Bidirectional interpretation of tense and aspect

James Thomas and Stephen Pulman*

University of Cambridge Computer Laboratory
New Museums Site
Cambridge CB2 3QG, UK
{James.Thomas,Stephen.Pulman}@cl.cam.ac.uk

Abstract

This paper describes an application of the theory of contextual interpretation described in [Pul94] to the area of English tense and aspect, partly based on the analysis in [Pul97b]. We describe an implemented system which allows a user to ask questions about the location and movements of individuals tracked by the Active Badge system [WHFG92], and which will also run in the reverse direction, using exactly the same linguistic specifications, to generate short narratives describing the same data.

Keywords: Tense, Aspect, Coercion, Bi-directional Analysis

1 Introduction

The benefits of reversible or bidirectional processing in syntax and semantics are well known ([Str94]). Aspects of contextual interpretation like reference or ellipsis resolution have not yet proved amenable to fully bidirectional treatments. In the theory described in [Pul94] a proposal was made as to how to achieve this in some respects at least. The current paper describes an application of this theory to the area of tense and aspect interpretation and generation. The theory has been implemented in an application which interprets and generates sequences of sentences concerned with a simple series of events reported by an 'Active Badge' system (described below) but the basic principles are intended to be application independent. As well as being a test of the theory, our attempt to carry out a program of descriptive analysis in a computationally rigorous and reversible way has led us to a new appreciation of the role of aspectual coercion as an important resource for the speaker in attempting to provide a linguistically coherent text even when the order of sentences does not mirror the order of the events described (thus disobeying the Gricean maxim of 'orderliness').

The structure of the paper is as follows. Firstly, we provide a brief summary of the theory of contextual interpretation we are assuming. Secondly we describe the approach

*also at SRI International Cambridge Computer Science Research Centre to tense and aspect taken here. The third section describes the application and the final section gives some sample output of the system and discusses some asymmetries between interpretation and generation.

2 Bidirectional interpretation

We assume that contextual interpretation and generation can be regarded as the interpretation or production of a 'quasi-logical form' (QLF) appropriate to a context, where a context is represented as a set of (usually resolved) logical forms with some kind of salience ordering defined over them. Quasi logical forms are interpreted by 'Conditional Equivalences', expressions of the form

```
QLF  <==> RLF
 if
  Condition1
  ...
  ConditionN
```

which relate them to (more) 'resolved' logical forms (RLF) provided that various conditions hold (or can be consistently assumed) in the context. The equivalence says: 'if the QLF/RLF will (higher order) unify with the expression on the left/right hand side of the arrow, and the conditions hold in the context, then the interpretation is as given on the right/left hand side of the arrow'.
Conditional equivalences (CE) are described and illustrated more fully in [Pul94, Pul97a]. They are related to the Abductive Equivalential Translation (AET) [Ray93] methods used in the Core Language Engine [Als92]. In other experimental implementations, equivalences have been written for pronoun references, verb phrase ellipsis, focus, quantifier scoping, definite descriptions and some other NP reference phenomena.

The major difficulty in implementation is the use of Higher Order Unification, which is not decidable in the general case. In order to sidestep these difficulties the current implementation uses a subset of higher order matching, defined as follows:

```
ho_match(X,Y):   % X and Y have the same type
  1. REPEAT
       rewrite X and/or Y according to the equivalence
             A=[\j.B](C)
       where C is a subterm of A containing no free variables that are
       not also free in A, and B is A with at least one instance of
       C replaced by j.
     UNTIL
       X and Y are isomorphic in function-arg structure.
  2. UNIFY (more or less) as if X and Y were first-order structures.

where:
 A is a subterm of A
 A is a subterm of B(C) if A is a subterm of B or C
 A is a subterm of \x.B if A is a subterm of B
```

The equivalences themselves are encoded in Rayner's Abductive Equivalential Translation (AET) code [Ray93][1] Equivalences and rules are written in a simple language which is compiled into Prolog and which may be easily interfaced with existing Prolog code.

Translation to and from NL sentences and QLF is accomplished by the Core Language Engine [Als92]. We simplify its QLFs into a simpler format for reasons given by Pulman [Pul94] and also so that the input to our theory could be generated by *any* NLP system. This gives a system overview:

```
               +-----+           +------------+            +----------+
    .-> NL  -> |     | -> QLF -> |  SIMPLIFY  | -> sQLF -> |          |
   /           |     |           +------------+            |   This   |
USER           |     |                                     |          |
   ^           | CLE |                                     |          |
   \           |     |           +------------+            |  theory  |
    .-- NL  <- |     | <- QLF <- | COMPLICATE | <- sQLF <- |          |
               +-----+           +------------+            +----------+
```

3 Tense and Aspect

For our treatment of tense we adopt a version of the theory of 'tense trees' developed in Episodic Logic [HS92, HS94] with some additional default discourse interpretation mechanisms derived from DRT [KR93], p.545). For reasons of space we say no more about this here.

