Collocations in Multilingual Natural Language Generation: Lexical Functions meet Lexical Functional Grammar

François Lareau **Mark Dras** **Benjamin Börschinger** **Robert Dale**
Centre for Language Technology
Macquarie University
Sydney, Australia
Francois.Lareau|Mark.Dras|Benjamin.Borschinger|Robert.Dale@mq.edu.au

Abstract

In a collocation, the choice of one lexical item depends on the choice made for another. This poses a problem for simple approaches to lexicalisation in natural language generation systems. In the Meaning-Text framework, recurrent patterns of collocations have been characterised by lexical functions, which offer an elegant way of describing these relationships. Previous work has shown that using lexical functions in the context of multilingual natural language generation allows for a more efficient development of linguistic resources. We propose a way to encode lexical functions in the Lexical Functional Grammar framework.

1 Introduction

Natural Language Generation (NLG) is the generation of natural language text from some underlying representation: numerical data, knowledge bases, predicate logic, and so on. Multilingual Natural Language Generation (MNLG) is NLG where the output is in more than one language. Most NLG systems are in some way modular (see Reiter and Dale (2000) for a discussion of typical architectures); one advantage to modularity is the scope for separating language-independent components from those which are language-dependent, making it possible to add multilinguality with much less work than would be involved in building a new system from scratch (Bateman et al., 1999). Such claims have been made since the very first MNLG systems; the FoG system generating weather forecasts in English and French (Bourbeau et al., 1990) is a case in point.

Consequently, MNLG has been applied for a large number of text types: government statistics reports (Iordanskaja et al., 1992), technical instruction manuals (Paris et al., 1995), fairy tales (Callaway and Lester, 2002), museum tours (Callaway et al., 2005), medical terminology (Rassinoux et al., 2007), codes of practice (Evans et al., 2008), and so on.

Marcu et al. (2000), in reviewing some of the earlier work, comment that MNLG systems need to abstract as much as possible away from the individual language generated:

> If an [MNLG] system needs to develop language dependent knowledge bases, and language dependent algorithms for content selection, text planning, and sentence planning, it is difficult to justify its economic viability. However, if most of these components are language independent and/or much of the code can be reused, an [MNLG] system becomes a viable option.

Bateman et al. (1999) similarly emphasise the importance of reducing language dependence.

One kind of abstraction generalises across language-specific collocations: for example, we might note that *heavy rain*, *strong wind* or *intense bombardment* all refer to the intensification of some phenomenon, as similarly does the French *pluie battante* ('beating rain'), but the particular intensifier used is determined by collocational appropriateness. These kinds of collocations are modeled within the Meaning-Text Theory (MTT) framework via lexical functions (LFs) (Mel'čuk, 1995); for some lexeme L, the above semantic notion of intensification or

François Lareau, Mark Dras, Benjamin Börschinger and Robert Dale. 2011. Collocations in Multilingual Natural Language Generation: Lexical Functions meet Lexical Functional Grammar. In *Proceedings of Australasian Language Technology Association Workshop*, pages 95−104

strength is represented by Magn(L). MTT-based MNLG systems, from the early works of Heid and Raab (1989) and Iordanskaja et al. (1992) onwards, have used LFs to abstract away from the specific collocational phenomena of individual languages.

In terms of resources developed and applications within the computational linguistics community, however, MTT has not been very prominent outside of NLG. In work that is more geared towards natural language understanding, other formalisms such as Lexical Functional Grammar (LFG) (Bresnan, 2001), Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994), Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) and Combinatory Categorial Grammar (CCG) (Steedman, 2000) have received much more attention. Of course, all of these frameworks have also been applied in NLG. LFG, for example, has a sophisticated grammar development environment called Xerox Linguistic Environment (XLE) (Maxwell and Kaplan, 1993) for both parsing and generation, with wide-coverage grammars for a number of languages such as English and German (Butt et al., 2002), and advanced statistical models for tasks such as realisation ranking (Cahill et al., 2007).

The existence of a wide-coverage English LFG grammar for XLE convinced us to build our resources on this platform for the MNLG project described below. However, LFG comes from a tradition quite different from MTT, and has no concept that corresponds to MTT's LFs. In this paper, we demonstrate that LFs cannot be straightforwardly introduced into the LFG formalism via direct manipulation of f-structures, and then show how glue semantics (Dalrymple, 2001) can incorporate them in an elegant way.

We first describe our MNLG system to provide a contextual background for the problem (§2), along with some basic notions on LFG (§3). We then describe LFs in more detail, and discuss how they have been used in other MNLG systems (§4), along with how they would fit into our system. We then return to LFG and glue semantics (§5), and present our proposal for incorporating LFs into LFG, using a running example (§6).

