

Packed Rewriting for Mapping Semantics to KR

Dick Crouch*

Palo Alto Research Center, California, USA
crouch@parc.com

This paper describes one component of a system being developed at PARC for deriving knowledge representation from free text. The component takes compositionally derived semantic representations and converts them to knowledge representations, intended for use by a back-end reasoning system. A resource sensitive rewriting system, originally developed for transfer in machine translation, is used for the conversion. This paper (i) discusses the system background and the need for an explicit mapping between semantic and knowledge representations, (ii) describes the rewriting system, with emphasis on its use of packing to manage ambiguity, and (iii) discusses the advantages of using such a system for semantics to KR mapping. Above and beyond a system component description, the prime goal of this paper is to show how the free-choice packing mechanisms developed for managing ambiguity in non-context free parsing [Maxwell and Kaplan, 1995] can profitably be extended beyond syntactic representations.

1 System Background

Conversion of text to KR proceeds as follows. The XLE is used to parse text against a broad-coverage, hand-written English grammar [Riezler *et. al.*, 2002]. The parse output is fed into a semantic interpreter, an implementation of the theory of “glue semantics” [Dalrymple, 2003] that produces fully scoped, higher-order intensional logical forms of the kind familiar to traditional formal semanticists. These logical forms are then flattened to a clausal form, similar in some respects to clausal forms obtained by skolemizing and flattening first order formulas. These clauses are then passed through the compo-

*This work builds on the efforts of numerous people. John Maxwell and Ron Kaplan for the library interface to XLE ambiguity management routines; Tracy King, Annie Zaenen, Anette Frank, Martin Forst, Stefan Riezler and Anubha Kothari for using and breaking the system while it was under development but coming back for more; Danny Bobrow, Cleo Condoravdi, Valeria Paiva, Reinhard Stolle, John Everett and Liz Coppock for work on KR; and Martin Kay for a prototype version of the rewrite system. Thanks also to the anonymous reviewers of this paper. This work was supported in part by the Advanced Research and Development Activity (ARDA)’s Advanced Question Answering for Intelligence (AQUAINT) Program.

ment described in this paper. This converts semantic representations induced by the linguistic structure of the sentence into an abstract knowledge representation more amenable to tractable reasoning. The abstract KR produced makes use of the CYC ontology [Lenat, 1995], and can be converted (with some loss of information) to a concrete KR formalism like CycL.

The entire system places an emphasis on robustness and dealing with ambiguity. All input texts produce an output KR, though sometimes only by virtue of using less precise back-off mechanisms. Ambiguities are packed into an efficient, chart-like representation that is common to all components of the system. This allows deferment of disambiguation decisions: soft, stochastic disambiguation can be left until after hard constraints from syntax, semantics and KR prune out impossible analyses.

1.1 Semantics to KR Mapping

Why is a mapping from semantic representations to knowledge representations required? Surely natural language, and hence natural language semantics, is the most natural knowledge representation there is? Given an ideal analysis of language and its semantics on the one hand, and an ideal analysis of knowledge representation and implementation of reasoning on the other, one might reasonably expect the two to coincide. A number of researchers have assumed no fundamental distinction between linguistic semantics and (the current state of the art in) KR. In some cases, this is a practical consequence of working on language understanding for limited domains. Here, linguistic analysis can be specially tailored to meet the domain, e.g. eliminating many forms of ambiguity at the outset, and/or avoiding anything other than special case analyses of inferentially troublesome higher-order or intensional constructions [Allen *et al*, 1996]. In other cases it is a consequence of the assumption that natural language semantics can be made to coincide with what is currently tractable in automated reasoning: Either through care and ingenuity in formulating a sophisticated first-order analysis [Hobbs, 1985, Blackburn *et al*, 2001], or through the assumption that cases falling outside of an essentially quantifier-free first-order logic are sufficiently rare as to be insignificant [Moldovan and Rus, 2001].

