Some Steps in Formalizing Events

Paper for Ling233, “Tense and Aspect”, Autumn 2005

Iddo Lev

October 2, 2006

1 Introduction

(Moens and Steedman, 1988) discuss various ways in which the aspectual class of an event may shift if the event is modified by various operators and adjuncts. However, they do not provide a formalization of their framework in terms of semantic representations. In this paper, I propose some steps towards such a formalization. I show that the representations need to be rich enough to capture the various cases that appear in language, and that meaning postulates and general world knowledge tie these representations together. I then discuss the issue of a gradual process and propose that it requires a further distinction in the representations, improving upon the proposals given in the Zucchi-Parsons debate.

2 Background: Aspectual Classes and Shifts

2.1 Aspectual Classes

(Moens and Steedman, 1988) give the following classification of eventualities into “aspectual classes”, a revision of (Vendler, 1967)’s classification. *Process* is Vendler’s *activity*, *culminated process* corresponds to his *accomplishment*, while his *achievement* is divided here into *culmination* and *point*, both atomic events, but they differ w.r.t. whether they result in a meaningful consequence state: reaching the top of a mountain results in the state of being at the top, while there is no standard state that hiccuping leads to.

(1)

	Events		States
	atomic	extended	
+conseq	culmination	culminated process	
−conseq	point	process	

Each combination of a predicate and its set of arguments is lexically specified for a certain aspectual class. The classification depends on the ontological type of the arguments as well as on some of their grammatical features. Thus, the basic classification says that intransitive *eat* is a process, *eat* + *an apple* is a culminated process, while *eat apples* is again a process (there is no specific amount of apples such that eating it can be considered as culminating the event). Other examples are: *know + NP* is a state, *arrive at NP* is an achievement, and *jump* is a point.

2.2 Aspectual Shifts

In general, every event belongs to one of the aspectual classes. An event's class can be coerced to another class if there is a clash between the event's class and the requirements of operators modifying the event. Grammatical operators that potentially cause a shift in an event's aspectual class include:

- present vs. non-present tense
- progressive vs. non-progressive
- perfect vs. non-perfect

In addition, various VP-modifiers may cause such shifts. I will briefly review here a few examples from (Moens and Steedman, 1988), and I will later return to them in more detail.

The basic class of *hiccup* is *point*, but the progressive in (2) requires a process.

(2) Sandra was hiccupping.

This clash results in a coercion of the point event to a process event. There are different kinds of possible coercions (as we shall see in section 4), and here the most likely one is iteration, which yields the process of Sandra repeatedly hiccuped. Moens & Steedman give the following informal representation:

(3) (progressive (process (iteration (point (*Sandra hiccup*)))))

To *run a mile* is a culminated process, but again, the progressive in (4) requires a process. This time, the clash is resolved by stripping the culmination and shifting to the preparatory process part of running. That the culmination point need not occur can be seen in the second part of (4).

(4) Roger was running a mile, but he gave up after two laps.

To *reach the top* is a culmination. The clash with the progressive in (5) is resolved by first shifting the culmination to a culminated process and then stripping the culmination. As (Rothstein, 2004, p.49) says, the added preparatory process is not determined lexically but by contextual factors and general world knowledge (there is more than one way to reach the top).

(5) Harry was reaching the top, when he slipped and fell to the bottom.

Aspectual shifts can be cascaded, as in:

(6) It took John two days to play the Minute Waltz in less than sixty seconds for more than an hour.

According to Moens & Steedman (p.21), *play the Minute Waltz* in (2) is a process, while the *in*-adverbial requires a culminated process. So the process needs to be coerced to a culminated process by adding the completion of the playing to the event. However, I think that *play the Minute Waltz* is a culminated process, just like *eat a cake* or *run a mile*, so no coercion is needed here. Moens & Steedman do not explain this point, but it is important to note that the resulting event, *play the Minute Waltz in less than 60 seconds*, is no longer a culminated process, as can be seen by the fact that it cannot be *easily* modified further by a *for*-adverbial (contrast this with a simple culminated process: *Mary ate a cake / was eating a cake for three minutes*). It is in fact a point.

The next modifier is *for more than an hour*, which requires a process as its input. So the point event needs to be shifted to a process by means of iteration, just as for *hiccuping* above. The resulting process is: repeatedly playing the entire waltz in less than 60 seconds each time. The result *play the Minute Waltz in less than sixty seconds for more than an hour* is again not a culminated process (since it cannot be readily modified by a *for* adverbial) but a point.

Finally, the modifier *it took . . . two days* needs to modify a culminated process. This is achieved by first shifting the point to a culmination, i.e. the achievement of becoming able to do the difficult feat of repeatedly playing the Minute Waltz in less than sixty seconds for more than an hour.

The representations given by Moens & Steedment (e.g. (3)) are informal because the definitions of the operators in terms of input, output, and conditions of use were not precisely given. In particular, it seems that *point* and *process* are not really operators in (3) but are given in the representations just to indicate the aspectual class at their point of use. My aim below is to work out some of these definitions and conditions in more detail.

3 Representation of Events

3.1 Obstacle

3.1.1 Accessing Events

What event representations should we use so that they would allow us to define what the operators and VP-modifiers do on these representations?

As an example, consider the progressive operator PROG. What this operator adds to the description of an event depends on the aspectual class of the event. Thus, "John was eating the cake" differs from "John ate the cake" in that the former has a strong implication for an incomplete event. However, in "John was hiccuping", the progressive adds an iteration. What kind of syntax-semantics interface would allow us to define this behavior of PROG?

An obstacle arises if we try to use conventional tools. If we use the representations as given in (7) and (8), then how can we define the PROG operator for the progressive?

(7) a. John eat the cake
    b. $\lambda e.pred(e) = cake \wedge agent(e) = john \wedge theme(e) = the(cake)$

(8) a. John hiccup
b. $\lambda e.pred(e) = hiccup \wedge experiencer(e) = john$

On (7), PROG would need to apply (9)a to get (9)b, whereas on (8), PROG would need to apply (10)a to get (10)b ($iteration(P)$ denotes the type of event that consists of a sequence of several event instances in the denotation of P).