2 Our System

The context for this work is a project involving an MNLG system for generating commentary-style textual descriptions of Australian Football League (AFL) games, in both English and the Australian Aboriginal language Arrernte. A typical sentence in a human-authored commentary for a game might look as follows:

> Led by Brownlow medallist Adam Goodes and veteran Jude Bolton, the Swans kicked seven goals from 16 entries inside their forward 50 to open a 30-point advantage at the final change—to that point the largest lead of the match.

For the games we want to describe, there is a corresponding database which contains quantitative and other data regarding the game: who scored which goal when, from where, and so on. The system will use handwritten grammars in the LFG formalism—for English, there is an already-existing wide-coverage one developed for XLE as part of the ParGram project (Butt et al., 2002)—and the research around the grammar development will have a number of foci. In particular, we are interested in exploring how to handle morphologically rich non-configurational languages such as Arrernte, which are not usually tackled in the field of language technology; these exhibit a number of interesting and complicated phenomena, as outlined by, for example, Austin and Bresnan (1996) or Nordlinger and Bresnan (2011). Given the radical language differences between such languages and those which are more typically the focus of NLG projects, we are particularly interested in investigating the possible extent of language independence (see §4 for a discussion). LFs are an important facet of the semantic abstractions we require. For example, in the short text cited above, the expression *kick a goal* would be rendered in Arrernte as *goal arrerneme*, where *arrerneme* literally means 'put'. We view *kick* and *arrerneme* as support verbs in these expressions.[1] These collocations exhibit the same syntactic structure, and express the same meaning; they are instances of the same pattern

[1] Note that in AFL one can only score goals by kicking the ball, so in this context, the semantic contribution of *kick* is weak; we believe that for practical purposes it can be viewed as empty.

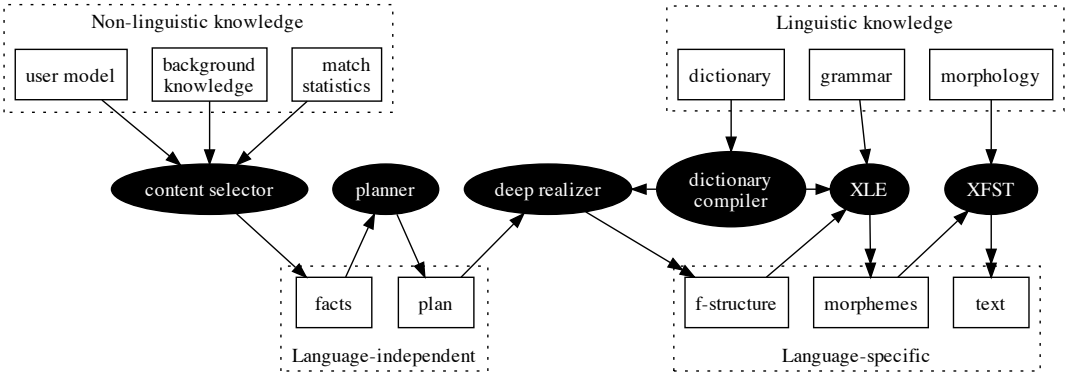

Figure 1: System architecture

of collocation, which is described in MTT by the LF $\texttt{Oper}_1(L)$. We will return to this example in §4.

Our system more or less follows the "consensus architecture" of Reiter and Dale (2000), as schematised in Figure 1. Data is selected using domain-specific knowledge and statistical methods, to produce a collection of facts to be expressed. These facts are organised into a document plan, which is then passed to a deep realiser that produces one or more f-structures for each sentence of the text (cf. §3). This is then passed to XLE, which uses linguistic knowledge to produce ordered lexical items with attached morphemes. This is finally passed to a two-level morphology model compiled with the Xerox Finite-State Tool (XFST) system, which produces fully inflected text. The double-headed arrows in Figure 1 indicate non-deterministic output; a stochastic reranking technique is used to select between alternative results.

The module which is the focus of the present discussion is the deep realiser, which maps a semantic representation to one or more f-structures, for which in turn XLE will produce textual realisations. It is not our purpose in this paper to discuss the technical details of the implementation of this component; rather, we focus on the mapping between semantic representations and f-structures from a theoretical point of view. To explain the approach, and to motivate the need for glue semantics, we now present an outline of LFG.

3 Lexical Functional Grammar

LFG is a formalism for a non-derivational theory of linguistic structure that posits at least two levels of representation: c(onstituent)-structure and f(unctional)-structure.[2] Mappings specify the relationship between the different levels of structure. C-structure is represented by phrase-structure trees, capturing hierarchical relationships between constituents and surface phenomena such as word order; while f-structure is represented by attribute–value matrices, describing more abstract functional relationships such as subject and object (indeed, syntactic dependencies). LFG assumes that these functional syntactic concepts are universally relevant across languages (Dalrymple, 2001, p. 3), and "so may be regarded as a major explanatory source of the relative invariance of f-structures across languages" (Bresnan, 2001, p 98). As an illustration, the c- and f-structures for the sentence (1) below are given in Figure 2.