Even if first-order logic were sufficient for NL semantics, there is still a clash of compositionality between semantics and KR to be overcome. Semantic representations must respect the syntactic composition of the texts from which they are derived, to achieve a general and systematic syntax-semantics mapping. Consequently, the semantic representations assigned to sentences tend to be more complex, and different, than the representations a knowledge engineer would assign on a case-by-case basis when targeting a particular knowledge base. Additional processing is required to overcome such “impedance mismatches”, which are more evident when one is trying

to produce, say, database queries in SQL from semantic representations [Rayner, 1993]. Semantics to KR transformations include:

1. Map words / word senses onto terms in the target ontology.
2. Make meaning postulates / lexical entailments explicit in the KR.

For example (1) currently gets the KR (2):

- (1) The technician prevented an accident.
- (2) (maximallySimilar context0 context2 action1)
(sub_concept technician3 TechnicianWorker)
(doneBy action1 technician3)
(sub_concept action1 Eventuality)
(sub_concept accident4 AccidentUnforeseenEvent)
(instantiated context0 technician3)
(instantiated context0 action1)
(uninstantiated context2 action1)
(instantiated context2 accident2)
(uninstantiated context0 accident2)

This encapsulates a possible worlds-style lexical expansion of “prevent”: If A prevents B, then in the actual world (context0) there is an instance of A but no instance of B, whereas in a counterfactual world (context2) just like the actual world apart from the absence of A, there is a B. The representation says that there is some sub-concept of an action performed by a technician that is instantiated in context0 but uninstantiated in context2. And it says that there is some sub-concept of accident (the type of action that was actually prevented) which is uninstantiated in context0 but instantiated in context2. As discussed in [Condoravdi *et al*, 2001], this analysis accounts for downward monotonicity and non-existence entailments that are problematic under various first-order accounts.

3. Canonicalize compositionally distinct but equivalent semantic representations onto the same KR. The sentences (3) receive different compositional semantic analyses, but are both mapped to the same KR (4):

- (3) The technician cooled the room.
The technician lowered the temperature of the room.
- (4) (decreasesCausally lower_ev22 room24 temperatureOfObject)
(doneBy lower_ev22 technician23)
(sub_concept technician23 TechnicianWorker)
(sub_concept lower_ev22 ScalarStateChange)
(sub_concept room24 RoomInAConstruction)
(instantiated context0 lower_ev22)

4. Eliminate ontologically ill-formed analyses. For example, in “John saw a man with Mary”, selectional restrictions in the ontology may rule out the parse where the prepositional phrase modifies the verb “see”.

5. Reformulate intensional and higher-order aspects of the semantic representation in a form more amenable for KR, as illustrated in (2).

1.2 Semantic Input to Mapping

Glue semantics [Dalrymple, 2003] treats semantic interpretation as deduction. Parsing a sentence produces a set of lexical premises. These pair expressions in the semantic representation language with formulas in a separate glue logic; here, propositional, implicational linear logic. A glue derivation then assembles the glue premises, and the Curry Howard correspondence is used to read off the resulting semantic representation from the structure of the propositional glue derivation. Semantic ambiguity, such as scope ambiguity, is the consequence of alternative derivations of meanings being possible from a single unambiguous set of premises.

An implementation of a chart-based theorem prover [Hepple, 1996] is currently used for performing glue derivations. But it is due to be replaced by an alternative theorem prover that exploits the skeleton-modifier structure pervasive in glue derivations [Gupta and Lamping, 1998], and makes full use of the XLE’s ambiguity management library (section 2.2). Packing of ambiguity in semantic representations is currently computed non-optimally: Each syntactic analysis is unpacked, interpreted separately, and the first few scopings for each analysis are retained. Each semantic analysis is then separately skolemized/flattened. The multiple clausal representations are then repacked together, using linkages between skolem terms and packed syntactic constituents to rejoin the analyses.

