IMPLEMENTING LEXICAL FUNCTIONS IN XLE

François Lareau
Macquarie University

Mark Dras
Macquarie University

Benjamin Börschinger
Macquarie University
Universität Heidelberg

Myfany Turpin
University of Queensland

Proceedings of the LFG12 Conference

Miriam Butt and Tracy Holloway King (Editors)

2012

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

Linguistic collocations such as *pay attention* or *heavy rain* are semi-compositional expressions that require a special treatment in symbolic grammars for NLP. Within the Meaning-Text Theory framework, recurrent patterns of collocations have been identified and described via so-called "lexical functions". Building on our previous attempt at importing such lexical functions into LFG using glue semantics, in this paper, we show how a workable approximation to this can be implemented in XLE. We explore and compare three different approaches: using lexical functions in the f-structure, modelling collocations in the σ-projection, and modelling them through transfer rules.

1 Introduction

Linguistic collocations such as *pay attention* or *heavy rain* are semi-compositional expressions that require a special treatment in symbolic grammars for natural language processing (NLP). Such expressions are very common in both oral and written speech, yet, their treatment in NLP remains often *ad hoc* and superficial. Much of the work within the Meaning-Text Theory (MTT) (Mel'čuk, 1973; Kahane, 2003) has focused on the lexicon, and the concept of collocations has been at the heart of this theory since the mid-60s. Consequently, this framework has a well-developed theory of the relations that exist between lexemes in the lexicon. In particular, it provides a convenient tool for the description of collocations, called lexical functions (LFs).

In terms of resources developed and applications within computational linguistics, MTT has not been very prominent outside of natural language generation (NLG), but LFs have proven useful especially for multilingual natural language generation (MNLG) (see, for instance, Heid and Raab, 1989; Bourbeau et al., 1990; Iordanskaja et al., 1992; Lareau and Wanner, 2007; Wanner et al., 2010). As it turns out, the context of our research is an MNLG project where we aim to produce Australian Football League game summaries in both English and Arrernte, an Australian language of the Pama-Nyungan family. The fact that sports news is very rich in collocations, and the bilingual nature of our system, made us look for a way to use LFs in our LFG grammars. More generally, having LFs defined in XLE would allow the LFG community to tap into existing lexical resources that focus on collocations and are built around the concept of LFs: DEC (Mel'čuk et al., 1984–1999), DiCo (Polguère, 2000), DicoInfo (L'Homme, 2005), and RLF (Lux-Pogodalla and Polguère, 2011) for French, DiCE (Alonso Ramos, 2003) for Spanish, the lexical component of ETAP-3 (Apresjan et al., 2003; Boguslavsky et al., 2004) for Russian, English and Arabic, and the multilingual lexical database architecture ILexiCon (LeFrançois and Gandon, 2011).

[†]We wish to thank Melanie Seiss for her help with transfer rules, and John Maxwell and Sina Zarrieß for their insight on generating from s-structures. We also acknowledge the support of ARC Discovery grant DP1095443.

Our first attempt at importing LFs into LFG (Lareau et al., 2011) was a theoretical approach to the problem. We showed that the f-structure alone was not sufficient to handle collocations properly, since the phenomenon pertains to the semantics-syntax interface, and we proposed a way to encode LFs in glue semantics. However, the current version of XLE, the platform that we use for our LFG grammar, does not implement glue semantics, so we searched for ways to put our ideas into practice. In this paper, we will explore and compare three different strategies: using lexical functions in the f-structure (§5), modelling collocations in the σ-projection (§6), and modelling them through transfer rules (§7). But before diving in, we will present briefly what exactly we mean by collocations (§2), introduce the concept of LF (§3), and expose our theoretical, glue-based solution (§4). We take for granted that the reader is familiar with LFG (Kaplan and Bresnan, 1982; Bresnan, 2001; Dalrymple, 2001; Falk, 2001), glue semantics (Dalrymple et al., 1993; Dalrymple, 1999, 2001; Andrews, 2010), and XLE (Maxwell and Kaplan, 1993; Crouch et al., 2011).

This paper expands on Lareau et al. (2011), and consequently there is a certain overlap between the two. This one does not entirely subsume the previous one however, since we have left aside all considerations linked too closely to our specific MNLG project, to gain in generality. Hence, although we have in mind the generation of texts rather than their interpretation, the ideas discussed here apply to both tasks (indeed, our implementation presently works better for parsing than generation). Also, although our examples are mostly related to football, our solution is not tied to any particular domain.

2 Collocations

A core phenomenon in the semantics-syntax interface is the mapping between meanings and lexemes—let us take a speech production perspective for a moment and call this *lexicalisation*. The lexicalisation of one meaning is usually independent of that of other meanings in the same sentence, but in the case of collocations, one lexicalisation interferes with another. A collocation is a semi-idiomatic expression where the choice of one lexeme, called the *base*, is free, but the choice of another lexeme, called the *collocate*, is context-sensitive and is constrained by the choice made for the base. Consider for example the phrases *strong preference*, *intense flavour*, *heavy rain* and *great risk*. While the lexemes PREFERENCE,[1] FLAVOUR, RAIN and RISK are chosen freely, the lexemes STRONG, INTENSE, HEAVY and GREAT are not. They carry roughly the same meaning of intensification, but their choice is tied to the lexeme they modify.

The concept of collocation is only fully understood when it is considered in the perspective of speech production rather than interpretation because there are collocations that are semantically transparent, yet the lexicalisation of their collocate is to a certain extent arbitrary. Compare for instance *strong taste* and *intense flavour*. They have very close meaning, and the semantics of STRONG and INTENSE is

[1] We use italics for word forms (e.g., *rain*) and small capitals for lexemes (e.g., RAIN).

transparent. Yet, although *strong flavour* sounds correct, the expression *intense taste* sounds odd. It is not really ungrammatical, nor is it semantically ill-formed, but it does not sound idiomatic; it is just not how English speakers would say it. The meaning of intensification is not lexicalised freely when it is used in the context of the lexeme TASTE: it must be lexicalised as STRONG. As further evidence that the relation between TASTE and STRONG is arbitrary, consider *mild taste*, which sounds idiomatic, as opposed to *weak taste*, which sounds odd, although WEAK is the antonym of STRONG. This is because there is a special relation in the lexicon of English between TASTE and STRONG or MILD that does not exist between TASTE and INTENSE or WEAK. It is these lexical relations that we aim to model; the question is how to describe these relations between lexemes in an LFG dictionary?

There are several types of collocations. We have mentioned examples where the collocate is a modifier of the base and expresses intensification or its opposite. There are adjunctive collocates that express a range of meanings: *black coffee* 'coffee without anything added to it', *green energy* 'energy that does not pollute', *decent meal* 'meal with enough food to satisfy', *wrong decision* 'bad decision', etc.