(1) Bradshaw kicked a beautiful goal.

The mapping between f- and c-structures is given by annotations on phrase structure and lexical rules as in (2) and (3) below.

(2) $\text{S} \rightarrow \underset{(\uparrow\text{SUBJ})=\downarrow}{\text{NP}} \quad \underset{\uparrow=\downarrow}{\text{VPall}}$

(3) *kicked* V $(\uparrow\text{PRED})=\text{`kick}\langle(\uparrow\text{SUBJ}),(\uparrow\text{OBJ})\rangle\text{'}$ $(\uparrow\text{TENSE})=\text{past}$

Lexical entries such as (3) are in fact only a different notation used for terminal nodes in c-structures; this could be written instead as in (4).

(4) $\text{V} \rightarrow \underset{\substack{(\uparrow\text{PRED})=\text{`kick}\langle(\uparrow\text{SUBJ}),(\uparrow\text{OBJ})\rangle\text{'} \\ (\uparrow\text{TENSE})=\text{past}}}{\textit{kicked}}$

[2] See Dalrymple (2001) or Bresnan (2001) for extensive discussions of LFG; here, we provide only the basic essentials required to understand our treatment.

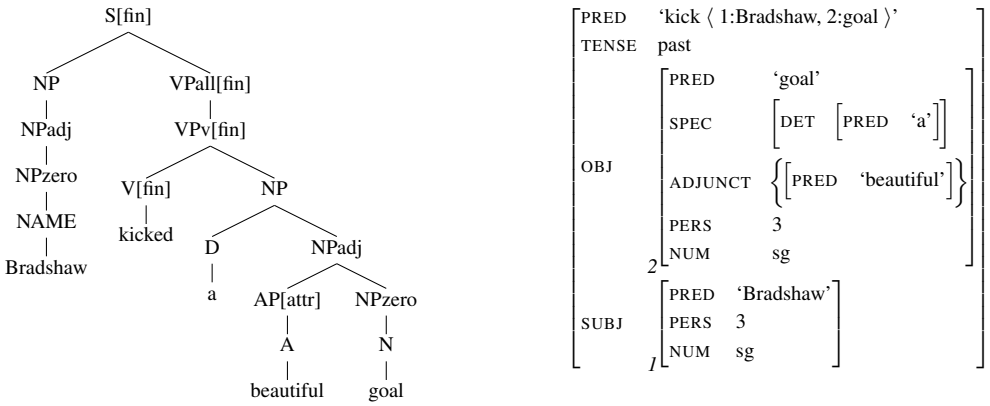

Figure 2: c-structure (left) and f-structure (right) for *Bradshaw kicked a beautiful goal*

The symbol ↑ is a metavariable representing the f-structure of the parent of a node in the c-structure, and ↓ the f-structure of that node itself. These can be followed by a sequence of attributes that specify a path to an element in the f-structure. For example, in rule (2), the annotation on the NP means that the SUBJ of the f-structure corresponding to the node above it in the c-structure (S) is the f-structure corresponding to this NP (in plain English: this NP is the subject of the sentence). The annotation on the VPall node means that it shares the same f-structure as its mother (i.e., it is the head of S). In the lexical rule for *kicked* (3), the annotation (↑OBJ) inside the PRED attribute refers to the f-structure numbered 2 in Figure 2, which represents *goal*. A less common way of referring to elements of an f-structure, albeit one that is necessary in a number of cases, is "inside-out" function application, where the ↑ or ↓ follows an attribute sequence. An annotation such as (OBJ↑) refers to the f-structure of which the current one is the OBJ.

First-order predicate logic is often used as the fundamental meaning representation in LFG, although other more expressive representations are also possible, such as intensional logic or Discourse Representation Theory. The issue then is how to relate the core LFG structures above to this meaning representation. Dalrymple (2001, p. 217) notes that early work in LFG took the f-structure element PRED to represent the locus of the semantics, with the PRED in fact originally being referred to as the *semantic form*. If our meaning representation for (1) were as in (5a) (ignoring here tense and number), the mapping to the f-structure in the right of Figure 2 would be

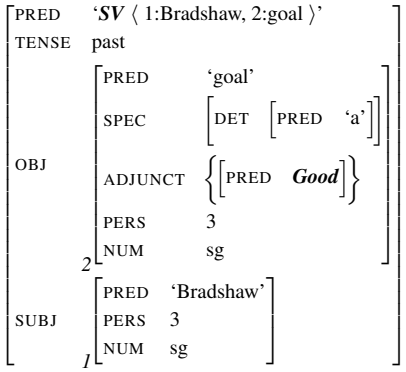

Figure 3: 'Quasi-'f-structure with variables

straightforward: the basic hierarchical structure of the f-structure is preserved, although predicativity is reversed in the case of the adjunct.

