Transfer Semantics for the Clear Parser

Richard Crouch, Richard.Crouch@nuance.com
Nuance Communications Inc.
1198 E Arques Ave, Sunnyvale, CA 95014, USA

Transfer semantics uses a rewriting (or transfer) system to map syntactic parses onto semantic representations [4, 5]. This form of semantic mapping was used to build broad coverage semantic text indexes at Powerset, and has been used in various other settings such as question answering for intelligence analysts and medical QA. It has been used for English, German [14] and Japanese [13]. However, in all these cases the transfer semantics has been applied to the output of just one kind of parser: LFG functional structures created by the XLE parser [6]. This paper describes the adaptation of transfer semantics to apply to the output of the Clear dependency parser [2].

The paper is organized as follows. Section 1 describes the transfer system, and how it has been used to construct semantic representations from LFG f-structures. Section 2 describes the differences between f-structures and Clear dependency trees, and what needs to be done to adapt f-structure rewriting rules to apply to Clear dependencies. Section 3 concludes with a discussion of the pros and cons of a rewrite-based approach to semantic interpretation, with a particular focus on modularity, re-usability, and (non) compositionalty.

Semantic Transfer

Transfer Rules

The transfer system was originally developed for machine translation by Martin Kay. It uses an ordered sequence of rewrite rules to consume input structures and produce output structures in their place. A significant feature of the transfer system is that it applies efficiently to packed, ambiguous representations [8]. This means that N-best parse outputs can be converted to N(+/-M) semantic representations without having to run the mapping individually on each output.

A somewhat contrived example of a rewrite rule is:

```
PRED(%V, eat), SUBJ(%V, %S), OBJ(%V, %O), -OBL(%V, %%)
==>
word(%V, eat, verb), role(Agent, %V, %S), role(Theme, %V, %O).
```

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.

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

Figure 1: Format of Rewrite Rules

A description of the effects of these rules can be found in [5]. For the present note that

- Rules are applied in a linear order. The first rule takes a set of input facts, and may delete some of the facts and add some new ones. The second rule operates on this updated set of facts, and so on.
- Facts are treated as resources. Sometimes the left hand side of a rule may match against and consume a single input fact in multiple ways. In such cases the different rule applications resolve the resource conflict by splitting the space of choices in which the output facts reside. For example:

```
PRED(%X, %P), +sense_map(%P, %Sense) ==> SENSE(%X, %Sense).
|- sense_map(bank, riverside).
|- sense_map(bank, financialInstitution).
```

```
Input =      in choice 1 : PRED(var(1), bank)
Output =     in choice A1: SENSE(var(1), riverside)
             in choice A2: SENSE(var(1), financialInstitution)
             where 1 = xor(A1,A2)
```

Here, the predicate "bank" can be consumed in two different ways corresponding to two different senses. This splits the original space of possible interpretations into two disjoint sub-spaces: one, A1, where the predicate is consumed to give the riverside sense, and the other A2 where it gives the financial institution sense. Other unconsumed input facts remain in the original choice space. Packing in the transfer system allows for efficient continued mapping within split choice spaces.

Semantic Rules

Semantic rewriting of f-structures is broken down into a number of stages (again, see [reference] for more details)