Our aspectual analysis is based on [Pul97b], in turn inspired by Moens and Steedman's well-known account of aspectual coercion [MS88]. Recall that *coercion* is the process by which the interpretation of an event is moved between aspectual classes, acquiring characteristics as it does so. A classic example is (1) in which a single sneeze is *coerced* into a series of sneezes in order to make modification by the temporal adverbial felicitous.

(1) a. John sneezed.

b. John sneezed for ten minutes.

Moens and Steedman [MS88] provide an account of coercion based on a transition network where nodes are aspectual classes and arcs represent coercions. Acceptable inputs are assigned to each aspectual operator—adverbials, progressives and perfectives—and the type of the output is also noted in some cases. (Compare Episodic Logic [HS94] where syntactic grammar rules for adverbials contain (head) features: telicity, boundedness and temporal-extent which are used to constrain application of the adverbials to verbs of the correct aspectual class.) Multiple paths through Moens and Steedman's transition network provide multiple aspectual readings for sentences such as *John swam for 2 hours.* " ... the felicity of any particular transition ... is conditional on support from knowledge and context" (p.18). Arbitrarily complex aspectual forms can be built up merely by traversing the arcs.

[1]Thanks to Ralph Beckett for assistance with this

Moens and Steedman propose five aspectual classes for sentences with their “basic” *point*, *process* and *state* types combining in a tripartite *nucleus* structure to give more complex types. A problem for Moens and Steedman’s formulation is that, although it provides an intuitive understanding of the nature of eventualities, the aspectual representations they use do not make explicit the intended interpretation for complex aspectual types. Specifically, complex types should be defined not only by their temporal ordering and contingency relationships, but also by how their subeventualities differ from basic aspectual types. The process in *John ran to the park*, for example, seems qualitatively different from that in *John sneezed for 10 minutes*. In particular, the former is ‘homogeneous’ in a way that the latter is not. Such features (in this case, *telicity*) play a critical role in constraining the interpretation of aspectual modifiers.

We will thus adopt an aspectual representation that enriches the notion of internal structure with features applying to basic and complex types. There is an implicit *temporal* relation of abuttal between the eventualities in a complex expression. We use telicity and persistence features as in Episodic Logic:

- *telicity*: An eventuality is *telic* if it is moving toward some *natural final point*[2]. Progress toward the expected culmination can be tracked along some trajectory (as described, for instance, by Langacker [Lan82]) over time. An eventuality is *atelic* (or *homogeneous*) if no such trajectory over time can be described.

- *persistence*: An eventuality is *persistent* if it can obtain without the exterior application of resources, *non-persistent* otherwise, although we note that the notion of persistence is one in which context has a role.

This gives us a taxonomy of eventualities like:

class	representation
state	state{-t,+p}:Verb
process	process{-t,-p}:Verb
point	pt{-t,-p}:Verb
accomplishment	[process{+t,-p}:Verb, state{-t,P}:have(Verb)]
achievement	[pt{+t,-p}:Verb, state{-t,P}:have(Verb)]

States are atelic and by default persistent, although non-persistent states are possible, e.g. the consequence of *John put up the tent for the summer*. Processes are non-persistent, since they require some consumable resource and will eventually terminate when that resource runs out. By default they are atelic, but in the context of a more complex aspectual type are often telic. Points are not usually subject to judgements of telicity or persistence but here we give them a default atelicity and non-persistence.[3] Accomplishments and

[2]Admittedly somewhat vague, this definition is an attempt to side-step the problems involved in defining telicity in terms of goals and then being unable to claim that, say, *the storm destroyed the city* is telic. The term *natural final point* comes from Smith [Smi91]

[3]In fact, this is not so far-fetched as it might appear since some authors, e.g. Passonneau [Pas88], suggest that point and process constitute a continuum.

achievements are analysed as complex, with (unless subject to coercion) a consequent state represented by 'have' (mnemonically suggested by the perfect auxiliary). The persistence of a *consequent* state is determined by context, or knowledge of expected duration, represented above by a variable, `P`.

