A full-fledged hierarchical lexicon in LFG: the FrameNet approach

Adam Przepiórkowski

Abstract. The aim of this paper is to propose a fully hierarchical organisation of valency information in Lexical Functional Grammar, inspired by recent LFG work on using templates to encode valency. The particular proposal rather closely follows FrameNet's inheritance hierarchy, makes heavy use of templates to encode multiple inheritance, and avoids the problem of multiple inheritance of semantic resources.

1 Introduction

In many constraint-based linguistic theories, as well as in some lexicographic projects, lexical information is organised hierarchically (e.g. Daelemans et al. 1992). In such a hierarchy, internal nodes represent various generalisations pertaining to various portions of the lexicon. These generalisations are inherited by 'lower' nodes. The 'lowest' nodes — the 'leaves' in the hierarchy — typically correspond to specific lexical items, which inherit generalisations from all the nodes on the way up to the root of the hierarchy, and only add truly idiosyncratic information such as the orthographic form. This approach to the lexicon is an important aspect of Head-driven Phrase Structure Grammar (cf., e.g., Flickinger 1987; Davis 2001), but similar proposals have also been made within Lexicalized Tree-Adjoining Grammar (Vijay-Shanker and Schabes 1992) and within Categorial Grammar (Linden 1992), *inter alia*. Hierarchical organisation is also an important feature of WordNet (Fellbaum 1998; Miller et al. 1990) and FrameNet (Fillmore and Baker 2015; Fillmore, Johnson, et al. 2003; Ruppenhofer et al. 2016). While in all these approaches hierarchies represent mainly syntactic and semantic generalisations, Network Morphology (Corbett and Fraser 1993), based on the lexical representation language DATR (Evans and Gazdar 1996), is concerned with morphological and morphosyntactic generalisations.

To the best of my knowledge, the possibility of adopting such a comprehensive taxonomic approach to the lexicon has never been seriously entertained within Lexical Functional Grammar (Bresnan 1982; Bresnan, Asudeh, et al. 2015; Dalrymple 2001). The aim of this paper is to propose an organisation of the LFG lexicon that is close to

The very model of a modern linguist. Edited by Victoria Rosén and Koenraad De Smedt. BeLLS Vol. 8 (2017), DOI 10.15845/bells.v8i1.1336.

that of FrameNet. The technical side of this proposal is relatively straightforward, assumes the Glue approach to LFG semantics (Dalrymple 1999, 2001), makes heavy use of templates (Asudeh et al. 2008, 2013; Dalrymple, Kaplan, et al. 2004), and does not require any formal extensions to the underlying LFG machinery, but does require some care to avoid the spurious multiple introduction of meaning constructors. In an accompanying paper (Przepiórkowski 2017a), which shares with the current paper most of the material of the initial three sections, I show that this approach to the lexicon also meshes well with my recent proposal *not* to distinguish arguments from adjuncts in LFG (Przepiórkowski 2016).

2 Inheritance in FrameNet

FrameNet organises lexical knowledge with reference to cognitive structures called *frames*. Various lexical items may evoke the same frame. For instance, the Apply_heat frame is evoked by verbs such as BAKE, FRY, GRILL, STEW, etc. (While FrameNet is concerned with various parts of speech, we only deal with the verbal domain here.) Frames also define *frame elements*, i.e. — simplifying a little — semantic roles which are normally expressed by dependents of lexical items evoking the frame. In the case of Apply_heat, typical frame elements are the Cook and the Food, but also the Container that holds the Food to which heat is applied, the Medium through which heat is applied to Food, etc. In examples (1) and (2), verbs evoking the Apply_heat frame are in boldface.[1]

(1) **Boil** [the potatoes]$_{\text{Food}}$ [in a medium-sized pan]$_{\text{Container}}$.

(2) [Drew]$_{\text{Cook}}$ **sauteed** [the garlic]$_{\text{Food}}$ [in butter]$_{\text{Medium}}$.

Frames are linked via a number of relations, including the hierarchical multiple-inheritance relation. For example, Apply_heat inherits semantic roles from both Activity and Intentionally_affect frames, and the latter inherits from Intentionally_act, which in turn inherits from Event (see Figure 1). It is not clear whether this is a design feature of FrameNet or just a reflection of its work-in-progress status, but it happens in current versions of FrameNet (including the latest at the time of writing this paper, version 1.7) that the same role is introduced multiple times in the hierarchy. For example, within the fragment of the inheritance hierarchy in Figure 1, the Agent role is introduced independently at Activity, at Objective_influence (where it is called Influencing_entity; see below) and at Intentionally_act.

Another feature of FrameNet is that inherited roles, as they acquire more specialised meanings, may change names.[2] For example, the agentive role introduced at Objec-

1 These made up examples are taken from the description of the Apply_heat frame at the FrameNet web interface, at https://framenet2.icsi.berkeley.edu/.

2 This correspondence between frame elements of different frames is currently not shown in the web interface to FrameNet, but it is explicitly defined in the distributed version of the lexicon, in the file frRelation.xml.

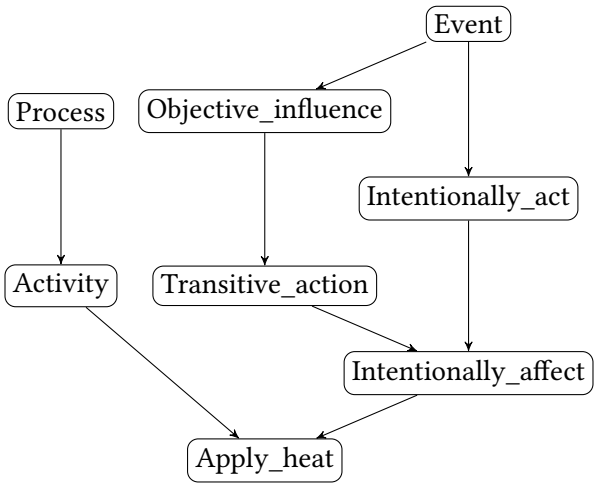

Figure 1: A fragment of the FrameNet 1.7 inheritance hierarchy — all frames from which Apply_heat inherits

tive_influence is actually called Influencing_entity there, but gets renamed to Agent when it is inherited by Transitive_action. The Agent roles of both Transitive_action and Intentionally_act correspond to the (single) Agent role of Intentionally_affect, but the role of Apply_heat corresponding to the Agent roles of both Activity and Intentionally_affect is renamed to Cook. Similarly, the Food role of Apply_heat corresponds to the Patient role of Intentionally_affect and Transitive_action above it (where it is renamed from the Dependent_entity role of Objective_influence). Below, I will simplify by adopting single names for roles related via the inheritance hierarchy. For example, instead of Cook and Food, the respective roles of Apply_heat will be called Agent and Patient, as on the superordinate frames. But, as always, it should be borne in mind that a role on a subordinate frame will usually carry more entailments than the homonymous role on a superordinate frame.