(9) a. $\lambda P \lambda e.P(e) \wedge complete(e) = -$
b. $\lambda e.pred(e) = cake \wedge agent(e) = john \wedge theme(e) = the(cake) \wedge complete(e) = -$

(10) a. $\lambda P.iteration(P)$
b. $iteration(\lambda e.pred(e) = hiccup \wedge experiencer(e) = john)$

The problem here is that in order to know which of (9)a or (10)a to apply, PROG needs access to the aspectual class of the events described by (7)b and (8)b. We could add a description of the aspectual class to the events, e.g. $aspclass(e) = accomplishment$, but how could PROG access that? As far as PROG is concerned, its input property is a "black box" that PROG cannot look into. Should PROG apply its input property on some dummy variable e' just to see if the result entails $aspclass(e') = process$ or $aspclass(e') = point$ etc.? This is very inconvenient. What we *really* want is for PROG to get as its input a *structured object* that has all the information it needs readily available.

There are two further complications. To determine the aspectual class of a basic predication, an operator needs access not only to the predicate and its arguments but also to the arguments' types and grammatical features. The first information is "buried" and inaccessible in any argument that comes from a quantified NP,[1] and the grammatical information is not even available in the semantic representation. The second complication is explained next.

3.1.2 Complex Events

Simple events can be described by a constellation of role-values standing in one relation over a certain interval of time. But we need to consider complex occurrences as well.

One kind of complex occurrence is brought about by coordinated events:

(11) a. For five minutes, John hugged Mary and Mary kissed John.
b. John hugged Mary and Mary kissed John. It lasted five minutes.

What happened for five minutes was not just the hugging or the kissing separately, but the complex occurrence, to which *it* is referring in (11)b.

Another kind of complex occurrence is brought about by quantifying over entities that participate in some role in the predicate:

(12) a. Every man gave a present to the teacher.
b. *It* happened yesterday.

[1]This will become more clear in section 4.1.

What does *it* refer to? Not to any particular giving of a present to the teacher by one man, but to the entire occurrence of all the givings.

A third kind of complex occurrence comes about when an event is repeated:

(13) John was hiccupping for 10 minutes.

The following representation is *not* appropriate for this sentence:

(14) $\exists e.hiccup(e) \wedge experiencer(e) = john \wedge length(time(e)) = 10minutes$

This says that John made one hiccup, and this event lasted 10 minutes. But in fact, (13) says that John was repeatedly hiccupping and that occurrence lasted 10 minutes.

The obstacle that this poses to the definition of operators that modify events is that it is now even less clear how they would determine the event's aspectual class by using standard tools. For example, if we were to represent (12)a as:

(15) $every(man, \lambda x.a(present, \lambda y.\exists e.give(e) \wedge agent(e, x) \wedge theme(e, y) \wedge recipient(e, the(teacher))))$

then how could a modifier such as *in less than 30 seconds* be able to determine whether the aspectual class of (15) is a culminated process?

3.2 A Solution

To overcome these obstacles, I propose to represent events in the syntax-semantics interface by using structured objects of two kinds. One kind will be called *event description* (*ed*) and the other *base event description* (*bd*). But first, a short interlude about time representation.

3.2.1 Time Points and Intervals

Our domain includes time points and intervals, where an interval is a contiguous set of time points. We use intuitive relations between them, such as:

- $t \in I$ is true iff time point t is included in time interval I.
- $I_1 < I_2$ is true iff all time points in I_1 are strictly before all time points of I_2.
- $during(I_1, I_2)$ is true iff time interval I_1 is included in time interval I_2.
- $meets(I_1, I_2)$ is true iff the latest time point of I_1 is the same as the earliest time point of I_2.

For other relations, see e.g. (Allen, 1991).

3.2.2 Event Descriptions

An **event description** is a structured object with the fields: *base*, *time*, *complete*, and *aspclass*. The first field can only take as its value a base event description (defined below). The second field takes a time interval (or point), the third field takes + or −, and the fourth field takes one of the five names of the aspectual classes. What an event description represents is that the base event occurred over the interval of time, and was completed or not depending on the value of the *completed* field. An event description may have further adjunct thematic roles such as *location* (whereas argument thematic roles appear inside a base event description below).

A **base event description** comes in different sub kinds:

1. Simple (s): This consists of a *pred* field with the name of a basic predicate, and then one field per thematic role that is relevant to *pred*, such as *agent*, *theme*, etc. Example: $[\mathit{pred} : \mathit{eat},\ \mathit{agent} : \mathit{john},\ \mathit{theme} : \mathit{the}(\mathit{cake})]$.

2. Proposition (p): A basic proposition has the form $\exists e.e \in T$, where T is of type *ed*. This says that an event e exists, which can be described as an instance of event-description T.
Example: $\exists e.e \in [\mathit{base} : [\mathit{pred} : \mathit{jump},\ \mathit{agent} : \mathit{john}],\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}]$.
This says that there is a complete event where John jumps, but gives no information about the time. Information on the time can be added by e.g.:
$\exists I \exists e.e \in [\mathit{base} : [\mathit{pred} : \mathit{jump},\ \mathit{agent} : \mathit{john}],\ \mathit{complete} : +,\ \mathit{time} : I] \wedge (I < \mathit{now})$
or simply:
$\exists e.e \in [\mathit{base} : [\mathit{pred} : \mathit{jump},\ \mathit{agent} : \mathit{john}],\ \mathit{complete} : +] \wedge (\mathit{time}(e) < \mathit{now})$

3. Quantified: This is a subtype of proposition. The form of this basic event description is $Q(R, S)$, where Q is a quantifier name, and R, S are properties of type $e \to p$ (individuals to propositions). For example, (part of) the representation for "Every man arrived" is:
$\mathit{every}(\lambda x.\mathit{kind}(x) = \mathit{man},$
$\quad \lambda y.\exists e.e \in [\mathit{base} : [\mathit{pred} : \mathit{arrive},\ \mathit{agent} : y],\ \mathit{complete} : +] \wedge (\mathit{time}(e) < \mathit{now}))$

4. Iteration: This has the form $\mathit{iteration}(T)$, where T is of type *ed*. For example:
$\mathit{iteration}([\mathit{base} : [\mathit{pred} : \mathit{hiccup},\ \mathit{experiencer} : \mathit{john}],\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}])$.