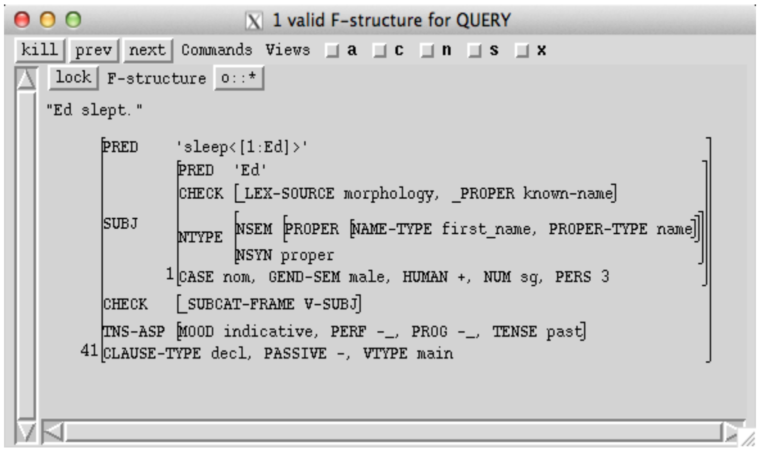
1. "Word-prime" semantics: map f-structures into basic predicate-argument structures. These are contexted predicate argument structures, in the sense that complement clauses and other constructions that take propositional argument introduce new semantic contexts.
2. Lexicalized semantics: consult lexical semantic information to e.g. replace word stem by senses, spell out connections between derivationally related nouns and verbs (e.g. destruction/destroy; [8]), determine the polarity/veridicality of arguments to verbs (know X vs believe X; forget that X vs forget to X[11,3]).
3. Contextually resolved semantics: anaphora resolution, etc.
4. Abstract KR: map the contextually resolved semantics to a format that efficiently supports certain kinds of subsumption-based entailment and contradiction reasoning [1]
5. KR: map the domain neutral semantics to a knowledge representation language and ontology better suited for reasoning in a particular application.

Fuller descriptions of the kinds of mapping performed at these various stages can be found in [references]. The adaptation of the mapping rules to other languages is described in [13,14]. For the current paper, the point to note is that there is a pinch point between the word-prime and the lexicalized semantics. If outputs from other parsers than XLE can be mapped similar predicate argument structures to those obtained from XLE, then the remaining mappings can apply from there.

Transfer Semantics for Clear

F-Structures and Dependencies

The functional structures produced by syntactic analysis in XLE represent both basic dependency relations like subject and object, but also a rich variety of other morphological and lexical information (e.g. tense, aspect, mood, gender) that is important for driving semantic interpretation. The Clear dependency structures by contrast appear very sparse.



```
"Ed slept."

   PRED      'sleep<[1:Ed]>'
             PRED   'Ed'
             CHECK  [LEX-SOURCE morphology, _PROPER known-name]
   SUBJ      NTYPE  NSEM  PROPER [NAME-TYPE first_name, PROPER-TYPE name]
                    NSYN  proper
           1 CASE nom, GEND-SEM male, HUMAN +, NUM sg, PERS 3
   CHECK     [SUBCAT-FRAME V-SUBJ]
   TNS-ASP   MOOD indicative, PERF -_, PROG -_, TENSE past
41 CLAUSE-TYPE decl, PASSIVE -, VTYPE main
```

Clear Dependencies

```
1 Ed     ed     NNP  _              2  nsubj  2:A0
2 slept  sleep  VBD  pb=sleep.01    0  root   _
3 .      .      .    _              2  punct  _
```

In order to re-use the mapping rules for constructing lexicalized semantics and beyond (steps 2—5 above), it is necessary to first map the clear dependencies to a word-prime semantics that is as similar as possible to the word-prime semantics derived from f-structures. This requires the reconstruction of the additional information that is explicitly represented in f-structure.

Additional Resources Needed

Tense, Aspect, and Mood

To reconstruct the tense and aspect information found in f-structure, the part of speech assignments and auxiliary structure of clear dependencies need to be inspected. A relatively simple set of patterns, coded up as transfer rules, can be used to make the appropriate assignments:

```
is_aux(%Aux, %V) :=                                 "Auxiliary verb macro"
   ( +dep(aux, %Aux, %V) | +dep(auxpass, %Aux, %V) ).

not_aux(%V) :=                                      "Main verb macro"
  -dep(aux, %V, %%),  -dep(auxpass, %V, %%).

has_no_aux(%V) :=                                   "Main verb with no auxiliaries"
  -dep(auxpass, %%,%V), -dep(aux, %%,%V), -dep(auxpass, %V,%%), -dep(aux,%V,%%).

+part_of_speech(%Aux, VBD), @is_aux(%Aux, %V)  ==> add_clause(%Aux, past(%V)).
+part_of_speech(%V, VBD),   @has_no_aux(%V)    ==> add_clause(%V, past(%V)).
+part_of_speech(%Aux, VBP), @is_aux(%Aux, %V)  ==> add_clause(%Aux, pres(%V)).
+part_of_speech(%V, VBP),   @has_no_aux(%V),
                -possible_imperative(%%, %V)   ==> add_clause(%V, pres(%V)).
+part_of_speech(%Aux, VBZ), @is_aux(%Aux, %V)  ==> add_clause(%Aux, pres(%V)).
+part_of_speech(%V, VBZ),   @has_no_aux(%V)    ==> add_clause(%V, pres(%V)).
+part_of_speech(%Aux, VBG), @is_aux(%Aux, %V)  ==> add_clause(%Aux, progr(%V)).
+part_of_speech(%V, VBG),   @not_aux(%V)       ==> add_clause(%V, progr(%V)).
+dep(aux, %Aux, %V), +stem(%Aux, have)         ==> add_clause(%Aux, perf(%V)).
```

