Argument Structure in TimeML

James Pustejovsky, Jessica Littman, Roser Saurí

Computer Science Department, Brandeis University
415 South St., Waltham, MA 02454 USA

{jamesp,jlittman,roser}@cs.brandeis.edu

Abstract. TimeML is a specification language for the annotation of events and temporal expressions in natural language text. In addition, the language introduces three relational tags linking temporal objects and events to one another. These links impose both aspectual and temporal ordering over time objects, as well as mark up subordination contexts introduced by modality, evidentiality, and factivity. Given the richness of this specification, the TimeML working group decided not to include the arguments of events within the language specification itself. Full reasoning and inference over natural language texts clearly requires knowledge of events along with their participants. In this paper, we define the appropriate role of argumenthood within event markup and propose that TimeML should make a basic distinction between arguments that are events and those that are entities. We first review how TimeML treats event arguments in subordinating and aspectual contexts, creating event-event relations between predicate and argument. As it turns out, these constructions cover a large number of the argument types selected for by event predicates. We suggest that TimeML be enriched slightly to include causal predicates, such as *lead to*, since these also involve event-event relations. We propose that all other verbal arguments be ignored by the specification, and any predicate-argument binding of participants to an event should be performed by independent means. In fact, except for the event-denoting arguments handled by the extension to TimeML proposed here, almost full temporal ordering of the events in a text can be computed without argument identification.

Keywords. Temporal annotation, event expressions, argument structure.

1 Introduction

The question to be addressed in this paper is not *whether* arguments should be included in the specification language of TimeML, but *which* arguments should be and *how* they should best be represented. We review the treatment of complex complementation in TimeML, whereby a proposition-denoting or event-denoting expression is linked to the predicate (event) introducing it by an explicit relational tag, the SLINK. This effectively binds these complements as arguments to their governing events. In fact, currently, any event-denoting expression appearing as an argument to a predicate, broadly speaking, is annotated explicitly in a link relation. In this paper, we wish to make the strategy

Dagstuhl Seminar Proceedings 05151
Annotating, Extracting and Reasoning about Time and Events
http://drops.dagstuhl.de/opus/volltexte/2006/449

explicit by which an argument to an event is annotated. We suggest that predicates selecting for situation, proposition, or event types should be part of the explicit annotation of an event. As a result, this requires expanding the specification to include causal predicates such as *lead to* and *induce*. Finally, we suggest the simplest way to incorporate entity-denoting arguments in the specification. While the specification language allows for bindings to entities, it does not require annotation of the entities for well-formed markup.

2 Overview of Current TimeML Specification

The TimeML specification language provides a standard for capturing all temporal information in a natural language text. This includes temporal expressions, events, and the relationships they share. To achieve such an annotation, TimeML uses four main tag types that fall into two categories, those that consume text and those that do not. `TIMEX3`, `SIGNAL`, and `EVENT` fall into the former group. The non-consuming tags are primarily of the `LINK` type with the exception of `MAKEINSTANCE`, a tag which completes the annotation of events. In the subsections that follow, we briefly describe each of these tags.

2.1 Temporal Expressions

TimeML expands on earlier attempts to annotate temporal expressions ([1], [2]), with the introduction of the `TIMEX3` tag. Specifically, `TIMEX3` adds functionality to the TIMEX2 standard [3].

Temporal expressions in TimeML fall into four categories: DATEs, TIMEs, DURATIONs, and SETs. A DATE is any calendar expression such as *July 3* or *February, 2005*. The annotation of such examples includes a `value` attribute that specifies the contents of the expression using the ISO 8601 standard. The example in (1) shows the annotation of a fully specified DATE `TIMEX3`.

(1) a. April 7, 1980
    b.
```
<TIMEX3 tid="t1" type="DATE" value="1980-04-07"
temporalFunction="false">
April 7, 1980
</TIMEX3>
```

April 7, 1980 is a fully specified temporal expression because it includes all of the information needed to give its value. Many temporal expressions are not fully specified and require additional information from other temporal expressions to provide their full `value`. We will say more about the annotation of these expressions shortly, but, for now, notice that the annotation in (1) includes an attribute called `temporalFunction` and that it is set to "false". When a temporal expression requires more information to complete its annotation, this attribute is set to "true" to indicate that a temporal function will be used. For more on this process, refer to the section below on temporal functions.

