

Closing TLink-Relations (Reasoning with Intervals)

Christian Kissig and Laura Rimell

June 23, 2005

Contents

1	Introduction	1
2	Reasoning with Intervals	2
2.1	Semantics of TLink relations	2
2.2	Inconsistencies and Dominance of TLinks	5
3	Implementation	5
3.1	Extracting TLinks for a Document	6
3.2	Constraint Propagation	6
4	Results	7
4.1	Density results	7
4.2	Inconsistencies	8
5	Conclusions	9

1 Introduction

One of the major standards to annotate textual documents with temporal information is the XML-based markup language TimeML. Along with other constructions it suggests to relate events and times temporally by so called TLinks. TimeML provides 14 classes, SIMULTANEOUS, BEFORE, AFTER, I-BEFORE (immediately before), I-AFTER (immediately after), INCLUDES, IS-INCLUDED, BEGINS, BEGUN-BY, ENDS, ENDED-BY, DURING, IDENTITY, and NONE.

Today, most of the TimeML document stocks, like the TimeBank base, are annotated manually. However, manual annotation is a very tedious task that might not always have an exact solution. Thus human annotation is likely to be error prone. Automatic (statistical) procedures are performing even worse. During our research for the 2nd assignment in this course [2] we could not find a classifier performing better than with a success rate of 0.75.

In our opinion, closing the TLink relation table symmetrically and transitively reveals annotation errors by exposing semantical inconsistencies. We will show how and where those inconsistencies are detected, and how they can be

used to trace annotation mistakes. A closure operation, furthermore, increases the density of relation table by adding newly deduced TLinks. We investigate this effect on the TimeBank document stock, as well.

In this report we will not treat all of those 14 TimeML TLink types, rather we used a smaller set of five basic TLink types matched by an appropriate partitioning of the TimeML TLink types.

2 Reasoning with Intervals

From a temporal viewpoint, events can be exhaustively described by mapping them on intervals in the real line delimited by a starting-point s and an end-point t with $s \leq t$. We have already mentioned in the introduction (section 1) that events can be compared pairwise, which can be expressed in terms of constraints over respective s and t . So, if e_1 happens before e_2 then e_1 ends before e_2 starts, written $t_1 < s_2$.

One can distinguish events from points in time. TimeML does so explicitly by referring to the respective entity as an event or a time (specifically, a TIMEX3). In TimeML, a point in time may actually be a named interval, such as a month or a year, in which case it can be analyzed in terms of a starting-point s and an ending-point t on the real line, like an event. In the literature, such as [1], intervals and times have been treated separately. In our opinion considerations can be simplified significantly if one unifies times and events, treating true points in time (a subset of time expressions in TimeML) as zero-length intervals, $s = t$. Then the inference rules applying to events apply to times as well. In the following we will thus transparently speak of events, even though we refer also to times.

Next, we will introduce a semantics of both TimeML TLinks and basic TLinks in terms of constraints on the intervals, and show how to map the TimeML TLinks to the basic ones. Furthermore, we will build an algebra for those TLinks and show how to close TLink relations algebraically using their semantics.

2.1 Semantics of TLink relations

As already mentioned events are semantically described as intervals $[s, t]$ in the time line. Allen suggested in [1] to temporally relate events by relating their start- and end-points w.r.t. $<, \leq, =, \geq,$ and $>$. It can be easily verified that this characterization is exhaustive. TimeML on the other hand introduced a well-chosen and common set of 14 relation types that can be completely described by Allen's characterization. The following table relates the TLink relations with their semantics in Allen's interval characterization.