Another common type of collocation is where the collocate is a verb that takes the base as one of its arguments. These are usually referred to as *light verbs* (Jespersen, 1946). Now, before we go any further, let us get terminological confusion out of the way. The LFG-aware reader may be familiar with the work on light verbs by Kim (1991, 1993), Matsumoto (1996), Butt (2003, 2010), Yokota (2005), Ivana and Sakai (2007) or Seiss (2009). These authors did not confine themselves to collocations, while we are not concerned with light verbs that combine freely, such as MAKE in *Mary made him read the book*. Here, there is no special lexical relation between MAKE and READ; this causative combines with any verb that is semantically compatible. This is different from collocations, where the choice of the causative depends on the choice of the lexeme it combines with, e.g., GIVE in *Mary gave him the flu*. That being said, there is an overlap with the mentioned works, and although LFs are intended to describe collocations, they could also describe non-idiomatic light verbs; see the paper by Dras et al. in this volume.

To illustrate what we have said, much of our discussion will be based on the example below:

(1) Mark kicked a beautiful goal.

The lexemes KICK and BEAUTIFUL, in this context, are both collocates of GOAL. *Beautiful goal* is typical of the "colourful" style that characterises sports news, and does not mean much more than a mere positive appreciation of the goal from the author. It could be replaced with, say, *spectacular goal* or *brilliant goal*, without significantly changing the meaning. BEAUTIFUL, BRILLIANT and SPECTACULAR are all instances of the same collocation pattern (and as we will see in the following section, LFs are all about identifying such patterns). The semantics of these collocations is $(\lambda x.\text{good}(x))(\lambda y.\text{goal}(y))$, where $\lambda x.\text{good}(x)$ gets lexicalised as BEAUTIFUL, BRILLIANT or SPECTACULAR when it is a modifier of GOAL.

Kick a goal may seem more compositional. Indeed, the player does have to kick the ball, but since it is the only way to score a goal in Australian football, the semantic contribution of KICK is marginal. This becomes more apparent when we consider the translation of this phrase into other languages, where the idea of kicking totally disappears: in Spanish, *hacer un gol*, lit. 'do a goal', in Arrernte, *goal arrerneme*, lit. 'put a goal', etc. In English, the phrase *score a goal* means essentially the same as *kick a goal*. Finally, the meaning conveyed by sentence (1) can also be expressed as a noun phrase, as in (2):

(2) a beautiful goal to/by Mark

All these facts suggest that KICK, in this context, is a collocational light verb and that its semantic contribution is weak enough to be considered null. So, the semantics of *kick a goal* in (1) should be the same as that of *goal* in (2), i.e., $\lambda x.\text{goal}(x)$. Note that we are only concerned with lexical situational meaning, leaving aside grammatical meaning and phenomena pertaining to the information structure, such as the difference between (1) and (2). Both expressions denote the same situation, but differ in their communicative perspective—a difference that we want to capture in a different structure.

Now, let us present the concept of LF.

3 Lexical functions

LFs were proposed by Žolkovskij and Mel'čuk (1967) as a way to describe collocations in the context of machine translation. The concept is based on the observation that collocations tend to be instances of a number of recurrent patterns that occur across languages. For example, the expressions *strong preference*, *gravely ill*, *intense flavour* and *win hands down* are all instances of a pattern of semantics-syntax mapping where the meaning of the base is intensified and, syntactically, the collocate is a modifier of the base. Žolkovskij and Mel'čuk's idea, then, was to give names to such patterns. This one was given the name `Magn` (from Latin MAGNUS 'big'). Then, the pattern is modeled as a relation[2] between the base and the collocate. Hence, `Magn`(PREFERENCE)=STRONG, `Magn`(FLAVOUR)=INTENSE, etc. Over the years, more than fifty basic recurrent patterns of this type, and a few hundreds of complex ones, have been identified across languages and given names. Detailed descriptions of LFs can be found elsewhere (Mel'čuk, 1995, 1996, 1998; Wanner, 1996b; Apresjan, 2000; Kahane and Polguère, 2001; Apresjan et al., 2002). We will see other patterns of collocations in the coming sections.

In §2, we identified in sentence (1) two collocations: *beautiful goal* and *kick a goal*. The first one is an instance of a pattern where the semantics of the collocate is $\lambda x.\text{good}(x)$, and where the collocate is realised as a syntactic modifier of the base. The LF for this pattern is called `Bon`, and we would like to have in our dictionary the

[2] In a sense, the term *lexical function* is unfortunate since in fact these are not true mathematical functions because there can be several values for the same relation applied to a given base.

ATTENTION [of X to Y]

Magn	close/whole/complete/undivided $\sim$
Func_2	X's $\sim$ is on Y
nonFunc_0	X's $\sim$ wanders
Oper_{12}	X gives his/pays $\sim$ to Y
Oper_2	Y attracts/receives/enjoys X's $\sim$
$\text{Oper}_2\text{+Magn}_{\text{quant-}X}$	Y is the center of $\sim$ (of many Xs)
IncepOper_{12}	X turns his $\sim$ to Y
IncepOper_2	Y gets X's $\sim$
ContOper_2	Y holds/keeps X's $\sim$
CausFunc_2	Z draws/calls/brings X's $\sim$ to Y
LiquFunc_2	Z diverts/distracts/draws X's $\sim$ from Y

Figure 1: Collocations controlled by ATTENTION described via LFs.

instruction Bon(GOAL)=BEAUTIFUL/BRILLIANT/SPECTACULAR. The second one is an instance of another pattern where the collocate has no meaning but serves only as a support verb to turn a noun into a verbal expression. Syntactically, the collocate takes as its object the base of the collocation, and as its subject the first semantic argument of the base. This pattern corresponds to the LF Oper_1. We would then like to have in our dictionary the instruction Oper_1(GOAL)=KICK/SCORE/BOOT.

In short, the whole idea is to seek patterns in collocations. There will always be subtle nuances between two synonymous collocations because collocates are never entirely stripped of their literal meaning in usage, so in order to recognise patterns we have to somehow reduce the meaning of collocations to rid them of such nuances. For practical purposes however, it is safe to do so, and the benefits we get in terms of grammar engineering and resource maintainability are greater than the loss in granularity (especially given the current state of computational semantics—we are not exactly at the stage of subtle nuances yet). Armed with such LFs, the lexicographer can quickly and conveniently describe collocations in a dictionary. For example, the entry for ATTENTION could look like the one in Fig. 1. We will not discuss each of these LFs here; the point is to illustrate the wide range of collocations that can be captured efficiently with LFs.

The patterns corresponding to each LF must be defined in the grammar. We discuss in the following sections how this can be conceptualised in LFG and implemented in XLE. Once this is done, the patterns can be reused across languages and domains with little or no modifications.