Semantic interpretation of (5) yields alternative analyses including (ignoring tense, collective vs distributive quantification, etc. for simplicity):

(5) Two terrorists intended to attack a factory.

```
A1: exists2(t. terrorist(t) &
      intend(t, exists(f. factory(f) & exists(a,attack(a,t,f))))))
A2: exists2(t. terrorist(t) & exists(f. factory(f) &
      intend(t,exists(a,attack(a,t,f))))))
A3: exists(f. factory(f) & exists2(t. terrorist(t) &
      intend(t,exists(a,attack(a,t,f))))))
```

where A1, A2 and A3 label different scopings, and constitute distinct, mutually exclusive choices of analysis. *Intend* is an intensional predicate taking a propositional argument. The packed, flattened version of these analyses is

```
(6) A2 v A3: ist(context3,factory(sk_factory2))
      ist(context0, terrorist(sk_terrorist1))
      ist(context0, cardinality(sk_terrorist1,2))
A1:      ist(context0,factory(sk_factory2))
A2 v A3: ist(context3,factory(sk_factory2))
      ist(context0, intend(sk_terrorist1, context3))
      ist(context3, attack(sk_attack3, sk_terrorist1, sk_factory2))
      skfn(sk_terrorist1, context0)
```

```

A2:      sk_function(sk_factory2, sk_terrorist1)
A3:      sk_function(sk_factory2, context0)
A2 v A3: sk_function(sk_factory2, context3)
          sk_function(context3, sk_terrorist1)
          sk_function(sk_attack3, context3)

```

where $ist(Context, Clause)$ states that $Clause$ is true in $Context$; $sk_function(Term, Args)$ states that $Term$ is a (skolem) function from $Args$. Note how the propositional argument to *intend* has been replaced by a function referring to a context, and where the contents of the embedded proposition are spelled out as clauses holding in this context. Note also that the shared clauses of the three analyses have been packed together, while choices label the clauses specific to particular analyses.

2 Packed Rewriting System

The rewrite system used to map flattened semantic representations to KR is a reimplemented version of a transfer component originally devised by Martin Kay for use in machine translation [Frank, 1999]. As with [Oepen *et al*, 2004, Wahlster, 2000] rewriting is a resource-sensitive process. The input is a set of clauses, and rewrite rules apply in an ordered, stepwise manner to progressively replace input clauses by output clauses. Its novel feature is that it is able to operate directly on packed input, such as (6).

A possible rule to map a word onto a concept (% marks variables) is

```

(7) ist(%C, terrorist(%X)) ==>
    sub_concept(%X, Terrorist), instantiated(%C,%X).

```

This rule would match the first clause in (6) and replace it with the two clauses “sub_concept(sk1, Terrorist)” and “instantiated(context0, sk1)”.

The BNF for rewrite rules is shown in fig 1. By prefixing a pattern on the left hand side of a rule with a “+”, one 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 are forbidden from directly manipulating the set of input clauses; instead they allow one to perform table lookup or tests on terms, such as subsumption checking in the Cyc generalization hierarchy. In addition to specifying a set of replacement clauses (possibly empty), the right hand side of a rule can specify that a whole analysis path should be deleted.

General, non-consumable atomic facts can be asserted. e.g.

```

(8) /- cyc_concept_map(terrorist, Terrorist).
    /- cyc_concept_map(technician, TechnicianWorker).

```

Rule	::=	LHS ==> RHS.	Obligatory rewrite
		LHS ?=> RHS.	Optional rewrite
		/- Clause.	Permanent, unresourced fact
LHS	::=	ClausePattern	Match & delete atomic clause
		+ClausePattern	Match & preserve atomic clause
		LHS, LHS	Boolean conjunction
		(LHS LHS)	Boolean disjunction
		—LHS	Boolean negation
		{ProcedureCall}	Procedural attachment
RHS	::=	ClausePatterns	Set of atomic replacement clauses
		0	Empty set of replacement clauses
		stop	Abandon the analysis

Figure 1: Format of Rewrite Rules

Such assertions allow a more general specification of concept mappings via (where $qp(\%P, [\%X])$ expresses quantification over predicates, $\%P(\%X)$):

- (9) $ist(\%C, qp(\%P, [\%X])), cyc_concept_map(\%X, \%Y) ==>$
 $sub_concept(\%X, \%Y), instantiated(\%X, \%C).$

One might alternatively have a table of concept mappings, and access it via means of a procedural call

- (10) $ist(\%C, qp(\%P, [\%X])), \{get_cyc_concept(\%X, \%Y)\} ==>$
 $sub_concept(\%X, \%Y), instantiated(\%X, \%C).$

The formalism also permits macros 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 [Oopen *et al*, 2004] for producing compact rule sets.