An important aspect of FrameNet is that frame elements correspond to both arguments and adjuncts. For example, among the roles associated with Apply_heat are roles realised by typical adjuncts, such as Manner, Time and Place. A FrameNet reflex of the argument/adjunct dichotomy is its categorisation of roles into core (corresponding to arguments) and non-core (corresponding to adjuncts), but the criteria used for deciding whether a role is core or not suffer from the usual problems (discussed, e.g., in Przepiórkowski 2016) of providing only partial tests or being pairwise incompatible.[3]

What is interesting is that inheritance may change the coreness status of a role. For example, at the Event frame the roles Time and Place are marked as core, proba-

3 See Przepiórkowski (2017a) for further discussion of the core/non-core distinction in FrameNet.

bly reflecting the intuition that verbs directly evoking this frame, such as HAPPEN or OCCUR, seem to require their presence: #*This event occurred* (but this impression is contested below). However, the same roles are treated as non-core on almost all of the 27 frames directly subordinate to Event.[4] The reverse situation happens in the case of the Existence frame, where Time and Place are non-core, but become core on its directly subordinate frame, Circumbscribed_existence. Such changes of coreness seem to bring non-monotonicity to the otherwise monotonic inheritance relation in FrameNet. Below, we will see how such apparently non-monotonic behaviour can be modelled via the monotonic means of Lexical Functional Grammar.

3 Valency in LFG

As is common in LFG, I assume the existence of a level of representation which encodes the semantic argument structure, i.e., which contains information about semantic (or thematic) roles such as Agent or Goal. Traditionally, semantic forms — values of PRED — served this purpose in Lexical Functional Grammar (Kaplan and Bresnan 1982). Alternatively, we could employ the distinct level of argument structure of Butt et al. 1997. Instead, I build here on more recent work and assume the formalisation of argument structure within the semantic structure (Asudeh and Giorgolo 2012; Asudeh, Giorgolo, and Toivonen 2014; Findlay 2016). For example, Asudeh and Giorgolo (2012, p. 78) propose the f-structure and s-structure in Figure 2 for the sentence *Kim tapped Sandy with Excalibur*.

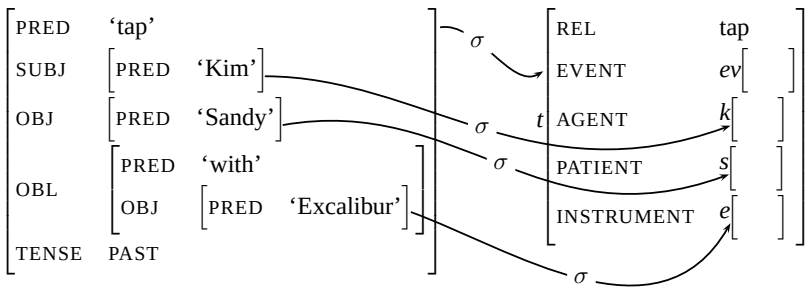

Figure 2: F-structure and s-structure for the sentence *Kim tapped Sandy with Excalibur*

Much of the work in this thread assumes that some of the semantic attributes — those which correspond to arguments in the scope of (Lexical) Mapping Theory (Bresnan and Kanerva 1989) as formalised in Findlay (2016) — are called ARG_1, ARG_2, etc., instead of the more mnemonic names such as AGENT and PATIENT, but — as I am not concerned with LMT in this paper — I will continue to use these more intuitive names.

4 The only exception is the Emergency frame, which treats Time as core.

What kind of lexical entries give rise to such f- and s-structures? Let us consider a simpler case, that of the transitive verb *devoured*; the first version of the lexical entry for this verb is shown in (3).

(3) *devoured* V $(\uparrow \textsc{pred}) = \text{`}\textsc{devour}\text{'}$
$(\uparrow_\sigma \textsc{rel}) = \textsc{devour}$
$\lambda e.\ \mathit{devour}(e) : (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$
$(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{agent})$
$\lambda P \lambda x \lambda e.\ P(e) \wedge \mathit{agent}(e, x) :$
$[(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \textsc{agent}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$
$(\uparrow \textsc{obj})_\sigma = (\uparrow_\sigma \textsc{patient})$
$\lambda P \lambda x \lambda e.\ P(e) \wedge \mathit{patient}(e, x) :$
$[(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \textsc{patient}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$
$(\uparrow \textsc{tense}) = \textsc{past}$
$\lambda P.\ \exists e\ P(e) \wedge \mathit{past}(e) : [(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

There are four natural parts of this lexical entry: 1) the idiosyncratic part, defining PRED,[5] as well as the corresponding s-structure attribute REL,[6] and introducing the basic meaning constructor containing the *devour* relation in its meaning representation; 2) the part saying that the subject grammatical function realises the agent semantic relation; 3) the analogous part defining the correspondence between the object and the patient; 4) the part adding tense information to the f-structure and to the meaning representation, as well as defining the existential closure over the event variable. In the case of the sentence *Godzilla devoured Kim*, this lexical entry gives rise to the f- and s-structures in Figure 3, as well as to the instantiated meaning constructors in (4).

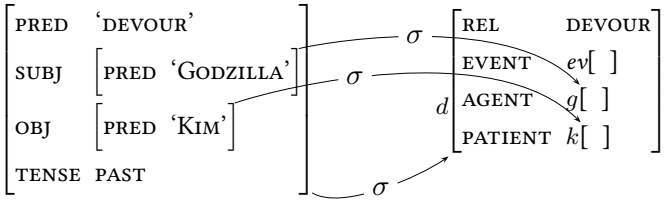

Figure 3: F-structure and s-structure for the sentence *Godzilla devoured Kim*

5 As noted already in Dalrymple, Hinrichs, et al. (1993, pp. 13–14) and Kuhn (2001, § 1.3.3), Glue makes PRED largely superfluous. Here I follow Asudeh and Giorgolo (2012) and retain PRED, but with values reflecting the predicate *sans* its valency — hence, the value of PRED in (3) is defined as 'DEVOUR' rather than 'DEVOUR⟨SUBJ, OBJ⟩'.