(5) a. *kick(bradshaw, beautiful(goal))*
b. *good(goal(bradshaw))*

However, simple semantic forms like this cannot represent many aspects of semantics, such as scope of modifiers or quantification. More particularly for our purposes, if we start from a semantic representation that abstracts away from the collocationally determined use of *beautiful* to characterise the goal, or from the collocational use of *kick* to refer to the goal event,[3] a suitable starting meaning representation might be as in (5b); what the mapping should look like is then much less clear.

A (quasi-)f-structure corresponding to this might be as in Figure 3. The top-level PRED could be a variable (***SV***) that would have to be instantiated to

[3]We might think of this as the action of "goaling"; another possible supporting verb would then be *score*.

a collocationally appropriate support verb; alternatively, for some types of action, it could indicate that both the top-level PRED and its object could be realised as a single verb through some kind of structure merging.[4] The adjunct PRED could similarly be a variable (***Good***) whose value is a word that retains the desired semantics but is collocationally determined. However, it is not possible for f-structures to have variable predicates, or to be of indeterminate structure, because of their role in ensuring the LFG wellformedness conditions of Coherence and Completeness; in addition, the mapping between meaning representation and f-structure is less straightforward, with quite different hierarchical relations.

In §6, we show how the mapping of such abstract meaning representations to f-structures can be done in an elegant way using glue semantics. First, we present more formally the MTT notion of LFs that these quasi-f-structure "variables" of Figure 3 are trying to capture, and we then give a brief description of glue semantics.

4 Lexical Functions and MNLG

An important step in the NLG task of surface realisation is lexicalisation, where specific lexemes are chosen to express the content of a message. Most of the time, this can be achieved by mapping either concepts or language-specific meanings to lexemes in a straightforward way. For example, the concept RAIN can be mapped to the lexemes *rain* (English), *pluie* (French), *lluvia* (Spanish), and so on. Often, however, concepts are lexicalised in a different way depending on the lexemes they appear with, as with our example from §3 above. Consider for example the phrases *strong preference*, *intense flavour*, *heavy rain* and *great risk*. While the lexemes *preference*, *flavour*, *rain* and *risk* are chosen freely according to their meaning, the lexemes *strong*, *intense*, *heavy* and *great* are not. They have roughly the same meaning of intensification, but their choice is tied to the lexeme they modify. Such collocations pose a non-trivial problem for lexicalisation in NLG systems. Under the MTT framework these are modeled via LFs. Collocations are viewed as instances of recurrent patterns of semantico-syntactic mappings, described in terms of functions (in a mathematical sense) between lexemes. Hence, these four collocations can be described in terms of a function f such that f(*preference*)=*strong*, f(*flavour*)=*intense*, etc. Over the years, more than fifty basic recurrent functions of this type, and hundreds of complex ones, have been identified across languages and given names; the one discussed above has been called `Magn`. Detailed descriptions of these functions can be found elsewhere (Mel'čuk, 1995; Wanner, 1996; Kahane and Polguère, 2001; Apresjan et al., 2002).

ATTENTION [of X to Y]

$\texttt{Magn}$	close/whole/complete/undivided $\sim$
$\texttt{Func}_2$	X's $\sim$ is on Y
$\texttt{nonFunc}_0$	X's $\sim$ wanders
$\texttt{Oper}_{12}$	X gives his/pays $\sim$ to Y
$\texttt{Oper}_2$	Y attracts/receives/enjoys X's $\sim$
$\texttt{Oper}_2$+$\texttt{Magn}_{\text{quant-}X}$	Y is the center of $\sim$ (of many Xs)
$\texttt{IncepOper}_{12}$	X turns his $\sim$ to Y
$\texttt{IncepOper}_2$	Y gets X's $\sim$
$\texttt{ContOper}_2$	Y holds/keeps X's $\sim$
$\texttt{CausFunc}_2$	Z draws/calls/brings X's $\sim$ to Y
$\texttt{LiquFunc}_2$	Z diverts/distracts/draws X's $\sim$ from Y

Figure 4: Dictionary entry for *attention*

The use of LFs allows the handling of lexicalisation in two steps. In the first step, unbound lexemes are chosen, and collocation patterns identified, while the actual value of the LF is only computed in the second step: INTENSE+RAIN $\rightarrow$ `Magn`(*rain*)+*rain* $\rightarrow$ *heavy*+*rain*.

The values for these functions are stored in the dictionary; for example, the entry for *attention* must contain the information in Figure 4. We will not discuss each of these functions here; the key point is that LFs offer a very efficient way of describing a wide range of collocations.

Each pattern must be defined in the grammar, but this is done only once for all languages and domains. We discuss in §6 how this can be done in LFG. The fact that the patterns must be defined only once for all languages makes this technique a cost-efficient way of developing MNLG resources by sharing parts of the grammar across languages, as advocated notably by Bateman et al. (1991), Bateman et al. (1999) and

[4] This kind of structure merging has not, to our knowledge, been implemented in LFG. In practice, it could be carried out by mapping from one f-structure to another, using the sort of mechanism found in XLE for use in machine translation. However, it would inelegantly add an unprincipled layer to the formalism.