TimeML TLink Relation	Semantics
<u>SIMULTANEOUS</u>	$s_1 = s_2 \wedge t_1 = t_2$
<u>BEFORE</u>	$t_1 < s_2$
<u>AFTER</u>	$s_1 > t_2$
<u>I-BEFORE</u>	$t_1 = s_2$
<u>I-AFTER</u>	$s_1 = t_2$
<u>INCLUDES</u>	$s_1 < s_2 \wedge t_1 > t_2$
<u>IS-INCLUDED</u>	$s_1 > s_2 \wedge t_1 < t_2$
<u>BEGINS</u>	$s_1 = s_2$
<u>BEGUN-BY</u>	$s_1 = s_2$
<u>ENDS</u>	$t_1 = t_2$
<u>ENDED-BY</u>	$t_1 = t_2$
<u>DURING</u>	$s_1 > s_2 \wedge t_1 < t_2$
<u>IDENTITY</u>	$s_1 = s_2 \wedge t_1 = t_2$
<u>NONE</u>	

In the assignment we were asked to find a reduction of the 13 TimeML TLink relation types to the five basic types BEFORE, AFTER, INCLUDES, IS-INCLUDED, and SIMULTANEOUS. This reduction should be a partition such that each basic TLink is associated with a set of TimeML TLinks. Furthermore, BEFORE and AFTER, INCLUDES and IS-INCLUDED are constrained to be inverse to each other.

Similar to the definition above, we decided to define the semantics of the five basic relations algebraically, like given in the following table.

Basic TLink Relation	Semantics
SIMULTANEOUS (SIM)	$s_1 = s_2 \wedge t_1 = t_2$
BEFORE (BEF)	$t_1 < s_2$
AFTER (AFT)	$s_1 > t_2$
INCLUDES (INC)	$s_1 < s_2 \wedge t_1 > t_2$
IS-INCLUDED (IS-I)	$s_1 > s_2 \wedge t_1 < t_2$

Then we tried to define the partitioning on the TimeML TLink relation types. Obviously, each of the basic TLink types can be associated the TLink of equal name. Then, there were only eight TimeML TLink types left over. Apparently, IDENTITY semantically coincides with SIMULTANEOUS. The other seven types, however, can not be associated, preseving semantics.

We could have ignored all those seven types, but decided to count I-BEFORE to BEFORE, and I-AFTER to AFTER, because the definitions of I-BEFORE and BEFORE, I-AFTER and AFTER coincide almost. Problems in the closing procedure occur rarely, and we will treat them later in this report.

Another solution would have been to weaken the definition of AFTER and BEFORE approprietly to make I-AFTER and I-BEFORE fit in accordingly. However, it turned out that this leads to counterintuitive results in the closure of the TLink relations. We will return to that issue later.

We decided to ignore the other five TimeML TLinks, as their interpretation is disjunctive with respect to the five basic categories.

The following mapping is the partition we use

SIMULTANEOUS \mapsto SIMULTANEOUS, IDENTITY
 BEFORE \mapsto BEFORE, I-BEFORE
 AFTER \mapsto AFTER, I-AFTER
 INCLUDES \mapsto INCLUDES
 IS-INCLUDED \mapsto IS-INCLUDED

The remaining TimeML TLink types are then dropped, i.e. associated to UNKNOWN, as follows.

UNKNOWN \mapsto BEGINS, BEGUN-BY, ENDS, ENDED-BY, NONE, DURING

If there is no TimeML TLink for an event pair, then the basic TLink for this pair is also considered to be UNKNOWN, which means there is no relation between them or the semantics of the relation is simply not known. Translating the events and relations into a labeled graph, there is also the notion of an inconsistent labeling for some edge (i.e. relation between events). In this case, the relation is considered to be EMPTY.

The reduction then associates a TimeML TLink type with the partition it occurs in. For the converse direction a basic TLink type is mapped to the TimeML TLink type of the same name, and thus the same semantics. This way we do not lose any more information writing the closed relations back into the TimeML documents.