5. Coordination: Two or more event descriptions can be coordinated, and the same goes for basic event descriptions. For example:
$[\mathit{pred} : \mathit{hug},\ \mathit{agent} : \mathit{john},\ \mathit{patient} : \mathit{mary}] \wedge [\mathit{pred} : \mathit{kiss},\ \mathit{agent} : \mathit{mary},\ \mathit{patient} : \mathit{john}]$
A related but distinct operator is $\supset\!\subset$: the notation $T_1 \supset\!\subset T_2$, where T_1 and T_2 are event descriptions, denotes a complex event description, where the event consists of a T_1 occurring, and then a T_2 occurring immediately after it.

3.3 Shorthand and Inference

The feature-structure notation introduced above can be seen as a shorthand for spelling out in a big conjunction the values of all the features:

(16) $e \in [\mathit{base} : T,\ \mathit{complete} : c,\ \mathit{aspclass} : a] \Leftrightarrow$
$\qquad \mathit{base}(e) = T \wedge \mathit{complete}(e) = c \wedge \mathit{aspclass}(e) = a$
$\mathit{base}(e) = [\mathit{pred} : p,\ r_1 : v_1,\ \ldots\ r_n : v_n] \Leftrightarrow \mathit{pred}(e) = p \wedge r_1(e) = v_1 \wedge \ldots \wedge r_n(e) = v_n$

Thus, just as with the standard event representation, we automatically get inferences which drop argument and adjunct information:

(17) John buttered the toast with the knife.
$\exists e.e \in [base : [pred : butter,\ agent : john,\ theme : the(toast),$
$instrument : the(knife)],$
$complete = +] \wedge time(e) < now$
implies: John buttered the toast.
$\Rightarrow\ \exists e.e \in [base : [pred : butter,\ agent : john,\ theme : the(toast)],\ complete = +]$

All the representations introduces above *denote* structured objects rather than functions, propositions, or truth values. Thus, the operators can traverse these structures and have full access to all the information in them. These structures correspond to what (Krifka, 1998, p.198) means by *conceptual structures*:

> [We can] assume that expressions are interpreted by elements of conceptual structures that in turn are related to real entities by some extralinguistic matching. [The structures are] attempts to capture certain properties of the way how we see the world, not as attempts to describe the world how it is.

4 Analysis of Cases

Let us see how this apparatus can be put to use. It is not a-priori clear in which order the quantifiers, PROG operator, and VP-modifiers should apply. Let us examine 18 cases that arise from considering three parameters: progressive vs. non-progressive; no modifiers vs. *in* vs. *for* adverbials; and aspectual class (process, accomplishment, achievement). Actually, we will consider an additional extra case of progressive + no-modifiers + point.

Note: In the examples below, such as (18), the classification to the three classes – *process*, *accomplishment*, and *achievement* – refers to the basic event (predicate + arguments), and not to the whole sentence.

4.1 Progressive, No Modifiers

(18) process: John was running.
accomplishment: John was eating two cakes.
achievement: John was reaching the top.

In all of them, the (preparatory) process occurred.
For process: incomplete, for others: strictly speaking unknown, with implication of incomplete.

4.1.1 Process

The initial representation for the first sentence is:

(19) $\exists e.e \in PROG([base : [pred : run,\ agent : john],\ complete : -,\ aspclass : process])$
$\wedge\ time(e) < now$

Because the input to PROG here is a process, PROG needs to do nothing:

(20) If $Q = [\mathit{base} : T,\ \mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}]$ then $PROG(Q) = Q$.

and the result is the same as (19) except with PROG dropped. The event description itself is obtained from the syntactic structure of the sentence by combining the predicate *run* with its arguments. The values of the *complete* and *aspclass* fields is determined by consulting a table, as will be explained in section 4.1.3. The part $I < \mathit{now}$ is contributed by the past tense operator. The prefix '$\exists e.e \in$' is a standard "wrapping" that is always added to a sentence after everything else has been calculated.

4.1.2 Achievement

The initial representation for the achievement sentence is:

(21) $\exists e.e \in PROG([\mathit{base} : [\mathit{pred} : \mathit{reach},\ \mathit{agent} : \mathit{john},\ \mathit{target} : \mathit{the}(\mathit{top})],$
$\mathit{complete} = +,\ \mathit{aspclass} : \mathit{achvmnt}]),$
$\wedge\, \mathit{time}(e) < \mathit{now}$

The PROG operator requires its input to be an event description with $\mathit{complete} : -$ and $\mathit{aspclass} : \mathit{process}$. If this condition is met, as it was for (19), PROG does nothing. If the condition is not met, PROG creates a new process event description whose details depend on the aspectual class of the input and on contextual factors. In the case of an achievement input, as here, one option is for PROG to create a new event description of the preparatory process of the achievement. This is done by using the *PreP* operator. So (21) expands into:

(22) $\exists e.e \in \mathit{PreP}([\mathit{base} : [\mathit{pred} : \mathit{reach},\ \mathit{agent} : \mathit{john},\ \mathit{target} : \mathit{the}(\mathit{top})],$
$\mathit{complete} = +,\ \mathit{aspclass} : \mathit{achvmnt}]),$
$\wedge\, \mathit{time}(e) < \mathit{now}$

The *PreP* operator takes its input event description and returns a new event description of an incomplete process:

(23) $\mathit{PreP}([\mathit{base} : T,\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{achvmnt}]) =$
$[\mathit{base} : T',\ \mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}]$
where T' depends on the content of T and on contextual factors.

The new event description T' should include a condition on the length of the event's time interval so that it is contextually close enough to the achievement point ("John was reaching the top" means he was about to reach the top, not just that he did some climbing towards the top). The operator *PreP* can easily access the content of T thanks to the fact that all the values here are structured objects rather than properties whose internals cannot be accessed directly. For example, *PreP* can access the *base.pred* value of its input. The final representation is then:

(24) $\exists e.e \in [\mathit{base} : T',\ \mathit{complete} = -,\ \mathit{aspclass} : \mathit{process}] \wedge \mathit{time}(e) < \mathit{now}$
where T' is a preparatory process of:
$[\mathit{base} : [\mathit{pred} : \mathit{reach},\ \mathit{agent} : \mathit{john},\ \mathit{target} : \mathit{the}(\mathit{top})],$
$\mathit{complete} = +,\ \mathit{aspclass} : \mathit{achvmnt}]$

4.1.3 Accomplishment

What happens with the accomplishment sentence is slightly different. The initial representation is