6 The existence of the semantic attribute REL is assumed — but not formally introduced — in various recent papers (including Asudeh and Giorgolo 2012; Asudeh et al. 2008, 2013; Asudeh, Giorgolo, and Toivonen 2014; Findlay 2016). The following lexical entries make the introduction of this attribute explicit.

(4)
1. $\lambda e.\ \textit{devour}(e) : \textit{ev} \multimap d$
2. $\lambda P \lambda x \lambda e.\ P(e) \wedge \textit{agent}(e, x) : [\textit{ev} \multimap d] \multimap g \multimap \textit{ev} \multimap d$
3. $\lambda P \lambda x \lambda e.\ P(e) \wedge \textit{patient}(e, x) : [\textit{ev} \multimap d] \multimap k \multimap \textit{ev} \multimap d$
4. $\lambda P.\ \exists e\ P(e) \wedge \textit{past}(e) : [\textit{ev} \multimap d] \multimap d$

These instantiated meaning constructors, together with the instantiated meaning constructors in (5), introduced by the lexical entries of *Godzilla* and *Kim*, may be used to derive the expected meaning representation for the whole sentence: $\exists e\ \textit{devour}(e) \wedge \textit{agent}(e, \textit{godzilla}) \wedge \textit{patient}(e, \textit{kim}) \wedge \textit{past}(e)$.

(5)
5. $\textit{godzilla} : g$
6. $\textit{kim} : k$

One possible proof is shown in (6).

(6)
7. $\lambda x \lambda e.\ \textit{devour}(e) \wedge \textit{agent}(e, x) : g \multimap \textit{ev} \multimap d$ (from 2 and 1)
8. $\lambda e.\ \textit{devour}(e) \wedge \textit{agent}(e, \textit{godzilla}) : \textit{ev} \multimap d$ (from 7 and 5)
9. $\lambda x \lambda e.\ \textit{devour}(e) \wedge \textit{agent}(e, \textit{godzilla}) \wedge \textit{patient}(e, x) : k \multimap \textit{ev} \multimap d$ (from 3 and 8)
10. $\lambda e.\ \textit{devour}(e) \wedge \textit{agent}(e, \textit{godzilla}) \wedge \textit{patient}(e, \textit{kim}) : \textit{ev} \multimap d$ (from 9 and 6)
11. $\exists e\ \textit{devour}(e) \wedge \textit{agent}(e, \textit{godzilla}) \wedge \textit{patient}(e, \textit{kim}) \wedge \textit{past}(e) : d$ (from 4 and 10)

Obviously, apart from the first — idiosyncratic — part of the lexical entry (3), the other three parts will also occur in many other lexical entries, so it makes sense to encode them as templates (Dalrymple, Kaplan, et al. 2004), as in (7)–(9).[7]

(7) $\textsc{agent} := (\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{agent})$
$\lambda P \lambda x \lambda e.\ P(e) \wedge \textit{agent}(e, x) :$
$[(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \textsc{agent}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

(8) $\textsc{patient} := (\uparrow \textsc{obj})_\sigma = (\uparrow_\sigma \textsc{patient})$
$\lambda P \lambda x \lambda e.\ P(e) \wedge \textit{patient}(e, x) :$
$[(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \textsc{patient}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$

(9) $\textsc{past} := (\uparrow \textsc{tense}) = \textsc{past}$
$\lambda P.\ \exists e\ P(e) \wedge \textit{past}(e) : [(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap \uparrow_\sigma$

7 In the actual templates, the parts defining the correspondence between a semantic argument and a grammatical function will be more complex, to allow for diathesis (see Asudeh, Giorgolo, and Toivonen 2014; Findlay 2016).

With these templates in hand, the lexical entry for *devoured* simplifies to the second (final) version in (10).

(10) *devoured* V $(\uparrow \textsc{pred}) = \text{`}\textsc{devour}\text{'}$
$(\uparrow_\sigma \textsc{rel}) = \textsc{devour}$
$\lambda e.\ \textit{devour}(e) : (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma$
@AGENT
@PATIENT
@PAST

LFG templates may call other templates, and in this sense they form a kind of hierarchy, although it is a very different kind of hierarchy than, say, the type hierarchy of Head-driven Phrase Structure Grammar (Pollard and Sag 1994). In the context of valency, this possibility was explored by Asudeh et al. (2008, 2013) in their account of English and Swedish *way*-constructions, as in: *Bill elbowed his way through the crowd.* Similarly, Asudeh, Giorgolo, and Toivonen (2014) make use of such embedded template calls for example when defining a prototypical transitive argument structure as in (11).

(11) AGENT-PATIENT := @AGENT @PATIENT

The proposal presented in the following section may be seen as taking the approach summarised above to its logical conclusion.

4 FrameNet-inspired extended valency in LFG

One important difference between LFG work on valency and FrameNet work on semantic roles (or — more generally — frame elements) concerns the argument/adjunct distinction: in LFG valency is understood traditionally, as concerned only with arguments, while FrameNet semantic roles correspond to both arguments and adjuncts. In Section 4.1 we will see that extending the LFG treatment of valency to all dependents — arguments and adjuncts alike — is relatively straightforward. Another important difference is that the idea of the inheritance of valency information has only been applied to a very specific construction in LFG, namely, to the *way*-constructions in a few languages, discussed in Asudeh et al. (2008, 2013), while it is a conspicuous feature of the whole lexicon in FrameNet. In Section 4.2 we will see how this holistic approach of FrameNet may be ported to LFG.

4.1 Frame elements via templates

In LFG, as in most other theories, arguments of a head are selected by this head, i.e., they are logical arguments, while adjuncts are not selected, acting instead as logical functors. In the neo-Davidsonian representations assumed here (Parsons 1990), this distinction is not visible in the final semantic representations, where each dependent — whether an argument or an adjunct — typically introduces a separate predicate (*agent*,

instrument, *beneficiary*, etc.). However, this distinction is present in the grammar and in the lexicon: verifying that a given head may combine with a given dependent is the responsibility of the head when the dependent is an argument, but it is the job of the dependent when the dependent is an adjunct. In practice, there is a lot of LFG work on the first scenario, i.e., on what arguments are required by what heads, but hardly any work on the second scenario, i.e., on what adjuncts are compatible with what heads. The often unspoken assumption is that particular adjuncts, e.g., manner or durative, select only those heads which are semantically compatible, but — to the best of my knowledge — this notion of semantic compatibility has never been formalised or made precise in LFG.

