

SEMANTICS VIA F-STRUCTURE REWRITING

Dick Crouch and Tracy Holloway King
Palo Alto Research Center

Proceedings of the LFG06 Conference
Universität Konstanz
Miriam Butt and Tracy Holloway King (Editors)

2006

CSLI Publications
<http://csli-publications.stanford.edu/>

Abstract

This paper discusses how the XLE general purpose ordered rewrite rule system is used to produce semantic representations from syntactic f-structures. The rules apply efficiently because they operate on the packed input of f-structures to produce packed semantic structures. In addition to rules which convert the syntactic structure to a semantic one, there are rules that use external resources to replace words with concepts and grammatical functions with roles. Although the system described here could by no means be described as a theory of semantics, from a practical stand point it can efficiently and robustly produce semantic structures from broad-coverage syntactic ones.

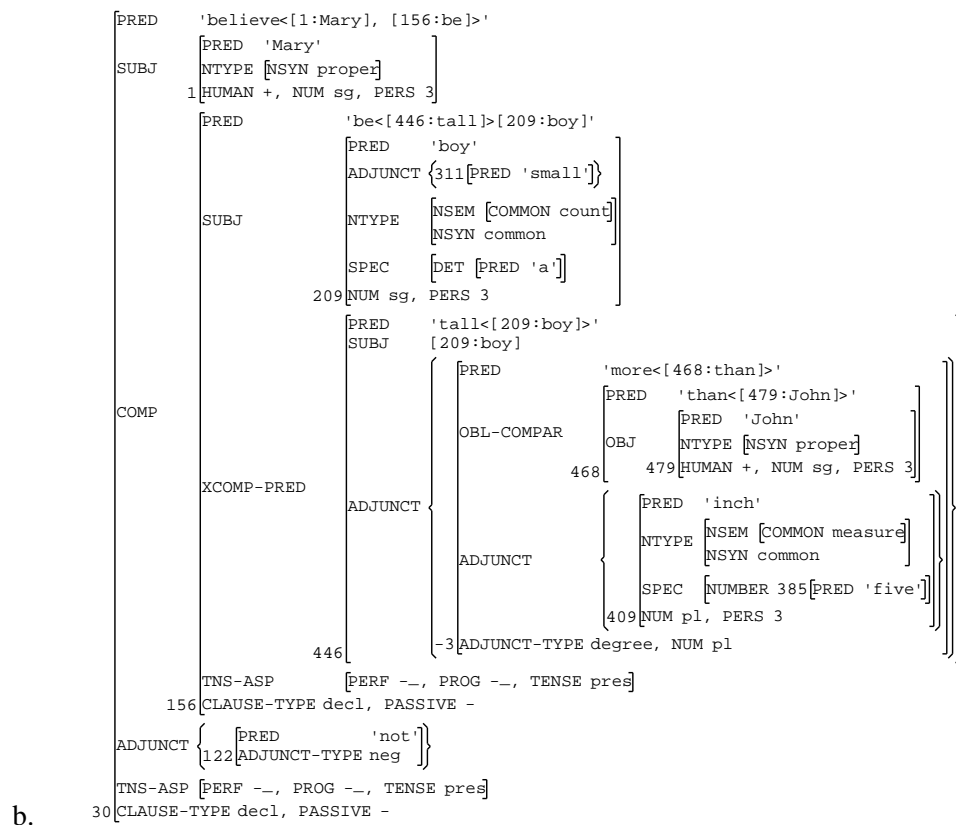
1 Introduction

This paper discusses the use of the XLE's [Crouch et al. (2006), Maxwell and Kaplan (1996)] transfer system [Crouch (2005), Frank (1999)] for mapping f-structures into semantic representations. The technique has been robustly applied to f-structures obtained by parsing open text, such as the Wall Street Journal and New York Times.

The semantics gives a flat representation of the sentence's predicate argument structure and the semantic contexts in which those predications hold. Contrast the f-structure and semantics in (1).

- (1) a. Mary does not believe that a small boy is five inches taller than John.

"Mary does not believe that a small boy is five inches taller than John."



- c. context_head(t,not:11)
 context_head(ctx(be:40),be:40)

```

context_head(ctx(believe:71),believe:71)
context_head(ctx(tall:55),tall:55)

in_context(t,cardinality(Mary:1,sg))
in_context(t,role(mod(degree),ctx(believe:71),not:11,normal))
in_context(ctx(be:40),pres(be:40))
in_context(ctx(be:40),role(copula,be:40,ctx(tall:55)))
in_context(ctx(believe:71),pres(believe:71))
in_context(ctx(believe:71),proper_name(Mary:1,person,Mary))
in_context(ctx(believe:71),role(Agent,believe:71,Mary:1))
in_context(ctx(believe:71),role(Theme,believe:71,ctx(be:40)))
in_context(ctx(tall:55),cardinality(John:67,sg))
in_context(ctx(tall:55),cardinality(boy:36,sg))
in_context(ctx(tall:55),specifier(boy:36,a))
in_context(ctx(tall:55),measure(inch:48,inch:48,five))
in_context(ctx(tall:55),proper_name(John:67,person,John))
in_context(ctx(tall:55),role(mod(degree),boy:36,small:30,normal))
in_context(ctx(tall:55),comparative_diff(tall:55,boy:36,John:67,pos,inch:48))

lex_class(believe:71,[vnclass(consider-29_9-2),prop-attitude])
lex_class(not:11,[sadv,impl_pn_np])
sortal_restriction(ctx(be:40),[1740])
sortal_restriction(Mary:1,[7899136,15024])
alias(John:67,[John])
alias(Mary:1,[Mary])
word(John:67,John,noun,0,67,ctx(tall:55),[[9487097]])
word(Mary:1,Mary,noun,0,1,t,[[9482706]])
word(be:40,be,verb,0,40,ctx(be:40),[[2579744], [2591280], [2629830], [2578719],
[2725216], [2639228], [2595485], [2422266]])
word(believe:71,believe,verb,0,71,ctx(believe:71),[[675183], [681247], [712804],
[676176], [675971]])
word(boy:36,boy,noun,0,36,ctx(tall:55),[[10131706], [9725282], [10464570], [9500236]])
word(inch:48,inch,noun,0,48,ctx(tall:55),[[13469892], [13533066]])
word(not:11,not,adv,0,11,t,[[24548]])
word(small:30,small,adj,0,30,ctx(tall:55),[[1443454], [1467170], [2419704], [1708858],
[1588010]])
word(tall:55,tall,adj,0,55,ctx(tall:55),[[2466583], [2088817], [786375], [678281]])

```

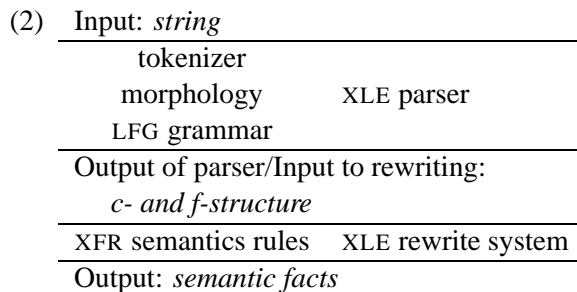
Note how each clause of the core of the representation is set within a context (*in_context*). Contexts are introduced by clausal complements (COMP, XCOMP) in f-structure, but can also be lexically introduced, as shown by the sentential adverb *not*. Nominal and event arguments are skolemized: instead of quantifiers binding variables, terms like *boy:12* are used in place of the bound variables. In addition, roles are introduced in place of grammatical relations and PREDs are replaced by concepts.