(25) $\exists e.e \in [\mathit{base} : \mathit{two}(\lambda y.\mathit{cake}(y),$
$\lambda x.\exists e'.e' \in \mathit{PROG}([\mathit{base} : [\mathit{pred} : \mathit{eat},\ \mathit{agent} : \mathit{john},\ \mathit{target} : x],$
$\mathit{complete} : +,\ \mathit{aspclass} : \mathit{acmplsh}]))]$
$\wedge\ \mathit{time}(e) < \mathit{now}$

This claims the existence of an event e which is described by a quantification statement. Again, as for the achievement, PROG notices that its input it not a process, and so one option is for it to call *PreP*. Now that operator is defined differently for accomplishment than for achievement:

(26) $\mathit{PreP}([\mathit{base} : T,\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{acmplshmnt}]) =$
$[\mathit{base} : T,\ \mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}]$

Since an accomplishment inherently has a process as part of it, the preparatory process of an accomplishment is described by using the same predicate-argument constellation but marking the event as incomplete (more on this in section 5). If we apply (26) on the representation, we get:

(27) $\exists e.e \in [\mathit{base} : \mathit{two}(\lambda y.\mathit{cake}(y),$
$\lambda x.\exists e'.e' \in [\mathit{base} : [\mathit{pred} : \mathit{eat},\ \mathit{agent} : \mathit{john},\ \mathit{target} : x],$
$\mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}])]$
$\wedge\ \mathit{time}(e) < \mathit{now}$

Note that the inner event description has $\mathit{complete} : -$, but the outer event has no information on completion. Thus, this says: there are two cakes and for each, John was eating it and did not complete the eating, but we don't know the completion status of the entire complex event.

One question to explain a bit more is: Where does the information '$\mathit{aspclass} : \mathit{acmplsh}$' that appears in (25) come from? Determining this depends on the type of the arguments, because if the object NP in the sentence were *cake* or *cakes* rather than *two cakes*, the aspectual class would be a process. The problem is that the object argument here is a variable x, which has no type or other information. This is the issue that was mentioned at the end of section 3.1.1. The answer is that the syntax-semantics interface has access to the entire syntactic structure, such as an F-structure in LFG (Dalrymple, 2001). That structure has enough information about the type of the arguments, which allows us to compute the aspectual class.

Just to illustrate this point a bit more concretely,[2] if we use LFG as our syntactic framework, we can specify the following rule to compute the aspectual class:

(28) If an f-structure G originates from a verb,
and the PRED of G is p,
and G has grammatical functions $f_1, \ldots, f_n$,
and these functions lead to the f-structures $H_1, \ldots, H_n$, respectively,
then the aspectual class of G is ASPTABLE$(p, \{f_1 : H_1, \ldots, f_n : H_n\})$.

[2]This paragraph is for illustration purposes only, and need not be taken as accurate.

ASPTABLE has a column for the basic predicate and one column for each grammatical function. In each row, an empty column indicates the grammatical function must not exist while a '_' mark indicates that the grammatical function must exist but we do not care what it contains:

PRED	SUBJ	OBJ	...	aspect class	example
love	_	_		state	John loves Mary.
jump	_			point	John jumped.
arrive	_			achievement	John arrived.
reach	_	NUM=sg		achievement	John reached the top.
eat	_			process	John ate.
eat	_	NUM=sg		accomplishment	John ate a cake.
eat	_	SPEC=generic		process	John ate cakes.
eat	_	COUNT=uncountable		process	John ate cake.

Now that we know how to compute the aspectual class, that information need not be placed in the f-structure itself, as it may be irrelevant for syntax. Instead, it can be placed only in the semantic structure that is *projected* off from the f-structure, e.g. via Glue Semantics (see (Dalrymple, 2001)).

4.1.4 Point

(29) point: John was jumping.

The event description of the basic predication is

(30) $[\mathit{base} : [\mathit{pred} : \mathit{jump},\ \mathit{agent} : \mathit{john}],\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}]$

Since PROG requires a process, there is a clash. In the case of a point event, the clash is resolved by iterating the point:

(31) $$\begin{aligned}&\mathit{PROG}([\mathit{base} : T,\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}]) = \\ &\quad [\mathit{base} : \mathit{iteration}([\mathit{base} : T,\ \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}]), \\ &\quad \mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}]\end{aligned}$$

Thus we get:

(32) $$\begin{aligned}&\exists e.e \in [\mathit{base} : \mathit{iteration}([\mathit{base} : [\mathit{pred} : \mathit{jump},\ \mathit{agent} : \mathit{john}], \\ &\qquad\qquad \mathit{complete} : +,\ \mathit{aspclass} : \mathit{point}]), \\ &\quad \mathit{complete} : -,\ \mathit{aspclass} : \mathit{process}] \wedge \mathit{time}(e) < \mathit{now}\end{aligned}$$

This section was particularly long because we introduced a lot of machinery. The analysis of the subsequent cases will therefore be much shorter.

From now on, to ease readability, we will omit field names if they can be uniquely understood from the field value, e.g. $[\mathit{run}, \mathit{john}]$ instead of $[\mathit{pred} : \mathit{run},\ \mathit{agent} : \mathit{john}]$.

4.2 Progressive, *for two minutes*

(33) process: John was running for two minutes.
accomplishment: John was eating two cakes for two minutes.
achievement: John was reaching the top for two minutes.

(Preparatory) process occurred for two minutes.
completion = strictly speaking unknown, with implication of incomplete.

The phrase *for two minutes* expects an event description T that has $complete : -$, and it outputs: $\exists e.e \in T \wedge length(time(T)) = 2minutes$.

Since a process event description has $complete : -$, it can be easily modified by *for two minutes*:

(34) $\exists e.e \in [base : [pred : run,\ agent : john],\ complete : -,\ aspclass : process]$
$\wedge\, length(time(e)) = 2minutes \wedge time(e) < now$

For the achievement case: because *for two minutes* expects a process event description, it can combine with an achievement only after the achievement is shifted to its preparatory process. That's why it can modify only the outer event of (24), and not the inner T' there. We get:

(35) $\exists e.e \in [base : T',\ complete = -,\ aspclass : process]),$
$\wedge\, length(time(w)) = 2minutes \wedge time(e) < now$
where T' is a preparatory process of:
$[base : [pred : reach,\ agent : john,\ target : the(top)],$
$complete = +,\ aspclass : achvmnt]$

As for the accomplishment case, there are two places in (27) where *for two minutes* can apply, corresponding to the scope ambiguity between it and *two cakes.* If it applies in the inner event, we get that for each of the two cakes, John was eating it for two minutes, whereas if it applies in the outer event, we get that for two minutes, John was eating the two cakes.

