Final project:
a Haskell implementation of tense and aspect semantics

Will Merrill

December 2016

1 Introduction

My final project uses a categorial grammar to interpret the semantics of complex tense and aspect constructions in English. For example, it can represent "john will have been laughing" as an expression of predicate logic. The compositional rules which accomplish this are based on the dual reference formalism developed by Hans Reichenbach in *The tenses of verbs* before the development of modern compositional semantics. In this paper, I will briefly summarize Reichenbach's understanding of tense and aspect and then define my own compositional system, which `TenseSemantics.hs` uses.

2 Reichenbach's approach to tense and aspect

Reichenbach suggests that aspectualized tenses can be thought of as relations between three points in time: the time of the event (i), the time of speech (r_0), and the time of reference (r_1).[1] Thus, tense-aspect constructions convey information about when an event is occurring relative to both when the sentence is uttered and some arbitrary reference time.

The purpose of the reference time is to establish a basis of comparison between distinct events. For example, consider the sentence "I had eaten when I arrived". By the pluperfect construction in the first sentence, we know that $i_e < r_1 < r_0$. The simple past in the second clause tells us that $i_a = r_1 < r_0$.

[1] I've formalized the relations from the original paper to deal with intervals as well as discrete points. The next section gives definitions for these relations.

Thus, we can derive easily that $i_e < i_a$. Interestingly, many awkward tense constructions like "I had eaten when I will arrive" produce contradictions under this paradigm.

More general theories have expanded Reichenbach's idea to allow arbitrarily many reference times (e.g. "I will have been about to have eaten"), but the formalism in my project will assume only two, since, practically speaking, that is enough to represent most commonly used English verbal constructions.

3 Formalizing events and times

3.1 Events

In order to capture the chronology of actions, we will need to modify the semantics of verbal predicates. Predicates will no longer map to truth values; instead, they will map to objects called events. An event represents the occurrence of an action without an associated temporal position. For example, $laugh(john)$ encodes the event of John laughing, but it does not say anything about when John is laughing.

3.2 Times

The next step is a formal way of situating events in time. To do this, we will define a time as a set which is a subinterval of the timeline $\mathbb{R}$.

An important property of times is their density, which intuitively corresponds to whether they are extended intervals or single points. Formally, we can define the function D which checks whether or not a time period t is dense:

$$D(t) \iff \int_t dx \neq 0 \tag{1}$$

In addition, we need a way of ordering events. To do this, we will define the temporal relations $<$ and $>$:

$$t_1 < t_2 \iff \forall n \in t_1 \forall m \in t_2 (n < m) \tag{2}$$

$$t_1 > t_2 \iff \forall n \in t_1 \forall m \in t_2 (n > m) \tag{3}$$

Set theoretic relations are useful when talking about times. The subset

relation will allow us to give a denotation for the present tense (the program writes $\subseteq$ as ~). Equality will be useful for talking about aspect.

Modeling Reichenbach, we will define r_0 as the contextually determined time of speech and r_1 as the time of reference.

3.3 Situating events in time

In order to unify events and times, we will define the @ relation:

$$e@t \iff \forall n \in t(e \text{ takes place at } n) \tag{4}$$

Using all of this machinery, we can finally give a formal denotation for the example sentence "john will have been laughing", a statement which intuitively tells us that John is laughing over some extended interval before a future reference time:

$$\exists i\big(laugh(john)@i \wedge (i < r_1) \wedge D(i)\big) \wedge (r_1 > r_0) \tag{5}$$

We see that this denotation captures all three of the possibilities which Reichenbach associates with the future perfect, namely $i < r_0 < r_1$, $r_0 \subseteq i < r_1$, and $r_0 < i < r_1$.

The remainder of the paper will discuss the implementation of a compositional system to generate such denotations.

4 Separating tense and aspect

English utilizes periphrastic constructions to encode tense and aspect. For example, in "John will have laughed", "will" encodes the future tense, and "have" encodes the perfect aspect. Thus, we need a way of building tense-aspect constructions from Reichenbach's paradigm compositionally. To do this, we will define tense and aspect as distinct properties.

4.1 Tense

Unaspectualized tense conveys information about when an action occurs relative to the present. We can define a tense as a lambda expression which relates the reference time to the speech time. We will define the following tenses:

$$past = \lambda r_1.(r_1 < r_0) \tag{6}$$
$$present = \lambda r_1.(r_0 \subseteq r_1) \tag{7}$$
$$future = \lambda r_1.(r_1 > r_0) \tag{8}$$

4.2 Aspect

Based on this definition of tense, our notion of aspect will relate event time to reference time. Formally, aspect relations are lambda expressions which take the following form:

$$perfect = \lambda t.\lambda r_1.(t < r_1) \tag{9}$$
$$simple = \lambda t.\lambda r_1.(t = r_1) \tag{10}$$

We will also have the following aspect-like modifier, which marks an event as occurring over an extended interval:

$$dense = \lambda t.D(t) \tag{11}$$

Denseness will be added to present participles in order to deal with constructions like "John was running".