One advantage of FrameNet is that it does model which predicates are compatible with which semantic roles expressed as adjuncts, i.e., it treats adjuncts just as arguments in this respect. For example, referring to the fragment of the inheritance hierarchy in Figure 1, the non-core Duration role is introduced high in the hierarchy, at the Event frame; Means and Purpose are introduced at a subordinate frame, Intentionally_act; Instrument is introduced even lower, at Intentionally_affect; and Medium — also a non-core frame element — is only introduced at a leaf in the hierarchy, at Apply_heat. Hence, in order to implement the FrameNet approach in LFG, templates should be provided not only for typical argument roles, such as AGENT and PATIENT in (7)–(8) above, but also for roles typically realised as adjuncts.

Let us start with *for*-benefactives, which — as discussed for example in Needham and Toivonen (2011, pp. 409–410, 417) — have mixed argument/adjunct properties and should perhaps be analysed as arguments of some predicates and adjuncts of other predicates. For this reason, the analogue of the equation $(\uparrow \textsc{subj})_\sigma = (\uparrow_\sigma \textsc{agent})$ in the above template (7) is the more complex equation $(\uparrow \{\textsc{obl}_{\textsc{ben}} \mid \textsc{adj} \in\})_\sigma = (\uparrow_\sigma \textsc{ben})$, which establishes the correspondence between the semantic attribute BEN(eficiary) and either an argument ($\textsc{obl}_{\textsc{ben}}$) or an adjunct. The definition of the first version of the BENEFICIARY template is given in (12).[8,9]

(12) $$\begin{aligned}\textsc{beneficiary} := \; & (\uparrow \{\textsc{obl}_{\textsc{ben}} \mid \textsc{adj} \in\})_\sigma = (\uparrow_\sigma \textsc{ben}) \\ & \lambda P \lambda x \lambda e.\ P(e) \wedge \textit{beneficiary}(e, x) : \\ & [(\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \textsc{ben}) \multimap (\uparrow_\sigma \textsc{event}) \multimap \uparrow_\sigma\end{aligned}$$

8 In fact, a more comprehensive definition of AGENT should also take into consideration two possible realisations — as the subject or as the agentive oblique: $(\uparrow \{\textsc{subj} \mid \textsc{obl}_{\textsc{ag}}\})_\sigma = (\uparrow_\sigma \textsc{agent})$; see Findlay (2016).

9 This template assumes that there is an independent mechanism, such as the traditional PRED coupled with the principles of completeness and coherence, which specifies whether a given predicate combines with an $\textsc{obl}_{\textsc{ben}}$ argument or not. In the current setup, with simpler PRED values and no coherence or completeness (see fn. 5), two different templates would have to be defined: one for benefactive arguments, and another for benefactive adjuncts. This technical inconvenience does not arise on the approach of Przepiórkowski (2016, 2017a).

One piece of information that is missing above is that this template is only concerned with *for*-benefactives. For the sake of concreteness, let us assume that *for* in sentences such as *Kim did it for Sandy* is an asemantic preposition, i.e., that it introduces the attribute PFORM with the value FOR, see (13), and that it is a co-head with the following nominal phrase, see the second disjunct under NP in (14).

(13) *for* P (↑ PFORM) = FOR

(14) PP → P NP
↓=↑ ↓ = (↑ OBJ) | ↓=↑

Then, a modification — using the local name %B — of the template in (12) will do. The second (final) version of the definition of the BENEFICIARY template is given in (15).

(15) BENEFICIARY := %B = (↑ {OBL$_{\text{BEN}}$ | ADJ ∈})
(%B PFORM) =$_c$ FOR
%B$_\sigma$ = (↑$_\sigma$ BEN)
$\lambda P \lambda x \lambda e.\ P(e) \wedge$ *beneficiary*(e, x) :
$[(\uparrow_\sigma$ EVENT) ⊸ $\uparrow_\sigma] \multimap (\uparrow_\sigma$ BEN) ⊸ (↑$_\sigma$ EVENT) ⊸ $\uparrow_\sigma$

This template, just as the templates for AGENT and PATIENT, introduces into the meaning representation a specific predicate, *beneficiary*, and the particular *for*-PP only provides an (e-type) argument for this predicate. The contribution of manner adjuncts, such as *nicely*, is different: it is the role of particular adverbs of manner, rather than the MANNER template, to introduce specific predicates, e.g., *nicely*. Assuming a lexical entry for *nicely* as in (16), the MANNER template may be defined as in (17).[10]

(16) *nicely* Adv (↑ PRED) = 'NICELY'
(↑$_\sigma$ REL) = NICELY
$\lambda e.$ *nicely*(e) : (↑$_\sigma$ REL) ⊸ $\uparrow_\sigma$

(17) MANNER := (↑$_\sigma$ MANNER) = (↑ ADJ ∈)$_\sigma$
$\lambda P \lambda Q \lambda e.\ P(e) \wedge Q(e)$:
$[(\uparrow_\sigma$ EVENT) ⊸ $\uparrow_\sigma] \multimap$
$[(\uparrow_\sigma$ MANNER REL) ⊸ (↑$_\sigma$ MANNER)] ⊸
(↑$_\sigma$ EVENT) ⊸ $\uparrow_\sigma$

An analogous template, shown in (18), is required for locative phrases such as *in Warsaw*.

(18) PLACE := (↑$_\sigma$ PLACE) = (↑ ADJ ∈)$_\sigma$
$\lambda P \lambda Q \lambda e.\ P(e) \wedge Q(e)$:

10 The following constraint may be added to the lexical entry of *nicely* to make sure that it only plays the semantic role of manner: (MANNER ↑$_\sigma$).

$$[(\uparrow_\sigma \ \textsc{event}) \multimap \uparrow_\sigma] \multimap$$
$$[(\uparrow_\sigma \ \textsc{place rel}) \multimap (\uparrow_\sigma \ \textsc{place})] \multimap$$
$$(\uparrow_\sigma \ \textsc{event}) \multimap \uparrow_\sigma$$

The only difference between *nicely* and *in Warsaw* is the kind of relation, Q, provided by the dependent: in the case of *nicely* it was $\lambda e.nicely(e)$, in the case of *in Warsaw* it should be $\lambda e.in(e, warsaw)$. This means that the lexical entry for the semantic preposition *in*, shown in (19), is a little more complex than that for *nicely*, as it must take care of the object of this preposition (here: *Warsaw*).[11]