4 Collocations in glue semantics

In the context of LFG, there have been several approaches to developing a compositional notion of semantics derived from the f-structure; we have chosen to base our work on Dalrymple et al. (1999) and Dalrymple (2001)'s view of glue

semantics, which we will assume the reader is familiar with. It should be noted that the exact form of the semantic representations does not have to be as in this paper; our analysis is not bound to it. Instead of the simple expressions we show here, one could use event semantics or frames, for instance. What does not vary is the linear logic expressions that control semantic composition. Linear logic differs from classical logic in that premises are treated as resources that are consumed in the process of the proof. This resource-sensitivity is appropriate when dealing with the linguistic expression of semantic content: the contribution of each lexeme and phrase to the meaning of a sentence is usually unique, and there should be no missing or redundant lexemes in terms of the meaning to be expressed. In the expressions that we are concerned with, however, the principle of compositionality is violated, and the mapping between meanings and lexemes can be rather complex.

Let us get back to our example (1). For a literal reading of this sentence, the lexical entries for *goal*, *kicked* and *beautiful* would be as follows:

goal N $(\uparrow\text{PRED})=\text{`goal'}$
$\text{goal} : \uparrow_\sigma$

kicked V $(\uparrow\text{PRED})=\text{`kick}\langle(\uparrow\text{SUBJ}),(\uparrow\text{OBJ})\rangle\text{'}$
$(\uparrow\text{TENSE})=\text{past}$
$\lambda x.\lambda y.\text{kick}(x,y) : (\uparrow\text{SUBJ})_\sigma \multimap [(\uparrow\text{OBJ})_\sigma \multimap \uparrow_\sigma]$

beautiful A $(\uparrow\text{PRED})=\text{`beautiful'}$
$\lambda x.\text{beautiful}(x) : (\text{ADJ} \in \uparrow)_\sigma \multimap (\text{ADJ} \in \uparrow)_\sigma$

This would yield the meaning ‘$\text{kick}(\text{Mark}, \text{beautiful}(\text{goal}))$’ (or, if you prefer, ‘$\text{kick}(e_1, m, g) \wedge \text{Mark}(m) \wedge \text{goal}(g) \wedge \text{beautiful}(e_2, g)$’, but we will stick to the former, simple notation). What we would like instead is ‘$\text{good}(\text{goal}(\text{Mark}))$’. For this, we need to change the lexical entries for these three word forms.

First, as we said in §2 and §3, *beautiful*, in (1), denotes a vague meaning of positive appreciation, which we could represent as ‘$\lambda x.\text{good}(x)$’. This is only true when *beautiful* modifies the noun GOAL (and perhaps other nouns with which it forms a collocation), so there must be a constraint in the entry that checks the context in which this modifier is used; this is what the second line does:

beautiful A $(\uparrow\text{PRED})=\text{`beautiful'}$
$((\text{ADJ} \in \uparrow)\ \text{PRED})=_\text{c}\text{`goal'}$
$\text{`}\lambda x.\text{good}(x)\text{'} : (\text{ADJ} \in \uparrow)_\sigma \multimap (\text{ADJ} \in \uparrow)_\sigma$

Second, *kicked a goal* does not mean more than ‘$\lambda x.\text{goal}(x)$’, i.e., the verb KICK simply recopies its object’s meaning, with the constraint that its object is the lexeme GOAL:

kicked V $(\uparrow\text{PRED})=\text{`kick}\langle(\uparrow\text{SUBJ}),(\uparrow\text{OBJ})\rangle\text{'}$
$(\uparrow\text{TENSE})=\text{past}$
$(\uparrow\text{OBJ PRED})=_\text{c}\text{`goal'}$
$\text{`}\lambda x.x\text{'} : (\uparrow\text{OBJ})_\sigma \multimap \uparrow_\sigma$

Finally, the meaning of *goal* should be a unary predicate: '$\lambda x.\text{goal}(x)$', i.e., 'x *goals*', so to speak. In the construction under consideration here, its semantic predicativity is not echoed in syntax, but this should not affect the representation of its meaning. In fact, this is precisely why a support verb is needed in the first place: KICK ties the noun GOAL to its semantic argument MARK and turns the noun into a verbal expression. This is rendered with a meaning constructor that checks that there is a meaning available for the subject of the verb of which GOAL is the object. This entry is only correct when used in the context of a support verb of which it is the object, so we also need a constraining equation here. This is inelegant, but we will see in §6 how we can get rid of it in the implementation.

goal N (↑PRED)='goal'
((OBJ↑) PRED)=$_c$'kick'
'$\lambda x.\text{goal}(x)$' : ((OBJ↑) SUBJ)$_\sigma$ $\multimap$ ↑$_\sigma$

As we have said above, *kick a goal* could be paraphrased as *score/boot a goal*, and *beautiful goal* could be replaced with *brilliant/spectacular goal*. The entries for the alternative collocates would be nearly identical to the ones we have just shown. There are generalisations to be made here, and we can capture them with lexical rules that we would use in the entries. And this is exactly where LFs come into play. The idea is to define lexical rules for `Oper`$_1$ and `Bon`, and then use them as follows:

kick V { @(Oper1 *goal*) | ... }
boot V { @(Oper1 *goal*) | ... }
score V { @(Oper1 *goal*) | ... }
beautiful A { @(Bon *goal*) | ... }
brilliant A { @(Bon *goal*) | ... }
spectacular A { @(Bon *goal*) | ... }

In the following sections we will explore three different ways or defining such lexical rules via templates in XLE.

5 An f-structure-based implementation

The simplest way to deal with collocations in XLE is to flatten their semantics by representing it in the f-structure. We achieve this by replacing collocates with the names of LFs in the PRED attribute, thus using in our representations "generalised lexemes", in the sense of Wanner (1996a), in a way similar to MTT's deep-syntactic representations, where LFs appear as nodes like other lexemes (Mel'čuk, 1988). For example, for the sentence (1), we would have in the f-structure `Oper`$_1$ and `Bon` instead of KICK and BEAUTIFUL, as in Fig. 2. By using LFs instead of lexical items, we abstract away from the collocates used in the sentence, which yields an f-structure that represents a range of paraphrases that have the same syntax and (roughly) the same meaning, but differ in the lexical choice of collocates. In this

example, we have two types of collocations: a modifier (Bon) and a support verb ($\texttt{Oper}_1$). These correspond to two types of lexical templates in XLE. Let us first look at some modifiers.

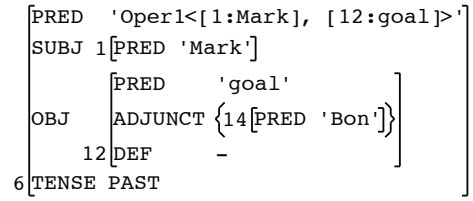

Figure 2: An f-structure for (1) with LF names in it.