The transfer system applies an ordered set of rewrite rules, which progressively consume the input f-structure replacing it by the output semantic representation. The system permits a form of input-limited recursion, where a rule can apply to its own output provided that each application consumes some more of the rules' original input (thus ensuring termination of the recursion). This is required for capturing the contextual structure induced by the recursive embedding of complements within

f-structure. The rewrite-based system can, and has, been used in place of components constructing semantic representations by more standard means, such as Glue Semantics [Dalrymple (2003)].

2 Brief System Introduction

In this section we provide a brief introduction to the XLE system that is used in producing XFR semantics. The syntactic component, including the morphology and tokenizer, is described in detail in [Riezler et al. (2002), Kaplan et al. (2004)]. XLE is described in [Maxwell and Kaplan (1996)] and many details are available in the on-line XLE documentation [Crouch et al. (2006)].



2.1 Types of Rewrite Rules

A somewhat contrived example of a rewrite rule is:

$$\begin{array}{l} (3) \text{ PRED}(\%V, \text{eat}), \text{SUBJ}(\%V, \%S), \text{OBJ}(\%V, \%O), \text{-OBL}(\%V, \%%) \\ ==> \\ \text{word}(\%V, \text{eat}, \text{verb}), \text{role}(\text{Agent}, \%V, \%S), \text{role}(\text{Theme}, \%V, \%O). \end{array}$$

This rule looks at a set of clauses describing an f-structure to see if there is some node %V (the % is used to indicate a variable), with a subject %S and object %O, but no oblique. If the left hand side of the rule is matched, the matching PRED, SUBJ and OBJ clauses are removed from the description, and are replaced by the word and role clauses on the right hand side of the rule.

More generally, the format for rewrite rules is shown in figure 1.

The left hand sides of rules contain boolean combinations of patterns over clauses. Clauses are atomic predicates heading a set of argument terms, where the terms may be non-atomic: e.g. “SUBJ(var(0), var(1))”, where SUBJ is the predicate and var(0) and var(1) are the non-atomic arguments. In patterns over clauses, some of the argument terms can be, or can contain, variables. For example, the pattern “SUBJ(%V, var(%Y))” will match “SUBJ(var(0), var(1))”, setting %V to var(0) and %Y to 1. Second-order quantification over atomic predicates is also available, where $qp(\%P, [\%V, \%Y])$ matches “SUBJ(var(0), var(1))”, setting the predicate variable %P to SUBJ and the list of argument variables [%V, %Y] to var(0) and var(1).

By prefixing a clause pattern on the left hand side of a rule with a “+”, you can indicate that the rule should check for the presence of a matching clause in the input without deleting the clause. Likewise, a prefix of “-” checks that a pattern is not matched by the input. Boolean combinations of clause patterns are possible, as are calls to external procedures. External procedures cannot directly manipulate the full set of input clauses; instead they allow you to perform table lookup or tests on terms, such as subsumption checking in the Cyc generalization hierarchy, or looking up the synset of a word in WordNet.

Rule	::= LHS ==> RHS.	<i>Obligatory rewrite</i>
	LHS ?=> RHS.	<i>Optional rewrite</i>
	LHS *=> RHS.	<i>Recursive rewrite</i>
	– Clause.	<i>Permanent, unresourced fact</i>
LHS	::= Clause	<i>Match & delete atomic clause</i>
	+Clause	<i>Match & preserve atomic clause</i>
	LHS, LHS	<i>Boolean conjunction</i>
	(LHS LHS)	<i>Boolean disjunction</i>
	–LHS	<i>Boolean negation</i>
	{ProcedureCall}	<i>Procedural attachment</i>
RHS	::= Clauses	<i>Set of replacement clauses</i>
	0	<i>Empty set of replacement clauses</i>
	stop	<i>Abandon the analysis</i>
Clause	::= Atom(Term,...,Term)	<i>Clause with atomic predicate</i>
	Atom	<i>Atomic clause</i>
	qp(Variable, [Term _i ,..., Term _n])	<i>Clause with unknown predicate and n arguments</i>
Term	::= Variable	
	Clause	

Figure 1: Format of Rewrite Rules

The right hand side of a rule can be a comma separated set of clause patterns, including the empty set represented as 0 . The right hand side can also be the directive *stop*, which means that the analysis path should be deleted.

Rules can be obligatory, optional or recursive rewrites, and can also introduce permanent non-consumable facts. If the left hand side of an obligatory rule is matched, then the consumed clauses (i.e. those not marked with a “+” or a “–”) have to be removed from the set of input clauses, and replaced by the clauses on the right hand side of the rule. For an optional rule, conceptually speaking, there is a fork in the set of output clauses. On one fork the rule applies, and the consumed clauses on the left hand side are replaced by those on the right hand side. On the other fork the rule does not apply, and the set of clauses remains unchanged. But instead of forking the sets of clauses, the choice space is split to record the alternatives where the rule is and is not applied (section 2.2). A recursive rule can re-apply to its own output, provided that each recursion also consumes some of the input that was present before the first recursive application; this ensures termination of the recursion.

Rules are ordered: rule 1 applies to the input, rule 2 applies to the output of rule 1, and so on. Rule ordering can be exploited, e.g. to encode sequences of defaults, but the feeding and bleeding behavior needs to be handled with care. The rule ordering also means that the scope of any recursion is strictly limited to a single rule. Note that the order of the discussion of the rules in this paper does not necessarily reflect their order in the system.

Clauses preceded by a |– are included as permanent, non-consumable facts. These are part of neither the input nor the output, but can be called on to provide tests or data. For example, a more sensible way of achieving the effects of (3) would be:

- (4) |– concept-map(eat, V-SUBJ-OBJ, Agent, Theme).
 |– concept-map(drink, V-SUBJ-OBJ, Agent, Theme).
 ...

```

PRED(%V, %P), SUBJ(%V, %S), OBJ(%V, %O), -OBL(%V, %%),
concept-map(%P, V-SUBJ-OBJ, %SR, %OR)
==>
word(%V, %P, verb), role(%SR, %V, %S), role(%OR, %V, %O).