Based upon the previously given semantics we can specify symmetry- and transitivity-like closure. Given for instance that events e_1 and e_2 happen simultaneously, i.e. $s_1 = s_2 \wedge t_1 = t_2$, we can infer for the inverse pair $\langle e_2, e_1 \rangle$ that $s_2 = s_1 \wedge t_2 = t_1$ and thus e_2 and e_1 happen simultaneously as well. Continuing with all of the five basic relations we get :

$\langle e_1, e_2 \rangle$	Semantics	$\langle e_2, e_1 \rangle$
e_1 SIMULTANEOUS e_2	$s_1 = s_2 \wedge t_1 = t_2$	e_2 SIMULTANEOUS e_1
e_1 BEFORE e_2	$t_1 < s_2$	e_2 AFTER e_1
e_1 AFTER e_2	$s_1 > t_2$	e_2 BEFORE e_1
e_1 INCLUDES e_2	$s_1 < s_2 \wedge t_1 > t_2$	e_2 IS-INCLUDED e_1
e_1 IS-INCLUDED e_2	$s_1 > s_2 \wedge t_1 < t_2$	e_2 INCLUDES e_1
e_1 UNKNOWN e_2		e_2 UNKNOWN e_1
e_1 EMPTY e_2		e_2 EMPTY e_1

We can draw similar conclusions for the transitivity-like closure. Take as an example three events e_1, e_2 , and e_3 with e_1 BEFORE e_2 SIMULTANEOUS e_3 . Then we know that $t_1 < s_2$ and $s_2 = s_3$ such that $t_1 < s_3$ and thus " e_1 BEFORE e_3 ". The transitivity table gives for all relations between e_1 and e_2 and all relations between e_2 and e_3 the inferred relation between e_1 and e_3 . For the above five basic relations the transitivity table looks like :

	e_1 SIM e_2	e_1 BEF e_2	e_1 AFT e_2	e_1 INC e_2	e_1 IS-I e_2
e_2 SIM e_3	e_1 SIM e_3	e_1 BEF e_3	e_1 AFT e_3	e_1 INC e_3	e_1 IS-I e_3
e_2 BEF e_3	e_1 BEF e_3	e_1 BEF e_3	e_1 UNK e_3	e_1 UNK e_3	e_1 BEF e_3
e_2 AFT e_3	e_1 AFT e_3	e_1 UNK e_3	e_1 AFT e_3	e_1 UNK e_3	e_1 AFT e_3
e_2 INC e_3	e_1 INC e_3	e_1 BEF e_3	e_1 AFT e_3	e_1 INC e_3	e_1 UNK e_3
e_2 IS-I e_3	e_1 IS-I e_3	e_1 UNK e_3	e_1 UNK e_3	e_1 UNK e_3	e_1 IS-I e_3

If either of the two relations, between e_1 and e_2 , or e_2 and e_3 , is UNKNOWN, no relation can be inferred for e_1 and e_3 . If either of the two relations is EMPTY, i.e. an inconsistency has already been identified, we also cannot infer any relation for e_1 and e_3 .

2.2 Inconsistencies and Dominance of TLinks

Closing the TLink relation table transitively might expose inconsistencies in the annotation. For an example consider e_1 BEFORE e_2 SIMULTANEOUS e_3 , and e_3 BEFORE e_1 . Then closing the relation between e_1 , e_2 , and e_3 transitively gives e_1 BEFORE e_3 . Closing e_3 BEFORE e_1 symmetrically then gives e_1 AFTER e_3 , contradicting the previous result that e_1 BEFORE e_3 .

Since all the relations in the semantics of the basic TLinks are strict we obtain an inconsistency if and only if we deduce for an event pair some relation type that is not the same as the one taken down before. An exception is posed by UNKNOWN which is dominated by the other five relations. So, if the TLink between two events was UNKNOWN and an expressive relation has been inferred then the TLink is overridden without claiming an inconsistency. This corresponds just to adding new knowledge about the event pair.

If an inconsistency is found, the algorithm does not write the result to the relation table. If it updated the TLink an error might be propagated throughout the relation table. This may then cache the actual source of the annotation flaw.

Earlier we mentioned that non-strict relations in the semantics of AFTER and BEFORE may cause problems. These become apparent actually when searching for inconsistencies. Let us suppose BEFORE was defined by the constraint $t_1 \leq s_2$, and let e_1 BEFORE e_2 and e_2 BEFORE e_1 . Then one could deduce that $s_1 = t_1 = s_2 = t_2$ which implies that e_1 SIMULTANEOUS e_2 and e_1 and e_2 are points in time. If we could rely upon the annotation this deduction would have been absolutely valid. But, if one has to expect flaws in the annotation then this rare case might point to a mistake.