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 in a number of useful ways (section 3), but it can sometimes be a nuisance. A way of marking blocks of rules that apply in parallel is due to be added to the system.

2.1 Packed Rewriting

There is nothing especially innovative about the range of rewrite rules available. What is significant is that the rules can be applied to packed input. Suppose that we have the following rule and packed input

- (11) Rule: $p(\%X), q(\%X) ==> r(\%X).$
Packed input: **C1**: $p(a)$ **C2**: $q(a)$ **C3**: $s(a)$

The rule matches the input only in the intersection of choices **C1** and **C2**, where both patterns on the left hand side can simultaneously be matched. This leads to the production of a new output fact, $r(a)$, only under this

intersected choice ($C1 \ \& \ C2$), and deletion of the matching input facts only under the same intersected choice. The matched input facts remain present under the residual choices that do not overlap with $C1 \ \& \ C2$. The input fact $s(a)$ is unaffected by the rule, and remains in its initial state. Thus the results of the rule application are

(12) Packed output:

$$\begin{array}{ll} C1 \ \& \ \neg C2: & p(a) & C2 \ \& \ \neg C1: & q(a) \\ C1 \ \& \ C2: & r(a) & C3: & s(a) \end{array}$$

Application of an optional rules conceptually splits an analysis into two disjoint parts. Rather than forking two separate rewrite sub-processes [Oopen *et al*, 2004], new choices are introduced to pack the two analysis paths together into the same process. For example:

(13) Rule: $p(\%X), q(\%X) \ ?=> \ r(\%X)$.

Packed input: $A1: p(a) \quad B2: q(a) \quad C3: s(a)$

The rule matches under choice $A1 \ \& \ B2$. This choice must now be split into two disjoint parts; $O1$ where the rule is applied, and $O2$ where the rule is not applied. This new split in the choice space can be represented as

(14) New choice: $(O1 \ \text{xor} \ O2) \ \text{iff} \ A1 \ \& \ B2$

which says that if you are under $C1 \ \& \ C2$ then you have a completely free choice of exactly one of $O1$ or $O2$, and that if you are under either $O1$ or $O2$, then you must also be under the choice $A1 \ \& \ B2$. The output of the rule is

(15) Packed output:

$$\begin{array}{ll} \neg(A1 \ \& \ B2) \ \vee \ O2: & p(a) & \neg(A1 \ \& \ B2) \ \vee \ O2: & q(a) \\ O1: & r(a) & C3: & s(a) \end{array}$$

Rewrite rules can also introduce new choices through the resolution of resource conflicts. Suppose that we have the following:

(16) $\text{/- cyc_concept_map}(\text{bank}, \text{BankFinancialInstitution})$.

$\text{/- cyc_concept_map}(\text{bank}, \text{BankEarthBarrier})$.

$\text{qp}(\%P, [\%X]), \text{cyc_concept_map}(\%P, \%C) \ ==> \ \text{sub_concept}(\%X, \%C)$.

Input: $1: \text{bank}(\text{sk3})$.

The two concept mapping facts mean that the rewrite rule can apply in two distinct ways to consume the input clause. But they cannot both simultaneously consume the one input clause. The rewriting system detects such competition over input resources, and resolves the conflict by introducing a new split in the choice space. One rule application will apply on one side of the split, and the other rule application on the other. Hence the output is

(17) $S1: \text{sub_concept}(\text{sk3}, \text{BankFinancialInstitution})$

$S1: \text{sub_concept}(\text{sk3}, \text{BankEarthBarrier})$

New choice: $(S1 \ \text{xor} \ S2) \ \text{iff} \ 1$

The rule ‘ambiguates’ the word “bank”, but ensures that the two senses sit under mutually exclusive choices. Thus “bank” cannot mean both financial institution and mound of earth, but means either one or the other.

2.2 Free Choice Packing

The packed rewriting system needs to compute efficiently boolean combinations of choices and determine whether they are satisfiable or not. Propositional satisfiability is an NP-complete problem. However, for typical language processing tasks, choice spaces can be put into a free-choice structure that allows for much more efficient boolean computation. The packing mechanisms in the XLE parser have been carefully constructed to exploit this possibility [Maxwell and Kaplan, 1995] and have been incorporated into the rewriting system. This section provides an overview of these mechanisms.