We further complicate our quasi-logical forms by adding a modifier list to give *Body-Modifiers*, where *Body* is of one of the forms already described and *Modifiers* is a set containing any predications of the entire eventuality that are added by aspectual modification, such as duration `dur` of an eventuality or temporal location `loc`. An example is given in (2) where the list order is not significant in the interpretation and is certainly not intended to reflect any scope order.

(2) John slept for 10 minutes on Sunday.
`proc:sleep-[dur(10),loc(sunday)]`

Having determined our representation we turn to the task of formalising Moens and Steedman's coercions. We took Moens and Steedman's transition network as a starting point, wrote a short computer program to generate interpretations based on the coercions available and examined them for incorrect or missing readings and inconsistencies. Our strategy was to create an efficient set whilst still accounting for as many readings as possible and to this end we added, revised and deleted coercions until we ended up with the operators below:

name	type
focus pp	acc/ach -> proc/pt
focus cs	acc/ach -> state
iterate	pt -> process
stretch	pt -> process
collapse	X -> point
try	ach -> acc
add-consq	pt/proc -> ach/acc

We have described these coercions as conditional equivalences with conditions on them to restrict application. This allows us to cater for dependency on factors such as context, world knowledge, lexical preferences, discourse conditions, aspectual features and aspectual class, by calls to context through specialised conditions for resolving particular questions. Of course, these conditions necessarily over-simplify.

The *collapse* operation applies mostly as a precondition to *iteration*, which then applies only to points and produces an atelic process consisting of iterated points. The *stretch* operator temporally elongates points into periods during which the punctual event is in progress. The *add-consq* operation coerces its input into an accomplishment; this will often involve the assumption that there is some relevance to the consequent state. *try* coerces a point into a telic process containing a set of events which lead up to the point. The two *focus* operations return either the point/process or consequent state from one of our complex eventualities.

To illustrate how these coercions are used we will examine the combination of an eventuality and a for-adverbial. We take the event analysis and any temporal modifiers as input and will use coercion and adverbial operators to obtain an interpretation as output. For example, (3a) is represented by the event and adverbial in (3b) and output as (3c).

(3) a. John sneezed for ten minutes.
b. `pt:sneeze(j), for(10, minutes)`
c. `proc:iter(n, pt:sneeze(j))-[dur(10,min)]`

This says roughly that a combination of a point description and a for-10-mins adverbial can be interpreted as denoting a process constituted by some number of iterations of those points within a duration of 10 minutes.

Pulman [Pul97b] represents the potential coercions as different interpretations of a QLF construct 'coerce' so that the input (below, on the left) can be interpreted as the output (below, right) as long as the result of the coercion chain is compatible with the durative adverbial. The coercion chain can, of course, be identity in some cases. Our treatment may conveniently be thought of in the same way. When we talk of coercion, we do not mean that events actually alter—merely that our perception of them alters. We can think of a coercion as mapping between alternative views of an event. A model theoretic semantics and illustrative fragment for this view of coercion was presented in [Pul97b], and the current account could easily be re-cast in these terms, although the actual notation is that given a little later.

```
coerce(pt:sneeze(j)), for(10, minutes)
  <==>
proc:iter(n, (pt:sneeze(j))-[dur(10,min)]
  if  C1....Cn
```

3.1 Some coercions and resolution rules

```
%  Collapse any non-point to a point:
---------------------------------
 Ev{T,P} <=> pt{T,-p}:coll(Ev''{T,-p})
  if
   not_point(Ev,Ev')        % an eventuality Ev can be collapsed as
   non_persistent(Ev',Ev'') % long as it is not already a point. The point
                            % produced retains the telicity of the
                            % original event and must be non-persistent
                            % (perhaps by coercion)
                            % For ach and acc, the non-persistence
                            % applies to the consequent state and
                            % the collapsed point will be atelic.

%  Iterate a Point -> Process
--------------------------
 Ev  <=> proc{-t}:iter(n, Ev')
  if
     point(Ev,Ev')              % either Ev is a point, or is coerced
                                % into one, Ev'. If so, it may be
```

```
                              % coerced into a process by iteration,
                              % i.e. postulating a series of
                              % repeated occurrences of the event.

     iteration_licenced(Ev, Ei) % check that the iteration is
                              % contextually sound.

% interpret for-modification
--------------------------
[for(T,U)| RestOfMods], AF <=> [RestOfMods], AF'-[dur(T,U)]
 if                           % AF = Aspectual Form
  state_or_process(AF, AF')   % an unmeasured, non-persistent,
  non_persistent(AF')         % atelic state or process may be
  unmeasured(AF')             % measured by a for-adverbial
  atelic(AF')                 % T is a number, U is the temporal unit
  T=start(AF') - end(AF')
```

The main function of collapsing to a point is to enable events to be iterated. It may not be obvious that an event needs to be collapsed before it is iterated and, indeed, this is often not explained in other treatments. Here, we regard a point as a kind of island which prevents modification of the eventuality inside it. The output of the iteration is said to be a process since the behaviour of a point under iteration is the same as a lexical process. Recall, however, that we are not suggesting that the physical construction of an iterated process is the same as a lexical process, merely the representation of it in context.