(19) *in* P $(\uparrow \textsc{pred}) = \text{`}\textsc{in}\text{'}$
$(\uparrow_\sigma \textsc{rel}) = \textsc{in}$
$(\uparrow_\sigma \textsc{loc}) = (\uparrow \textsc{obj})_\sigma$
$\lambda x \lambda e.\ in(e, x) \ : \ (\uparrow_\sigma \textsc{loc}) \multimap (\uparrow_\sigma \textsc{rel}) \multimap \uparrow_\sigma$

Let us take a look at these templates and lexical entries in action, in the sentence *Kim danced nicely for Sandy in Warsaw*. Assuming appropriate syntactic rules, standard lexical entries for proper names, and a lexical entry for *danced* which obligatorily calls the AGENT template and optionally calls the tempates BENEFICIARY, MANNER and PLACE (perhaps among many others), the f-structure and s-structure shown in Figure 4 will result. Moreover, the instantiated meaning constructors in (20) will be added to the

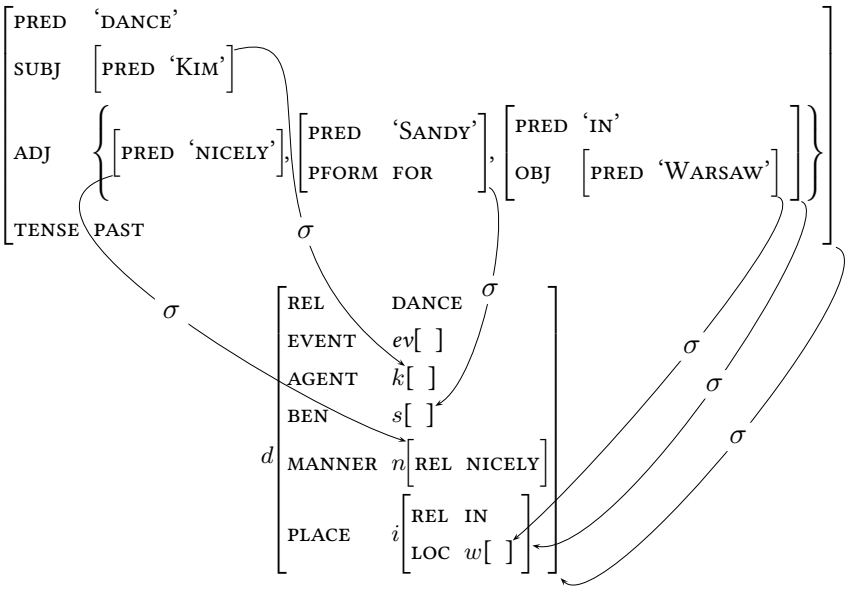

Figure 4: F-structure and s-structure for the sentence *Kim danced nicely for Sandy in Warsaw*

11 Again, the following constraint may be added to the lexical entry for *in* to make sure that the PP based on this lexical entry only plays the semantic role of place: $(\textsc{place} \uparrow_\sigma)$.

pool (1–3 from the lexical entries of proper names, 4 directly from the lexical entry of *danced*, 5 from the PAST template called there, 6–9 from the AGENT, BENEFICIARY, MANNER and PLACE templates called there, 10 from the lexical entry of *nicely* and 11 from the lexical entry of *in*).

(20)
1. $\mathit{kim} : k$
2. $\mathit{sandy} : s$
3. $\mathit{warsaw} : w$
4. $\lambda e.\ \mathit{dance}(e) : \mathit{ev} \multimap d$
5. $\lambda P.\ \exists e\ P(e) \wedge \mathit{past}(e) : [\mathit{ev} \multimap d] \multimap d$
6. $\lambda P \lambda x \lambda e.\ P(e) \wedge \mathit{agent}(e, x) : [\mathit{ev} \multimap d] \multimap k \multimap \mathit{ev} \multimap d$
7. $\lambda P \lambda x \lambda e.\ P(e) \wedge \mathit{beneficiary}(e, x) : [\mathit{ev} \multimap d] \multimap s \multimap \mathit{ev} \multimap d$
8. $\lambda P \lambda Q \lambda e.\ P(e) \wedge Q(e) : [\mathit{ev} \multimap d] \multimap [(n\ \textsc{rel}) \multimap n] \multimap \mathit{ev} \multimap d$
9. $\lambda P \lambda Q \lambda e.\ P(e) \wedge Q(e) : [\mathit{ev} \multimap d] \multimap [(i\ \textsc{rel}) \multimap i] \multimap \mathit{ev} \multimap d$
10. $\lambda e.\ \mathit{nicely}(e)\ :\ (n\ \textsc{rel}) \multimap n$
11. $\lambda x \lambda e.\ \mathit{in}(e, x)\ :\ w \multimap (i\ \textsc{rel}) \multimap i$

It is easy to see that these constructors give rise to the expected meaning representation for this sentence, cf. (21).

(21)
12. $\lambda e.\ \mathit{in}(e, \mathit{warsaw})\ :\ (i\ \textsc{rel}) \multimap i$ (from 11 and 3)
13. $\lambda x \lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, x) : k \multimap \mathit{ev} \multimap d$ (from 6 and 4)
14. $\lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) : \mathit{ev} \multimap d$ (from 13 and 1)
15. $\lambda x \lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, x) : s \multimap \mathit{ev} \multimap d$ (from 7 and 14)
16. $\lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) : s \multimap \mathit{ev} \multimap d$ (from 15 and 2)
17. $\lambda Q \lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) \wedge Q(e) :$ $[(n\ \textsc{rel}) \multimap n] \multimap \mathit{ev} \multimap d$ (from 8 and 16)
18. $\lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) \wedge \mathit{nicely}(e) : \mathit{ev} \multimap d$ (from 17 and 10)
19. $\lambda Q \lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) \wedge \mathit{nicely}(e) \wedge Q(e) :$ $[(i\ \textsc{rel}) \multimap i] \multimap \mathit{ev} \multimap d$ (from 9 and 18)
20. $\lambda e.\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) \wedge \mathit{nicely}(e) \wedge$ $\mathit{in}(e, \mathit{warsaw}) : \mathit{ev} \multimap d$ (from 19 and 12)
21. $\exists e\ \mathit{dance}(e) \wedge \mathit{agent}(e, \mathit{kim}) \wedge \mathit{beneficiary}(e, \mathit{sandy}) \wedge \mathit{nicely}(e) \wedge$ $\mathit{in}(e, \mathit{warsaw}) \wedge \mathit{past}(e) : d$ (from 5 and 20)

In summary, it is possible to define templates introducing various types of dependents, both arguments and adjuncts. In Section 4.2 we will see how to call such templates in a way that does not cause massive redundancy in the resulting description.