```

In this way, a large number of lexical mappings can be asserted permanently, and a single rule takes care of the concept mapping for transitive verbs.

The formalism also allows macros to be used to parameterize commonly occurring patterns in rules, and templates to parameterize commonly occurring sequences of rules. This is an alternative to using a type hierarchy [Oepen et al. (2004)] for producing compact rule sets.

2.2 Packing

Although constraints can sometimes be applied at the semantics level (or subsequent mapping to knowledge representation) to resolve syntactic ambiguities, others will pass through to the semantics level, and yet more may be introduced by things such as word sense ambiguity.

Alternative interpretations are represented in a packed form. An example will give an idea of what these packed representations are like (shown somewhat abbreviated, e.g. cardinality and sortal restriction facts, which would all be in the top choice 1, are not shown):

(5) John saw a man with a telescope.

(6) choice: (A1 xor A2) iff 1

```

1 alias(John:1,[John])
1 context_head(t,see:32)
1 in_context(t,past(see:32))
1 in_context(t,specifier(man:12,a))
1 in_context(t,specifier(telescope:23,a))
1 in_context(t,proper_name(John:1,person,John))
1 in_context(t,role(Experiencer,see:32,John:1))
1 in_context(t,role(Stimulus,see:32,man:12))
A1 in_context(t,role(preposition,man:12,telescope:23))
A2 in_context(t,role(preposition,see:32,telescope:23))
1 lex_class(see:32,[vnclass(see-30_1-1)])
1 word(John:1,John,noun,0,1,t,[[9487097]])
1 word(man:12,man,noun,0,12,t,[[10133569], [10423788], [10135377], [2449786],
[10135514], [10135101]])
1 word(see:32,see,verb,0,32,t,[[2109658], [583923], [2109242], [1620934],
[682517], [591374], [2131231], [911004]])
1 word(telescope:23,telescope,noun,0,23,t,[[4351615]])

```

The standard prepositional attachment ambiguity is reflected in the semantics (6) by two alternative role restrictions: the telescope either modifies the seeing event, or the man. The two alternatives are labeled by the distinct choices *A1* and *A2*. As the first line in the representation states, *A1* and *A2* are mutually exclusive (xor = exclusive or) ways of partitioning the true choice labeled *1*. Most parts of the representation are common to both possible interpretations, and are thus labeled with the choice *1*. It is only the two role assignments for *telescope:23* that are put under distinct choice labels.

A slightly more complex case of prepositional attachment ambiguity gives rise to the following semantic representation (shown somewhat abbreviated):

(7) John saw a man in a park with a telescope.

(8) choice: (A1 xor A2) iff 1
choice: (B1 xor B2 xor B3) iff A1
choice: (C1 xor C2) iff A2
1 alias(John:1,[John]),
1 context_head(t,see:42),
1 in_context(t,past(see:42)),
1 in_context(t,specifier(man:12,a)),
1 in_context(t,specifier(park:21,a)),
1 in_context(t,specifier(telescope:33,a)),
1 in_context(t,proper_name(John:1,person,John)),
1 in_context(t,role(Experiencer,see:42,John:1)),
1 in_context(t,role(Stimulus,see:42,man:12)),
A1 in_context(t,role(preposition,man:12,park:21)),
A2 in_context(t,role(preposition,see:42,park:21)),
B3 in_context(t,role(preposition,man:12,telescope:33)),
or(B2,C2) in_context(t,role(preposition,park:21,telescope:33)),
or(B1,C1) in_context(t,role(preposition,see:42,telescope:33)),
1 word(John:1,John,noun,0,1,t,[[9487097]]),
1 word(man:12,man,noun,0,12,t,[[10133569], [10423788], [10135377],
[2449786], [10135514], [10135101]]),
1 word(park:21,park,noun,0,21,t,[[8494974], [8495199], [2756453], [11059588],
[8495445], [3847283]]),
1 word(see:42,see,verb,0,42,t,[[2109658], [583923], [2109242], [1620934],
[682517], [591374], [2131231], [911004]]),
1 word(telescope:33,telescope,noun,0,33,t,[[4351615]])

Here there are interactions between the attachments: if the location of the man is the park (*A1*), then *with a telescope* can modify either the seeing (*B1*), the park (*B2*), or the man (*B3*). But if the location of the seeing event is the park (*A2*), then *with a telescope* can only modify either the seeing (*C1*) or the park (*C2*). This is reflected in the choice structure, which says that *A1* and *A2* are a disjoint partition of *1*, and that *A1* is in turn partitioned into *B1*, *B2*, and *B3*, while *A2* is partitioned into *C1* and *C2*.

Note that the five possible readings for (7) are represented in not much more space than the two readings for (5). It is possible to count the number of readings by looking only at the choice space: *A1* has three alternatives sitting under it, *A2* has two, and *A1* and *A2* are disjoint, so there are $3+2 = 5$ alternatives altogether.

For more detail on packing in the rewriting system see [Crouch (2005)] and for packing in XLE in general see [Maxwell and Kaplan (1991)].

3 Flattening of Context-relative Predications

The semantics rules use input limited recursion to capture, and flatten, the structural embeddings in f-structure as context-relative predications. Flattening replaces embedded expressions with complex internal structure, such as clausal complements, with atomic first order terms, contexts. The information about the level of embedding of an expression is preserved by associating its content with the corresponding context. Negation and intensional operators also trigger the introduction of new con-

texts. Contexts thus serve as scope markers since their use enables globally represented information, such as the scope of operators, to be made locally accessible.

3.1 Flattening of Verbal F-structures

We will illustrate the use of recursive rules to flatten out the contextual structure implicit in f-structure. F-structures are recursive, with one node being embedded inside another, yet it is straightforward to represent this as a flat set of clauses. For (9) these might be along the (abbreviated) lines of (10).

(9) Mary knew that Ed ate turnips.

(10)	PRED(var(0), know),	
	SUBJ(var(0), var(1)),	PRED(var(1), Mary)
	COMP(var(0), var(3)),	PRED(var(3), eat)
	SUBJ(var(3), var(2)),	PRED(var(2), Ed)
	OBJ(var(3), var(4)),	PRED(var(4), turnip)
	TNS-ASP(var(0), var(5)),	TENSE(var(5), past)
	TNS-ASP(var(3), var(6)),	TENSE(var(6), past)

The f-structure node var(3) is embedded under var(0), and var(2) is in turn embedded under var(3). But not all of the f-structure embeddings lead to context embeddings in the semantics: in fact, it is only the nodes var(0) and var(3) that introduce semantic contexts.