While the DATE type is used to annotate most calendar expressions, the TIME type is used to capture expressions whose granularity is smaller than one day. Examples of this include *4:20* and *this morning*. Example (2) shows the annotation of a fully specified TIME `TIMEX3`. Notice that for a TIME to be fully specified, it must include date information as well.

(2) a. 10:30am April 7, 1980
b.
```
<TIMEX3 tid="t1" type="TIME" value="1980-04-07T10:30"
temporalFunction="false">
10:30am April 7, 1980
</TIMEX3>
```

Expressions such as *for three months* include a DURATION `TIMEX3`. The `value` attribute of a DURATION again follows the ISO 8601 standard. For example, *three months* receives a `value` of "P3M". Occasionally, a DURATION will appear anchored to another temporal expression. Since TimeML strives to annotate as much temporal information as possible, this information is also included in the annotation of a DURATION with the `beginPoint` and `endPoint` attributes as shown in (3).

(3) a. two weeks from December 17, 2005
b.
```
<TIMEX3 tid="t1" type="DURATION" value="P2W"
beginPoint="t2" endPoint="t3">two weeks</TIMEX3>
```
c.
```
<TIMEX3 tid="t2" type="DATE" value="2005-12-17">
December 17, 2005</TIMEX3>
```
d.
```
<TIMEX3="t3" type="DATE" value="2005-12-31"
temporalFunction="TRUE" anchorTimeID="t1"/>
```

The example in (3a) contains to temporal expressions separated by a signal (see subsection 2.3). The first, *two weeks*, is annotated as a DURATION. The second, *December 17, 2005*, is a fully specified DATE. Every `TIMEX3` annotation includes an identification number. This number is used to relate the temporal expression to other TimeML objects. In this case, the identification value in (3c), "t2", is included in the annotation of *two weeks* as the `beginPoint` of the duration. With this information, the `endPoint` of the duration can be calculated. An additional `TIMEX3` is created to hold its value. This is the `TIMEX3` given in (3d). Since the value of the new `TIMEX3` must be calculated, `temporalFunction` is set to "true" and a temporal anchor is suppled. This new attribute will be explained below.

The final type of `TIMEX3` is used to capture regularly recurring temporal expressions such as *every three days*. This type, SET, uses the attributes `quant` and `freq` to annotated quantifiers in an expression and the frequency of the expression, respectively. An example is given in (4).

(4) a. two days every week
b.
```
<TIMEX3 tid="t1" type="SET" value="P2D" quant="EVERY"
freq="1W">two days every week</TIMEX3>
```

Temporal Functions When a temporal expression is not fully specified, it requires the use of a temporal function to calculate its value. In a manual annotation, the user provides a particular anchor time ID that supplies the missing information. The user then gives the correctly calculated value for the `TIMEX3`. In automatic annotation, a library of temporal functions is used to perform the calculation.

The example in (3d) shows an annotation that uses a temporal function. In this case, the end point of a duration was calculated using the `beginPoint` and `value` of the duration given in (3b). For the new temporal expression in (3d), the `temporalFunction` attribute is set to "TRUE" and the `tid` for the duration is given as the `anchorTimeID`. Finally, the correct `value` is supplied. This same process is used for temporal expressions that are missing information such as *April 7*, which is missing the year, and for relative temporal expressions such as *today*.

2.2 Events

Events that can be anchored or ordered in time are captured with TimeML. Such events are predominantly verbs, but nouns, adjectives, and even some prepositions can also be eventive. The annotation of TimeML events is a two part process. First, they are tagged with the `EVENT` tag. This tag has two attributes: an ID number and an event class. The classification of an event can help determine what relationships that event may participate in. For example, an event classified as REPORTING will be the first element of an evidential `SLINK` (see the subsection on Subordinating Links in section 2.4). There are seven event classes:

- REPORTING: *say, report, tell*
- PERCEPTION: *see, watch, hear*
- ASPECTUAL: *initiate, terminate, continue*
- I_ACTION: *try, investigate, promise*
- I_STATE: *believe, want, worry*
- STATE: *on board, live, seek*
- OCCURRENCE: *land, eruption, arrive*

Several of these classes introduce an event argument and are of particular interest to the work in this paper. The TimeML Annotation Guidelines [4] detail exactly which events fall into which classes.