The rule above licences a coercion from point to process. The `iteration_licenced` condition checks that the events are "close enough" together, a concept which will depend on the event, context and knowledge of the expected gap between iterations. The example below shows *collapse* and *iterate* combining on the sentences in (4). In (4a) the most likely reading involves multiple twinkles rather than a single long one. (4b) will admit an iteration reading if the *bake* process is collapsed although could equally be modified by the for-adverbial directly since it is a process. (4c) will also undergo many coercions of varying degrees of plausibility: processes and points are easily iterated since they do not have a consequent state, but some events like this may not be repeated without a lot of extra context.

(4) a. ... she twinkled for many years ... in shows that ranged from "This Year of Grace" and "One Damn Thing After Another" ... (from the LOB corpus, C16 162)

b. Delia baked for the whole afternoon.

c. John left the office for ten minutes.

```
coercion chain                         associated conditions
------------------------------------------------------------------------
'...she twinkled for many years...'
a. pt{-t}:twinkle
-> proc{-t}:iter(n, pt:twinkle)        3 months is a suitable length
```

```
                                          of time for repeated twinklings
                                          to be considered the same event
                                          (based on plays running for 3 months)
b. proc{-t}:bake
-> pt{-t}:coll(proc:bake)
-> proc{-t}:iter(pt:coll(..))
                                          we require separate baking events,
                                          it is not usual to repeatedly
                                          bake  the same item

c. John left the office for ten minutes.

a. [pt:leave, state{-p}:left]
-> state{-p}:left                             focus-cs
-> state{-p}:left-[dur(10,m)]                 measure

b. [pt{+t}:leave, state:left]
-> pt{+t}:leave                               focus-pp
-> proc{-t}:str(pt:leave)                     stretch
-> proc{-t}:str(pt:leave)-[dur(10,m)]         measure

c. [pt{+t}:leave, state:left]
-> [proc{-t}:str(pt:leave), state:left]       stretch,
-> proc{-t}:str(pt:leave)                     focus-pp
-> proc{-t}:str(pt:leave)-[dur(10,m)]         measure

d. [pt{+t}:leave, state:have(leave)]
-> pt{+t}:leave                               focus-pp
-> proc{-t}:iter(pt:leave)                    iterate
-> proc{-t}:iter(pt:leave)-[dur(10)]          measure

e. [pt:leave, state:have(leave)]
-> pt{+t}:([pt, state])                       collapse
-> proc{-t}:iter(pt:([..]))                   iterate
-> proc{-t}:iter(pt:([..]))-[dur(10)]         measure
```

Theoretically, we can think of all possible combinations of coercions as applying in parallel and often multiple plausible paths will be possible. Even an eventuality that provides a naturally compatible interpretation may yield another reading through coercion. [4]

4 The Active Badge System

As Mann [Man87] notes, deciding what to say is in general very difficult. One can side-step it by working in a domain where content can be obtained independently of the linguistic component and this is the approach that we adopt. The domain in which our implementation is placed concerns the movements of staff around the University of Cambridge

[4]For example, *John swam for 2 days* can be either a continuous swim or several shorter swims over a two-day period.

Computer Laboratory. Voluntarily, some staff in the Computer Laboratory, and affiliated research labs like SRI or Olivetti/Oracle, wear an Active Badge [WHFG92, HFG93] which transmits a signal every 15 seconds enabling them, through a network of sensors around the departments, to be located at any time—with obvious exceptions. Usually it is not permitted to log the movement of staff members over a prolonged period of time, but for this project several volunteers allowed their movement data to be collected for a whole day. This will be referred to as the AB data. The domain was chosen for several reasons. It is:

- real data, used daily in the Computer Laboratory
- simple—only (person, location, time) tuples
- a useful approximation of many domains in which a stream of time-dependent events are generated, e.g. newswires
- non-linguistic in origin
- not dependent on a complex knowledge base

One disadvantage is that this type of data has no intentional structure and thus (as one of our reviewers poited out) there will be some components of the interpretation of aspect that we are unable to address.