The f-structure to semantics rewrite rules therefore need to make a recursive traversal of the f-structure, linking each f-structure node to the nearest dominating node that introduces a semantic context. This is achieved in three stages. First, nodes introducing a context are identified and labeled (where $c(\dots)$ is wrapped around a node to indicate its context):

(11) $+COMP(\%N1, \%N2)$
 \implies
 $new_context(\%N2, c(\%N2)), in_context(\%N2, c(\%N2)).$

Second, all immediate links between f-structure nodes are labeled:

(12) $+SUBJ(\%N1, \%N2) \implies link(\%N1, \%N2).$
 $+OBJ(\%N1, \%N2) \implies link(\%N1, \%N2).$
 $+COMP(\%N1, \%N2) \implies link(\%N1, \%N2).$
 $+TNS-ASP(\%N1, \%N2) \implies link(\%N1, \%N2).$

Finally, a recursive rule traverses the links propagating the *in_context* labels.

(13) $+in_context(\%N1, \%C), link(\%N1, \%N2), -new_context(\%N2, \%C)$
 \implies
 $in_context(\%N2, \%C).$

Each recursive step will consume one of the $link(\dots, \dots)$ facts, ensuring that the recursion terminates. The recursion will simultaneously start at all the nodes initially labeled as being *in_context* by rule (11), and the negative test on *new_context* ensures that nodes are only connected back to their immediately dominating context. It is important to remember that this rule only applies recursively to

its own output. This is unlike more general recursion in a set of unordered rules, where rules can recursively apply to the output of other rules.

Modals such as *can* and *should*, behave similarly to other context inducing verbs, despite their distinctive syntactic structure. Since modals take XCOMPs in the f-structure, the rules described above apply relatively straightforwardly to them. Later processing, such as mapping to KR, can make further distinctions among the different modals for applications.

3.2 Negation and Other Context-inducing Adverbs

In the f-structure, sentential negation is an adverb in the ADJUNCT set. However, in the semantic structure, negation introduces a context. Thus a sentence like (14a) has a simplified f-structure like (14b) but a simplified semantics like (14c).

(14) a. Jane did not hop.

b.
$$\left[\begin{array}{ll} \text{PRED} & \text{'hop<SUBJ>'} \\ \text{SUBJ} & \left[\text{PRED} \text{ 'Jane'} \right] \\ \text{ADJUNCT} & \left\{ \left[\begin{array}{ll} \text{PRED} & \text{'not'} \\ \text{ADJUNCT-TYPE} & \text{neg} \end{array} \right] \right\} \end{array} \right]$$

c. *context_head*(*t*,not:10)

context_head(*ctx*(*hop*:17),*hop*:17)

in_context(*t*,role(mod(degree),*ctx*(*hop*:17),not:10,normal))

in_context(*ctx*(*hop*:17),role(Theme,*hop*:17,*Jane*:1))

word(*Jane*:1,*Jane*,noun,0,1,*t*,[[9482706]])

word(*hop*:17,*hop*,verb,0,17,*ctx*(*hop*:17),[[1948772], [2076532], [1823521], [2076385], [2076247], [2076113]])

word(not:10,not,adv,0,10,*t*,[[24548]])

There are other context inducing adverbs: these are ADJUNCTS in f-structure but introduce a context in the semantics. The rules for these are similar to those for sentential negation and are lexicalized to apply only to adverbs of this class. Examples of such adverbs include sentential uses of *necessarily*, *possibly*, *probably*, *maybe*, and *certainly*, as in (15). These can combine with each other and with negation, as in (16), in which case a series of embedded contexts is created by the semantic rewrite rules.

(15) a. Jane probably left.

b. Jane certainly left.

(16) Jane certainly did not leave.

The rules to introduce the contexts work as follows. The rules need to make the clause the adverbs modify the first argument to the modifier. If there is a sequence of sentential modifiers, the modified clause is made the first argument of the last modifier in the sequence, which is itself the first argument of the penultimate modifier, and so on. This is done using limited recursion to build up a list of sentential modifiers. First an empty list is created, by (17) for anything that has an appropriate adjunct, is negative, or is imperative or interrogative (imperatives and interrogatives also introduce contexts).

(17) +PRED(%A, %%),
 (+ADJUNCT(%A,%%)
 | +is_negated(%A,%%)
 | +CLAUSE-TYPE(%A,imp)
 | +CLAUSE-TYPE(%A,int)
)
 ==>
 sentential_mods([], %A, %A).

The empty sentential modifiers list is then filled with the sentential modifiers in order by the rules in (18). (18a) first puts negation on the list. Then (18b) puts the other sentential modifiers on the list, checking for the relative scope of the adjuncts where the scope is provided by the f-structure *scopes* fact and reflects the linear order of the adjuncts.

(18) a. sentential_mods([], %A, %A), is_negated(%A, %NMod)
 ==>
 sentential_mods([%NMod], %A, %A).
 b. +ADJUNCT(%H,%M), in_set(%N,%M), sentential_mod(%N),
 -(in_set(%N1, %M), sentential_mod(%N1), scopes(%N1, %N)),
 sentential_mods(%Mods,%H, %H)
 * ==>
 sentential_mods([%N|%Mods],%H, %H).

Finally, the rules do head switching of the modifiee and the last modifier: everything that was expecting the modifiee as an argument now takes the last modifier. When the adverbs modify the main (root) clause, the rules indicate that the node of the last modifier becomes the root node since it is important that all semantic structures, like f-structures, are rooted and connected.

4 Canonicalization of F-structures

A number of syntactic constructions are canonicalized very early in the semantic rules. These are relatively straight-forward rewrites that then feed the more complex semantic rules (section 5).

Passive constructions are turned into their corresponding actives. For passives with overt agent *by* phrases, as in (19a), this results in a structure similar to that of their active counterpart. For passives without overt agents, as in (19b), a special agent pronoun is put into the subject role. The ordered rules to achieve this are shown in (20).

(19) a. The cake was eaten by John.
 b. The cake was eaten.
 (20) a. +VTYPE(%V, %%), +PASSIVE(%V,+), SUBJ(%V, %LogicalObj)
 ==>
 OBJ(%V, %LogicalObj).
 b. +VTYPE(%V, %%), +PASSIVE(%V,+),
 OBL-AG(%V, %LogicalSubj), PFORM(%LogicalSubj,%%)
 ==>
 SUBJ(%V, %LogicalSubj).

c. +VTYPE(%V, %%), +PASSIVE(%V,+), -SUBJ(%V,%%)
 ==>
 SUBJ(%V,%AgtPro), PRED(%AgtPro,agent_pro), PRON-TYPE(%AgtPro, null).

(20a) takes the subject of a passive verb and makes it the object. Then (20b) takes the OBL-AG of a passive verb and makes it the subject. Finally, (20c) creates a dummy subject for any passive verb that does not have one provided by (20b).