Free choice packing is a generalization of the use of charts in context-free parsing. Context-freeness guarantees that alternative analyses of disjoint word spans are independent and do not interact. Hence one can freely combine any analyses of disjoint spans; charts exploit this to compute all complete analyses in cubic time and quadratic space. For non-context free grammars (e.g. LFG), analyses of disjoint word spans are not always independent. But they are mostly independent, so that interactions are minimal. Under these conditions one can efficiently refactor the analyses to obtain an equivalent representation that guarantees free choice of alternatives.

To illustrate, consider a schematic free-choice representation of the 4-way syntactically ambiguous sentence “The deer ate the cabbage”.

- (18) Free choice structure: $(A1 \text{ xor } A2) \text{ iff } 1$
 $(B1 \text{ xor } B2) \text{ iff } 1$
- Choced clauses:
- | | |
|----------------------|-------------------|
| A1: sg(deer) | A2: pl(deer) |
| B1: count(cabbage) | B2: mass(cabbage) |
| 1: eat(deer,cabbage) | |

The choice structure is defined independently of the clauses that sit under the various choices — these could be syntactic, semantic or KR clauses, or a mixture. The choice structure says that starting under the top level choice 1, you have a completely free choice of exactly one of A1 or A2 (singular/plural deer), and a completely free choice of exactly one of B1 or B2 (count/mass cabbage), and that the A and B choices have no hidden interactions. The structure gives two independent 2-way choices, i.e. $2 \times 2 = 4$ analyses.

Suppose that semantic and KR processing said that if several deer are eating a single (count) cabbage, the cabbage must be a large one. This additional fact introduces interactions between the previously independent choices. But one can refactor the choice space to preserve freedom of choice:

- (19) Free choice structure: $(A1 \text{ xor } A2) \text{ iff } 1$
 $(B1 \text{ xor } B2) \text{ iff } A1$
 $(C1 \text{ xor } C2) \text{ iff } A2$

Choiiced clauses:

A1:	sg(deer)	A2:	pl(deer)
B1 v C1:	count(cabbage)	B2 v C2:	mass(cabbage)
C1:	large(cabbage)	1:	eat(deer,cabbage)

The new choice space is free in that starting under choice 1 you have a completely free choice of A1 or A2 (singular/plural deer). Then, depending on which you have chosen, you have a completely free choice of either B1/B2 or C1/C2 (count/mass cabbage). Under C1 (plural deer, single/count cabbage) is the further fact that the cabbage is large.

Free choice spaces have two important properties. First, they allow you to efficiently compute the number of analyses without having to enumerate each one: e.g. two disjoint 2-way choices in (19) leads to $2+2=4$ analyses. This is key for efficient stochastic disambiguation of packed structure [Riezler and Vasserman, 2004]. Second, propositional satisfiability is trivial to check. Choice structures correspond to and/or trees. A conjunction of two choices is unsatisfiable iff the lowest node in the and/or tree dominating both of them is an or-node. This is key to efficient packed rewriting.

But there is a price to be paid for these properties. Re-arranging the choice space to account for additional boolean combinations of choices increases the size of the choice space, and the complexity of the choices decorating individual clauses. In the worst case, where all choices interact with all other choices, the choice space can expand to a size proportional to the (exponential) total number of possible analyses. That is, in the worst case packing degrades into a disjunctive normal form enumeration of each analysis. The Maxwell-Kaplan wager is that these bad cases rarely arise for linguistically typical input. Choices from distant parts of sentences or text usually do not interact (coordinations and long distance dependencies being exceptions), and free choice spaces normally remain a manageable size.

Packing differs from underspecification as a technique for ambiguity management. Underspecification delivers a set of fragmentary analyses plus a set of constraints recording interactions between fragments and limiting the ways they can be put together. Packing delivers a chart-like structure that records all completely assembled analyses, and does not require further constraint satisfaction checks to read out or count individual analyses. Underspecification is a form of procrastination: work is not done on evaluating constraints until it is necessary, in the expectation that for many of the constraints it will never become necessary. Packing computes everything, whether it needs to or not, on the basis that it is often easier to do everything at once than it is to carefully distinguish what needs to be computed now from what can be left until later. While underspecification in semantics has attracted a lot of attention, packing has been relatively and unjustly neglected. To keep packed representations a manageable size, it is important to pull disjunctions of meanings out into the choice space, and not to represent

them explicitly as modalized disjunctions within the semantic representation (c.f. [Ramsay, 1999]). The latter approach tends to multiply out the size of representations when disjunctions get embedded and/or distributed in other disjunctions.