The set of events which we aim to describe comes virtually direct from the Active Badge system. A few manipulations are necessary to translate from the logging data, which is highly redundant (locating each person every minute), but this is taken care of with some simple Unix shell scripts. This gives us data like the following, where eventualities are given an initial lexically assigned aspectual category:

```
ach([enter,john, t44], 10:01),
state([in,john, t44], 10:01, 10:05),

ach([enter,john, t44], 10:08),
ach([leave,john, t44], 10:11),

state([enter,john, t44], 10:25)
state([in,john, t44], 10:25, 10:35)
```

We expect the user to be able both to query the system and receive natural language responses as in (5) although we do not look at the problem of generating co-operative responses [Gaa92, ART94].

(5) a. User: What did Bill do on Monday afternoon?
    b. System: Bill was in his office for 2 hours from 12:15. At 2:15 he went to the tea room and left the lab at 3:00.

5 Some examples of the system in operation

We now give an example interpretation and generation. First consider (6) which we will feed into our implementation.

(6) a. John entered T45 at 10:05.
  b. Bill was working.
  c. John left T45 for 3 hours.
  d. He returned (to T45) at 1:30.
  e. He had met Mary.

(6a) is an achievement with telic point and non-persistent state. No coercion is required as the at-adverbial applies directly to the punctual event. The corresponding tense tree is generated and the past tense tells us that:

```
before(pt(enter,john,T45):[at(10,5,morning)],now)
```

(6b) is progressive. Our interpretation strategy says that the progressive stativises a process and hence *work* is straightforwardly transformed, again without coercion. As the result is a state it includes the previous event (another default interpretation) giving us the relations:

```
include(state(prog(process(work,bill))),
            pt(enter,john,T45):[at(10,5,morning)])
same-time(start(process(work,bill)),
                 end(state(prog(process(work,bill)))))
same-time(end(process(work,bill)),
                 end(state(prog(process(work,bill)))))
```

There is no explicit statement in the input discourse that Bill is in T45 when John enters there and so our system does not infer that this is the case although it is obviously a real-life implicature in the context.

(6c) licences 26 coercion chains (in this version of the implementation). A selection is given below where the syntax of `NewType:[Coercion, OldType]` should be viewed as the aspectual class of an eventuality, `OldType`, becomes `NewType` after coercion `Coercion`. The initial type in each case is the base achievement from (5b) and the coercion chains can be read off the right hand edges.

```
abut(pt(leave,john,t45),state(consq(pt(leave,john,t45)))-[dur(3,hr)])
before(pt(enter,john,T45):[at(10,5,morning)],pt(leave,john,t45))

1. proc([tel(n)]):[iter,
       pt([tel(n)]):[collapse,
           ach([tel(y)],[temp(y)])]]

2. proc([tel(n)]):[iter,
       pt([tel(n)]):[collapse,
          proc([tel(n)]):[iter,
               pt([tel(n)]):[collapse,
                    ach([tel(y)],[temp(y)])]]]]

3. state([temp(y)]):[focuscs,
       ach([tel(y)],[temp(y)])]


4. state([temp(y)]):[focuscs,
       acc(tel(y),[temp(y)]):[add-consq,
           proc([tel(n)]):[iter,
                 pt([tel(n)]):[collapse,
                      ach([tel(y)],[temp(y)])]]]]]
```

We choose (3) as the most likely interpretation because it has the shortest chain, only a single coercion. (It is not difficult to imagine more sophisticated strategies for disambiguation but most elude implementation.) In this instance it is the correct solution focussing on the temporary consequent state. The number of possible solutions is restricted by constraining the length of the coercion chain. The chosen event is added to the temporal context.

Once the pronoun in (6d) has been resolved, the aspect and tense tree processing is straightforward although note that, again, there is no way to connect the *return* event and the state of *being out of the office*. We can infer that the state started sometime before, but not that the end of the state was coincident with, the return to T45. The perfective in the discourse-final sentence (6e) requires an event with a consequent state as input and, as we treat *meet* as a point, this means that we must infer a consequence. The coercion chain

```
state([temp(y)]):[consq,ach([tel(y)],[temp(y)]):[add-consq,pt([tel(y)])]]
```

accomplishes this and the tense tree specifies that the perfective event took place before the discourse-prior event:

```
before(pt(meet,john,mary), pt(return,john,t45):[at(1,30,evening)]
salient(consq-state(meet,john,mary))
abut(pt(meet,john,mary),state(consq(pt(meet,john,mary))))
```

We turn now to the generation direction. Given a set of time-dependent events we will decide which of them to generate from by attempting to reduce the number of logical forms required to describe the same set.