A number of constructions are assigned null pronominal subjects in the f-structure. In many cases, the semantics substitutes in the most likely subject instead. Some example constructions are shown in (21) with the rule for (21a) shown in (22).

- (21) a. Before leaving, I fixed it. (=I leaving)
 b. To open it, John broke the seal. (=John to open it)
 c. Having arrived early, Mary sat down. (=Mary arrived early)
 d. Broken by the wind, the gate fell. (=the gate broken by the wind)

(22) +SUBJ(%Main,%MainSubj), +ADJUNCT(%Main,%Adj), +OBJ(%Adj,%AdjObj),
 SUBJ(%AdjObj,%AdjObjSubj), arg(%AdjObj,1,%AdjObjSubj),
 PRON-TYPE(%AdjObjSubj,null), PRED(%AdjObjSubj,%%)
 ==>
 SUBJ(%AdjObj,%MainSubj), arg(%AdjObj,1,%MainSubj).

(22) looks for an f-structure %Main which has a subject and an adjunct. That adjunct must take an object with a null subject (in examples like (21a) *leaving* is the object of *before*). This subject is replaced by the subject of the main clause. Note that the + in front of the first fact +SUBJ(%Main,%MainSubj) ensures that the main clause subject is not deleted.

Similar canonicalization occurs for comparatives, measure phrases, and related scalars. These have a number of different overt expressions in the syntax which are all regularized to allow the semantics to operate on them more directly.

- (23) a. John is happier.
 b. John is happier than Mary.
 c. John is much happier than Mary.

(24) in_context(ctx(happy:14),comparative_diff(happy:14,John:1,Mary:23,pos,much:10))

(24) shows the canonicalized semantic form for the comparative adjective *happier* in (24c). *Mary:23* is the comparison class, the *pos* indicates that it is more happy as opposed to less happy, and *much:10* indicates the amount of difference. The comparison class and the amount of difference can be *unspecified*, e.g. for sentences like (24a).

5 Semantic Rewrites

There are a set of rules in the semantic rewrite rules which correspond to more traditional, theoretical semantic rules. These include treating coordination by semantic instead of syntactic type, creating structures for deverbal nouns, and providing appropriate scope and canonicalizations for quantifiers. In this section we discuss how these are done in the rewrite rules.

5.1 Quantifiers

Unlike in the Glue semantic [Dalrymple (2003)] approach to manipulating f-structures to create semantic structures, using the semantic rewrite rules does not provide a theoretically motivated treatment of quantifier scope possibilities. Instead, the rules determine one scope.¹ For example, the scope of indefinite subjects is raised relative to that of negation. Relatedly, the modals *must*, *ought*, and *should* are rescoped relative to sentential negation, while other modals are not.

One interesting rule for quantifiers derives negative sentential modifiers from downward monotone nominal arguments. The basic idea is to mark anything with a downward monotone nominal argument as negated, and then introduce a new *not* context. For example, the sentence *No girl hopped*. has a semantics as in (25) in which there are two contexts, one introduced by *no* (ctx(hop:15)), and in which there is a word fact similar to that for sentential negation.

```
(25) context_head(t,not:n(147)
      context_head(ctx(hop:15),hop:15)
      in_context(t,role(mod(degree),ctx(hop:15),not:147,normal))
      in_context(ctx(hop:15),past(hop:15))
      in_context(ctx(hop:15),cardinality(girl:4,sg))
      in_context(ctx(hop:15),proportion(girl:4,no))
      in_context(ctx(hop:15),role(Theme,hop:15,girl:4))
      word(girl:4,girl,noun,0,4,ctx(hop:15),[[9979060], [9934281], [9844392], [9979885],
      [9979646]])
      word(hop:15,hop,verb,0,15,ctx(hop:15),[[1948772], [2076532], [1823521], [2076385],
      [2076247], [2076113]])
      word(not:147,not,adv,0,147,t,[[24548]])
```

The semantic rewrite rule to handle these cases first looks for the appropriate quantified arguments of a verb, introducing a fact that the verb is negated. Then a second rule creates the negative adjunct for the verb which then triggers the same rule that introduces the context for sentential negation (section 3.2).

5.2 Coordination

In the f-structure, coordination is represented as a set with a feature indicating what level in the c-structure the coordination occurred at (e.g., N, NP). Coordination of nominals indicates the resolved number and person of the set but otherwise is identical to coordination of verbs and sentences. The f-structure for *Mary and Jane hopped*. is shown in (26).

```
(26) [ PRED 'hop<SUBJ>'
      [ SUBJ { [ PRED 'Mary'
                NUM sg
              ]
              [ PRED 'Jane'
                NUM sg
              ]
              NUM pl
              COORD-FORM and
              COORD-LEVEL NP
            }
      ]
```

¹Multiple scopes can be produced in certain situations by using optional rules. However, this has not proven a useful or efficient strategy in using the rewrite rules to produce semantic representations.

In contrast, the semantics differentiates between nominal, verbal, and adjunct coordinations. The rules first determine which type of coordination is present, typing them as nominal, sentential, verbal, predicative, number, or adjunct. A typed PRED is then created for the coordinate structure (note in (26) that the SUBJ f-structure has not PRED of its own). It is this new PRED that will then act as an argument, with its elements listed as additional semantics facts, as in (27).

```
(27) context_head(t,hop:18)
      in_context(t,cardinality(Jane:1,sg))
      in_context(t,cardinality(Mary:10,sg))
      in_context(t,cardinality(group_object:2,pl))
      in_context(t,is_element(Jane:1,group_object:2))
      in_context(t,is_element(Mary:10,group_object:2))
      in_context(t,role(Theme,hop:18,group_object:2))
      word(Jane:1,Jane,noun,0,1,t,[[9482706]])
      word(Mary:10,Mary,noun,0,10,t,[[9482706]])
      word(group_object:2,group_object,implicit,0,2,t,[[1740]])
      word(hop:18,hop,verb,0,18,t,[[1948772], [2076532], [1823521], [2076385], [2076247],
      [2076113]])
```

The syntax treats all coordinators similarly. However, the semantics differentiates between coordinators like *and* which do not introduce new contexts, as seen in (27), and ones like *or* which do. Compare the analysis of *Jane or Mary hopped.* in (28) to that in (27) for *Jane and Mary hopped.*

```
(28) context_head(t,hop:20)
      context_head(ctx(Jane:1),Jane:1)
      context_head(ctx(Mary:9),Mary:9)
      in_context(t,coord_or([ctx(Jane:1),ctx(Mary:9)]))
      in_context(t,cardinality(group_object:2,sg))
      in_context(t,role(Theme,hop:20,group_object:2))
      in_context(ctx(Jane:1),cardinality(Jane:1,sg))
      in_context(ctx(Jane:1),is_element(Jane:1,group_object:2))
      in_context(ctx(Mary:9),cardinality(Mary:9,sg))
      in_context(ctx(Mary:9),is_element(Mary:9,group_object:2))
      word(Jane:1,Jane,noun,0,1,ctx(Jane:1),[[9482706]])
      word(Mary:9,Mary,noun,0,9,ctx(Mary:9),[[9482706]])
      word(group_object:2,group_object,implicit,0,2,t,[[1740]])
      word(hop:20,hop,verb,0,20,t,[[1948772], [2076532], [1823521], [2076385], [2076247],
      [2076113]])
```

There are two contexts related to the top context *t* by the COORD_OR fact. The *group_object* is still the Theme of *hop* but which element (*Jane* or *Mary*) is in that set depends on the context.