4.3 Non-Progressive, No Modifiers

(36) process: John swam.
accomplishment: John ate a cake.
achievement: John reached the top.

(Preparatory) process occurred.
For process, $complete = -$, unless there is some contextually salient amount of running that got completed, in which case this is actually an accomplishment.
For achievement, $complete = +$:

(37) # John reached the top, but he didn't go all the way to the top.

For accomplishment, this is sometimes less strong:

(38) a. # John ran a mile yesterday, but he didn't run the whole mile.
b. (?) John ate a cake yesterday, but he didn't eat the whole cake.
c. John read a book yesterday, but he didn't read the whole book.

Also, sometimes an accomplishment that prefers $complete = +$ when it is not followed by a VP-modifier changes its preference to $complete = -$ when it is modified by a *for*-adverbial:

(39) a. Dafna drew a circle. (preference: complete)
b. Dafna drew a circle for one minute. (preference: incomplete)

I suggest that, depending on the particular predicate and argument types, some accomplishments will have $complete = +$ and some will have this only as a defeasible fact, prefixed with a $\eth$, i.e. $\eth(complete(e) = +)$. The important thing about the logic of $\eth$ is the following pair of inference rules:

(40) $\Gamma \vdash \eth\varphi \quad ; \quad \Gamma \nvdash \neg\varphi \quad \Rightarrow \quad \Gamma \vdash \varphi$
$\Gamma \vdash \eth\varphi \quad ; \quad \Gamma \vdash \neg\varphi \quad \Rightarrow \quad \Gamma \nvdash \varphi$

In other words, if we can defeasibly conclude φ from Γ, and we cannot conclude $\neg\varphi$, then we can conclude φ ; but if we can conclude $\neg\varphi$, then this defeats $\eth\varphi$, and we cannot conclude φ.

As a possible benefit, this can simplify the analysis of an accomplishment with a progressive in (25)-(27) as follows. The PROG operator can simply *assert* that $complete : -$. For a process, this is consistent with the given event description. For an achievement or a point, this contradicts their $complete : +$ and so $PreP$ or $iteration$ are triggered. But for an accomplishment that has $\eth(complete : +)$, the PROG simply asserts $complete : -$ and overrides this defeasible fact, effectively shifting the accomplishment to its preparatory process without resorting to $PreP$.

Note that we get the desired inferences for the so-called "imperfective paradox" (Dowty, 1979; Rothstein, 2004, p.38):

(41) a. John was swimming. $\Rightarrow$ John swam.
b. John was building a house. $\not\Rightarrow$ John built a house.

The first two sentences are both processes marked with $complete : -$. In contrast, the third sentence has $complete : -$ (section 4.1), whereas the fourth accomplishment sentence has a (defeasible) $complete : +$ and so is not entailed by the third sentence. This is a simple matter of event description, and there is no need to resort to Dowty's inertia worlds or an intensional analysis (more on this in section 5).

4.4 Non-Progressive, *in two minutes*

(42) process: John swam in two minutes.
accomplishment: John ate the cake in two minutes.
achievement: John reached the top in two minutes.

The modifier *in two minutes* requires an accomplishment event T, and it outputs $\exists e.e \in T \wedge complete(e) = + \wedge length(time(t)) = 2minutes$. So this asserts that an event e of type T occurred and it lasted for 2 minutes. It also asserts that it was complete, and so it changes the status of $complete : +$ to definite if this fact is defeasible inside T. For the accomplishment example here we get:

(43) $\exists e.e \in [base : [eat,\ john,\ the(cake)],\ complete : +,\ aspclass : acmplsh]$
$\wedge\, length(time(e)) = 2minutes \wedge time(e) < now$

The *in two minutes* modifier creates a clash if the event description is a process or an achievement: the first lacks a culmination while the second lacks a preparatory process. We have already seen how an achievement event can be shifted to its preparatory process by using the $PreP$ operator, but here we need both the preparatory process and the culmination. So in the case of an achievement, we have:[3]

(44) If $Q = [base : T,\ complete : +,\ aspclass : achvmnt]$ then
$(in\ two\ minutes(Q)) =$
$\exists e.e \in [base : (PreP(Q) \supset\subset Q),\ complete : +,\ aspclass : acmplsh]$
$\wedge\ length(time(e)) = 2minutes)$

For a process, we have two options. One option is to focus on the end of the process and to add a culmination. The definition of the shift would be dual to (44), except we would use the $CulP$ operator which takes a process event and adds a culmination point. Just like the $PreP$ operator, it depends on the particular predicate and arguments of the event as well as contextual factors. $CulP$ could actually make the shift by claiming there is a hidden *theme* argument for the process that measures it, e.g. "John ran in two hours" could mean "John ran a mile in two hours".

The second option is to focus on the beginning of the process as the point of interest, and consider it a culmination of some preparatory process. So in this case, we would use:

(45) If $Q = [base : T,\ complete : -,\ aspclass : process]$ then one option is:
Let $Q' = [base : [pred : start,\ theme : Q],\ complete : +,\ aspclass : point]$ and
$(in\ two\ minutes(Q)) =$
$\exists e.e \in [base : (PreP(Q') \supset\subset Q'),\ complete : +,\ aspclass : acmplsh]$
$\wedge\ length(time(e)) = 2minutes)$

This second option is highly unlikely for the accomplishment and achievement cases (this would mean: "In two minutes, John started eating the cake, or started doing the process that would lead him to reach the top"), simply because it would require too many coercions that go against the simplest, strongest, and most natural interpretation.

4.5 Progressive, *in two minutes*

(46) process: John was running in two minutes.
accomplishment: John was eating a cake in two minutes.
achievement: John was reaching the top in two minutes.

This is unclear because there is a clash between the adverbial that implies completion and the progressive that implies an ongoing process. So this requires coercion. Consider:

(47) John was playing the Minute Waltz in two minutes.

There are two orders in which PROG and *in two minutes* could be applied:

(48) a. (in two minutes (progressive (John play the Minute Waltz))))
b. (progressive (in two minutes (John play the Minute Waltz))))

[3]The operator $\supset\subset$ was defined in section 3.2.2.