Instances of Events Besides the classification of an event, natural language documents supply much more information about events that we need to represent in an accurate annotation. In addition to the head of the event that is captured in the text, an event may include further tense and aspect indicators or modifiers that affect its modality or polarity. This information is captured with `MAKEINSTANCE`, a non-consuming timeML tag. Every event in TimeML has at least one instance annotated with this tag. A separate tag is used because

one mention of an event in text can actually refer to multiple instances, as in example (5).

(5) John swims on Monday and Tuesday.

Here, there is one mention of *swim* that is tagged as an OCCURRENCE `EVENT`. TimeML will try to link this event to the temporal expressions also present in the sentence. However, it is clear that the *swim* event that takes place on *Monday* is not the same one that takes place on *Tuesday*. Instead, it is an instance of the event that is anchored to each temporal expression.

Instances of events can also have different tense, aspect, polarity, or modality properties. Again, this information is captured with the `MAKEINSTANCE` tag. Once an event has an instance annotated, that instance is elligible to take part in a `LINK` tag to show what relationship it has with other temporal objects (refer to section 2.4).

2.3 Signals

When temporal objects are related to each other, there is often an additional word present whose function is to specify the nature of that relationship. These words are captured with the `SIGNAL` tag, which has one attribute that provides an identification number. Example (6) shows a typical use of preposition *at* as `SIGNAL`, and a complete annotation of all the temporal objects present.

(6) a. The bus departs at 3:10 pm.

b.
```
The bus
<EVENT eid="e1" class="OCCURRENCE">
departs
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e1" pos="VERB"
tense="PRESENT" aspect="NONE" polarity="POS"/>
<SIGNAL sid="s1">
at
</SIGNAL>
<TIMEX3 tid="t1" type="TIME" value="XXXX-XX-XXT15:10">
3:10pm
</TIMEX3>
```

2.4 Links

TimeML uses three varieties of `LINK` tag to represent relationships among temporal objects. In all cases, the `LINK` tag is non-consuming as there may not be any explicit text to capture or the relationship could be between objects whose locations vary greatly. Each link tag comes with a set of relation types to specify the nature of the relationship. In the following paragraphs, we briefly describe each of these tags: `TLINK`, `ALINK`, and `SLINK`.

Temporal Relationships All temporal relationships are represented with the TLINK tag. TLINK can be used to annotate relationships between times, between events, or between times and events. In this way, TimeML can both anchor and order temporal objects. A signalID can also be used in a TLINK if it helps to define the relationship. The TLINK in example (7) completes the annotation of *The bus departs at 3:10pm.*

(7) <TLINK lid="l1" eventInstanceID="ei1" relatedToTime="t1" signalID="s1" relType="IS_INCLUDED"/>

The possible relType values for a TLINK are based on Allen's thirteen relations [5]. TLINK is also used to assert that two event instances refer to the same event using the IDENTITY relType.

Aspectual Links Events classified as ASPECTUAL introduce an ALINK. The ALINK represents the relationship between an aspectual event and its argument event. This is an example of one way that TimeML already deals with event arguments.

Subordinating Links As mentioned in section 2.2, certain event classes introduce a subordinated event argument. Some examples are verbs like *claim, suggest, promise, offer, avoid, try, delay, think*; nouns like *promise, hope, love, request*; and adjectives such as *ready, eager, able, afraid*. In the following sentences, the events selecting for an argument of situation or proposition type appear in bold face, whereas the corresponding argument is underlined:

(8) a. The Human Rights Committee **regretted** that discrimination against women persisted in practice.
b. Uri Lubrani also **suggested** Israel was **willing** to withdraw from southern Lebanon.
c. Kidnappers **kept** their **promise** to kill a store owner they took hostage.