5.3 Deverbal Nouns

Deverbal nouns, or nominalizations, can pose serious challenges for knowledge-based systems. Sentences (29) and (30) describe the same event of destruction, which has the same two participants in both cases. However, the event is expressed by a verb in the first case and a noun in the second case.

(29) Alexander destroyed the city in 332 BC.

(30) Alexander's destruction of the city happened in 332 BC.

The f-structure for these differ significantly. However, these are canonicalized in the semantics so that both resemble events with roles from VerbNet (section 6.2) and verbal concepts from WordNet (section 6.1).

An external database of deverbal nouns is created indicating the noun, the corresponding verb, the type of deverbal, and how lexicalized it is.² Note that gerunds are treated productively by the syntax and so do not need entries in the database.

(31) deverbal(destruction, destroy, null, only).
deverbal(parolee, parole, ee, only).
deverbal(teacher, teach, er, both).

The semantic rewrite rules first identify deverbal nouns and indicate whether they have a COMP or XCOMP argument. Then a set of rules determine for each type of deverbal (e.g., *null*, *er*) how the different specifiers and adjuncts map to the argument of the verb. In (30) the POSS maps to the subject and the *of* adjunct to the object. There are several of these rules to account for the different types of deverbals with different combinations of arguments. Consider the rule for deverbals like *a teacher of poetry* shown in (32).

(32) may_be_deverbal(%N, %V, er, %HasComp),
@isPrepAdjunct(%N,of,%Obj),
@hasTransMapping(%V, %HasComp, %SubCat)
==>
is_deverbal(%N, %V, %SubCat, needs_arg, %Obj), is_er_deverbal(%N).

(33) states that a noun %N, which has been identified as an *er* deverbal, has a prepositional *of* adjunct and that the corresponding verb form %V has a transitive mapping. This creates a new fact *is_deverbal* with the transitive frame, no subject (*needs_arg*), and the *of* phrase as the object. A later rule then creates a subject for the verb from the *er* deverbal itself. That is, the subject of the *teach* event is the *teacher* while the object is the *poetry*, as in (33).

(33) in_context(ctx(teach:3),role(Agent,teach:3,teacher:3))
in_context(ctx(teach:3),role(Recipient,teach:3,implicit_arg:2))
in_context(ctx(teach:3),role(Topic,teach:3,poetry:14))
lex_class(teach:3,[vnclass(transfer_mesg-37_1-1-1)])
word(implicit_arg:2,implicit,implicit,0,0,ctx(teach:3),[[1740]])
word(poetry:14,poetry,noun,0,14,ctx(teach:3),[[6995243], [6995943]])
word(teach:3,teach,verb,0,3,ctx(teach:3),[[820277], [270355]])
word(teacher:3,teacher,noun,0,3,t,[[10533902], [5781275]])

5.4 Other Rules

There are rules which are not strictly speaking semantic from a theoretical perspective but which are useful for applications and for deeper processing such as mapping to knowledge representation ([Crouch (2005)]). An example of these is the alias fact. Proper nouns receive an alias fact which ties the skolem of that noun to the surface form. For example, the proper noun *John* would receive a fact as in (34).

²Extremely lexicalized deverbals like *building* are not mapped on to events.

(34) alias(John:67,[John])

This records the string that corresponds to the proper noun, thereby providing more identification information. This is necessary because the concept for all proper nouns referring to male people is the same (and similarly for companies, locations, and other proper noun classes).

In addition, the alias fact for multiword proper nouns contains variants of the noun that are useful for applications. For example, a name like *Mr. John Smith* would have an alias fact like that in (35).

(35) alias(John:67,[John, Smith, John Smith, Mr. John Smith])

This allows the proper noun to be more easily matched with occurrences in other sentences and texts.

6 Incorporation of Lexical-semantic Resources

This section discusses how further lexical-semantic resources (in particular, WordNet and VerbNet) can be used in the semantic rules. Incorporation of external lexical resources saves time in lexical development, but it comes at a cost both for finding ways in which to integrate the resources and in dealing with errors in those resources.

6.1 WordNet Concepts

WordNet is used to assign concepts to words. WordNet [Fellbaum (1998)] contains words with their part of speech organized into synonym sets (synsets) which represent underlying lexical concepts; these synonym sets are linked by relations such as hypernyms (e.g., *auction* is a type of *sell* which is a type of *exchange*, *change*, *interchange* which is a type of *transfer*). The semantic rewrite rules assign concepts to words by looking up the word with its associated part of speech. Some words will belong to just one synset while others belong to many. To accommodate words with more than one synset, the synset concepts are stored as a list. The concept becomes part of the word fact associated with a skolem. An example is shown in (36).

(36) word(hop:17,hop,verb,0,17,t,[[1948772], [2076532], [1823521], [2076385], [2076247], [2076113]])

The word facts contain the string, part of speech, and context as well as the skolem and synsets. This information is all stored for use in further processing, such as in the mapping to KR.

In theory, the contents of WordNet could be dumped into a lexicon of non-resourced facts or an external database that the semantic rewrite rules could refer to. However, instead the rules call the WordNet interface directly via a procedural attachment (indicated in the rewrite rules by { }). This calling of an external resource directly contrasts with the way in which VerbNet is incorporated into the rules, as described below.

In addition to the general lookup of word concepts in WordNet, certain classes of words are assigned WordNet synsets in a more constrained fashion. These lookups are done before the more general lookups since they are the more specific case and take precedence over the default case. For example, proper nouns are not looked up based on their string form³ but instead by their proper noun type as assigned by the morphological analyzer that is used by the syntax. Proper nouns can be classed as locations, organizations, companies, male persons, female persons, etc. These are assigned a synset appropriate for this class by rules such as (37b) in conjunction with the non-resourced fact in (37a).

³WordNet does have entries for many proper nouns. However, these entries are spotty and can result in rather unexpected behavior in applications.