In (48)a, the progressive shifts the accomplishment of playing the Minute Waltz to a process by stripping the culmination. This process is the input of *in two minutes*. As we saw above, this could invoke either a coercion to the preparatory process of starting the process, i.e. (49)a, or adding a culmination, i.e. (49)b. Because the second option could be expressed much more simply by just saying (49)b, we get that (49)a is a more likely interpretation of (47) than (49)b is.

(49) a. It took John two minutes to start playing the Minute Waltz.
b. John played the Minute Waltz in two minutes.

There are also two options for what (48)b could mean. They depend on what we do with the event $E1$="John play the Minute Waltz in two minutes" formalized as:

(50) $\exists e.e \in [base : [eat,\ john,\ the(cake)],\ complete : +,\ aspclass : acmplsh]$
$\wedge\ length(time(e)) = 2minutes$

One option is to consider this event as a point by putting it inside:

(51) $[base : E1,\ complete : +,\ aspclass : point]$

and to iterate it, i.e. apply (31) on (51). So the sentence would mean that John repeatedly performed the feat of playing the Minute Waltz in two minutes each time. Another option is to consider $E1$ as an achievement by putting it inside:

(52) $[base : [pred : become\text{-}possible,\ theme : E1],\ complete : +,\ aspclass : achvmnt]$

and then apply $PreP$ on (52) to get its preparatory process – what John had to do in order to be able to perform the feat of playing the Minute Waltz in two minutes. But because (52) is more complicated than (51), this second option is less likely as an interpretation for (47). Or another explanation is that $E1$ includes an existential quantifier over an event, and $PreP$ does not like to get such a thing in its input because when shifting to a preparatory process, the claim of existence of the culmination event is actually withdrawn. This was not a problem in (22) or (44) because there, $PreP$ got a pure event description (which does not involve an existential claim) as its input.

The order ambiguity between the progressive and the VP-modifier suggests that this ambiguity also exists for the cases with *for two minutes* discussed in section 4.2. Thus, *John was playing the Minute Waltz for two minutes* may mean he was in the process of playing it once and did not finish:
(for two Minutes (progressive (John play Minute Waltz incomplete)))
or he was in the process of repeatedly playing the Waltz, each time for two Minutes:
(progressive (iteration (point (for two Minutes (John play Minute Waltz)))))
But again, the second option is more complex and so is less likely.

4.6 Non-Progressive, *for two minutes*

(53) process: John swam for two minutes.
accomplishment: John ate a cake for two minutes.
achievement: John reached the top for two minutes.

For process, this usually means the same as the progressive.[4]

For achievement, this is unclear, because of the clash between *complete* = + of the achievement and *complete* = − of the VP-modifier. A possible way to understand it is by adding iteration of the "John reach the top" event, which is then modified by the VP-modifier – this is a little similar to the first option that was discussed for (48)b above.

For accomplishment, there is dispute in the literature whether this combination is possible. (De Swart, 1998) puts a '#' before (54)b, saying that it cannot have the same meaning as (54)a, only an iterative meaning.

(54) a. Eve was drawing a circle for three hours.
b. Eve drew a circle for three hours.

But I think she confused herself because of the very long duration of three hours which is unlikely for drawing one circle. Indeed, (Zucchi, 1998) accepts "John baked the cake for five minutes" and (Rothstein, 2004, p.40) accepts "Neta painted a picture for an hour". Also, many examples can be found on the web, such as:

(55) You wouldn't believe it! It was so nice outside yesterday, I just sat on the patio and read a book for three hours!! [5]

In such cases, Rothstein says that there is no implication the event was completed. Since we took care in section 4.3 to make the fact '*complete* : +' defeasible for an accomplishment, the for-adverbial is free to override this fact with *complete* : −, and no coercion is required.

4.7 Comparison to (De Swart, 1998)

(De Swart, 1998) also presents a formalization of aspectual classes and shifts. However, in contrast to the DRS representations that she uses, my representations distinguish between event descriptions and claims of event existence. In particular, embedded events, for example the input to the PROG and $PreP$ operators, are usually descriptions and do not have an event variable. This is an advantage over her representation because we do not want to claim the existence of the embedded event in these cases. The only way that she can apply her operators is on an event variable, and that variable must be existentially quantified.

Also, the definition that de Swart gives for her PROG and aspectual shift operators do nothing more than claim that the aspectual class of the event is shifted. In contrast, I spelled out the details of what these operators do (in (20), (21)-(22), (26), (31)), and in particular, the more complex event descriptions that they wrap around their input events. This further supports some predictions about which possible readings are more or less likely than others based on the complexity of coercion, (see e.g. section 4.5), an issue which she does not discuss.

[4](De Swart, 1998, p.356) says that "Andrew was swimming for three hours" does not entail "Andrew swam for three hours", but I disagree.

[5]http://www.dallasnews.com/sharedcontent/dws/fea/breakroom/columnists/mwixon/stories/102105brhumorme.111f9d917.html

5 Processes that Modify an Object

5.1 Background

Consider the following quote from (Zucchi, 1999):

> Parsons's analysis of the progressive runs into another problem. Consider sentence [(56)] and Parsons's translation [(57)]:
>
> (56) Mary is building a house.
>
> (57) $\exists e \exists I \exists t.[I = now \wedge t \in I \wedge building(e) \wedge Agent(e, Mary) \wedge$
> $\exists e.[house(x) \wedge Theme(e, x)] \wedge Hold(e, t)]$
>
> (58) $\exists x.house(x)$
>
> If [(57)] is true, [(58)] must be true. Thus, by Parsons's analysis, if Mary is building a house, there is a house. But suppose that so far she only built the foundations. Then, she is building a house, but there is no house yet.

Zucchi goes on to quote Parsons: "We should stick to the prediction of the theory: if Mary is building a house, then there is a house. If the building process is interrupted, the house exists, but is unfinished." The same goes for the existence of a circle in "John is drawing a circle" even if he only drew an arc.

Since Zucchi is unsatisfied with this analysis, he proposes instead an intensional analysis, where it is only the intension of a house that stands in the role of *theme* of the building event. Only if the event culminates then an extra "building principle" says that the house actually exists.

I think the problem with this discussion is that Zucchi and Parsons are considering only binary predicates. For Parsons, the house exists throughout the event. For Zucchi, only after the event's culmination (if the event indeed culminates). I think neither view is satisfactory, and what we need to allow is non-binary degrees of truth in the interval $[0, 1]$. With this, the problem disappears (or at least very much reduced): If John was building a house or drawing a circle for a little while, but then stopped before finishing, then indeed something came into existence which we would be willing to describe to some extent, though not totally, as a house or a circle. In this section I aim to formalize this intuition for creation and destruction verbs.