In TimeML, subordination relations between two events are represented by means of a Subordinating Links (or SLINKs). The SLINK tag is perhaps the best example of the current treatment of arguments in TimeML. Reference to each event is expressed by a pointer to them (through the attributes eventInstanceID and subordinatedEventInstance), and the relation type is conveyed by means of the attribute relType, which captures the type of modality projected in each case onto the event denoted by the subordinated clause. relType can be any of the following types:

1. FACTIVE: When the argument event is entailed or presupposed. Here is an annotated example:[1]

[1] For the sake of simplicity, in this and the following examples we obviate the annotation of MAKEINSTANCE tags.

(9) a. The Human Rights Commitee regretted that discrimination against women persisted in practice.

b.
```
The Human Rights Committee
<EVENT eID="e1" class="I_ACTION">
regretted
</EVENT>
that discrimination against women
<EVENT eID="e2" class="ASPECTUAL">
persisted
</EVENT>
in practice.
<SLINK eventInstanceID="e1" subordinatedEventInstance="e2"
relType="FACTIVE"/>
```

2. COUNTERFACTIVE: When the main predicate presupposes the non-veracity of its argument:

(10) a. A Time magazine reporter avoided jail at the last minute...

b.
```
A Time magazine reporter
<EVENT eID="e1" class="I_ACTION">
avoided
</EVENT>
<EVENT eID="e2" class="STATE">
jail
</EVENT> at the last minute...
<SLINK eventInstanceID="e1" subordinatedEventInstance="e2"
relType="COUNTERFACTIVE"/>
```

3. EVIDENTIAL: Typically introduced by REPORTING or PERCEPTION events, such as *tell, say, report* and *see, hear*, respectively.
4. NEGATIVE_EVIDENTIAL: Introduced by REPORTING and PERCEPTION events conveying negative polarity; e.g., *deny*.
5. MODAL: For annotating events introducing a reference to possible world.

(11) a. Uri Lubrani also suggested Israel was willing to withdraw from southern Lebanon.

b.
```
Uri Lubrani also
<EVENT eID="e1" class="I_ACTION">
suggested
</EVENT>
Israel was
<EVENT eID="e2" class="I_STATE">
willing
</EVENT>
to
<EVENT eID="e3" class="OCCURRENCE">
withdraw
</EVENT>
from southern Lebanon.
<SLINK eventInstanceID="e1" subordinatedEventInstance="e2"
```

```
relType="MODAL"/>
<SLINK eventInstanceID="e2" subordinatedEventInstance="e3"
relType="MODAL"/>
```

The following section goes into the detail of how SLINKs account for some arguments.

3 Events and their Participants

We will assume for our discussion that events can be represented as first order individuals, existentially quantified in a neo-Davidsonian manner where participants to the event are conjoined relations between individuals and the event ([6], [7]). For each event, e, we will identify the participants to this event with a three-place relation, Arg:

(12) λk: intλx: indλe: event$[Arg(k, e, x)]$

Rather than labeling arguments with specific named semantic functions, such as agent, patient, and instrument, we identify the argument by an index, k. The idea is that a post-parsing procedure will identify the appropriate semantic role played by an argument.

Both named entity arguments and event arguments are expressible in this fashion. For example, for the sentence in (13a), the participants are directly identified by their indices 1 and 2, respectively, but not functionally, as *Agent* and *Patient*.

(13) a. John kissed Mary.
b. $\exists e[kiss(e) \wedge Arg(1, e, j) \wedge Arg(2, e, m)]$

Notice that the current TimeML representation of (13a) identifies the event predicate but not its arguments.

(14)
```
John
<EVENT eid="e1" class="OCCURRENCE">
kissed
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e1" pos="VERB"
tense="PAST" aspect="NONE" polarity="POS"/>
Mary.
```

With the addition into TimeML of an Arg-relation, we would be able to identify the entity participants as represented in (13b) above. This should be done cautiously, however, without complicating the specification language or making the annotation task more difficult than it already is. We will take up this issue in Section 5 below.