Reconstruction of mood (interrogative, declarative, imperative) is similar, though a little more involved. It requires identifying the main verb of the sentence and looking at its part of speech, or that of any auxiliaries attaching to it to determine if the sentences is finite (declarative, interrogative) or infinitival (imperative, provided that the main verb is missing its subject). To distinguish declaratives from interrogatives the presence of subject-auxiliary inversion and/or the presences of Wh-terms (who, what, why, where, etc) needs to be determined. Also interrogative Wh-terms needs to be distinguished from those used for form relative clauses. But this can all be extracted from the Clear parses with a handful of rules encoding the necessary patterns.

Argument Structure

Though not illustrated in the simple "Ed slept" example, f-structures also make explicit elements of argument structure arising from raising and control verbs that are completely missing from Clear dependencies.[1] For example

(1) John promised Mary to leave.
(2) John persuaded Mary to leave.

In (1) John is the implicit subject of the embedded leave clause, whereas in (2) Mary is the subject of the leaving. The clear parser assigns both sentences the same dependency structure, and so simple pattern

[1] Clear's semantic role labeling does fill in extra argument structure not encoded by pure syntactic dependencies. But it does so in a sometimes erratic way.

matching rules on the dependencies will not reconstruct the missing information in the way that they could for tense, aspect, and mood.

To overcome this deficit, it helps for the transfer rules to incorporate subcategorization information from the XLE lexicon. In this case the XLE lexicon assigns the frame V-SUBJ-OBJ-XCOMPinf to “persuade”, and V-SUBJ-OBJ-XCOMPinf_scon to “promise”. Both frames correspond to the same syntactic valence, but the addition “scon” for promise indicates that the verb’s subject (rather than object) should act as the subject of the subordinate clause.

Likewise, similar additional lexical information can be “stolen” to recover information about gender, name types, and so forth.

Evaluation

How does one judge how well a set of transfer mapping rules is working? The work reported here arose from a set of circumstances that suggest at least one evaluation strategy. The starting point was a language processing pipeline that used XLE plus transfer semantics to map user utterances to a conversational user interface for an entertainment system into formal representations that drive processing in the back-end system. The question was whether another parser could be plugged into the pipeline in place of the XLE. To answer the question, one therefore needs to plug in another parser and see how the overall system behavior changes.

Fortunately, this kind of in situ evaluation can be decomposed. The XLE grammar and semantics comes with a regression set of a few thousand sentences designed to have reasonably broad coverage of a range of syntactic and semantic phenomena. These were used to track and guide the initial development of transfer rules to map Clear dependencies onto word-prime, contexted predicate argument structures. The regression sentences were pushed through the XLE pipeline up to the level of word prime semantics, and the results stored. The same sentences were (repeatedly) pushed through the clear pipeline, and differences noted, prioritized, and corrected.

After about one person-month of rule development, the Clear and XLE word-prime pipelines were in exact agreement on over 30% of the test sentences, with very minor discrepancies on another 40%. Most of the remaining cases were due to XLE and Clear assigning completely incompatible syntactic structures to the sentences, from which one would not ever expect to derive similar semantic analyses. Genuine and significant differences between semantic representations were confined to certain kinds of construction (e.g. partitive noun phrases, tag questions) which had been marked as relatively unimportant for the application domain, and so no effort had been put into the mapping rules.