Cahill et al. (2000). This was the approach taken by Lareau and Wanner (2007) for the system MARQUIS, which generated air quality reports in eight European languages (Catalan, English, Finnish, French, German, Polish, Portuguese and Spanish). Their use of LFs was an important factor that contributed to the low number of language-specific rules they reported for the deeper modules of their grammars, making the addition of new languages in their framework relatively cheap. It is LFs such as these that we wish to incorporate into LFG.

5 Glue Semantics

In the context of LFG, there have been several approaches to developing a compositional notion of semantics derived from the f-structure; one that is well developed, and is the basis of our work, is glue semantics (Dalrymple, 2001; Andrews, 2010). We give only a brief summary here; for a full treatment, see Dalrymple (2001).[5]

Glue semantics is based on linear logic. This differs from classical logic in its resource-sensitivity, in that premises are treated as resources that can be kept track of. For example, consider the statements *If you have $1, you can get an apple* and *You have $1*. In classical logic, you can deduce that you can get an apple, but the original premises would still be true, i.e. you would still have $1, and you could still get an apple. In linear logic, these premises are resources that will be consumed in the process of deduction, and therefore not available for further proof;[6] notationally, the implication above in linear logic is written $\$1 \multimap apple$.

This resource-sensitivity is particularly appropriate when we are concerned with the linguistic expression of semantic content: the contribution of each word and phrase to the meaning of a sentence is unique, and there should be no missing or redundant words in terms of the meaning to be expressed.

We illustrate how this works by showing the more straightforward mapping between our example sentence (1) and its literal meaning representation (5a). The lexical item representing *Bradshaw* is given below in (6). The first line contains the wordform, its part of speech, and an annotation asserting that the f-structure corresponding to the N node immediately dominating the lexical item has an attribute PRED whose value is the semantic form 'Bradshaw'. The second line ($\mathit{Bradshaw} : \uparrow_\sigma$) contains what is termed the *meaning constructor*, as it gives instructions on how to construct meanings. These are pairs: the lefthand (meaning) side represents the meaning, and the righthand (glue) side represents a logical formula over semantic structures corresponding to those meanings.

(6) *Bradshaw* N (↑PRED)='Bradshaw'
$\mathit{Bradshaw} : \uparrow_\sigma$

The present example is trivial: it should be read as *the semantic projection of the mother node is the meaning* ***Bradshaw***. A σ subscript indicates the semantic projection of a node, so the notation $\uparrow_\sigma$ gives the corresponding element of the semantics via the projection of the mother node to the semantics. The element *goal* is similar:

(7) *goal* N (↑PRED)='goal'
$\mathit{goal} : \uparrow_\sigma$

The verb *kick* is transitive, so it will have the following form.

(8) *kicked* V (↑PRED)='kick⟨(↑SUBJ),(↑OBJ)⟩'
(↑TENSE)=past
$\lambda X.\lambda Y.\mathit{kick}(X, Y)$:
$(\uparrow \text{SUBJ})_\sigma \multimap [(\uparrow \text{OBJ})_\sigma \multimap \uparrow_\sigma]$

In the meaning constructor, the semantics of the action of kicking is represented by a lambda term, on the left; the righthand glue side is given in terms of the linear logic implication operator $\multimap$. The first implication says that if $(\uparrow \text{SUBJ})_\sigma$ is available (i.e., if we have already built the semantic projection for the verb's subject—in our example, ***Bradshaw***), it will be consumed and will saturate the first variable of the lambda expression, to produce the new premise that follows the first $\multimap$ symbol, leaving us with $\lambda Y.\mathit{kick(Bradshaw,Y)} : (\uparrow \text{OBJ})_\sigma \multimap \uparrow_\sigma$. This in turn consumes the semantic projection for the object (in our case, ***goal***) to reduce the lambda term, and produces the semantic resource $\uparrow_\sigma$, i.e., the semantic projection for the verb and its complements, ***kick(Bradshaw,goal)***.

[5] It should be noted that the current version of XLE cannot directly handle glue semantics.

[6] Somewhat counterintuitively, even the premise *If you have $1, you can get an apple* is consumed.

For the remaining two elements, we would have the following.