By design, TimeML treats predicates that select for event arguments differently from those taking named entities. For example, the event-embedding

predicate *see*, in most cases, allows the same simple conjunctive representation over arguments that we saw in (13b), assuming the argument is extensional.[2]

(15) a. John saw Mary fall.
b. $\exists e_1 \exists e_2 [see(e_1) \wedge Arg(1, e_1, j) \wedge Arg(2, e_1, e_2) \wedge fall(e_2) \wedge Arg(1, e_2, m)]$

In the next section, we turn to the question of how to generalize the encoding of an event argument as expressed in TimeML through SLINKs.

3.1 SLINK Encodes Partial Argument Structure

According to the TimeML specification, predicates in natural language that are encoded as introducing SLINKs in fact already identify the embedded complement as an argument to the verb.

For example, the TimeML markup of (16a) explicitly identifies the embedded complement (verb) as a subordinated argument to the event *regret*.

(16) a. John regretted that Sue marrried Bill.
b.
```
John
<EVENT eID="e1" class="I_ACTION">
regretted
</EVENT>
that Sue
<EVENT eID="e2" class="OCCURRENCE">
married
</EVENT>
Bill.
<SLINK eventID="e1" subEventID="e2" relType="FACTIVE"/>
```

As it happens, with a factive predicate such as *regret* we can existentially quantify the event representing the embedded complement of the SLINK predicate. A first-order neo-Davidsonian representation of this sentence would, therefore, look like the following:

(17) $\exists e_1 \exists e_2 [regret(e_1) \wedge Arg(1, e_1, j) \wedge Arg(2, e_1, e_2) \wedge marry(e_2) \wedge Arg(1, e_2, s) \wedge Arg(2, e_2, b)]$

The current TimeML representation of this sentence, however, expressed as a first-order expression, is closer to that shown in (18), since no entity arguments are represented in TimeML.

(18) $\exists e_1 \exists e_2 [regret(e_1) \wedge Arg(2, e_1, e_2) \wedge marry(e_2)]$

For all other modality-introducing predicates, TimeML is generally descriptively adequate in differentiating the modal force of the complement expression. For example, the SLINK predicate *believe* is annotated as (19b) below.

[2] We assume that the typing on the Arg relation can be generalized to allow events as arguments.

(19) a. John believes that Bill went to Japan.
b.
```
Mary
<EVENT eID="e1" class="I_ACTION">
believes
</EVENT>
that Bill
<EVENT eID="e2" class="OCCURRENCE">
went
</EVENT>
to Japan.
<SLINK eventID="e1" subEventID="e2" relType="MODAL"/>
```

The modal subordination introduced by the propositional attitude predicate *believe* is represented by an SLINK with a `relType` value of MODAL. To model this, we will introduce a special first order variable, $\hat{e}$, effectively encoding the modality of the event and the domain of its subordination. On this strategy, a first order expression representing the partial argument structure of (19b) would be that shown in (20).[3]

(20) $\exists e \exists \hat{e}[believe(e) \wedge Arg(2, e, \hat{e}) \wedge go(\hat{e})]$

4 Encoding Causation in TimeML

We move now to the class of predicates introducing causal relations between events explicitly in their lexical semantics. We believe that there should be an explicit representation of this relation in an event ordering markup language such as TimeML, and we provide such a relation here.

The representation of causation between event denoting expressions within the same sentence is common in natural languages. For example, the following sentences express causal (and hence temporal) relations between events, which are largely ignored in TimeML.

(21) a. [The rain]$_{e1}$ caused [the flooding]$_{e2}$.
b. [The rioting]$_{e1}$ led to [curfews]$_{e2}$.
c. [Fifty years of peace]$_{e1}$ brought about [great prosperity]$_{e2}$.