Below are the templates for three different patterns of collocations where the base is modified by an adjectival or adverbial collocate. The difference between them is semantic, which is captured by the names of the LFs in the PRED attribute: Bon for modifiers that denote a positive appreciation from the speaker (*beautiful goal*, *superior quality*); Epit for pleonastic modifiers that contribute little or nothing to the meaning of the phrase, merely repeating something that is included in the meaning of the base (*happy victory*, *safe haven*); and Magn for modifiers that denote intensification in a broad sense (*considerable amount*, *intense flavour*). This is what the first instruction of these patterns encodes. It also gives the syntactic valence of the lexeme, which is trivial in the case of modifiers like here (they do not have syntactic arguments). The second instruction ensures that we are really dealing with a collocation, and not a normal modifier. It checks that the adjective/adverb is an adjunct of a specific lexeme, the base of the collocation, passed as an argument to the template. This effectively models restricted lexical co-occurrence.

```
Bon(Base) =
   (^PRED)='Bon'
   ((ADJUNCT ^) PRED FN) =c Base.

Magn(Base) =
   (^PRED)='Magn'
   ((ADJUNCT ^) PRED FN) =c Base.

Epit(Base) =
   (^PRED)='Epit'
   ((ADJUNCT ^) PRED FN) =c Base.
```

Now, describing common adjunct-type collocations in the dictionary is a matter of calling such templates. Adjectives and adverbs have their normal description, corresponding to their literal readings (given below by the trivial templates @ADJ and @ADV—their exact definition is irrelevant to us), to which are added a number of LF templates for collocations they are involved in. Both the literal and the idiomatic readings are thus available; we leave it to an external process to choose the right interpretation.

```
beautiful A { @ADJ | @(Bon goal) }.
brilliant A { @ADJ | @(Bon goal) }.
happy A { @ADJ | @(Epit victory) }.
spectacular A { @ADJ | @(Bon goal) }.
easily Adv { @ADV | @(Magn win) }.
hands_down Adv @(Magn win).
```

This is equivalent to saying that Bon(GOAL)=BEAUTIFUL/BRILLIANT/SPECTACULAR, Epit(VICTORY)=HAPPY, and Magn(WIN)=EASILY/⌜HANDS DOWN⌝. Note that ⌜HANDS DOWN⌝ has no literal interpretation; it can only be used as an intensifier of WIN (and perhaps a few other lexemes that we have ignored here).

Below are the templates for three different patterns of support verbs: $\texttt{Func}_0$ (*the wind blows*, *the rain pours*) and $\texttt{Oper}_1$ (*perform an operation*, *take a nap*), which are semantically empty verbs, and $\texttt{LiquFunc}_0$ (*snap a streak*, *eradicate a disease*), which means 'to cause the end of'. The templates for support verbs are similar to the ones above, but their PRED must encode their syntactic valence, since such collocates always have at least one syntactic argument; this is what the first line of these templates is for. The second line gives the position of the base in relation to the support verb; unlike the templates above, where the base of the collocation was always the lexeme being modified by the collocate, the base of a support verb can be any of its syntactic arguments. Hence, the difference between a $\texttt{Func}_0$ and an $\texttt{Oper}_1$ is purely syntactic: the former is an intransitive verb that takes the base as its subject, whereas the latter is a transitive verb that takes the base as its object. The difference between an $\texttt{Oper}_1$ and a $\texttt{LiquFunc}_0$, on the other hand, is semantic, and it is captured by the PRED attribute. This is not very transparent however, because patterns that have the same semantics (such as $\texttt{Func}_0$ and $\texttt{Oper}_1$, both semantically empty) also have a different PRED functor.

```
Func0(Base) =
   (^PRED)='Func0<(^SUBJ)>'
   (^SUBJ PRED FN) =c Base.

Oper1(Base) =
   (^PRED)='Oper1<(^SUBJ)(^OBJ)>'
   (^OBJ PRED FN) =c Base.

LiquFunc0(Base) =
   (^PRED)='LiquFunc0<(^SUBJ)(^OBJ)>'
   (^OBJ PRED FN) =c Base.
```

Then, the collocations *boot/kick/score a goal*, *get a mark*, *get the victory* and *snap a streak* are described in the dictionary as below (@TRANS is the usual template for transitive verbs with a literal reading):

```
boot V { @TRANS | @(Oper1 goal) }.
get V { @TRANS | @(Oper1 mark) | @(Oper1 victory) }.
kick V { @TRANS | @(Oper1 goal) }.
score V { @TRANS | @(Oper1 goal) }.
snap V { @TRANS | @(LiquFunc0 streak) }.
```

With the lexical entries discussed above, and a few trivial ones not shown here, we can parse (1) to obtain the f-structure in Fig. 2. Because `Oper1` and `Bon` are abstractions on the lexemes KICK and BEAUTIFUL that appeared in the parsed sentence, we can regenerate all paraphrases that only differ in the lexical choice for these collocates:

(3) Mark booted/kicked/scored a beautiful/brilliant/spectacular goal.

This is useful for shallow paraphrasing where lexical items are changed. However, it is not possible to produce paraphrases that differ in their syntactic structure. For this, we need to have a proper semantic structure. The obvious way to get one in XLE is to use the σ-projection mechanism; let us now discuss this approach.

6 An s-structure-based implementation

XLE allows us to define new projections in addition to the built-in ϕ-projection. We use this mechanism to derive from the f-structure an s-structure where we encode the semantics of expressions. Since we now have a separate structure for meaning, the attribute PRED does not have to capture semantic information anymore. It is not obsolete however; it stores lexico-syntactic information about lexemes, that is, the name of the lexeme and its syntactic valence in the expression under consideration. Our f-structure now looks like the usual ones, sticking to a more superficial description of the actual words used in the sentence, regardless of whether they have an idiomatic or a literal reading. Thus, the f-structure for sentence (1) is the one in Fig. 3.

```
⎡PRED  'kick<[1:Mark], [12:goal]>'                  ⎤
⎢SUBJ 1⎡PRED 'Mark'⎤                                ⎥
⎢      ⎡PRED    'goal'                            ⎤⎥
⎢OBJ   ⎢ADJUNCT {14⎡PRED 'beautiful'⎤}             ⎥⎥
⎢   12 ⎣DEF     -                                  ⎦⎥
⎣6 TENSE PAST                                       ⎦
```

Figure 3: A normal f-structure for (1).

Our s-structure is a connected direct acyclic graph that encodes predicate-argument relations only. We do not use semantic roles; we only encode the salience of a predicate's arguments by numbering them from 1 up in decreasing order of salience, as is done in MTT's semantic representations (Mel'čuk, 2004). This has the advantage of having more generic relations between meanings, which makes it easier to find recurrent patterns. The graph is encoded in XLE as an attribute-value matrix (AVM) where nodes are rendered as structures, and the relations between the nodes as attributes ARG1, ARG2, etc., that have as their value an embedded structure. The labels of the nodes, given by the attributes SEM in the AVMs, are either the name of a lexeme when it has a literal reading, or the name of a LF when it is a meaningful

collocate (e.g., ‘Bon’ instead of ‘good’, ‘Magn’ instead of ‘big’ or ‘intense’). We use LF names here because we want to underline the fact that we are pointing to an idealised meaning, stripped of the nuances that may exist between instances of a collocation pattern. Hence, for sentence (1), we want to have the s-structure in Fig. 4, equivalent to the expression $(\lambda x.\text{Bon}(x))((\lambda y.\text{goal}(y))(\text{Mark}))$. To the left is a graphical representation of the meaning of (1); to the right is its encoding as an AVM in XLE (note that structure 2 is actually embedded in structure 4 even if it does not appear so visually—this is how XLE displays the result for technical reasons that are irrelevant here). We leave aside grammatical meanings.

