Events in Glue Semantics

Research Memo #glue-3
Iddo Lev
April 6, 2006

1 The Basic Analysis

1.1 The Basic Idea

The basic proposal here follows (Fry, 1999a,b) with some modifications, except that we present it in the new Glue Semantics notation (as in (Dalrymple, 2001) and later work) rather than the old style (as in (Dalrymple, 1999) and earlier work).

A verb's semantics, after all its arguments are combined with it, is a property of events. For example, "John greeted Mary" describes the property of events: $\lambda e.greet(e) \wedge past(e) \wedge agent(e, john) \wedge patient(e, mary)$.[1] VP modifiers contribute conjuncts to this property. For example, "on Monday" adds a conjunct to this property: $\wedge\ time(e) \sqsubseteq Monday$. In a declarative sentence, the entire property is eventually closed off by an existential quantifier $\exists e$ at the sentential level. However, the property is not existentially closed off in some cases where the clause is embedded (e.g. "John cried when Mary left". See (Francez and Pratt, 1997)).

We revise the semantics of verbs to yield not a truth value but an event predicate. For example:

(1) [[John]$_2$ greeted [Mary]$_3$ [on [Monday]$_5$]$_4$]$_1$
  a. $johh : e_2$
  b. $mary : e_3$
  c. $\lambda x \lambda y \lambda v.greet(v) \wedge agent(v, x) \wedge patient(v, y) : e_2 \rightarrow e_3 \rightarrow ep_1$
  d. $\lambda P.\lambda v.P(v) \wedge on(v, monday) : ep_1 \rightarrow ep_1$
  e. $\lambda P.\lambda v.P(v) \wedge past(v) : ep_1 \rightarrow ep_1$
  f. $\lambda P.\exists v.P(v) : ep_1 \rightarrow t_1$

We use here a notation for the right-hand-side following (Kokkonidis, 2006). It is different from what we have used in previous documents.[2] We do this for convenience: in the other style, we would say that the category **1** is of type t, but it has a sub-label that represents the event predicate. We may want to call this sub-label ep, but we may also want to use ep for the type designation. So instead of writing $\mathbf{1}^{ep}_{ep}$, we just write ep_1 here.

The statements (1)a-b are as usual. The statement (1)c is contributed by the verb. Once it is applied on (1)a-b, we get the event predicate (2)a. The statement (1)d comes from the VP modifier "on Monday" and when applied on (2)a it yields (2)b. The statement (1)e is contributed by the tense morpheme, yielding the extended event predicate (2)c (the value of having this contribution separate from the verb's is more clear when the tense resides in a separate auxiliary such as *did* or *will*). Finally, (1)f is contributed by the declarative sentence, producing the final meaning (2)d.

[1] We ignore for now the "time of utterance". It should be an argument of $past(e)$.

[2] Perhaps the notation in all the documents should be changed accordingly.

(2) a. $\lambda v.greet(v) \wedge agent(v, john) \wedge patient(v, mary) : ep_1$.
b. $\lambda v.greet(v) \wedge agent(v, john) \wedge patient(v, mary) \wedge on(v, monday) : ep_1$
c. $\lambda v.greet(v) \wedge agent(v, john) \wedge patient(v, mary) \wedge on(v, monday) \wedge past(v) : ep_1$
d. $\exists v.greet(v) \wedge agent(v, john) \wedge patient(v, mary) \wedge on(v, monday) \wedge past(v) : ep_1$

This idea can be extended in various ways to account for topic time and aspect (see (Francez and Pratt, 1997; Fry, 1999b)).

Note that this account interacts correctly with quantifiers. Because the second argument of a quantifier modifies the t resource of a clause, the event variable in the clause's meaning must first be existentially closed before that meaning is given as argument to the quantifier. For example:

(3) Every man saw some woman. ('every' wide scope reading:)
$every(man, \lambda x.some(woman, \lambda y.\exists e.see(e) \wedge experiencer(e, x) \wedge theme(e, y) \wedge past(e)))$

1.2 Fleshing It Out

Several things remain to be explained. First, *ep* stands for "event predicate", and is effectively a shorthand for the type $ev \rightarrow t$, where ev is the type of events. The statement (1)c is a bit unusual compared to all we have seen so far, because while e_2 is associated with x and e_3 with y, ep_1 is associated not with v but with the entire result $\lambda v.[\ldots]$.[3] (We use v for the event variable rather than the conventional e so as not to cause confusion with the type e).

Second, the generic lexical entry for any verb includes the statement

(4) $\lambda P \exists v.P(v) : ep_l \rightarrow t_l$ where l is my clause's label

which closes off the event property with an existential quantifier. In addition, each verb contributes its core meaning:

(5) intransitive verb m:
$\lambda x \lambda v.\ m(v) \wedge agent(v, x) : e_j \rightarrow e_k \rightarrow ep_l$
transitive verb m:
$\lambda x \lambda y \lambda v.\ m(v) \wedge agent(v, x) \wedge patient(v, y) : e_j \rightarrow e_k \rightarrow ep_l$
...
where l is my clause's label, j is my agents's label, k is my patient's label ...