4.2 Frame inheritance via template inheritance

The formalisation of FrameNet's multiple-inheritance hierarchy within LFG is relatively straightforward, although some care needs to be taken to avoid multiple introduction of glue resources. I will illustrate such a formalisation with the Apply_heat frame evoked by BOIL.

FrameNet lists 15 semantic roles of the Apply_heat frame. Many of these roles, including Time and Place, but also Agent and Patient, are elements of many different frames. However, an inheritance hierarchy makes it possible to avoid redundancy by introducing particular semantic roles only once or a couple of times in the appropriate place(s) of the hierarchy. Following FrameNet, I assume that the maximally general frame Event introduces the following seven roles potentially realised as dependents of lexical units evoking this frame:[12] Place, Time, Duration, Explanation, Frequency, Manner and Timespan. As mentioned in Section 2, the first two, Place and Time, are marked as core. As also mentioned there, the criteria used to distinguish core and non-core frame elements in FrameNet mirror the vague and pairwise incompatible criteria usually invoked to distinguish arguments from adjuncts. Here, I assume that the intuition behind coreness strongly correlates with the intuition of obligatoriness: the frame elements which are marked as core are usually either syntactically or semantically obligatory in some sense. This also seems to be the reason for marking Place and Time as core: at first sight verbs which directly evoke the Event frame, like HAPPEN or OCCUR, seem to require Time and/or Place. However, this intuition is more naturally explained with a reference to Grice's Maxim of Quantity — arguments similar to those in Goldberg and Ackerman (2001) may be given showing that the requirement of Time or Place is purely pragmatic and may be overriden, as in the following sentence adduced by an anonymous reviewer: *That long-anticipated event didn't occur after all*, or the attested:[13] *Scientists manipulate brains of mice to make them think fake event really occurred*. Obviously, the Event frame pertains to situations which normally occur at some time and at some place, so in this sense Time and Place are semantically obligatory, but the same can be said about all frames inheriting from Event, on which, however, Time and Place are not marked as core. For this reason, I will treat Place and Time as non-core frame elements of Event, just as these roles are treated in FrameNet on frames subordinate to Event. More generally, I am not aware of convincing cases of a role changing its status from core on a superordinate frame to non-core on a subordinate frame, so I will not model this possibility below.

In LFG, frames are naturally encoded as templates which call particular templates corresponding to frame elements. We will assume that templates corresponding to core frame elements are called obligatorily, and those corresponding to non-core

12 I ignore here another type of FrameNet roles, 'core unexpressed', not realisable by dependents.

13 http://www.independent.co.uk/news/science/is-it-inception-total-recall-no-science-fact-false-implanted-in-mice-brains-8732466.html

frame elements are called optionally.[14] In the case of the Event frame, we define the EVENT_FRAME template in (22) which optionally calls templates such as PLACE (defined above), DURATION, etc.

(22) EVENT_FRAME := (@PLACE) (@TIME) (@DURATION) (@EXPLANATION) (@FREQUENCY) (@MANNER) (@TIMESPAN)

An immediately subordinate frame, Intentionally_act, introduces a few additional roles, including the obligatory Agent and the optional Domain, and inherits most of the roles introduced by Event, apart from Timespan.[15] This may be represented via the template in (23).

(23) INTENTIONALLY_ACT_FRAME := @EVENT_FRAME $\neg(\uparrow_\sigma$ TIMESPAN) @AGENT (@DOMAIN) ...

Note that the way Timespan is *not* inherited, via the negative constraint $\neg(\uparrow_\sigma$ TIMESPAN),[16] actually preserves the monotonicity of inheritance, as this semantic role is defined on the higher frame as optional. Similarly, the obligatoriness of a role which is specified as optional on a superordinate frame may be ensured by an existential constraint such as $(\uparrow_\sigma$ MANNER),[17] as in the case of frames evoked by verbs such as BEHAVE, TREAT or WORD, for which the expression of Manner is obligatory.[18]

Another frame, inheriting from Event and adding a few more semantic roles including the obligatory Agent,[19] is Objective_influence, shown in (24), which is a superordi-

14 This is a vast oversimplification, as there is no strong assumption in FrameNet that all core roles are syntactically obligatory; syntactic obligatoriness is just one of a number of partial criteria assumed in FrameNet for deciding whether a role is core or non-core. Moreover, semantic obligatoriness cuts across the class of non-core roles and is the basis of the distinction within this class between peripheral roles (semantically obligatory but — unlike core roles — not central to the meaning of the frame) and extrathematic roles (semantically optional). As discussed in Przepiórkowski (2016, pp. 562–563), dependents may be obligatory in various ways and for various reasons, some of them of pragmatic nature, and I believe that the issue of the proper modelling of these various aspects of obligatoriness requires substantial research.

15 Also Duration is not mentioned in the description of Intentionally_act in the November 2016 release of FrameNet. It is not clear to me whether these omissions are intentional, but I will use the lack of Timespan here to illustrate how such an apparently non-monotonic aspect may be modelled in LFG.

16 As noted by an anonymous reviewer, it would be more elegant not to 'unpack' the TIMESPAN template this way, but rather have a constraint such as ¬@TIMESPAN. However, this would have the effect of negating a sequence — normally interpreted as conjunction — of two statements, one of which is a constructor. As I am not sure what this would mean (but see Przepiórkowski 2017b for a suggestion), I propose this clearer — even if less elegant — encoding.

17 In this case the more elegant obligatory call to the MANNER template would also do (cf. the previous footnote). However, if such obligatory manners are treated as arguments, a separate template would have to be defined (cf. fn. 9).

18 This is not what actually happens in the current release of FrameNet, as there frames evoked by such verbs are not integrated in the inheritance hierarchy.

19 In FrameNet, this role is called Influencing_entity on this frame, but it is renamed to Agent in the subordinate frame Transitive_action.

nate frame of Transitive_action, shown in (25), which in turn introduces the obligatory Patient role.

(24) OBJECTIVE_INFLUENCE_FRAME := @EVENT_FRAME @AGENT
(@CIRCUMSTANCES) ...

(25) TRANSITIVE_ACTION_FRAME := @OBJECTIVE_INFLUENCE_FRAME
@PATIENT ...