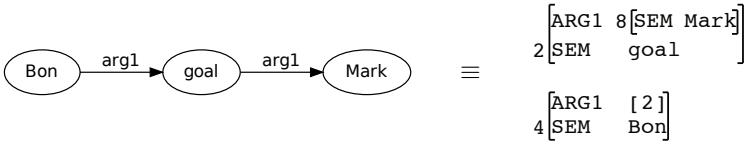

Figure 4: An s-structure for (1).

It is in the projection between f-structure and s-structure that the complex mapping between collocations and their semantics takes place. Again, we use templates to define LFs; we will look below at the templates for `Bon`, `Epit` and $\texttt{Oper}_1$. As in §5 above, modifier-type templates such as `Bon` or `Epit` have a trivial PRED and these collocates are always adjuncts of their base, while support verbs like $\texttt{Oper}_1$ have a PRED that reflects their syntactic sub-categorisation, and their base may be any of their syntactic arguments. This time however, the PRED functor is the lexical stem, since we do not need to represent the semantics of collocates in the f-structure anymore. This is ensured by the `s::` instructions, which construct the σ-projection. In the `Bon` pattern, the first semantic instruction projects the idealised meaning ‘Bon’ as the SEM attribute in the s-structure. The second instruction gives the mapping between the semantic and syntactic relations: the meaning ‘Bon’ is a semantic predicate that becomes in f-structure an adjunct of its first semantic argument—i.e., the σ-projection of the collocate has a first semantic argument in s-structure which is the σ-projection of the lexeme it is an adjunct of in f-structure. In the case of `Epit`, because it is a pleonastic adjunct that does not contribute significant meaning to the phrase, there is simply no semantic information. This is not possible for support verbs like $\texttt{Oper}_1$, although they are also semantically empty, because they stand at the root of the clause, so they must provide a σ-projection for the outermost f-structure. Also, even if they do not contribute meaning, they do perform a rather complex remapping of semantic/syntactic arguments. The meaning of an $\texttt{Oper}_1$ is that of its base, which is always its direct object (cf. the second line of the $\texttt{Oper}_1$ template); this is what the first semantic instruction models. The last instruction handles the remapping of arguments: the first semantic argument in s-structure becomes the subject of the support verb in f-structure.

```
Bon(Base) =
   (^PRED) = '%stem'
```

```
   ((ADJUNCT ^) PRED FN) =c Base
   (s::^ SEM) = Bon
   (s::^ ARG1) = s::(ADJUNCT^).

Epit(Base) =
   (^PRED) = '%stem'
   ((ADJUNCT ^) PRED FN) =c Base.

Oper1(Base) =
   (^PRED)='%stem<(^SUBJ)(^OBJ)>'
   (^OBJ PRED FN) =c Base
   s::^ = s::(^OBJ)
   (s::^ ARG1) = s::(^SUBJ).
```

These templates are used in the same way as in §5 above. Fig. 5 illustrates graphically what is happening in the semantics-syntax interface for the `Oper1` and `Bon` patterns. The elements in bold are the ones actively built by the rules above. Note how the `Oper1` does not really realise any meaning from the s-structure, but merely links syntactically a predicate to its first semantic argument. These graphs bear a striking resemblance to Polarized Unification Grammars (PUGs) (Kahane and Lareau, 2005; Lareau, 2008).

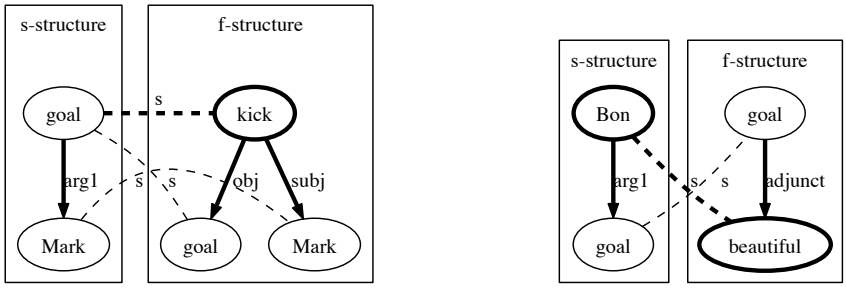

Figure 5: σ-projection and argument mapping for `Oper1` and `Bon`.

Let us look at a more complex example involving an idiomatic causative verb:

(4) A beautiful goal to Mark gave the victory to Sydney.

Its f-structure, in Fig. 6, has nothing particularly interesting. Its semantics, however, is not completely compositional. Here, *gave* means roughly 'cause', and *beautiful* expresses positive appreciation by the speaker. As we did for the previous example, we replace the collocational meanings by the names of the LFs that correspond to them, to highlight the fact that we are dealing with idealised meanings. Then, the s-structure for (4) should be something like in Fig. 7.

The light verb GIVE, besides having a non-literal meaning in this context, "steals" the first semantic argument of 'victory', which becomes its oblique object (the choice of this syntactic function is flexible). This corresponds to Mel'čuk's `CausFunc1`, which we define below. The first line gives its syntactic sub-categorisation and the

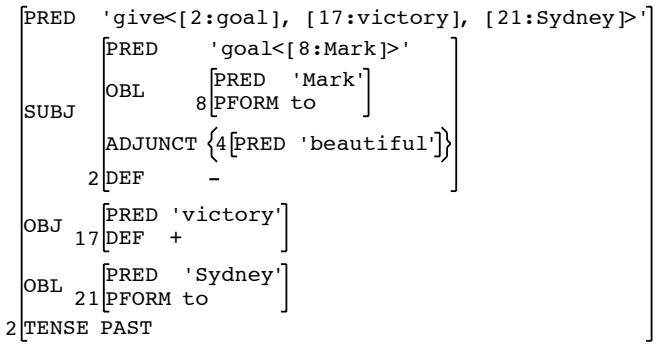

Figure 6: An f-structure for (4).