5.2 Low-Level Representation

Let us say that a *time-space function* is a function from time points to (not necessarily contiguous) space regions. We can use them in our representation of individuals in order to handle successfully creation, destruction, and other change events.

It is tempting to take the denotation of "John" to be a time-space function $john$ such that for all time points t, $john(t)$ is the region of space that John occupies. If John does not exist at t then $john(t)$ is the empty region $\emptyset$. And it is tempting to take the denotation of "chair" to be a predicate $chair$ such that for all time-space functions f, $chair(f)$ holds iff for all time points t such that $f(t) \neq \emptyset$, we would be willing to describe the space region $f(t)$ as (being occupied by) a chair. (We may furthermore require f to be smooth, i.e. if t and t' are close to each other then so are $f(t)$ and $f(t')$).

But this does not capture well the change that individuals and objects undergo. If a house is being constructed, at what point would we be willing to describe the space it occupies as holding a house? Surely if the house is finished we would be willing to do so, but what about one hour before it is completely assembled, and one of its parts is disconnected from it and occupies a space region one meter away from it? It would be misleading to say that those regions are completely not a house.

First, let us reify kinds as elements in our domain. Thus our domain will includes the kinds $Chair$, $House$, etc. We will also treat individual people as (singleton) kinds, e.g. $John$. We then use the function *holding* which takes a space region, a time point, and a kind, and returns a number between 0 and 1 that designates to what degree we would be willing to describe the space region at that time as being occupied by an instance of that kind. For example, if s is the region of space that John occupies at a certain moment t then $holding(s, t, John) = 1$. If s is the region of space that is occupied by an unfinished house at time t then $0 < holding(s, t, House) < 1$.

(In this paper, I will not discuss objects that do not have a physical manifestation, such as kinds that have no instances in the real world (e.g. unicorns), or abstract concepts such as love. Instances of the former can still be said to occupy a region of some imaginary space.)

We define the following for convenience:

- $true(K, f, I) \equiv \forall t \in I.[holding(f(t), t, K) = 1]$
  The time-space function f over the entire time interval I gives regions of space that can be fully described as holding an instance of kind K.

- $partial[\uparrow](K, f, I) \equiv$
  $\forall t_1, t_2 \in I.[t_1 < t_2 \rightarrow holding(f(t_1), t_1, K) \leq holding(f(t_2), t_2, K)]$
  The time-space function f over the time interval I returns a sequence of space regions that hold an object of kind K that is gradually coming into existence.

- $partial[\downarrow](K, f, I) \equiv$
  $\forall t_1, t_2 \in I.[t_1 < t_2 \rightarrow holding(f(t_1), t_1, K) \geq holding(f(t_2), t_2, K)]$
  Similar to the above but the object is gradually being destroyed.

An example of how this can be used:

(59) John was sitting on a chair yesterday.
$\exists I \exists f_1 \exists f_2.[true(John, f_1, I) \wedge true(Chair, f_2, I) \wedge I < now \wedge during(I, yesterday)$
$\wedge \exists e.e \in [base : [pred : sit,\ agent : f_1], on : f_2,\ time : I,\ complete : -]$

(60) John was drawing a circle yesterday.
$\exists I \exists f_1 \exists f_2.[true(John, f_1, I) \wedge partial[\uparrow](Circle, f_2, I) \wedge I < now \wedge during(I, yesterday)$
$\wedge \exists e.e \in [base : [pred : draw,\ agent : f_1,\ theme : f_2],\ time : I,\ complete : -]$

(61) John was eating a cake yesterday.
$\exists I \exists f_1 \exists f_2.[true(John, f_1, I) \wedge partial[\downarrow](Cake, f_2, I) \wedge I < now \wedge during(I, yesterday)$
$\wedge \exists e.e \in [base : [pred : draw,\ agent : f_1,\ theme : f_2],\ time : I,\ complete : -]$

What (60) says is: there was a sequence of space regions over the interval I, call it f_2. This sequence of regions over time occupied (part of) a circle to a gradually increasing degree. There is no guarantee that the circle fully came into existence.

In contrast, the representation of the sentence “John drew a circle yesterday” has *complete* : + instead of *complete* : − in the description of the event to indicate is was completed. We can conclude from this that a circle (fully) came into existence after the event by using the following general postulate:

(62) For a creation predicate p,
$partial[\uparrow](K, f, I) \wedge \exists e.e \in [[p,\ theme : f], I, +] \rightarrow holding(f(max(I)), max(I), K)$
For a destruction predicate p,
$partial[\downarrow](K, f, I) \wedge \exists e.e \in [[p,\ theme : f], I, +] \rightarrow f(max(I)) = \emptyset$

This says that if the *theme* of a creation (resp. destruction) event is represented by a time-space function f of kind K, then at the end of the time interval of the event, $max(I)$, the space region of f at that time is occupied by an instance of K (resp. is empty).

This solution allows us to block the derivation “John was drawing a circle” → “John drew a circle” (i.e. completed drawing a circle) while at the same time allowing the possibility that an object came into existence which we would be willing to describe partially as a circle, or as a partial circle.

In this way, we do not need to resort to (Landman, 1992)’s or Zucchi’s intensional analysis. “John is eating the cake” is true in virtue of there being, over an interval of time I, a sequence f of space regions that occupy an object which we would be willing less and less to describe as a cake ($partial[\downarrow](Cake, f, I)$ holds), and it stands in the relation *eat* with John. Saying that such a sequence existed is much more clear than Landman’s analysis that says there is a continuation of stages in possible worlds that lead to John eating the whole cake, even if it turns out that in the real world he did not. The former relies on facts that actually happened in the world while the latter relies on the not well-defined notions of “natural continuation” and “closest world”.

As a side note, we relativized events to time intervals while relativizing objects to time-space functions. Should we do so for events as well? (Cooper, 1985) suggests that we should. For example, an event of two dogs chasing each other in the garden has a location that can be expressed using a time-space function, i.e. the exact locations of the dogs as they go around, which is more specific than the location of the garden. But then what is the location of the event “The astronomers saw the star”?[6] This is another case of an unclear ontological concept: does this event take place in two non-contiguous regions of space (where the astronomers are located and where the star is located) or in some “tunnel” between them as well? Therefore, I localize events only to time and general space regions (e.g. “in the garden”) but not time-space functions. The locations of the two dogs during the chasing event can be inferred from the time-space functions of the dogs themselves, and there is no need to get the event involved in this as well.