To capture this relation, we introduce a new link type we call CLINK, to express the causal relation between two events.

```
<CLINK>
attributes ::= [lid] [origin] [eventInstanceID] signalID
               subordinatedEventInstance relType
lid ::= ID
{lid ::= LinkID
```

[3] This is somewhat similar to the first order representations in DAML for modal subordination; cf. http://www.daml.org/ontologies/.

```
LinkID ::= l<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
subordinatedEventInstance ::= IDREF
{subordinatedEventInstance ::= EventInstanceID}
signalID ::= IDREF
{signalID ::= SignalID}
relType ::= 'CAUSES'
```

This solution can be adopted for the following verbs, in their causative senses: *cause, stem from, lead to, breed, engender, hatch, induce, occasion, produce, bring about, produce, secure*.

Now, a sentence such as (22a) can be explicitly annotated as involving a causal relation, as follows:

(22) a. The rioting led to curfews on November 22, 2004.

b.
```
The
<EVENT eid="e1" class="OCCURRENCE">
rioting
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e2" tense="NONE"
aspect="NONE"/>
<EVENT eid="e2" class="CAUSE">
led
</EVENT>
<MAKEINSTANCE eiid="ei2" eventID="e2" tense="PAST"
aspect="NONE"/>
to
<EVENT eid="e3" class="OCCURRENCE">
curfews
</EVENT>
<MAKEINSTANCE eiid="ei3" eventID="e2" tense="NONE"
aspect="NONE"/>
on
<TIMEX3 tid="t1" type="DATE value="2004-11-22">
November 22, 2004
</TIMEX3>.

<CLINK eventInstanceID="ei1" relatedToEvent="ei3"
relType="CAUSES" signalID="ei2"/>
<TLINK eventInstanceID="ei3" relatedToTime="t1"
reltype="IS_INCLUDED"/>
```

Note that both the subject and object event expressions are syntactically arguments to the causal predicate. In this case, the Arg relation is not operative since the matrix predicate is itself a realization of a $Cause$ relation directly:

(23) a. The rioting led to curfews.
b. $\exists e_1 \exists e_2 [rioting(e_1) \wedge Cause(e_1, e_2) \wedge curfews(e_2)]$

It is important to note that there are many cases where causation, expressed through explicit causative predicates such as those mentioned above, is not syntactically a relation between two events, but a relation between an individual and an event. Consider the sentences below:

(24) a. [John]$_x$ caused [a fire]$_{e2}$.
b. [The drug]$_x$ induced [a seizure]$_{e2}$.

In such cases of event metonymy ([8], [9]), we will introduce a skolemized event instance, $ei1$, to act as the proxy in the causation relation. Hence, the TimeML for (24a) would be as follows:

(25)
```
John
<MAKEINSTANCE eiid="ei1" eventID="NONE" tense="NONE"
aspect="NONE"/>
<EVENT eid="e2" class="CAUSE">
caused
</EVENT>
<MAKEINSTANCE eiid="ei2" eventID="e2" tense="PAST"
aspect="NONE"/>
a
<EVENT eid="e3" class="OCCURRENCE">
fire
</EVENT>
<MAKEINSTANCE eiid="ei3" eventID="e3" tense="NONE"
aspect="NONE"/>

<CLINK eventInstanceID="ei1" relatedToEvent="ei3"
relType="CAUSES" signalID="ei2"/>
```

The interpretation of *John* as the agent of an event involved in the causation is out of the scope of TimeML; it would be the responsibility of subsequent semantic interpretation to bind the entity *John* to the causing event.

5 Binding Entity Arguments in TimeML

In this section, we propose an extension to the current specification of TimeML to accommodate the treatment of entity arguments. Our goal is to avoid any explicit mention of entities within the TimeML markup. There are two reasons for this move: first, entity arguments are not temporally sensitive text extents, unlike event-denoting predicates and temporal expressions; secondly, we wish to avoid complicating the specification and subsequent annotation task for human or machine tagging. Therefore, our strategy will be to accomplish the argument binding independent of the event tag itself. Currently, the EVENT tag is defined as follows:

```
<Event>
     attributes ::= eid class
     eid ::= ID
     {eid ::= EventID
     EventID ::= e<integer>}
     class ::= 'OCCURRENCE' | 'PERCEPTION' | 'REPORTING'
               'ASPECTUAL' | 'STATE' | 'I_STATE' | 'I_ACTION'
```