The *lexical* syntax-semantic interface links thematic descriptions such as "my agent's label" to descriptions in terms of syntactic function such as "my subject's label".

Third, the contribution of an adverb has the following basic structure:[4]

[3] There is a general issue in the syntax-semantics interface of glue semantics, whether to allow semantic resources of such higher types, or whether to always spell out such types explicitly and only use semantic resources of atomic types. For example, in the semantics of nouns and adjectives, we could use just one glue resource of type et, which is a shorthand for $e \rightarrow t$, instead of two separate resources of types e and t. If higher types are allowed, then after all the glue statements are collected, they may need to undergo a step of conversion (to expand such shorthands) before the computation of derivations, although that depends on how the derivations are calculated. In any case, this issue is orthogonal to the main points of this paper.

[4] For an analysis of gradable VP modifiers such as "quickly", see (Lev, 2005a).

(6) adverb m:
$\lambda P \lambda v.P(v) \wedge m(v) : ep_l \rightarrow ep_l$
where l is the label of the clause of the VP that I modify

A tense morpheme can contribute information in a similar manner:

(7) +ed on a verb
$\lambda P \lambda v.P(v) \wedge past(v) : ep_l \rightarrow ep_l$
where l is the verb's clause's label

The basic entry for a preposition is:

(8) preposition p:
$\lambda x \lambda v.p(v, x) : e_k \rightarrow pred_l$
where k is the label of my NP argument and l is the label of my PP

If we take *pred* to be some generic or polymorphic type of predicate, then this entry could work for both PPs that modify VPs and PPs that modify NPs. The grammar rules themselves[5] would contribute glue statements:

(9) "VP $\rightarrow$ VP PP_j" adds:
$\lambda Q \lambda P \lambda v.P(v) \wedge Q(v) : pred_j \rightarrow ep_l \rightarrow ep_l$
where l is my clause's label

"N' $\rightarrow$ N' PP_j" adds:
$\lambda Q \lambda P \lambda v.P(v) \wedge Q(v) : pred_j \rightarrow (e_{v(l)} \rightarrow t_l) \rightarrow (e_{v(l)} \rightarrow t_l)$
where l is the label of my highest N'

For the particular sentence in (1), this is instantiated as:

(10) $monday : e_5$
$\lambda x \lambda v.on(v, x) : e_5 \rightarrow pred_4$
$\lambda Q \lambda P \lambda v.P(v) \wedge Q(v) : pred_4 \rightarrow ep_1 \rightarrow ep_1$
$\Rightarrow \lambda v.on(v, monday) : ep_1 \rightarrow ep_1$

1.3 Dealing with Duplicate Derivations

Finally, note that there is actually another non-equivalent normal derivation for (1), where first (1)e combines with (1)a-c, and the result combines with (1)d. But this derivation yields a logically-equivalent semantics: $\exists v.greet(v) \wedge agent(v, john) \wedge patient(v, mary) \wedge past(v) \wedge on(v, monday)$. In general, with n VP modifiers (including tense, aspect, etc.) we will have $n!$ non-equivalent derivations that all yield logically-equivalent results. This problem also arises with N' modifiers (adjectives, PPs, and relative clauses), except that some of those modifiers may have a non-intersective semantics so it is important to restrict their order of combination according to their order in the sentence.

One way to deal with this is to extend the idea that was mentioned in section 3.4 of (Lev, 2006) (cf. (Crouch and van Genabith, 1999)): Each basic glue statement is associated with a scope label, and scoping constraints between these labels constrain

[5] Or some other part of the grammar, such as the f-structure in LFG.

the order in which these statements may be used in a derivation. Thus, with VP modifiers, we may constrain the glue statement from the first VP modifier to be used before the one from the second VP modifier, and so on, and constrain all of these to be used before the contribution from the tense morpheme (this, by the way, does not affect at all the possibility of NPs inside the VP modifiers from taking scope around the clause or their relative scoping).

A variant of this idea which does not require altering the glue derivation algorithm to take account of scoping constraints is to tag type categories with more indices. In general, change sets of statements like (11)a to (11)b, which has a unique normal derivation. This can be done when instantiating the labels in generic lexical contribution of glue statements by letting the instantiating procedure know about the linear position of the modifiers in the sentence.

(11) a. $a \to b,\ b \to b,\ \ldots\ ,\ b \to b,\ b \to c$
b. $a \to b_1,\ b_1 \to b_2,\ \ldots\ ,\ b_{n-1} \to b_n,\ b_n \to c$

Yet another idea is to define the glue contribution of a verb to already include all its arguments and modifiers (including even the tense and aspect modifiers). In fact, this idea is related to the following idea.