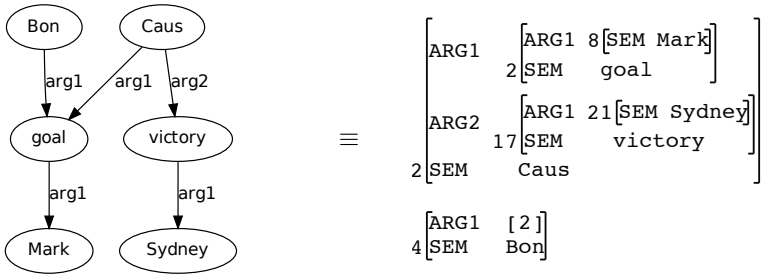

Figure 7: An s-structure for (4).

second gives the position of the base that controls this collocation. The last four lines model the tricky semantics of this light verb. First, it projects its meaning, the idealised meaning 'Caus'. Then, it gives the mapping of the first and second semantic arguments of 'Caus', which become respectively the subject and object of GIVE. Finally, it maps the first semantic argument of the σ-projection of its object to its oblique (that is, it "steals" an actant from 'victory'). This template is then used in the entry for GIVE as below. Fig. 8 gives a visual representation of the complex mapping between semantic and syntactic elements performed by this template, where the bold elements are the ones actively involved in the rule.

```
CausFunc1(Base) =
   (^PRED)='%stem<(^SUBJ)(^OBJ)(^OBL)>'
   (^OBJ PRED FN) =c Base
   (s::^ SEM)=Caus
   (s::^ ARG1)=s::(^SUBJ)
   (s::^ ARG2)=s::(^OBJ)
   (s::(^OBJ) ARG1)=s::(^OBL).

give V { @DITRANS | @(CausFunc1 victory) }.
```

The s-structure solution yields a higher level of abstraction than the previous one, so that even expressions with a completely different syntactic structure get the

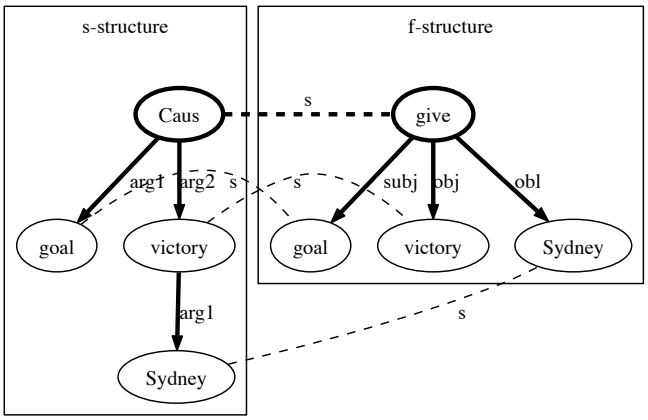

Figure 8: σ-projection and argument mapping for $\texttt{CausFunc}_1$.

same analysis. Consider for example sentence (1) and the noun phrase (2). When we parse them, we get different f-structures, but the same s-structure, as in Fig. 9. However, it is not possible to (re)generate from this s-structure, because additional projections in XLE do not have their own structure but are instead encoded as special attributes in the f-structure. This means that generating from an s-structure amounts to generating from an underspecified f-structure where all the attributes except the S:: ones are missing. XLE can accept underspecified f-structures as input for generation, but there must be a PRED attribute in each structure, because of choices made during the implementation of the platform. So this approach works for parsing, but will not work for generation or paraphrasing in the current implementation of XLE. In order to generate from s-structures like the ones we have just looked at, we have tried a different approach, using transfer rules.

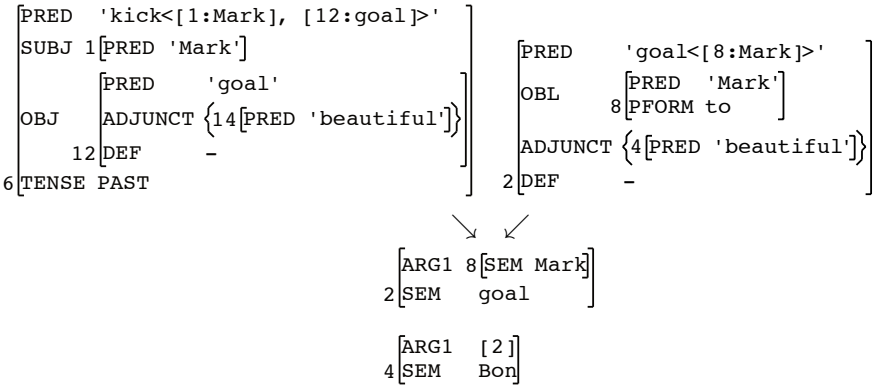

Figure 9: Two synonymous expressions yielding the same s-structure.

7 A transfer-based implementation

The Packed Rewriting System (PRS) was first intended as the transfer module of an XLE-based machine translation system. It applies rewrite rules that produce new f-structures from existing ones. Since s-structures are in fact encoded as f-structures, it is possible to hijack this mechanism to model the semantics-syntax interface. Indeed, Crouch and King (2006) and Zarrieß and Kuhn (2010) have done it for, respectively, parsing and generation. We have tried to use it in a similar way to generate collocations, with mixed results.

Transfer rules rewrite attributes of an f-structure. Rules can be optional, which allows for backtracking and provides a set of output f-structures. Our input is an s-structure like the one in Fig. 4, where the only attributes are SEM and ARG1, ARG2, etc. These will be the elements on the left-hand side of the transfer rules. On the right-hand side, we have attributes of a normal f-structure: PRED, SUBJ, OBJ and ADJUNCT. There is also a predicate ARG that encodes the position of the syntactic arguments in PRED's value, as well as IN_SET, which builds sets of adjuncts.

Below are the transfer-based templates for `Bon` and $\texttt{Oper}_1$. Whereas in the s-structure approach we looked for the base of the collocation in the f-structure with the instruction `((ADJUNCT ^) PRED FN) =c Base`, in the transfer-based approach we must encode lexical restrictions in semantics. Indeed, the left-hand side of a rule can have contextual elements, denoted by the `+` symbol; these elements are not consumed by the rule. There is no way to specify such contextual elements on the right-hand side of the rules to avoid building new items in the f-structure. This means that in this version of our grammar, collocations are controlled by meanings rather than lexemes, which is a dangerous setting since most of the lexical relations that we are trying to describe actually exist between lexemes, not meanings. For the rest, these templates do essentially the same things as the ones we have discussed in §6 (cf. Fig. 5). In the case of `Bon`, the meaning 'Bon' and its first semantic argument are consumed by the rule to produce an adjunct in f-structure. For $\texttt{Oper}_1$, only the relation ARG1 is consumed, and it is realised in syntax by the support verb and its subject and object.

```
bon(%Base, %Collocate) ::
  SEM(%X, Bon),
  ARG1(%X, %Y),
  +SEM(%Y, %Base)
  ?=>
  PRED(%X, %Collocate),
  ADJUNCT(%Y, %Adjs), in_set(%X, %Adjs).

oper1(%Base, %Collocate) ::
  +SEM(%X, %Base),
  ARG1(%X, %Y)
  ?=>
  PRED(%Z, %Collocate),
  arg(%Z, 1, %Y), SUBJ(%Z, %Y),
  arg(%Z, 2, %X), OBJ(%Z, %X).
```