At this point, evaluation moved to the consideration typical sentences for the application domain. For these, the Clear and XLE parses were pushed all the way through the transfer pipeline to the point of generating abstract KR representations. On a test set of 50 sentences, exact matches were obtained for 40 of them, with 6 of the remaining differences being due to incompatible parses.

Thus, taking the XLE representations as a baseline, a relatively modest investment of effort got the Clear and XLE semantic representation to a level of around 90% agreement (when parses are compatible).

Since this initial evaluation, the Clear semantic pipeline has been employed in a medical question answering application, both to process questions, and also to analyze medical documents in order to construct a semantic search index in which to look for answers. No focused evaluation of the efficacy of the Clear pipeline in this application has yet been carried out, but anecdotally it has not been the cause of major problems.

Discussion

In practical terms, the use of a rewriting system to map syntactic representations into semantic ones has shown itself to be flexible, scalable and robust. A core set of rewriting rules has been used for (a) different application domains, (b) different languages, and (c) the outputs of different kinds of parser. The semantic rules have broad coverage, and so long as the parser is up to it, can deal with input ranging from short questions, to Wikipedia pages, to medical monographs.

But in theoretical terms there is a lingering suspicion of inadequacy. Compared to the elegance of compositional or type-driven syntax-semantics interfaces, the application of largely unconstrained and moreover ordered rewrite rules to the task just doesn't look mathematically very respectable. This is not to impugn the theoretical interest and coherence of an ambiguity-packed, resource-sensitive and ordered rewrite system like the Transfer system. But the mathematical underpinnings of such a rewrite system are not the topic of this paper, and nor do they obviously have a bearing on the theoretical applicability of such a system to the syntax-semantics interface.

But the usefulness and flexibility of the rewrite approach does lead one to wonder if there is in fact any theoretical insight to be gained. A few observations, pro and con, are in order.

Modularity: An ordered sequence of rules feeding and bleeding into one another imposes very little natural discipline on a rule writer. It is all too easy to construct a tangled, unmaintainable mess. But just because the formalism allows you to do such things, it does not follow that you are obliged to do. As an analogy, typing in programming languages can impose a very helpful discipline on programmers. But this does not mean that you can't write clean modular code in an untyped language, any more than it means that you can't write a tangled unmaintainable mess in a typed language. With effort, and the core set of transfer rules have been subject to considerable effort and repeated refactoring, it is possible to come up with a configurable and re-usable set of rules.

That being said, the greatest drawback to the rewriting approach to semantics in terms of modularity is its lack of reversibility. Unification-based grammars (such as LFG) and unification-based semantics, for example, allow exactly the same components to be used in a forward (analysis) direction, and a reverse (generation) direction [reference SHD gen]. The same cannot be said of resource consuming rewrite rules: there is absolutely no guarantee that one can run a set of rules in reverse in such a way as to reproduce the input that was consumed by those rules going in a forward direction. In practice, one has to write a separate set of generation rules.

Ambiguity: The rewriting approach extends naturally and efficiently to dealing with packed, ambiguous input. Given the prevalence of ambiguity in natural language, this is a significant feature. But it is a topic that is strikingly missing from most discussions of compositional or type-driven semantics. In principle it may be possible to pack a type theory, since as [9] points out packing can be superimposed on a variety of base logics. But to my knowledge this has yet to have been attempted.

Compositionality: Transfer semantics enforces no kind of compositionality. In the absence of a fixed syntactic representation, compositionality can be something of a vacuous constraint [12]. But in the case of XLE and Clear, the syntax is fixed in advance of the semantics, so that the constraint does have some bite. But to go out on a limb, it is unclear how beneficial compositionality really is.

Alternate compositionalities: When semantic representations are generated for an additional purpose, like reasoning in a back-end system, there is frequently a mapping from the semantic representation to an inference optimized representation. The structure of the semantic representation reflects the language input, whereas the structure of the inference engine reflects the operations and sub-formula decomposition performed by the inference engine. That is, language and inference impose their own compositional and decompositional structures on representations. Especially with the current state of automated inference (no one actually reasons with Montague's higher order intensional logic), there is no reason to expect these structures to coincide. The transfer mapping was originally introduced just for mapping between compositionally derived semantic representations and more inferentially tractable representations [4]. Since then, it has proved convenient to push the non-compositionality further back in the pipeline.