Generation begins with a set of time-stamped events taken from the AB data which give basic *enter* and *leave* occurrences for person, time and room. We want to translate these basic events into something from which we can generate discourses. This will be done by applying conditional equivalences to build *higher-level events*—such as *visit, meet* and *return*—from the *basic* ones. In terms of the traditional distinction, this constitutes our strategic generation stage, and is not our major concern here. Tactical generation consists simply of the reverse of the interpretation approach outlined above. As far as tense is

concerned, we assign tense wrappers based on input event ordering. We assume that our default interpretation relations are also valid for generation, i.e. if we have two events in a succession relation to generate, we can simply generate them discourse-successively. If, however, the events overlap then we must signal this, perhaps by the addition of temporal adverbials to either or both of the events.
For illustration, we will generate versions of the text we have just interpreted. Before we can do this, we must unpack the event description given by resolution:

```
1. ach([enter,john,t45],      10:5)
2. proc([work,bill],          _, _)              start(2) < 1
3. ach([leave,john, t45],    10:30)                end(2) > 1
4. pt([meet, john, mary],        _)                     4 < 5
5. ach([return_to,john,t45], 13:30)
```

to simulate direct AB data, i.e. we swap *return* for *enter*, replace *meet* with the two states which we define comprise a chance meeting (choosing some time before 13:30 and some meeting place which is not T45) and fill in compatible times for those events which were underspecified in the interpretation. This results in the event sequence:

```
 ach([enter,john,t45],            10:5)
 proc([work,bill],            8:0,12:0)
 ach([leave,john,t45],           10:30)
 state([be,john,[in,t46]], 12:46,12:47)
 state([be,mary,[in,t46]], 12:46,12:47)
 ach([enter,john,t45],           13:30)
```

We do not substitute *work* for its underlying state[5] as we do not know Bill's location.

Having the input set, we can now begin the strategic generation process and attempt to reduce the size of the input set whilst retaining informational equivalence. The application of replacement equivalences is non-deterministic, any given equivalence succeeding if its local conditions are satisfied—there are no global constraints. Here are the results of six different executions on the input set:

```
1.                                     2.
pt([enter,john,t45]),[cardinal(2)]     ach([enter,john,t45]),[]
proc([work,bill]),[]                   proc([work,bill]),[]
ach([leave,john,t45]),[]               ach([leave,john,t45]),[]
pt([meet,john,mary]),[]                pt([meet,john,mary]),[]
                                       ach([enter,john,t45]),[]

3.                                     4.
ach([enter,john,t45]),[]               ach([enter,john,t45]),[]
proc([work,bill]),[]                   proc([work,bill]),[]
ach([leave,john,t45]),[]               ach([leave,john,t45]),[]
ach([enter,john,t45]),[]               ach([enter,john,t45]),[]
```

[5]In generation, to give variation, we allow the state of being alone in an office to mean that a person is working.

```
pt([meet,john,mary]),[]                pt([meet,john,mary]),[]

5.                                     6.
pt([enter,john,t45]),[cardinal(2)]     ach([enter,john,t45]),[]
proc([work,bill]),[]                   proc([work,bill]),[]
ach([leave,john,t45]),[]               ach([leave,john,t45]),[]
state([be,john,[in,t46]]),[]           pt([meet,john,mary]),[]
state([be,mary,[in,t46]]),[]           ach([return_to,john,t45]),[]
```

In (1) and (5) a cardinal has been created from the two *enter* events. It is treated as a point and located at the position occupied by the initial *enter*. As we noted earlier, no heed is taken of the possible discourse implications of the cardinal and so its position in the list of events is not of great concern at this time. (1), (2), (3), (4) and (6) all reconstitute *meet* from the states of John and Mary being in T46 while only (6) sees a new *return* event being generated. The reasons for this are that the non-determinism (implemented with random numbers) that governs application of equivalences has not chosen the *return* equivalence and, in the case of (1) and (5), the second *enter* has already been used by a prior application of the cardinal.

For discourses (3), (4) and (5) we have altered the order of the input events slightly so that it reflects the order of the sentences in the original discourse, i.e. so that the meeting occurs later in the list of events than John's return to T45, although still occurring prior in time. This does not affect the substitution process but will be seen to be relevant in the strategic stage when events are added to a tense tree. Finally, note that we have only reduced the number of events very slightly in this example. With larger sets of events, the reduction averages around 50%.