Note that, unlike the EVENT_FRAME template, which corresponds to a root frame, the above two templates contain calls to templates corresponding to immediately superordinate frames, thus encoding inheritance.

Both Transitive_action and Intentionally_act are immediately superordinate frames of Intentionally_affect, as formalised in (26).

(26) INTENTIONALLY_AFFECT_FRAME := @INTENTIONALLY_ACT_FRAME
@TRANSITIVE_ACTION_FRAME ...

The technically unfortunate effect of this is that the template call @AGENT is inherited twice.[20] This is a potential problem as this template includes a meaning constructor — see the second and third lines of (7) — so two copies of this constructor will be present whenever the @INTENTIONALLY_AFFECT_FRAME template is called.[21] There is a straightforward solution to this problem, though, consisting in the following modification of the AGENT template shown in (27), to be compared with the previous definition in (7).

20 Also various other templates are inherited twice, but since they are optional in the first place, this will not lead to the problem discussed here.

21 An anonymous reviewer contests the view that this is a potential problem, saying that "the problem with multiple introduction of glue resources seems to come from a confusion between the mechanism (templates) that express generalizations over entries for lexical and morphological formatives, the structures that those entries describe, and the operations that apply to those structures. Inheritance by template invocation in LFG just gives pieces of text that are used to specify entries that then enter into the grammar and Glue interpreters. I don't know that there is an assumption anywhere that 2 copies of the same text in a lexical scope leads to different behavior than a single instance, even if the entry ultimately has resource sensitive components. The confusion is between the template mechanism for specifying lexical formatives (which basically operates by manipulation and substitution of text strings) and whatever interpretation (like glue deductions) applies to the so-specified formatives. @AGENT @AGENT (or any other stutter) should be the same as a single @AGENT in the specification how to create a formative before it enters into the combinatorial operations of the grammar or semantics…" If this is right, than the slight complication introduced below is not necessary. However, as LFG lacks a comprehensive mathematical formalisation comparable to the formalisation of Head-driven Phrase Structure Grammar (as provided in Richter 2000), it is not clear to me whether a repeated call to a template containing a semantic resource should be interpreted as resulting in one or multiple copies of the resource, and for this reason I provide a solution for the worst-case scenario. This and other questions about the formal status of meaning constructors and semantic structures are addressed in Przepiórkowski (2017b).

(27) AGENT := ($(\uparrow \text{SUBJ})_\sigma = (\uparrow_\sigma \text{AGENT})$
$\lambda P \lambda x \lambda e.\ P(e) \wedge \mathit{agent}(e, x)$:
$[(\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma] \multimap (\uparrow_\sigma \text{AGENT}) \multimap (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$)
$(\uparrow_\sigma \text{AGENT})$

In the above template, the content of the previous version of this template is made jointly optional (see the parentheses in bold), and a non-optional constraint is added ensuring that the semantic AGENT feature is defined. Note that multiple occurrences of the $(\uparrow_\sigma \text{AGENT})$ constraint have exactly the same effect as a single occurrence, so multiple inheritance of this part of the template is not harmful. The only place in the grammar that assigns a value to the AGENT feature is the optional part of the new AGENT template in (27), so — in case of multiple calls to this template — at least one copy of this optional part must actually be used. On the other hand, at most one may be used, as more would introduce multiple copies of the meaning constructor within the optional part. Each such constructor causes a consumption of the glue resource introduced by the subject, corresponding to the value of the AGENT attribute, and only one such resource is introduced by the subject. Hence, exactly one of the multiple copies of the optional part of the AGENT template will actually be used.

Returning to the running example, the Apply_heat frame in (28) also inherits from two superordinate frames, Intentionally_affect and Activity (the latter not discussed here), and also introduces a few specific roles such as Container and Medium.

(28) APPLY_HEAT_FRAME := @INTENTIONALLY_AFFECT_FRAME
@ACTIVITY_FRAME (@CONTAINER) (@MEDIUM) ...

With such a hierarchy of templates, the lexical entry for *boiled*, introducing the many possible dependents of this verb, boils down to (29).

(29) *boiled* V $(\uparrow \text{PRED}) = \text{'BOIL'}$
$\lambda e.\ \mathit{boil}(e) : (\uparrow_\sigma \text{EVENT}) \multimap \uparrow_\sigma$
@APPLY_HEAT_FRAME
@PAST

In practice, different verbs belonging to the same frame may additionally introduce specific — possibly different — morphosyntactic constraints on the realisation of the same role, as is the case with the verbs BEGIN and ENTER evoking the Activity_start frame: only BEGIN may realise the Activity role as an infinitival phrase (*begin to negotiate*) and only ENTER may realise it as an *into*-PP (*enter into negotiations*).

5 Conclusion

In this paper I proposed to carry over the main ideas of FrameNet to LFG: introduce all kinds of dependents lexically (which does not preclude constructional analyses like

that of Asudeh et al. 2013), and organise such extended valency information hierarchically, so as to avoid redundancy and capture generalisations. This proposal may be seen as pushing to the limit some of the ideas presented in Asudeh and Giorgolo (2012), Asudeh, Giorgolo, and Toivonen (2014) and Asudeh et al. (2008, 2013). While the present paper proposes a way to encode a FrameNet-like hierarchical valency lexicon in standard LFG, an accompanying paper (Przepiórkowski 2017a) shows that this approach to valency meshes particularly well with my earlier proposal *not* to distinguish arguments from adjuncts in LFG (Przepiórkowski 2016) and helps remove the last vestiges of the ill-defined argument/adjunct distinction from this linguistic framework.

Acknowledgments

This paper is dedicated to Helge Dyvik, who is obviously no stranger not only to LFG and formal semantics, but also to hierarchical lexicons (Dyvik 2004). Many thanks are due to the three anonymous reviewers of this volume, whose comments have led to numerous improvements; I also benefitted from comments by Michael Ellsworth, Jamie Y. Findlay, and the audience of the 23rd South of England LFG meeting (in May 2017). The research reported here is partially supported by the Polish Ministry of Science and Higher Education within the CLARIN ERIC programme 2016–2018 (http://clarin.eu/).

References

Asudeh, Ash, Mary Dalrymple, and Ida Toivonen (2008). "Constructions with Lexical Integrity: Templates as the Lexicon–Syntax Interface". In: *The Proceedings of the LFG'08 Conference*. Ed. by Miriam Butt and Tracy Holloway King. University of Sydney, Australia: CSLI Publications, pp. 68–88.