The lexical items in this kind of grammar contain only the information necessary for the semantics-syntax interface. Below are some of the entries relevant for the meaning 'goal' (`@n` and `@n_obl` are trivial templates that simply realise a meaning as a noun or a noun with an oblique complement).

```
@bon(goal, beautiful).
@bon(goal, brilliant).
@bon(goal, spectacular).
@oper1(goal, boot).
@oper1(goal, kick).
@oper1(goal, score).
@n_obl(goal).
@n(goal).
```

The most interesting characteristic of this approach is that it allows us to group the collocations by their base instead of having them described in the lexical entries for the collocates as in §5 and §6. This is a lot more convenient for the lexicographer. There are three main drawbacks, however. First, as we have mentioned above, the collocations in this grammar have to be controlled by meanings instead of lexemes. Second, we have to set all our rules as optional. When there is no obligatory rule, it is always a valid solution to leave some or all of the elements of the input structure untouched in the output. We end up with chimeras that contain stock from the semantic and functional levels of representation, and we cannot do anything with these. While it would be possible to filter out the invalid output structures, this is obviously not an elegant solution. Finally, the transfer rules are applied in the order in which they appear in the file. This is unfortunately incompatible with what we are trying to do. So we have not yet overcome the limitations of the current implementation of XLE so as to allow for a direct application of the analysis presented here to NLG. We are currently exploring other strategies to solve this problem by means of external processing that are of no particular interest from a linguistic perspective.

8 Conclusion

MTT's lexical functions offer an elegant and convenient solution to the treatment of collocations in NLP. We showed that this device could be used also in the LFG framework, both from a conceptual and implementational point of view. Glue semantics offers the kind of expressive power we need to handle the complex mappings between meanings and lexemes, as well as their arguments, in semi-idiomatic expressions. However, our glue-based solution involves adding information in the lexical entry of the base of collocations in a way that is not entirely satisfactory, since it spreads out the information across several entries. In XLE, we tried and compared three different strategies for the implementation of LFs.

The f-structure based implementation lacks the depth needed to fully model a phenomenon that indeed belongs to the semantics-syntax interface. By using

abstract LF names instead of actual lexemes in the f-structure, we managed to represent the functional structure of a range of paraphrases that differ in the choice of the collocates. Despite its inherent limitations, this implementation works for both parsing and generation, and allows shallow paraphrasing.

The s-structure based implementation is the one that most closely matches the power of glue semantics. It is the most elegant of our three solutions to the problem of describing collocations, as it allows us to parse synonymous expressions with radically different lexico-syntactic structures and get the semantic representation that we expect. However, because of how XLE is implemented, we cannot generate from s-structures using this approach.

As a complementary strategy for generation, we have used transfer rules to handle the semantics-syntax interface. This strategy is different from the other two in that it isolates the semantics-syntax interface from the rest of the model. It is the most compelling solution for lexicographers because it allows them to group collocations by their base, rather than forcing them to describe the collocations in the entries of the collocates. However, technical considerations force us to seek other strategies to handle generation from s-structures. One strategy that we plan to explore is to use Bohnet et al. (2000)'s generic graph-transducer, MATE, to handle the mapping between s-structure and f-structure. This approach could be combined with our f-structure implementation.

We view this work as developing a set of tools for the LFG community, rather than an actual grammar. For the f-structure and s-structure strategies, we have defined 222 templates corresponding to a range of LFs (modifiers, dummy support verbs, causative light verbs, inchoative light verbs, etc.), which we will make available at `http://web.science.mq.edu.au/~ayeye`.

For future work, we plan to extend the coverage of our patterns to include more LFs. We also want to explore how we can make these patterns even more generic, and we hope to make them compatible with Butt et al. (2002)'s ParGram architecture.

References

Alonso Ramos, Margarita. 2003. Hacia un diccionario de colocaciones del español y su codificación. In M.A. Martí (ed.), *Lexicografía computacional y semántica*, pages 11–34, Barcelona: Edicions de l'Universitat de Barcelona.

Andrews, Avery. 2010. Propositional glue and the correspondence architecture of LFG. *Linguistics and Philosophy* 33, 141–170.

Apresjan, Juri. 2000. *Systematic lexicography*. Oxford: Oxford University Press.

Apresjan, Jury, Boguslavsky, Igor, Iomdin, Leonid, Lazursky, Alexander, Sannikov, Vladimir, Sizov, Victor and Tsinman, Leonid. 2003. ETAP-3 linguistic processor: a full-fledged NLP implementation of the Meaning-Text Theory. In *Proceedings of MTT 2003*, pages 279–288, Paris.

Apresjan, Jury, Boguslavsky, Igor, Iomdin, Leonid and Tsinman, Leonid. 2002.

Lexical functions in actual NLP applications. In *Computational linguistics for the new millennium: divergence or synergy?*, pages 55–72, Frankfurt: Peter Lang.

Boguslavsky, Igor, Iomdin, Leonid and Sizov, Victor. 2004. Multilinguality in ETAP-3: reuse of lexical resources. In *Proceedings of the Workshop on Multilingual Linguistic Ressources (MLT'04)*, pages 7–14, Stroudsburg, PA.

Bohnet, Bernd, Langjahr, Andreas and Wanner, Leo. 2000. A development environment for an MTT-based sentence generator. In *Proceedings of the first international conference on Natural Language Generation*, volume 14, pages 260–263, Mitzpe Ramon.

Bourbeau, Laurent, Carcagno, Denis, Goldberg, Eli, Kittredge, Richard and Polguère, Alain. 1990. Bilingual generation of weather forecasts in an operations environment. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, pages 90–92.

Bresnan, Joan. 2001. *Lexical-functional syntax*. Oxford, UK: Blackwell.

Butt, Miriam. 2003. The light verb jungle. In G. Aygen, C. Bowern and C. Quinn (eds.), *Harvard Working Papers in Linguistics. Papers from the GSAS/Dudley House Workshop on Light Verbs*, pages 1–49.

Butt, Miriam. 2010. The light verb jungle: still hacking away. In M. Amberber, B. Baker and M. Harvey (eds.), *Complex predicates: cross-linguistic perspectives on event structure*, pages 48–78, Cambridge: Cambridge University Press.

Butt, Miriam, Dyvik, Helge, King, Tracy Holloway, Masuichi, Hiroshi and Rohrer, Christian. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Crouch, Dick, Dalrymple, Mary, Kaplan, Ron, King, Tracy, Maxwell, John and Newman, Paula. 2011. *XLE documentation*. Palo Alto Research Center, Palo Alto.

Crouch, Dick and King, Tracy Holloway. 2006. Semantics via f-structure rewriting. In *Proceedings of the LFG06 Conference*, Konstanz.

Dalrymple, Mary (ed.). 1999. *Semantics and syntax in lexical functional grammar: the resource logic approach*. Cambridge, MA: The MIT Press.

Dalrymple, Mary. 2001. *Lexical functional grammar*, volume 42 of *Syntax and Semantics Series*. New York: Academic Press.

Dalrymple, Mary, Gupta, Vineet, Lamping, John and Saraswat, Vijay. 1999. Relating resource-based semantics to categorial semantics. In Dalrymple (1999).