On our approach, this need not change. Rather than add an argument list to the event —similar to the SUBCAT list in HPSG [10]— we will treat the binding of particpants to events in a parallel fashion to the treatment of event ordering; by introducing a new linking relation, called ARGLINK. This will encode, in TimeML, the binding accomplished by the Arg relation defined in (12) above.

```
<ARGLINK>
attributes ::= alid [origin] eventInstanceID ArgID
alid ::= ID
{alid ::= ArgLinkID
ArgLinkID ::= al<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
ArgID ::= IDREF
{ArgID ::= EntityID}
```

Now let us see how the two participants in sentence (26),

(26) John kissed Mary.

can be represented, using the ARGLINK tag. Recall that the desired logical form for this sentence is:

(27) $\exists e[kiss(e) \wedge Arg(1, e, j) \wedge Arg(2, e, m)]$

Assuming that the named entities in (26) have been identified and indexed, we can express the bindings shown in (27) as the two ARGLINKs below:

(28)
```
John (ai1)
<EVENT eid="e1" class="OCCURRENCE">
kissed
</EVENT>
<MAKEINSTANCE eiid="ei1" eventID="e1" pos="VERB"
tense="PAST" aspect="NONE" polarity="POS"/>
Mary (ai2).
<ARGLINK alid="al1" eventInstanceID="ei1" ArgID ="ai1"/>
<ARGLINK alid="al2" eventInstanceID="ei1" ArgID ="ai2"/>
```

This allows us to take advantage of entity tagging information from other resources, while binding these values to the events identified and marked up within TimeML

6 Conclusions

In this paper, we discussed the role of arguments in an event annotation specification language. We first described how TimeML handles event arguments in subordinating and aspectual contexts, where SLINKs and ALINKs create event-event relations between a predicate and an event-denoting argument. We proposed that TimeML be enriched slightly to include causal predicates, such as *lead to*, since these also involve event-event relations. Finally, we introduced a linking mechanism that allows entities to be identified with the event they participate in, while not including named entity tagging as part of TimeML.

Acknowledgements

The authors would like to thank the participants of the 2005 Dagstuhl workshop. We would also like to thank the members of the TERQAS and TANGO Working Groups on TimeML for their contribution to the specification language presented here. This work was performed in support of the Northeast Regional Reseach Center (NRRC) which is sponsored by the Advanced Research and Development Activity in Information Technology (ARDA), a U.S. Government entity which sponsors and promotes research of import to the Intelligence Community which includes but is not limited to the CIA, DIA, NSA, NIMA, and NRO. It was also funded in part by the Defense Advanced Research Projects Agency as part of the DAML program under Air Force Research Laboratory contract F30602-00-C-0168.

References

1. Mani, I., Wilson, G.: Robust temporal processing of news. In: Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000), New Brunswick, New Jersey (2000) 69–76
2. Schilder, F., Habel, C.: From Temporal Expressions To Temporal Information: Semantic Tagging Of News Messages. In: ACL-EACL-2001, Toulose, France (2001) 65–72
3. Ferro, L., Mani, I., Sundheim, B., Wilson, G.: Tides temporal annotation guidelines. Technical Report Version 1.0.2, MITRE Technical Report (2001) MTR 01W0000041.
4. Saurí, R., Littman, J., Knippen, R., Gaizauskas, R., Setzer, A., Pustejovsky, J.: TimeML Annotation Guidelines, http://www.timeml.org. (2005)
5. Allen, J.: Towards a general theory of action and time. Artificial Intelligence **23** (1984) 123–154
6. Davidson, D.: The logical form of action sentences. In: The Logic of Decision and Action. (1967)
7. Parsons, T.: Events in the Semantics of English. MIT Press, Cambridge, MA (1990)
8. Pustejovsky, J.: Current issues in computational lexical semantics. In: ACL89. (1989) xvii–xxv
9. Pustejovsky, J.: The Generative Lexicon. MIT Press, Cambridge (1995)
10. Pollard, C., Sag, I.: Head-Driven Phrase Structure Grammar. CSLI, Stanford, CA (1994)