs. Linguistically speaking, the fusion mechanisms between the old LTs and CTs were schematically depicted as vertical processes in which the lexical piece was subsumed into the constructional template. As Figure 5.26 illustrates, FunGramKB seems to follow a distinct procedure in which various levels cooperate to produce a syntactically projectable conceptual pattern.[1]

Our goal in this dissertation was to show the extent to which linguistic information can in fact be computationally treatable through the FunGramKB project. Future research will thus be concerned with the population of the L1-Constructicon by carrying the constructional analysis carried out in section 4 over onto the grammatical level of this lexico-conceptual knowledge base. Although so far, only five out of the approximate number of forty-five constructions recognized to exist in English have been computationally treated in FunGramKB, this project is a truly promising step in the development of NLP systems. Thus, we would like to close this chapter by providing an image of how the AP resultative construction looks like in the L1 English Constructicon.

[1] Since levels 2, 3 and 4 of the LCM have not been the focus of our study, we refer the reader to Mairal, Ruiz de Mendoza and Periñán (2011) for more information on other levels.


English L1-Constructicon

- ⊞ ✗ Abstract cause subject alternation
- ⊞ ✗ Apart-reciprocal alternation (intran
- ⊞ ✗ Apart-reciprocal alternation (transi
- ⊞ ✗ As alternation
- ⊞ ✗ Attribute object alternation
- ⊞ ✗ Benefactive alternation
- ⊞ ✗ Body-part possessor ascension alter
- ⊞ ✓ Causative/inchoative alternation
- ⊞ ✓ Caused-motion construction (intran
- ⊞ ✗ Caused-motion construction (transi
- ⊞ ✗ Cognate object construction
- ⊞ ✗ Conative alternation
- ⊞ ✗ Container subject alternation
- ⊞ ✗ Creation and transformational alter
- ⊞ ✗ Creation and transformational alter
- ⊞ ✗ Dative alternation
- ⊞ ✗ Fulfilling alternation
- ⊞ ✗ Image impression alternation
- ⊞ ✗ Induced action alternation
- ⊞ ✓ Instrument subject alternation
- ⊞ ✗ Location subject alternation
- ⊞ ✗ Locative alternation
- ⊞ ✗ Locatum subject alternation

5 / 45 constructions

Examples:

```
Raid kills bugs. >>> Raid kills bugs
dead.
```

Meta-variables:

```
<C> Concept
<VAR> Variable
<CLS> Conceptual logical structure
<STA> State
<ACT> Activity
<ACC> Accomplishment
<ACH> Achievement
```

Source CLS:

```
<CLS-x, y>
```

Target CLS:

```
<CLS-x, y> (z)
```

COREL scheme:

```
+(e1: <EVENT> (f1: (e2: +BECOME_00 (x1: y)Theme (x2: z)Attribute))Result)
```

Variable: Macroroles: Variable: z Thematic role: Attribute

Update [✓ Done]

Figure 5.27: The computationally implementation of the English AP resultative.

Let us conclude by explaining this figure. According to Mairal, Ruiz de Mendoza and Periñán (2011), in the argument structure layer, it is possible for a given construction, say, the AP resultative pattern exemplified in *Tobias ate the bowl empty*, to alter both the CLS and the COREL schema (see Figure 5.27 above) as shown in (37) and (38).

(1) a. Tobias ate the bowl.

b. $<_{IF}$ DECL $<_{TNS}$ PAST <do (%TOBIAS_00-Theme [+EAT_00 (%TOBIAS_00-Theme, %BOWL_00-Referent)])

c. +(e1: past +EAT_00 (x1: %TOBIAS_00)Theme (x2: %BOWL_00)Referent)

(2) a. Tobias ate the bowl empty.

b. $<_{IF}$ DECL $<_{TNS}$ PAST <do (%TOBIAS_00-Theme [+EAT_00 (%TOBIAS_00-Theme, %BOWL_00-Referent)])] ($EMPTY_00-Attribute)

c. +(e1: past +EAT_00 (x1: %TOBIAS)Theme (x2: %BOWL_00)Referent (f1: (e2: +BECOME_00 (x2)Theme (x3: $EMPTY_00)Attribute))Result)

Hence, in the case at hand, the resultative construction, whose formal representation is that of (39), permits the extension of the CLS in (37b) and the COREL representational schema in (37c) into (38b) and (38c) respectively:

(3) Source CLS: <CLS- x, y>
Target CLS: <CLS- x, y> (z), where Z =Attribute
COREL scheme: +(e1: <EVENT> (f1: (e2: +BECOME_00 (x1: y)Theme (x2: z)Attribute))Result)

Whereas the conceptual metavariable <CLS- x, y> stands for any type of CLS that includes these two variables (x, y), the conceptual metavariable <EVENT> should be understood as "any event from the ontology, together with its TF" (Mairal, Ruiz de Mendoza and Periñán 2011). Speaking in linguistic terms, in the Source CLS we have two participants (Tobias and the bowl, i.e. (x) and (y) respectively). In turn, the Target CLS would augment the valence, i.e. (z) being an attribute (e.g. *empty*), which is not part a natural part of *eat*. Finally, the COREL scheme would read as: there is an event, i.e. 'eating' in which (x1: Tobias) acts on (y: the bowl) so that as a result the bowl (x2) becomes empty, i.e. (z).