– (2013). "Constructions with Lexical Integrity". In: *Journal of Language Modelling* 1.1, pp. 1–54.

Asudeh, Ash and Gianluca Giorgolo (2012). "Flexible Composition for Optional and Derived Arguments". In: *The Proceedings of the LFG'12 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, pp. 64–84.

Asudeh, Ash, Gianluca Giorgolo, and Ida Toivonen (2014). "Meaning and Valency". In: *The Proceedings of the LFG'14 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, pp. 68–88.

Bresnan, Joan, ed. (1982). *The Mental Representation of Grammatical Relations*. MIT Press Series on Cognitive Theory and Mental Representation. Cambridge, MA: The MIT Press.

Bresnan, Joan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler (2015). *Lexical-Functional Syntax*. 2nd ed. Blackwell Textbooks in Linguistics. Wiley-Blackwell.

Bresnan, Joan and Jonni M. Kanerva (1989). "Locative Inversion in Chichewa: A Case Study of Factorization in Grammar". In: *Linguistic Inquiry* 20.1, pp. 1–50.

Butt, Miriam, Mary Dalrymple, and Anette Frank (1997). "An Architecture for Linking Theory in LFG". In: *The Proceedings of the LFG'97 Conference*. Ed. by Miriam Butt and Tracy Holloway King. University of California, San Diego: CSLI Publications.

Corbett, Greville G. and Norman M. Fraser (1993). "Network Morphology: a DATR Account of Russian Nominal Inflection". In: *Journal of Linguistics* 29, pp. 113–142.

Daelemans, Walter, Koenraad De Smedt, and Gerald Gazdar (1992). "Inhertance in Natural Language Processing". In: *Computational Linguistics* 18.2. Ed. by Walter Daelemans and Gerald Gazdar, pp. 205–218.

Dalrymple, Mary, ed. (1999). *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. Cambridge, MA: The MIT Press.

– (2001). *Lexical Functional Grammar*. San Diego, CA: Academic Press.

Dalrymple, Mary, Angie Hinrichs, John Lamping, and Vijay Saraswat (1993). "The Resource Logic of Complex Predicate Interpretation". In: *Proceedings of ROCLING 1993*, pp. 3–21.

Dalrymple, Mary, Ronald M. Kaplan, and Tracy Holloway King (2004). "Linguistic Generalizations over Descriptions". In: *The Proceedings of the LFG'04 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, pp. 199–208.

Davis, Anthony R. (2001). *Linking by Types in the Hierarchical Lexicon*. Stanford, CA: CSLI Publications.

Dyvik, Helge (2004). "Translations as Semantic Mirrors: From Parallel Corpus to Wordnet". In: *Advances in Corpus Linguistics: Papers from the 23rd International Conference on English Language Research on Computerized Corpora (ICAME 23), Göteborg, 22–26 May 2002*. Ed. by Karin Aijmer and Bengt Altenberg. Vol. 49. Language and Computers, pp. 311–326.

Evans, Roger and Gerald Gazdar (1996). "DATR: A Language for Lexical Knowledge Representation". In: *Computational Linguistics* 22.2, pp. 167–216.

Fellbaum, Christiane, ed. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: The MIT Press.

Fillmore, Charles J. and Collin Baker (2015). "A Frames Approach to Semantic Analysis". In: *The Oxford Handbook of Linguistic Analysis*. Ed. by Bernd Heine and Heiko Narrog. 2nd. Oxford: Oxford University Press, pp. 791–816.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck (2003). "Background to FrameNet". In: *International Journal of Lexicography* 16.3, pp. 235–250.

Findlay, Jamie Y. (2016). "Mapping Theory without Argument Structure". In: *Journal of Language Modelling* 4.2, pp. 245–289.

Flickinger, Daniel (1987). "Lexical Rules in the Hierarchical Lexicon". Ph.D. Thesis. Stanford, CA: Stanford University.

Goldberg, Adele E. and Farrell Ackerman (2001). "The Pragmatics of Obligatory Adjuncts". In: *Language* 77.4, pp. 798–814.

Kaplan, Ronald M. and Joan Bresnan (1982). "Lexical-Functional Grammar: A Formal System for Grammatical Representation". In: *The Mental Representation of Grammatical Relations*. Ed. by Joan Bresnan. MIT Press Series on Cognitive Theory and Mental Representation. Cambridge, MA: The MIT Press, pp. 173–281.

Kuhn, Jonas (2001). "Resource Sensitivity in the Syntax-Semantics Interface: Evidence from the German Split NP Construction". In: *Constraint-Based Approaches to Germanic Syntax*. Ed. by Detmar Meurers and Tibor Kiss. Stanford, CA: CSLI Publications, pp. 177–215.

Linden, Erik-Jan van der (1992). "Incremental Processing and the Hierarchical Lexicon". In: *Computational Linguistics* 18.2, pp. 219–238.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller (1990). "Introduction to WordNet: An Online Lexical Database". In: *International Journal of Lexicography* 3.4, pp. 235–244.

Needham, Stephanie and Ida Toivonen (2011). "Derived Arguments". In: *The Proceedings of the LFG'11 Conference*. Ed. by Miriam Butt and Tracy Holloway King. Stanford, CA: CSLI Publications, pp. 401–421.

Parsons, Terence (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. Cambridge, MA: The MIT Press.

Pollard, Carl and Ivan A. Sag (1994). *Head-driven Phrase Structure Grammar*. Chicago, IL: Chicago University Press / CSLI Publications.

Przepiórkowski, Adam (2016). "How *not* to Distinguish Arguments from Adjuncts in LFG". In: *Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*. Ed. by Doug Arnold, Miriam Butt, Berthold Crysmann, Tracy Holloway King, and Stefan Müller. Stanford, CA: CSLI Publications, pp. 560–580.

– (2017a). "Hierarchical Lexicon and the Argument/Adjunct Distinction". Submitted to LFG 2017 proceedings.

– (2017b). "Some Doubts about Meaning Constructors and Semantic Structures in LFG + Glue". Unpublished manuscript.

Richter, Frank (2000). "A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar". Ph.D. Thesis. Universität Tübingen.

Ruppenhofer, Josef, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk (2016). *FrameNet II: Extended Theory and Practice*. Revised November 1, 2016.

Vijay-Shanker, K. and Yves Schabes (1992). "Structure Sharing in Lexicalized Tree-Adjoining Grammars". In: *Proceedings of the 14th International Conference on Computational Linguistics (COLING 1992)*. Nantes, pp. 205–211.