Meaning sensitivity: In a strictly compositional approach like Glue Semantics [7] the derivation of meaning representations proceeds independently of the actual lexical meanings of the words: everything is by type combinatorics. This allows for useful optimizations of composition algorithms, but is a nuisance when you actually want the meaning structures to depend on the meanings of words. A case in point is the differing behaviors of the English perfect tense when combining with verbs of different aspectual classes. It is extremely difficult to code up the necessary lexical variations without in effect having a collection of if-then rules based on lexical properties. The ability to do precisely this in transfer constitutes a liberating release from compositionality.

Syntax neutrality (rule-to-rule): Many approaches to compositional semantics assume a rule-to-rule hypothesis, whereby semantic construction is driven by the grammar rules used. This is problematic in the case of statistical parsers where it can be hard to identify the rules used, or indeed in porting a compositional semantics from one grammatical framework to another. Transfer semantics does not require a rule-to-rule hypothesis, which greatly improves portability. However, in the right setting for compositional semantics this kind of portability can still be achieved. For example, Lev [10] used the transfer system to derive glue premises through the inspection of output XLE f-structures (description-by-analysis, rather than a rule-to-rule style co-description). Given the premises, glue composition could proceed in a meaning insensitive (but at least reversible way). Moreover, a different set of lexical transfer rules could in principle be applied to generate similar glue premises for the outputs of other parsers.

References

(1) Bobrow, Danny, Condoravdi, Crouch, Kaplan, Karttunen, King, de Paiva, Zaenen, 2005. "A basic logic for textual inference" *Proc. AAAI Workshop on Inference for Textual Question Answering*.

(2) Choi, Jinho and Martha Palmer, 2011. "Getting the most of transition-based dependency parsing" *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*: short papers - Volume 2, pp687—692.

(3) Condoravdi, Cleo, Crouch, van den Berg, Stolle, de Paiva, Everett & Bobrow, 2001. "Preventing existence" *Proc. Conference on Formal Ontologies in Information Systems*.

(4) Crouch, Richard, 2005. "Packed rewriting for mapping semantics to KR", *Proc. 6th Int. Workshop on Computational Semantics*, pp 103—114, Tilburg.

(5) Crouch, Dick and Tracy Holloway King, 2006. "Semantics via f-structure rewriting", *Proc. LFG06 Conference*, pp 145—165.

(6) Crouch, Dalrymple, Kaplan, King, Maxwell, Newman, 2008. *XLE Documentation*, Palo Alto Research Center.

(7) Mary Dalrymple, 1999. *Semantics and syntax in Lexical Functional Grammar: The resource logic approach*. MIT Press.

(8) Gurevich, Olya, Crouch, King, de Paiva, 2008. "Deverbal nouns in knowledge representation" *Jnl. Logic and Computation*, 18(3) pp.385—404, OUP.

(9) Ron Kaplan and John Maxwell, 1995. "A method for disjunctive constraint satisfaction" in *Formal Issues in Lexical-Functional Grammar*. CSLI Press

(10) Lev, Iddo. 2007. *Packed computation of exact meaning representations*. PhD dissertation, Stanford University.

(11)Nairn, Rowan, Condoravdi and Karttunen, 2006. "Computing relative polarity for textual inference" *ICoS-5*.

(12) Partee, Barabara, 1984. "Compositionality" in *Varieties of Formal Semantics* vol3, pp 281—311.

(13) Umemoto, Hiroshi and Keigo Hattori, 2011. "Experiments of FX for NTCIR-9 RITE Japanese BC Subtask", *Proc. NTCIR-9 Workshop*, Tokyo.

(14) Zariess, Sina, 2009. "Developing German semantics on the basis of parallel LFG Grammars", *Proc. Workshop on Grammar Engineering across Frameworks*, pp 10—18.