The tactical operation now takes over these sets of events and translates them into a set of logical forms for realisation as sentences. In order for this to occur, the events must be related to each other temporally by the addition of tense, adverbials and so on. This process is driven by the tense tree and a notion of default relationships between discourse-successive events and states. The discourses which result from the six strategic stage outputs above are:

```
1.                                      2.
John entered T45 twice.                 John entered T45 at 10 5 a m.
Bill worked from 8 a m to twelve p m.   Bill worked from 8 a m to twelve p m.
John left T45 at 10 30 a m.             John left T45 at 10 30 a m.
He met Mary at 12 45 p m.               He entered T45 after he met Mary.

3.                                      4.
John entered T45 at 10 5 a m.           John entered T45 at 10 5 a m.
Bill worked from 8 a m to twelve p m.   Bill was working.
He left T45 at 10 30 a m.               John left T45 at 10 30 a m.
He entered T45 at 13 30 p m.            He entered T45 at 13 30 p m.
He had met Mary at 12 45 p m.           He met Mary at 12 45 p m.

5.                                      6.
John entered T45 twice.                 John entered T45 at 10 5 a m.
```

```
Bill worked from 8 a m to twelve p m.   Bill worked from 8 a m to twelve p m.
John left T45 at 10 30 a m.             John left T45 at 10 30 a m.
He was in T46 afterwards.               He returned to T45
Mary was in T46.                              after he met Mary at 12 45 p m.
```

There are two major points of difference between them: First, consider (2) and (4) where the first event is added to an empty tense tree and given an at-adverbial to locate it in time. The next event, `work(bill)`, includes the first (the default relation is succession) and hence something must show that this is the case. In (2) extra adverbials are added and in (4) a progressive is used to locate Bill's working around John's entry. Tense trees are constructed as in the interpretation direction with the exception that no temporal relations need to be inferred.

The other major difference between the generated discourses concerns the final two sentences. Recall that (3), (4) and (5) have a slightly different input event ordering which forces the last two events to be non-successive. In (3) this results in a perfective being generated, in (4) we have extra adverbials specifying the non-default relation[6] and in (5) the problem does not arrive, the second *enter* being subsumed by the cardinal.

It is also interesting to note that in (5), the two states have their respective adverbials because the first state occurs after the event rather than as the default inclusion and the second state occurs in overlap and so requires no extra adverbial.

6 Asymmetries between analysis and generation

We are striving for bidirectionality in our theories of contextual resolution. But there are some obvious asymmetries between the interpretation and generation tasks. To take the most obvious example, the rule for interpreting deictic pronouns like 'I' and 'you' cannot be the same in both directions [PropAtt]. There are some pragmatic asymmetries: consider a discourse in which *Kate* is introduced in sentence 2 , and thereafter only male characters are referred to until sentence 30 when the pronoun *she* is used. Assuming that the whole discourse is part of the same *discourse segment*, e.g. [Sid86, GS86], we would, however much we disliked it, assume that the two terms referred to the same person. However, in the generation direction, we very much would not want to introduce a pronoun such a long way removed from the first time at which the fully-specified form is mentioned. While there is undoubtedly cross-over, the input direction must be prepared to cope with ill-formed or unusual input while the generation should be well-formed and generate easily comprehensible ouput.

In essence, it is the interpretation and generation tasks which are not equivalent. We must be flexible in interpretation in order to get as much information as possible from what data we have, but do not want to licence the same degree of flexibility in generation where it is important to be correct in order that a reader can comprehend the intended message.

[6]The adverbial on the second *enter* was added by the processing for the *meet* event. If there was no contradiction of the default, then the second *enter* would need no adverbial, e.g. (2).

There are more subtle differences between the two directions, however, when the role of aspectual coercions is considered. It becomes apparent that the set of coercions which are treated in the same way in interpretation actually have different functions in generation. Some, such as iteration, reduce the number of event descriptions to generate by combining a series of similar events into a single event description (operating at the strategic stage) whilst the others operate to smooth the construction of a natural discourse by fitting events into their (temporal) context (i.e. the tactical stage). The same split is observed in the adverbials we cover: cardinal and frequency adverbials provide a way to reduce the number of event descriptions to generate whilst *for* and *in* adverbials are more cosmetic, operating to ensure a cohesive discourse. This separation of roles is simple to handle in the implementation. We write equivalences for the coercions based on their semantics and interpretation rules, i.e. an iteration looks for a series of the same events satisfying the `iteration_licenced` condition

Similarly, the progressive, which is treated with the adverbials in interpretation, in generation has a role that enables events which violate the default temporal ordering to be added to a discourse, as in the first two sentences of discourse 4.
Perfectives enable us to stativise events that occurred prior to the last event at the past node: the last two sentences of discourse 3 exemplify this.