- (37) a. |− name_synset(company, −, −, company, 7948427).
 b. +in_context(%proper_name(%NameSk,%Type,%)),
 node_label(%V,%NameSk), NTYPE(%V,%),
 name_synset(%Type,%G,%H,%WNWord,%S),
 {wn_all_hypers(%WNWord,noun,%S,%HL,%SS)},
 @get_word_context(%V,%Ctx),
 {%NameSk = %Pro:n(%SN,%N)}
 ==>
 synsets(%NameSk,%SS), word(%NameSk,%Pro,noun,%N,%SN,%Ctx,%HL).

Pronouns work similarly in that they are assigned synsets based on an appropriately predefined set of features. Some of the nonresourced facts used in the pronoun mapping are shown in (38).

- (38) |− pronoun_synset(he, male‘ person, 9487097).
 |− pronoun_synset(she, female‘ person, 9482706).
 |− pronoun_synset(we, person, 7626).
 |− pronoun_synset(you, person, 7626).
 |− pronoun_synset(it, entity, 1740).

6.2 VerbNet Roles

VerbNet is used to assign roles to the arguments of verbs. VerbNet [Kipper et al. (2000)] classifies verbs according to Levin verb classes [Levin (1993)]. It includes syntactic subcategorization information, information about thematic roles (e.g., agent, patient), and basic lexical semantics. In order to use the VerbNet material, we extracted it into a Unified Lexicon ([Crouch and King (2005)]) which contained information about the XLE syntactic subcat frames, WordNet synsets, and VerbNet, as well as some lexical class information used in the later semantics to KR mapping rules.

As discussed in [Crouch and King (2005)], there were some problems in converting VerbNet into a format that could be used by the semantics rules. The first was converting VerbNet subcategorization frames into ones that were compatible with the XLE syntactic lexicon. This was difficult because the VerbNet subcategorization information is listed not as grammatical function information but as abstractions over a canonical phrase structure tree. This extraction becomes extremely involved for verbs which take NP small clauses, particles, expletives, or verbal complements. The second issue in the VerbNet extraction was ensuring that a verb belonging to a particular VerbNet class inherited all the correct role restrictions from the classes above it. The final issue with VerbNet was that many verb frames have implicit roles. These roles are determined by looking at the semantics provided for the verb. If there is a thematic role mentioned that is preceded by a ?, e.g. ?Topic, then it is implicitly present in the verb frame and may have role restrictions on it. For example, the transcribe-25.4 class for *The secretary transcribed the speech* has an implicit Destination role which is restricted to being concrete. Note that this role is overt in other frames for this verb, as in *The secretary transcribed the speech into the record*.

In the semantics rules, the VerbNet role mapping works by looking up the head word and subcat frame in the UL to see whether there are VerbNet roles associated with the arguments. In the building of the UL, words with subcat frames which did not have a direct listing in VerbNet are sometimes assigned a VerbNet mapping from a similar frame (e.g. a version with an oblique prepositional phrase may be able to use the mapping for the base transitive, with a guessed role for the oblique). If there are roles in the UL, then the grammatical functions are converted to the relevant VerbNet roles. Thus, the f-structure SUBJ and OBJ for a sentence like *The girl ate the cake*. are mapped into the roles in

(39) by a simplified rule like (40) using the UL entry in (41). The UL itself is stored as a database that the rules access ([Crouch et al. (2006)]).

- (39) `in_context(t,role(Agent,eat:22,girl:5))`
`in_context(t,role(Patient,eat:22,cake:18))`
- (40) `+word(%VSk,%Verb,verb,%%,%%,%%,%%),`
`verb_mapping(%Origin,%Verb,%SubCat,%Source,%VerbClass,%WN,%VSk,%Ctx,`
`%GF1,%Restr1,%Sk1,`
`%GF2,%Restr2,%Sk2,`
`%Mapping),`
`@get_subcat(%VSk,%SubCat),`
`@get_gf(%VSk,%Sk1,%GF1),`
`@check_selectional_restriction(%Sk1,%Restr1,%%C1),`
`@get_gf(%VSk,%Sk2,%GF2),`
`@check_selectional_restriction(%Sk2,%Restr2,%%C2)`
`==>`
`%Mapping,`
`lookup_subcat(%VSk,%SubCat,[%Origin,%Source]),`
`wordnet_classes(%VSk,%WN),`
`source_of_concept_mapping(%VSk,%Source),`
`lex_class(%VSk,%VerbClass),`
`sortal_restriction(%Sk1,%Restr1),`
`sortal_restriction(%Sk2,%Restr2).`
- (41) `verb_map(eat,v-SUBJ-OBJ,verbnet,[vnclass(eat-39_1-1),`
`[wn(1155228,verb(consumption)),wn(1157345,verb(consumption)),`
`wn(1168626,verb(consumption))],%Ev,[],`
`subj,[15024,4359],%subj,`
`obj,[1740],%obj,`
`implicit_args(%Ev,[]),`
`concept_for(%Ev,eat),`
`source_of_concept(%Ev,guessed_verb),`
`verbnet_role(Agent,%Ev,%subj),`
`verbnet_role(Patient,%Ev,%obj),`
`vn_sem(take_in(during(%Ev),%subj,%obj))).`

First consider the UL entry in (41). It corresponds to the word *eat* in its transitive v-SUBJ-OBJ use and has a VerbNet class with WordNet correspondences. The subj and obj arguments have selectional restrictions of WordNet synsets [15024, 4359] and [1740]. Then the VerbNet mapping indicates that there are no implicit arguments and that the concept of the event is the string *eat* (a semantics rewrite rule will assign the appropriate WordNet synset (section 6.1)). The *verbnet_role* facts assign the Agent role to the subject and the Patient role to the object. Finally, the VerbNet semantics is recorded; this is not currently used by the semantic rules.

Even with the UL tuned for the XLE semantic rules, there are some discrepancies which must be taken into account. For example, many VerbNet mappings refer to OBL(ique) arguments. Due to the f-structure assignments, these may be either a syntactic OBL or in the ADJUNCT set as prepositional

phrases. As such, the rewrite rules look first for an OBL and if none is found, then for a PP in the ADJUNCT set.

The UL entry and the semantic rewrite rule that does the mapping mention sortal restrictions (*eat* subj = [15024, 4359]; obj = [1740]). VerbNet, and other resources, provide sortal restrictions on the arguments of verbs. However, applying sortal restrictions obligatorily may be very dangerous because if the argument does not match the sortal restriction, then the only mapping of the verb can be eliminated. Mismatches on sortal restrictions can come from a variety of sources. Sometimes the concept for the restriction or for the argument is incorrect. More often, the mismatch reflects a type of coercion; for example, organizations are often treated as volitional and/or animate. As such, sortal restrictions should be implemented with an optimality mechanism similar to that used in XLE parsing ([Frank et al. (2001)]). Currently, such a mechanism is not in place in the rewrite system used by the semantic rules and so although the sortal restrictions are recorded, they are not enforced in the rules.