(9) *beautiful* A (↑PRED)='beautiful'
$\lambda X.\mathit{beautiful}(X)$:
$(\text{ADJ} \in \uparrow)_\sigma \multimap (\text{ADJ} \in \uparrow)_\sigma$

(10) *a* D (↑PRED)='a'
$\lambda X.X : (\text{DET} \uparrow)_\sigma \multimap (\text{DET} \uparrow)_\sigma$

For *beautiful* in (9), the notation $(\text{ADJ} \in \uparrow)_\sigma$ differs in two ways from that introduced earlier. First, it uses an "inside-out" function to refer to the semantic structure of the phrase it modifies; and second, it uses set membership notation, as modifiers are typically represented by sets (as in the f-structure of Figure 2). The expression thus refers to the semantic structure corresponding to the f-structure in which ↑ appears as a member of the modifier set. In terms of the glue side, all modifiers have this structure: they take and return the same type of element. We treat the determiner in (10) similarly; further, it does not add any meaning element.[7]

All these combined together, then, give the literal semantics of (5a). Such mechanics are more complicated than is necessary for this simple example, which was used only for illustrative purposes here. However, they can equally well provide the more abstract semantics of (5b), as we show in §6.

6 Adding Lexical Functions to LFG

There are a number of changes necessary to incorporate LFs, both in a less straightforward use of glue semantics and in other aspects of the definitions of lexical entries. The lexical entry for the proper noun *Bradshaw* still has the same simple meaning constructor as above in (6). By contrast, *goal* in (11), is a unary predicate: $\lambda X.\mathit{goal}(X)$, i.e., '$X$ *goals*', so to speak. However, in the construction under consideration here, its semantic predicativity is not echoed in syntax, since there is no verb *to goal* in standard English. This is precisely why a support verb is needed in the first place: *kick* ties the noun *goal* to its semantic argument *Bradshaw*. This is rendered in the lexical entry in (11) below with a meaning constructor that checks that there is a meaning available for the subject of the verb of which *goal* is the object.

[7]The determiner could be considered as a quantifier, which would require a much more sophisticated treatment.

(11) *goal* N (↑PRED)='goal'
$\lambda X.\mathit{goal}(X)$:
$((\text{OBJ}\uparrow)\ \text{SUBJ})_\sigma \multimap \uparrow_\sigma$

The lexeme *kick* serves only as a support verb to turn *Bradshaw's goal* into a verbal expression, so that it forms a clause. It is a collocation of *goal* that, in the context of football match summaries, does not contribute to the meaning of the sentence in a significant way. Hence, X *kicks a goal* means nothing more than $\lambda X.\mathit{goal}(X)$, that is, the verb *kick* simply recopies its object's meaning, with the constraint that its object is the lexeme *goal* in (12):

(12) *kicked* V (↑PRED)='kick⟨(↑SUBJ),(↑OBJ)⟩'
(↑OBJ PRED)=$_c$'goal'
(↑TENSE)=past
$\lambda X.X : (\uparrow\text{OBJ})_\sigma \multimap \uparrow_\sigma$

In this example, the second line is a constraining equation, which is LFG's way of handling collocational constraints; it specifies that this rule can only be applied if the predicate of the object of *kick* is *goal*. And just as for the determiner in (10), the meaning side adds nothing to the overall semantics.

We note here that the semantic description provided by glue semantics does not render obsolete the PRED function. It is still needed to encode purely syntactic information: the name of the lexeme and its sub-categorisation. The verb *kick* could control its own collocations, so we need to have access to the name of the lexeme.

Beautiful, in (1), could be replaced with *spectacular* or *brilliant*, for instance. In these kinds of texts, the semantic difference between these expressions is not significant. The adjectives *beautiful*, *brilliant* and *spectacular*, when they modify *goal*, merely denote a positive appreciation: $\lambda X.\mathit{good}(X)$.

(13) *beautiful* A (↑PRED)='beautiful'
((ADJ ∈ ↑) PRED)=$_c$'goal'
$\lambda X.\mathit{good}(X)$:
$(\text{ADJ} \in \uparrow)_\sigma \multimap (\text{ADJ} \in \uparrow)_\sigma$

The second line is again an LFG constraining equation, which specifies that this rule can only be applied if *beautiful* modifies the lexeme *goal*; the semantic element $\lambda X.\mathit{good}(X)$ will be realised in other ways in different contexts.

The extra lines in the lexical entries are regular, and can be captured using templates, which are the XLE instantiation of LFG's lexical rules. These in fact then correspond very closely to MTT's LFs. For example, for the LF $\texttt{Oper}_1(L)$, which represents the use of support verbs in contexts such as that of *kick* in our examples, the following template could be defined:

(14) @OPER1(L)=
(↑PRED)='%stem⟨(↑SUBJ),(↑OBJ)⟩'
(↑OBJ PRED)=$_c$'L'
$\lambda X.X : (\uparrow \text{OBJ})_\sigma \multimap \uparrow_\sigma$

The constraining equation on the second line restricts the support verb to the particular lexical element with which it is invoked. The third line constructs the meaning by just passing along the meaning of the existing components with no additions. The template is then invoked in the dictionary:

(15) *kick* V @(OPER1 *goal*)
suffer V @(OPER1 *loss*)
have V @(OPER1 *cold*)