1.4 Reducing Duplicate Work and Adding Robustness

If we define a separate template for a glue statement for each verb subcategorization frame, there will be a lot of repetitive work. For example, intransitive, transitive, and di-transitive verbs in active voice all share the facts that their first argument is the agent and their result is an event predicate. It seems like a good grammar engineering practice to have just one general entry for verbs that takes care of all possibilities.

Moreover, to increase robustness of the system, we do not want it to fail on a verb that is used with an unexpected set of arguments for which the lexicon does not include an appropriate glue statement. We want our glue semantics statements to potentially work with parses given by a statistical parser. Even if we have a precise parse based on a hand-built grammar, there is no need to duplicate in the glue lexicon the work of subcategorization from the syntactic lexicon – it is enough to know that the parser produces a syntactic structure and to collect all the verb's arguments and modifiers as they appear in that structure.

To achieve this, we can have the following general template for verbs:

(12) verb m:

(a) $\lambda x_1 \ldots \lambda x_n \lambda v.m(v) \wedge role_1(v, x_1) \wedge \ldots \wedge role_n(v, x_n) : c_1 \to \ldots \to c_n \to ep_l$
where l is my clause's label, n is the number of arguments I have, and for $1 \leq i \leq n$, c_i is the typed category of my ith argument, and $role_i$ is the thematic role of my ith argument.

(b) $\lambda P_1 \ldots \lambda P_n \lambda P \lambda v.P(v) \wedge P_1(v) \wedge \ldots \wedge P_n(v) : c_1 \to \ldots \to c_n \to ep_l \to ep_l$
where l is my clause's label, n is the number of modifiers I have, and for $1 \leq i \leq n$, c_i is the typed category of my ith modifier.

We separate the combination of arguments from the combination of modifiers just for convenience. We let the semantics of modifiers (including adverbs, tense, and aspect) simply be predicates that can apply on the event variable rather than being modifiers of an event-predicate (so e.g.: $\lambda v.past(v)$ instead of $\lambda P \lambda v.P(v) \wedge past(v)$).

If one does not wish to combine all the verb's arguments into one statement as in (12), one can use a basic verb statement $m : ep_l$ and a separate statement $\lambda x \lambda P \lambda v.P(v) \wedge role_i(v, x) : c_i \rightarrow ep_l \rightarrow ep_l$ for each argument. To prevent $n!$ spurious permutations of using argument and VP-modifier contributions, all these $ep_l \rightarrow ep_l$ modifiers can be marked as saying that their order of application is unimportant. The algorithm that computes glue derivations could then assign them some arbitrary order, expressed using scoping constraints (Lev, 2006, sec. 3.4).

The only exception to the idea above is equi and raising verbs. See (Asudeh, 2002) for an analysis. For a raising verb as in "Gonzo seemed to leave", the entry for "seem" should not include the typed category of its subject since it does not participate in the semantics $seem(leave(Gonzo))$. For an equi verb as in "Gonzo tried to leave", the entry for "try" should not include the typed category c of its clausal complement, but instead, it should include $d \rightarrow c$, where d is the typed category of the complement's subject. The semantics of the complement is the property $\lambda x.leave(x)$, and this is the second argument of "try" giving the semantics: $try(Gonzo, \lambda x.leave(x))$.

References

Asudeh, Ash. 2002. A resource-sensitive semantics for equi and raising. In *The construction of meaning*, ed. David Beaver, Stefan Kaufmann, Brady Clark, and Luis Casillas. CSLI Publications.

Crouch, Richard, and Josef van Genabith. 1999. Context change, underspecification, and the structure of glue language derivations. In (Dalrymple, 1999).

Dalrymple, Mary, ed. 1999. *Semantics and syntax in lexical functional grammar: The resource logic approach*. MIT Press.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics Series*. Academic Press.

Francez, Nissim, and Ian Pratt. 1997. Derivation of temporal preposition meanings in LFG's glue-language approach. In *Proc. of LFG'97*.

Fry, John. 1999a. Resource-logical event semantics for LFG. Talk given at the LFG'99 conference.

Fry, John. 1999b. Resource-logical event semantics for LFG. Unpublished draft.

Kokkonidis, Miltiadis. 2006. First-order glue. *Journal of Logic, Language and Information* To appear.

Lev, Iddo. 2005a. Comparative constructions. Addendum to Lev (2005b).

Lev, Iddo. 2005b. Gradable comparatives: Syntax and syntax-semantics interface. Paper for Ling221B, Stanford University. http://www.stanford.edu/∼iddolev/.

Lev, Iddo. 2006. Introduction to the syntax-semantics interface using glue semantics. Research Memo #glue-1.