7 Discussion, and Conclusions

7.1 Comparison to Glue

The rewrite-based system is used in place of components constructing semantic representations by more standard means, such as Glue Semantics [Dalrymple (2003)]. A comparison of Glue and rewrite semantic construction reveals a number of theoretical and practical pros and cons.

In practical terms, the transfer/rewriting system provides a relatively straightforward tool for efficiently manipulating f-structures, which should be accessible to grammarians with little or no background in formal theories of semantic construction like Glue- or Montague-semantics. This tool comes with the XLE.

Theoretically, the rewrite system imposes few interesting constraints on what kinds of representation can be constructed. Unlike Glue, for example, there is no elegant account of scope ambiguity derived from the theory of semantic construction. The rewrite approach allows semantic construction to be sensitive to features of the meaning language in a way that Glue expressly forbids. This can be convenient when quickly developing a broad-coverage system, but it provides few constraints and guides to the rule writer, allowing for theoretically unsound analyses and implementations.

The feeding and bleeding nature of the rewrite rules means that, unlike unification- or Glue-based semantics, the semantic construction rules cannot easily be run in reverse for generation purposes. Given that the XLE LFG grammars can be run in both the parsing and generation direction, having an accompanying semantics for both parsing and generation is desirable and opens a broader range of applications. Without inherent reversibility, the semantic rewrite rules have to be written in two sets and maintenance becomes a serious issue since a change in the f-structure to semantics rules can necessitate a corresponding change in the semantics to f-structure rules.

Thus, while it would be a stretch to call the rewrite approach a *theory* of semantic construction, it does provide a powerful, practical and efficient tool for the task.

7.2 Cross-linguistic Application

The technique of mapping from f-structures to semantics described in this paper can be applied cross-linguistically. Given the degree of abstraction and generality already present in f-structures for different languages [Butt et al. (1999)], one can port semantic rules from one language to another. As an experiment, an initial port of the rules to Japanese, using the Japanese ParGram grammar as a basis ([Masuichi and Ohkuma (2003)]), was successfully conducted ([Umemoto (2006)]).

Some of the rules described above, such as the context-inducing adverbs, are lexicalized. These need to be ported to apply to the corresponding lexical items in other languages. In addition, some constructions that are present in English and that need to be manipulated by the semantic rules may not be present in other languages. Leaving rules for these in the semantics will not hurt in that they will never trigger, but for clarity they should be removed. Similarly, other languages may have constructions which are not covered by the rules described here because English does not have them. We anticipate that it should be possible to use the rewrite system to process such constructions efficiently.

A more serious problem in the cross-linguistic application of this approach is in the mapping of concepts and roles which depended on WordNet and VerbNet respectively. There are WordNets, and ontologies, for a variety of languages and so for these languages it is possible to incorporate these resources as the English semantics uses WordNet. Broad-coverage role mapping resources like VerbNet are much rarer for other languages and so if this type of role is needed for the semantics, then the lexical resources may need to be boot-strapped in some way. Fortunately, however, the semantic rules can run independently of concepts and roles, applying only the rules which convert f-structures into semantic structures, such as the context-introducing and flattening rules (section 3). For many purposes, these may be sufficient even without concepts and roles.

7.3 Summary

This paper discussed how the XLE general purpose ordered rewrite rule system is used to produce semantic representations from syntactic f-structures. The rules apply efficiently because they operate on the packed input of f-structures to produce packed semantic structures. In addition to rules which convert the syntactic structure to a semantic one, there are rules that use external resources to replace words with concepts and grammatical functions with roles. Although the system described here could by no means be described as a theory of semantics, from a practical stand point it can efficiently and robustly produce semantic structures from broad-coverage syntactic ones.

References

- [Butt et al. (1999)] Butt, Miriam, Stefani Dipper, Anette Frank, and Tracy Holloway King. 1999. Writing large scale parallel grammars for English, French and German. In *Proceedings of LFG99*.
- [Crouch (2005)] Crouch, Dick. 2005a. Packed rewriting for mapping semantics to KR. In *Proceedings of the Sixth International Workshop on Computational Semantics*.
- [Crouch et al. (2006)] Crouch, Dick, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2006. XLE documentation. http://www2.parc.com/isl/groups/nlft/xle/doc/xle_toc.html.
- [Crouch and King (2005)] Crouch, Dick and Tracy Holloway King. 2005. Unifying lexical resources. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*.
- [Dalrymple (2003)] Dalrymple, Mary. 2003. *Lexical Functional Grammar*. Academic Press. Syntax and Semantics, vol 34.
- [Oepen et al. (2004)] et al., Stefan Oepen. 2004. Som a hoppe etter wirkola? unpublished manuscript.

- [Fellbaum (1998)] Fellbaum, Christiane, ed. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.
- [Frank (1999)] Frank, Anette. 1999. From parallel grammar development towards machine translation. In *Proceedings of the MT Summit VII*.
- [Frank et al. (2001)] Frank, Anette, Tracy Holloway King, Jonas Kuhn, and John T. Maxwell III. 2001. Optimality theory style constraint ranking in large-scale lfg grammars. In P. Sells, ed., *Formal and Empirical Issues in Optimality Theoretic Syntax*. CSLI Publications.
- [Kaplan et al. (2004)] Kaplan, Ron, John T. Maxwell III, Tracy Holloway King, and Richard Crouch. 2004. Integrating finite-state technology with deep LFG grammars. In *Proceedings of the Workshop on Combining Shallow and Deep Processing for NLP (ESLLI)*.
- [Kipper et al. (2000)] Kipper, Karin, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *AAAI-2000 17th National Conference on Artificial Intelligence*.
- [Levin (1993)] Levin, Beth. 1993. *English Verb Classes and Alternations*. Chicago University Press.
- [Masuichi and Ohkuma (2003)] Masuichi, Hiroshi and Tomoko Ohkuma. 2003. Constructing a practical Japanese parser based on Lexical-Functional Grammar. *Journal of Natural Language Processing* 10:79–109. in Japanese.
- [Maxwell and Kaplan (1991)] Maxwell, John and Ron Kaplan. 1991. A method for disjunctive constraint satisfaction. *Current Issues in Parsing Technologies* .
- [Maxwell and Kaplan (1996)] Maxwell, John and Ron Kaplan. 1996. An efficient parser for LFG. In M. Butt and T. H. King, eds., *Proceedings of the First LFG Conference*. CSLI On-line Publications.
- [Riezler et al. (2002)] Riezler, Stefan, Tracy Holloway King, Ron Kaplan, Dick Crouch, John Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- [Umemoto (2006)] Umemoto, Hiroshi. 2006. Implementing a Japanese semantic parser based on Glue approach. In *Proceedings of The 20th Pacific Asia Conference on Language, Information and Computation*.