Such templates need only be described once for all languages. For example, the Arrernte dictionary contains the following entry for the expression *goal arrerneme* (literally 'put (a) goal'):

(16) *arrerneme* V (OPER1 *goal*)

One problem with this approach is that collocations must be described in the collocate's entry, which is not very elegant and obfuscates the lexicographer's work. Indeed it is a lot easier, for example, to answer the question "how do you intensify *smoker*?" than "what lexemes can *heavy* intensify?". However, this problem can easily be resolved by writing the dictionary in the format of Figure 5 (similar to the *attention* example in Figure 4), where all collocations are listed under their base headword, and using a compiler to build the corresponding XLE lexical entries.

goal [of X]

Bon beautiful/spectacular/brilliant ∼
$\texttt{Oper}_1$ X kicks/scores/gets/makes a ∼

Figure 5: Dictionary entry for *goal*

In the example we have considered so far, the English expression *kick a goal* and its Arrernte equivalent *goal arrerneme* have the same structure. However, this need not be the case. For example, consider the contrast between the following two sentences:

(17) John abandons the baby.

(18) John-le ampe-Ø ipmentye-Ø
John-ERG baby-NOM abandonment-NOM
iwe-me
leave-N.PST
'John abandons the baby'

Both sentences express the meaning *abandon(John,baby)*, but in English, the predicate is expressed by a single verb, while in Arrernte it is expressed by a noun with a support verb. This construction corresponds to an LF called $\texttt{Labor}_{12}$, which denotes a support verb that takes as its subject the first semantic argument of the base of the collocation (here, *John*), the second argument as its direct object (*baby*), and the base itself as its second object (*ipmentye*).[8] The template for $\texttt{Labor}_{12}$ would look like this:

(19) @LABOR12(L)=
(↑PRED)='%stem⟨(↑SUBJ),(↑OBJ),(↑OBJ2)⟩'
(↑OBJ2 PRED)=$_c$'L'
$\lambda X.X : (\uparrow \text{OBJ2})_\sigma \multimap \uparrow_\sigma$

And just as we did for *goal* in (11), we also need a specific entry for *ipmentye* that reflects its behaviour in this collocation, as well as an entry for *iweme* that says it is the $\texttt{Labor}_{12}$ of *ipmentye*:

(20) *ipmentye* N
(↑PRED)='ipmentye'
$\lambda X \lambda Y.$*abandon*(X, Y) :
((OBJ2↑) SUBJ)$_\sigma$ $\multimap$
[((OBJ2↑) OBJ)$_\sigma$ $\multimap$ $\uparrow_\sigma$]
iweme V @(LABOR12 *ipmentye*)

Hence, given the same meaning as input, the grammar produces different structures, as appropriate for the language being processed.

[8] Since both *ampe* and *ipmentye* are in the nominative form, it is hard to determine which is the first and which is the second object, but this question is largely irrelevant here.

Of course, LFs have their limitations too. In the context of MNLG, there are two problems related to lexicalisation that are worth mentioning here. One is that languages sometimes diverge at the semantic level. For example, there is no direct equivalent to the verb *teach* in Arrernte; one has to say *akaltye antheme*, literally 'give knowledge'. This is a collocation of the noun *akaltye* 'knowledge' that can be captured by an LF; but the problem here lies in the fact that the semantic input in Arrernte should be *cause(X,know(Y,Z))*, while in English it would be *teach(X,Y,Z)*. This is different from the abandonment case discussed above: there, one language uses a straightforward realisation, while the other uses a light verb, but there is no need to decompose the meaning of these expressions to see that they are identical; they both have the same semantic representation. For *teach*∼*akaltye antheme*, the two languages do not conceptualise the world in the same way, and these conceptual/semantic differences must be dealt with early in the generation process; the deep realiser must produce different semantic representations depending on the language. At this stage, LFs are irrelevant because we are operating on concepts rather than at the lexical level, where LFs come into play.

Another limitation is that LFs are designed to describe recurrent patterns of collocations. Although most collocations found in languages are instances of a few common patterns, there are many that either express unusual meanings, or that exhibit a very peculiar syntactic or morphological structure. For example, the expression *winning goal* could be described as a collocation. However, the meaning expressed here by *winning* is very specific to this domain, and it cannot be reduced to a recurrent pattern across languages (beyond the equivalent expressions for *winning goal*). *Ad hoc* LFs can still be defined for such collocations, but their use will only be a viable solution in the context of an application within a restricted domain (such as ours).

7 Conclusion

We have proposed a technique for the description of collocations in LFG based on MTT's concept of LFs, in order to solve the problem of complex lexicalisation in NLG. We showed that a direct treatment within LFG's f-structure is not possible because it would require variable values for the attribute PRED, which is not allowed. Also, the semantics of support verbs in particular is tricky and cannot be captured satisfactorily with a PRED attribute. We proposed a treatment using glue semantics, which handles more elegantly the complex correspondence between the semantics and syntax of collocations. The rules that describe collocates use constraining equations so that they apply only in the context of the base of a collocation. We also showed how templates could be used in XLE to define recurrent patterns, effectively defining any given LF once for all languages. The result is an elegant way of describing collocations within the LFG framework. This technique simplifies the task of preparing resources for MNLG by sharing these patterns across languages.