3 Affordances of Packed, Resourced Rewriting

Coverage and robustness are major issues when deriving KR from free text. Full, detailed KR coverage of texts is not likely to be achieved, but gaps in coverage should not lead to a failure to produce an analysis. Resourced rewriting allows one to specify an ordered sequence of backoffs and defaults to call on rules and external data of varying degrees of reliability. First, one applies carefully hand-coded rules to consume as much of the input as possible. Then a first level of backoff is applied to mop up some of the remaining input. This calls on language-to-KR mapping rules imported from Cyc. A second level of backoff consults other external data, such as WordNet and VerbNet in order to guess plausible concepts for unknown words bearing some similarity to known words. Finally, for any original input still remaining, concepts names are systematically invented. The most extreme form of backoff is to leave elements of the input unchanged. Invented concepts are of lesser use to a backend reasoning system with no ontological information about them, though they still enable inferences based on strict identity of concepts. Similar levels of backoff are used to deal with other phenomena, such as clausal complement verbs that set up contexts that are veridical (e.g. “know”), averidical (e.g. “believe”) or anti-veridical (e.g. “refute”). Unknown contexts default to being averidical.

Packing and conflict resolution handles ambiguous mappings, e.g. (16). The system can also eliminate ambiguities. A rule for mapping the verb “hire” and its grammatical arguments onto concepts and roles is

```
(20) hire(%E), subj(%E,%X), obj(%E,%Y),
      +sub_concept(%X,%CX), +sub_concept(%Y,%CY),
      {genls(%CX, Organization), genls(%CY,Person)}
      ==>sub_concept(%E, EmployingEvent),
      performedBy(%E,%X), personEmployed(%E,%Y).
```

Consider “the bank hired Smith”, assuming that rules have already enforced a choice split and assigned the nominal arguments to concepts, and the “bank” corresponds to either BankFinancialInstitution or BankEarthBarrier. Given that a procedural call to the genls hierarchy confirms that BankFinancialInstitution is a subtype of Organization, but not BankEarthBarrier, the rule will only apply under the choice where “bank” maps to BankFinancialInstitution. Applying the rule will consume the hire, subj and obj inputs, but leave them in place under the alternate choice where

bank maps to BankEarthBarrier. After various other concept and role mapping rules come disambiguation rules like

(21) subj(%,%) ==> stop.

If by the time the rule applies, there are any choices under which the original subj input still remains, these choices are abandoned. If no other rules have used up the subj input under the EarthBarrier choice, then the rule eliminates this reading.

Softer, stochastic disambiguation can in principle be applied to the packed output KR, using exactly the same mechanisms as for MaxEnt disambiguation of f-structure [Riezler *et. al*, 2002]. This has yet to be done, and depends on (a) defining a set of disambiguation features, and (b) obtaining lightly annotated material to train weights on these features.

4 Further Work

Evaluation material is not yet available for assessing the *quality* of the full text to KR system. Gold standard annotations are in preparation for seven articles taken from the Aquaint New York Times / Associated Press corpus (three for development, four for testing). But these are not complete, and the correct KR for these is still something of an undefined target. Other evaluation material like PropBank does not go as deeply as required, though it will be used to assess argument selection in the parser. In terms of *quantity*, full coverage is obtained.

The best that can be done in the meantime is to report, somewhat anecdotally, on the utility of the rewriting system. It is distributed as a fully integrated part of the XLE, under the XLE's research license. It has been used for a variety of tasks inside and outside PARC, including sentence condensation [Riezler *et. al*, 2003, Crouch *et al*, 2004], converting the TIGER dependency annotations to f-structures [Forst, 2003], constructing FrameNet annotations for German [Frank and Semecky, 2004], and also machine translation. Timing figures are only moderately informative, but with the current set of mapping rules packed semantic input from 30-40 word sentences can typically be converted in well under a second on a 2GHz processor.