[6]This was pointed out to me by Stanley Peters.

5.3 Higher-Level Representation

Above we have completely reduced objects and individuals to the space they occupy at each time point and to the degree to which we would be willing to describe those spaces as being occupied with an object of a certain kind. This doesn't immediately mesh well with the kinds of representations we saw before, specifically *john* and *cake* in (27) and (43).

The solution is to use a higher-level setup and to allow individuals back into our conceptualization, but to equip them with additional properties:

- *total-time*(x) is the interval of time in which x can be said to exist to some degree.
- *creation-time*(x) is the interval of time in which x can be said to come into existence.
- *fully-time*(x) is the interval of time in which x can be said to fully exist.
- *destruction-time*(x) is the interval of time in which x can be said to come out of existence.
- *time-space*(x) is that partial function f with domain *total-time*(x) such that $f(t)$ is the space region that x occupies at time t.

Furthermore, when we write *cake*(y), we do not mean that y is a full instance of a cake throughout its lifetime. Instead, only during *fully-time*(y) we would be completely willing to describe y as a cake, whereas during *creation-time*(y) and *destruction-time*(y), we would be willing to describe *time-space*(y) as a cake being created or destroyed. Thus we have the postulates:

(63) $K(x) \rightarrow \mathit{true}(K, \mathit{time\text{-}space}(x), \mathit{fully\text{-}time}(x))$
$K(x) \rightarrow \mathit{partial}[\uparrow](K, \mathit{time\text{-}space}(x), \mathit{creation\text{-}time}(x))$
$K(x) \rightarrow \mathit{partial}[\downarrow](K, \mathit{time\text{-}space}(x), \mathit{destruction\text{-}time}(x))$

We also have the following postulates:

(64) $\mathit{total\text{-}time}(x) = \mathit{creation\text{-}time}(x) + \mathit{fully\text{-}time}(x) + \mathit{destruction\text{-}time}(x)$
$\mathit{meet}(\mathit{creation\text{-}time}(x), \mathit{fully\text{-}time}(x))$
$\mathit{meet}(\mathit{fully\text{-}time}(x), \mathit{destruction\text{-}time}(x))$
$\forall t \in_{strict} \mathit{total\text{-}time}(x).[\mathit{time\text{-}space}(x)(t) \neq \emptyset]$

The relation $\in_{strict}$ means $\in$ but not equal to the minimum or maximum point.

Above we explained why we should take *John* to be a kind. But we can also re-introduce a constant *john* with the postulate:

(65) $John(x) \;\leftrightarrow\; x = john$

Now (59) becomes:

(66) $\exists I.(I < \mathit{now}) \wedge during(I, yesterday) \wedge \exists x.Chair(x) \wedge$
$\quad \exists e.e \in [base : [pred : sit,\ agent : john],\ on : x,\ time : I,\ complete : -]$

How do we guarantee in (66) that both John and the chair existed during the sitting event, i.e. $during(I, total\text{-}time(x))$ and $during(I, total\text{-}time(john))$? We can have a general rule that says: if a object participates in an event with time interval I then that object exists (at least to some extent) over that time interval:

(67) $e \in [base : [pred : p,\ r : x],\ time : I] \ \rightarrow\ during(I, total\text{-}time(x))$

For many predicate-role pairs, this can be strengthened to using *fully-time*, i.e. the object totally exists over the time interval. For example, the agent of a sitting event totally exists over the entire length of the event's time interval.

In contrast, the theme of a creation event only partially exists over the time interval. For example, now instead of (60) we have:

(68) $\exists I.(I < now) \wedge during(I, yesterday)$
$\qquad \wedge \exists x.Circle(x) \wedge \exists e.e \in [[draw,\ john,\ theme : x], I, -]$

Remember that $\exists x.Circle(x)$ does not necessarily claim the existence of a full circle, only the existence of some conceptual instance x located in $time\text{-}space(x)$ in which there are circle-parts. Only if the accomplishment culminates do we have a full circle. This is guaranteed by the following meaning postulates:

(69) For a creation predicate p,
$e \in [[p,\ theme : x], I, -] \ \rightarrow\ during(I, creation\text{-}time(x))$
$e \in [[p,\ theme : x], I, +] \ \rightarrow\ I = creation\text{-}time(x)$

By following previous definitions (including that of *during*), we can prove:

(70) For a creation predicate p,
$e \in [[p,\ theme : x], I, -] \wedge K(x) \ \rightarrow\ partial[\uparrow](K, space\text{-}time(x), I)$
$e \in [[p,\ theme : x], I, +] \wedge K(x) \ \rightarrow\ true(K, space\text{-}time(x), max(I))$

6 Conclusion

Surely more work remains to be done in analyzing more cases and constructions, including an investigation of the interaction between aspectual classes and the various VP-modifiers and operators. But I think the proposals here provide a flexible and rich framework to work with. I plan to integrate the proposal here with my recent work on developing a lexicon and a syntax-semantics interface in Glue Semantics (as was briefly mentioned in section 4.1).

References

Allen, James F. 1991. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems* 6.

Cooper, Robin. 1985. Aspectual classes in situation semantics. Report No. CSLI-84-14C, CSLI, Stanford University.

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

De Swart, Henriëtte. 1998. Aspect shift and coercion. *Natural Language and Linguistic Theory* 16:347–385.

Dowty, David. 1979. *Word meaning and Montague grammar*. D. Reidel.

Krifka, Manfred. 1998. The origins of telicity. In *Events and grammar*, ed. Susan Rothstein, 197–235. Dordrecht: Kluwer.

Landman, Fred. 1992. The progressive. *Natural Language Semantics* 1:132.

Moens, Marc, and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics* 14:15–28.

Rothstein, Susan. 2004. *Structuring events*. Malden, MA: Blackwell.

Vendler, Zeno. 1967. Verbs and times. *Linguistics and Philosophy* 97–121.

Zucchi, Sandro. 1998. Aspect shift. In *Events and grammar*, ed. Susan Rothstein, 349–370.

Zucchi, Sandro. 1999. Incomplete events, intensionality and imperfective aspect. *Natural Language Semantics* 7:179–215.