Dalrymple, Mary, Lamping, John and Saraswat, Vijay. 1993. LFG semantics via constraints. In *Proceedings of the 6th Meeting of the European Association for Computational Linguistics*, Utrecht.

Dras, Mark, Lareau, François, Börschinger, Benjamin, Dale, Robert, Motazedi, Yasaman, Rambow, Owen, Turpin, Myfany and Ulinski, Morgan. 2012. Complex predicates in Arrernte. In M. Butt and T. H. King (eds.), *Proceedings of the LFG12 Conference*, Stanford: CSLI Publications.

Falk, Yehuda. 2001. *Lexical-functional grammars*. Stanford: CSLI Publications.

Heid, Ulrich and Raab, Sybille. 1989. Collocations in multilingual generation. In *Proceedings of the fourth conference of the European chapter of the Association*

for Computational Linguistics (EACL'89), pages 130–136.

Iordanskaja, Lidja, Kim, Myunghee, Kittredge, Richard, Lavoie, Benoît and Polguère, Alain. 1992. Generation of extended bilingual statistical reports. In *Proceedings of COLING'92*, pages 1019–1023, Nantes.

Ivana, Adrian and Sakai, Hiromu. 2007. Honorification and light verbs in Japanese. *Journal of East Asian Linguistics* 15(3), 171–191.

Jespersen, Otto. 1946. *A modern English grammar on historical principles. Part VI: morphology.*. London: Allen & Unwin.

Kahane, Sylvain. 2003. The Meaning-Text theory. In V. Agel, L. M. Eichinger, H.-W. Eroms, P. Hellwig, H. J. Heringer and H. Lobin (eds.), *Dependenz und Valenz: Ein internationales Handbuch der zeitgenössischen Forshung/ Dependency and Valency: An International Handbook of Contemporary Research*, volume 1, pages 546–570, Berlin / New York: Walter de Gruyter.

Kahane, Sylvain and Lareau, François. 2005. Meaning-Text Unification Grammar: modularity and polarization. In *Proceedings of MTT 2005*, pages 163–173, Moscow.

Kahane, Sylvain and Polguère, Alain. 2001. Formal foundation of lexical functions. In *Proceedings of ACL 2001*, Toulouse.

Kaplan, Ronald M. and Bresnan, Joan. 1982. Lexical-functional grammar: a formal system for grammatical representation. In J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pages 173–281, Cambridge, MA: The MIT Press.

Kim, Jay. 1991. *A lexical-functional grammar account of light verbs*. Ph.D. thesis, University of Hawaii.

Kim, Jeong-Ryeol. 1993. Parsing light verb constructions in lexical-functional grammar. *Language Research* 29(4), 535–566.

Lareau, François, Dras, Mark, Börschinger, Benjamin and Dale, Robert. 2011. Collocations in multilingual natural language generation: lexical functions meet lexical functional grammar. In *Proceedings of ALTA'11*, pages 95–104, Canberra, Australia.

Lareau, François and Wanner, Leo. 2007. Towards a generic multilingual dependency grammar for text generation. In *Proceedings of Grammar Engineering Across Frameworks (GEAF'07)*, pages 203–223, Stanford.

Lareau, François. 2008. *Vers une grammaire d'unification sens-texte du français: le temps verbal dans l'interface sémantique-syntaxe*. Ph.D. thesis, Université de Montréal/Université Paris 7.

LeFrançois, Maxime and Gandon, Fabien. 2011. ILexicOn: toward and ECD-compliant interlingual lexical ontology described with semantic web formalisms. In *Proceedings of MTT 2011*, pages 155–164, Barcelona.

L'Homme, Marie-Claude. 2005. Conception d'un dictionnaire fondamental de l'informatique et de l'Internet : sélection des entrées. *Le langage et l'homme* 40(1), 137–154.

Lux-Pogodalla, Veronika and Polguère, Alain. 2011. Construction of a French lexical network: Methodological issues. In *Proceedings of the International*

Workshop on Lexical Resources (WoLeR 2011), ESSLLI, pages 55–62, Ljubljana.

Matsumoto, Yo. 1996. A syntactic account of light verb phenomena in Japanese. *Journal of East Asian Linguistics* 5(2), 107–149.

Maxwell, John T. and Kaplan, Ronald M. 1993. The interface between phrasal and functional constraints. *Computational Linguistics* 19(4), 571–590.

Mel'čuk, Igor. 1973. Towards a linguistic "Meaning-Text" model. In F. Kiefer (ed.), *Trends in Soviet Theoretical Linguistics*, pages 33–57, Dordrecht: Reidel.

Mel'čuk, Igor. 1988. *Dependency syntax: theory and practice*. SUNY series in linguistics, Albany: State University of New York Press.

Mel'čuk, Igor. 1995. The future of the lexicon in linguistic description and the Explanatory Combinatorial Dictionary. In I.-H. Lee (ed.), *Linguistics in the morning calm*, volume 3, Seoul: Hanshin.

Mel'čuk, Igor. 1996. Lexical functions: a tool for the description of lexical relations in a lexicon. In Wanner (1996b), pages 37–102.

Mel'čuk, Igor. 1998. Collocations and lexical functions. In A. P. Cowie (ed.), *Phraseology. Theory, analysis, and applications*, pages 23–53, Oxford: Clarendon.

Mel'čuk, Igor. 2004. Actants in semantics and syntax. I: Actants in semantics. *Linguistics* 42(1), 1–66.

Mel'čuk, Igor et al. 1984–1999. *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques*, volume I–IV. Montréal: Presses de l'Université de Montréal.

Polguère, Alain. 2000. Une base de données lexicales du français et ses applications possibles en didactique. *LIDIL* 21, 75–97.

Seiss, Melanie. 2009. On the difference between auxiliaries, serial verbs and light verbs. In M. Butt and T. H. King (eds.), *Proceedings of LFG'09*, pages 501–519, Trinity College, Cambridge, UK.

Wanner, Leo. 1996a. Introduction. In Wanner (1996b), pages 1–36.

Wanner, Leo (ed.). 1996b. *Lexical functions in lexicography and natural language processing*. Amsterdam/Philadelphia: John Benjamins.

Wanner, Leo, Bohnet, Bernd, Bouayad-Agha, Nadjet, Lareau, François and Nicklaß, Daniel. 2010. MARQUIS: generation of user-tailored multilingual air quality bulletins. *Applied Artificial Intelligence* 24(10), 914–952.

Yokota, Kenji. 2005. The structure and meaning of Japanese light verbs. *Language Sciences* 27(3), 247–280.

Zarrieß, Sina and Kuhn, Jonas. 2010. Reversing f-structure rewriting for generation from meaning representations. In *Proceedings of the LFG10 Conference*, Ottawa.

Žolkovskij, Aleksandr and Mel'čuk, Igor. 1967. O semantičeskom sinteze. *Problemy kibernetiki* 19, 177–238.