Major effort is still required to build up the semantics-to-KR rules to minimize the number of times mapping backs off to default rules. There is also scope for further improvement in the rewriting system, including parallel rule application, improved rule indexing, further optimization of the order in which LHS patterns are matched, and tape limited recursion to allow a rule to reapply to its own output while preserving decidability.

References

[Allen *et al*, 1996] Allen, J., Miller, Ringger & Sikorski. A robust system for natural spoken dialogue. *Proc. 34th ACL*.

- [Blackburn *et al*, 2001] Blackburn, P., Bos, Kohlhase & Nivelde. Inference in computational semantics. In Bunt *et al* (eds) *Computing Meaning, vol 2*, Kluwer.
- [Condoravdi *et al*, 2001] Condoravdi, C., Crouch, van den Berg, Stolle, Paiva, Everett & Bobrow. Preventing existence. *Proc. Conference on Formal Ontologies in Information Systems*.
- [Crouch *et al*, 2004] Crouch, R., King, Zaenen & Riezler. Exploiting F-structure Input for Sentence Condensation. *Proc. LFG04*.
- [Dalrymple, 2003] Dalrymple, M. *Lexical Functional Grammar*. Syntax and Semantics, vol 34. Academic Press.
- [Forst, 2003] Forst, M. Treebank Conversion — Creating an f-structure bank from the TIGER Corpus. *Proc. LFG03*.
- [Frank, 1999] Frank, A. From parallel grammar development towards machine translation. *Proc. MT Summit VII*, Singapore.
- [Frank and Semecky, 2004] Frank, A. and Semecky, J. Corpus-based induction of an LFG syntax-semantics interface for frame semantic processing” *Proc. 5th Int. Conf. Linguistically Interpreted Corpora*.
- [Gupta and Lamping, 1998] Gupta, V. and Lamping, J. Efficient linear logic meaning assembly. *COLING-ACL’98, Montreal, Canada*.
- [Hepple, 1996] Hepple, M. A compilation-chart method for linear categorial deduction. *COLING-96, Copenhagen, Denmark*.
- [Hobbs, 1985] Hobbs, J. Ontological promiscuity. In *Proc 23rd ACL*.
- [Kaplan *et. al*, 2004] Kaplan, R., King, Riezler, Maxwell, Vasserman & Crouch. Speed and accuracy in shallow and deep stochastic parsing *HLT-NAACL’04, Boston, MA*.
- [Maxwell and Kaplan, 1995] Kaplan, R. and Maxwell, J. A method for disjunctive constraint satisfaction. *Formal Issues in Lexical-Functional Grammar*. CSLI Press.
- [Lenat, 1995] Lenat, D. Cyc: a large-scale investment in knowledge infrastructure.” *CACM 38:11*.
- [Moldovan and Rus, 2001] Moldovan, D and Rus, V. Logic form transformation of WordNet and its applicability to question answering. *ACL’01*.
- [Oepen *et al*, 2004] Oepen, S. *et al*. Som a hoppe etter Wirkola?. Ms.
- [Ramsay, 1999] Ramsay, A. Dynamic & underspecified semantics without dynamic and underspecified logic. In Bunt *et al* (eds) *Computing Meaning vol 1*, Kluwer.
- [Rayner, 1993] Rayner, M. *Abductive Equivalential Translation and its application to Natural Language Database Interfacing*. Ph.D. thesis, Royal Institute of Technology, Stockholm.
- [Riezler *et. al*, 2002] Riezler, S., King, Kaplan, Maxwell, Crouch & Johnson. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. *ACL’02, Philadelphia, PA*.
- [Riezler *et. al*, 2003] Riezler, S., King, Crouch & Zaenen. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. *Proc. HLT-NAACL’03*, Edmonton, Canada.
- [Riezler and Vasserman, 2004] Riezler, S. and Vasserman, A. Incremental feature selection and l1 regularization for relaxed maximum-entropy modeling *Proc. EMNLP’04*, Barcelona, Spain.
- [Wahlster, 2000] Wahlster, W. (ed) . *Verbmobil: Foundations of Speech-to-Speech Translation*