5 Inflecting verbs

In the categorial grammar defined in `TenseSemantics.hs`, each allomorph of every verb has its own lexical rule. However, to model the morphological relationship between related allomorphs, I defined a lambda term for each verbal root. The meaning of each allomorph in the categorial grammar is written as a function of its root term. For example:

```
Lex "had" ((S :\ NP) :/ Sup) (inflect have simple past)
Lex "saw" ((S :\ NP) :/ NP) (inflect see simple past)
Lex "laughed" (S :\ NP) (inflect laugh simple past)
```

Thus, the denotations of allomorphs are produced from their root meanings systematically. Even in the case of the auxiliary "had", we get the desired pluperfect meaning by inflecting `have` in the same way that we would to get the past tense meaning of normal verbs.

5.1 Root denotations

As an example, the denotation of the root "laugh-" is defined as:

$$[\![\text{laugh}-]\!] = \lambda a.\lambda t.\lambda u.\lambda r_1.\exists i\big(laugh(u) \wedge a(i)(r_1) \wedge t(r_1)\big) \tag{12}$$

6 Temporal conjunctions

Following Reichenbach, we will define the denotations of conjunctions like "before", "while", and "after" as imposing temporal relations between the reference times in conjoined clauses. My implementation does this by creating two new reference times using an `Append` term. For example, consider the following implementation of "before":

```
Lex "before" ((S :/ S) :\ S) (Lambda "s1" $ Lambda "s2" $
  Lambda "r*" $ Conj (Conj (Appl (Var "s1") (Append (Var
  "r*") "_0")) (Appl (Var "s2") (Append (Var "r*") "_1")))
  (Before (Append (Var "r*") "_0") (Append (Var "r*")
  "_1")))
```

Calling the method `reduce` on a lambda term substitutes all `Append` chains with flat-string-named variables. Thus, we can generate a different reference time variable of each of arbitrarily many clauses in a sentence.

7 Implementation notes

I adapted the categorial grammar parser which we used in lecture. In order to make it easy to write entries for inflected verbs, I added three new base syntactic categories to the parser: `Sup`, `Inf`, `Adj`. `Sup` corresponds to a verbal supine, or past participle used in the context of a "have" construction. `Inf` corresponds to the infinitive, and `Adj` gives the present participle.

To represent the semantics of linguistic expressions, I used an extended version of the untyped lambda calculus implemented in `Combinators.hs`. I added `Term` constructors for $<$, $=$, $>$, $\subseteq$, D, and @. In addition, I created the aforementioned `Append` term for encoding the presence of new reference times bound by conjunctions, and wrote the `reduce` function for converting it to single-variable form.

7.1 Interpreting text

The function in `TenseSemantics.hs` for interpreting sentences according to a categorial grammar is `interpret`. Some example calls are as follows:

```
interpret eng "john will have been laughing"
> (E i ((((laugh john) @ i) & (i < r1)) & (D i)) & (r1 > r0))


interpret eng "is"
> (\u'. (((r1 (\s. (\r1'. (s = r1')))) (\r1'. (r0 ~ r1')))
  u'))


interpret eng "john does"
> (((r1 (\s. (\r1'. (s = r1')))) (\r1'. (r0 ~ r1'))) john)


interpret eng "john does laugh"
> (E i (((laugh john) @ i) & (i = r1)) & (r0 ~ r1))


interpret eng "john laughed after mary laughed"
> (((E i (((laugh john) @ i) & (i = r1_0)) & (r1_0 < r0)) &
  (E i (((laugh mary) @ i) & (i = r1_1)) & (r1_1 < r0))) &
  (r1_0 > r1_1))
```

7.2 List of tokens

- john, mary
- have, having, has, had
- be, being, been, is, was
- does, did
- will
- laugh, laughed, laughs, laughed (participle)
- see, seen, seeing, sees, saw
- while, before, after

8 Interesting results

One of the most interesting aspects of this program is that I didn't have to hard-code any information about the pluperfect or future perfect tenses into my grammar; instead they are built compositionally. We get the pluperfect meaning of "had" if we systematically take the past tense of "have". Similarly, we get the future perfect meaning of "will have" just by defining the semantics of "will" and "have" independently.

Another interesting result is that the meaning of "will" is generated by taking the future tense of "do-". At first glance, this might appear troubling since "will" seems to have non-conditional "would" (e.g. "Last year, John didn't know Donald Trump would be president") as a past tense form, which is clearly not the same thing as "does".

However, further analysis reveals that the English future is actually made up of two aspectualized tenses, which Reichenbach calls posterior present and simple future. Both these tenses represent events occurring after the speech time, but they diverge in where they place the reference time:

$$posteriorPresent = \lambda t.\lambda r_1.(r_0 \subseteq r_1 < t) \tag{13}$$

$$simpleFuture = \lambda t.\lambda r_1.(r_0 < r_1 = t) \tag{14}$$

Inflecting "do" with the future gives us a simple future denotation. On the other hand, treating "I will" as equivalent to "I am going to"[2] gives us the posterior present, to which "would" is related. In my program, all future constructions are given simple future meaning.

9 Potential extensions

The most obvious extension of this project would be attempting to implement a tense/aspect system with arbitrarily many reference points. This would allow it to handle odd future constructions like "John will have been going to laugh".

[2] "going to" is not implemented in my program because a proper implementation requires more than two reference times ("I am going to be going to be going to laugh" is syntactically allowable, whereas "I had had had laughed" is not).

10 References

Reichenbach, H. (2005). *The tenses of verbs.* The language of time: A reader, 71-78.