Acknowledgments

We acknowledge the support of ARC grant DP1095443, and thank Mark Johnson for his feedback on the idea.

References

Avery Andrews. 2010. Propositional Glue and the Correspondence Architecture of LFG. *Linguistics and Philosophy*, 33:141–170.

Jury Apresjan, Igor Boguslavsky, Leonid Iomdin, and Leonid Tsinman. 2002. Lexical functions in actual NLP applications. In *Computational Linguistics for the New Millennium: Divergence or Synergy?*, pages 55–72. Peter Lang, Frankfurt.

Peter Austin and Joan Bresnan. 1996. Non-configurationality in Australian aboriginal languages. *Natural Language and Linguistic Theory*, 14(2):215–268.

John Bateman, Christian Matthiessen, Keizo Nanri, and Licheng Zeng. 1991. The re-use of linguistic resources across languages in multilingual generation components. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, volume 2, pages 966–971, Sydney.

John Bateman, Christian Matthiessen, and Licheng Zeng. 1999. Multilingual Natural Language Generation for Multilingual Software: A Functional Linguistic Approach. *Applied Artificial Intelligence*, 13(6):607–639.

Laurent Bourbeau, Denis Carcagno, Eli Goldberg, Richard Kittredge, and Alain Polguère. 1990. Bilingual Generation of Weather Forecasts in an Operations Environment. In *Proceedings of the 13th International Con-*

ference on Computational Linguistics (COLING'90), pages 90–92.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell, Oxford, UK.

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

Lynn Cahill, Christie Doran, Roger Evans, Rodger Kibble, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. 2000. Enabling resource sharing in language generation: an abstract reference architecture. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens.

Aoife Cahill, Martin Forst, and Christian Rohrer. 2007. Stochastic realisation ranking for a free word order language. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG'07)*, pages 17–24, Schloss Dagstuhl, Germany.

Charles B. Callaway and James C. Lester. 2002. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252.

Charles Callaway, Elena Not, Alessandra Novello, Cesare Rocchi, Oliviero Stock, and Massimo Zancanaro. 2005. Automatic Cinematography and Multilingual NLG for Generating Video Documentaries. *Artificial Intelligence*, 165(1):57–89.

Mary Dalrymple. 2001. *Lexical Functional Grammar*, volume 42 of *Syntax and Semantics Series*. Academic Press, New York.

Roger Evans, Paul Piwek, Lynne J. Cahill, and Neil Tipper. 2008. Natural language processing in CLIME, a multilingual legal advisory system. *Natural Language Engineering*, 14(1):101–132.

Ulrich Heid and Sybille Raab. 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.

Lidja Iordanskaja, Myunghee Kim, Richard Kittredge, Benoît Lavoie, and Alain Polguère. 1992. Generation of Extended Bilingual Statistical Reports. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'92)*, pages 1019–1023. Nantes, France.

Aravind Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin.

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

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

Daniel Marcu, Lynn Carlson, and Maki Watanabe. 2000. An Empirical Study in Multilingual Natural Language Generation: What Should A Text Planner Do? In *Proceedings of the 1st International Conference on Natural language Generation (INLG'00)*, pages 17–23.

John T. Maxwell and Ronald M. Kaplan. 1993. The Interface between Phrasal and Functional Constraints. *Computational Linguistics*, 19(4):571–590.

Igor Mel'čuk. 1995. The future of the lexicon in linguistic description and the explanatory combinatorial dictionary. In I.-H. Lee, editor, *Linguistics in the morning calm*, volume 3. Hanshin, Seoul.

Rachel Nordlinger and Joan Bresnan. 2011. Lexical-Functional Grammar: interactions between morphology and syntax. In R. Borsley and K. Börjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*. Wiley-Blackwell, Chichester.

Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. 1995. A Support Tool for Writing Multilingual Instructions. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1398–1404, Montreal.

Carl Pollard and Ivan Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press, Chicago.

Anne-Marie Rassinoux, Robert H. Baud, Jean-Marie Rodrigues, Christian Lovis, and Antoine Geissbühler. 2007. Coupling Ontology Driven Semantic Representation with Multilingual Natural Language Generation for Tuning International Terminologies. In *Proceedings of the 12th World Congress on Health (Medical) Informatics (MEDINFO'07)*, pages 555–559, Brisbane.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Leo Wanner, editor. 1996. *Lexical functions in lexicography and natural language processing*, volume 31 of *Studies in Language Companion Series*. John Benjamins, Amsterdam/Philadelphia.