When the system makes either strategic or tactical errors the resulting discourses are much less natural and comprehensible:

```
John entered T45 twice.
Bill worked from 8:00 to 12:00.
John was in T44 after he left T45 at 10:30.
Mary was in T44 for a minute.
```

7 Conclusion

We have described the application of a theory of bidirectional contextual interpretation to the domain of aspectual coercion, and illustrated an implemented system within a small but representative application domain. Discussion of some of the components (in particular the treatment of tense, but also many of the details of our analysis of aspect) has been necessarily brief. We also have not had space to describe an experiment to evaluate the accuracy and comprehensibility of the texts generated by the system: the results of this experiment are extremely encouraging and we hope to report them elsewhere. A fuller description of all of these issues, and of the implementation will be available in [Tho99].

References

[AL94] Nicholas Asher and Alex Lascarides. Intentions and information in discourse. In *Proceedings of the ACL*, 1994.

[Als92] Hiyan Alshawi, editor. *The Core Language Engine.* MIT Press, 1992.

[ART94] I. Androutsopoulous, G. D. Ritchie, and P. Thanisch. Natural Language interfaces to databases—an introduction. *Journal of Natural Language Engineering*, 1994.

[BMR94] Harry Bunt, Reinhard Muskens, and Gerrit Rentier, editors. *International Workshop on Computational Semantics.* Tilburg University, Netherlands, 1994.

[Gaa92] Theresa Gaasterland. *Generating Co-operative Answers in Deductive Databases.* PhD thesis, University of Maryland, 1992.

[GLS87] Roger Garside, Geoffrey Leech, and Geoffrey Sampson, editors. *The Computational Analysis of English: a Corpus-based Approach.* Longman, 1987.

[GS86] B.J. Grosz and C.L. Sidner. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[HFG93] Andy Hopper, Andy Harter, Tom Blackie The Active Badge system. Interchi '93, p. 533-534

[HS92] Chung Lee Hwang and Lenhart K. Schubert. Tense trees as the "fine structure" of discourse. In *Proceedings of the ACL*, 1992.

[HS94] Chung Lee Hwang and Lenhart K. Schubert. Interpreting tense, apsect and time adverbials: A compositional, unified approach. In *First International Conference on Temporal Logic*, 1994.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic.* Studies in Linguistics and Philosophy. Kluwer Academic, 1993.

[PropAtt] Hans Kamp. *Prolegomena to a structural theory of belief and other attitudes.* in Propositional Attitudes, ed. C A Anderson and J Owens, CSLI Lecture Notes 20, Stanford University, pages 27–90.

[Lan82] Ronald W. Langacker. Remarks on English aspect. In Paul Hopper, editor, *Tense-Aspect: Between Semantics and Pragmantics*, pages 265–304. John Benjamins Publishing Company, Amsterdam, 1982.

[Man87] William C. Mann. Text generation: The problem of text structure. Technical report, University of Southern California, 1987.

[MS88] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–27, June 1988.

[Pas88] Rebecca J. Passonneau. A computational model of the semantics of tense and aspect. *Computational Linguistics*, 14(2):44–60, 1988.

[Pul94] Stephen G. Pulman. A computational theory of context dependence. In Bunt et al. [BMR94], pages 161–170.

[Pul97a] Stephen G. Pulman. Higher order unification and the Interpretation of Focus. Linguistics and Philosophy, 20, 73-115.

[Pul97b] Stephen G. Pulman. Aspectual shift as type coercion. *Transactions of the Philological Society*, 95(2):279–317, 1997.

[Ray93] Manny Rayner. *Abductive Equivalential Translation and its application to Natural Language Database Interfacing.* PhD thesis, Stockholm University, Sweden, October 1993. Also available as SRI Cambridge Technical Report CRC-052.

[Sid86] Candace L. Sidner. Focusing in the comprehension of definite anaphora. In B.J. Grosz, K. Spark Jones, and B.L. Webber, editors, *Readings in Natural Language Processing*, pages 363–394. Morgan Kaufman, 1986.

[Smi91] Carlotta S. Smith. *The Parameter of Aspect.* Kluwer Academic, 1991.

[Str94] Tomek Strzalkowski, editor. *Reversible Grammar in Natural Language Processing.* Kluwer Academic, 1994.

[Tho99] James Thomas. *Stretching a point.* PhD thesis, University of Cambridge Computer Laboratory, 1999. (forthcoming).

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The Active Badge location system. Technical Report 92-1, Olivetti Research Limited, 1992.