For a converse argument, it is apparently unlikely that annotator says e_1 I-BEFORE e_2 and e_2 I-BEFORE e_1 instead of using SIMULTANEOUS. However, if he does, the algorithm points out this pair as critical. The annotator could then still ignore the warning.

3 Implementation

Our program for this assignment needed two major infrastructural ingredients, a parser system that reads TLink relations for TimeBank documents and datastructures for the closure algorithm. Although writing these components from scratch might have lead us to a simpler program, we decided to reuse the framework provided in the program we wrote for the second assignment.

3.1 Extracting TLinks for a Document

For the extraction of TLinks we used parser and handler provided by the SAX2 API. We reimplemented the handler (`TLinkDocumentModelBuilder`) to extract from the `<tlink>` tags a hash table relating pairs of event and time ids with values for the `relType` attribute. (In fact, due to the nature of TimeML coding, event instance ids are used.)

Then, the TLink extractor `TLinkExtractor` (resp. `TLinkExtractor5Classes`) fetches the hash tables and maps the values to the 5 temporal relations, plus UNKNOWN, given in the assignment (EMPTY relations are not stored in the hash table). The result is a TLink hash table (`TLinkHash`) associating pairs of event and time id's with their respective TLink class.

Later this TLink hash table is closed reflexively (`closeReflexively()`) and symmetrically (`closeSymmetrically()`) for normalization, and transitively (`closeTransitively()`) afterwards.

3.2 Constraint Propagation

We performed three types of closure operations on the TLINKs in the Time-Bank corpus: reflexive closure, symmetric closure, and transitive closure.¹ Each of these closure operations was performed on a per-document basis; since the documents were presumed to be about unrelated stories, and there was no seed set of TLINKs across documents, temporal closure of TLINKs across documents would be neither meaningful nor possible. The process of *normalization* is considered to consist of reflexive and symmetric closure, while transitive closure is a separate process.

In reflexive closure, performed by the `closeReflexively()` function in `TLinkHash.java`, a TLINK is created between every event or time and itself, with the relation type SIMULTANEOUS. (Note that in addition to the hash table containing the TLINKs, the program also collects a list of entities, which includes all the times and events in the document, even those that do not participate in TLINKs. Reflexive closure is performed on all of these entities. Of course, such entities play no role in symmetric or transitive closure, which are based on existing TLINKs. But, the total number of entities is used when describing link density.)

In symmetric closure, performed by the `closeSymmetrically()` function in `TLinkHash.java`, TLINKs are created in the reverse order that they appear in the document. For example, if the document contains a link stating that e1 is BEFORE t5, but no link stating the relation between t5 and e1, then a link is added to show that t5 is AFTER e1. A simple lookup provides the reverse relation types.

The method for transitive closure, performed by the `closeTransitively()` function in `TLinkHash.java`, is based on the Floyd-Warshall shortest paths algorithm. It is an $O(n^3)$ algorithm that takes each entity participating in the

¹Note that these three operations together do not constitute complete temporal closure. To obtain the fullest possible temporal closure, we would have had to analyze the time expressions referred to by the TLINKs, in order to isolate any temporal relations between them that had not been identified by the human taggers. For example, in some document there might be no TLINK relating times t1 and t2, but if t1 referred to July 1st, 2003, and t2 referred to September 27th, 2004, then it could be inferred that t1 was BEFORE t2. However, this analysis was not part of the present project.

TLINKs, and creates links between each of its predecessors and each of its successors. The relation type in the new TLINK (equivalent to the edge label in a graph) is looked up in a transitivity table, as described above.

In each case, the actual creation of a new TLINK is done by a function in `TLinkHash.java` called `setAndCheckLink()`. This function abstracts out the process of checking for TLINK consistency before creating a new link. The process proceeds in the following way. The function is passed the entities which are to be linked and the intended value of the new relation between them. If there is no existing link between those entities (in that order), or there is a link but its type is `Unknown`, then the new link can be added (or the `Unknown` value replaced with the new value). However, if there was already a relation between the two entities, further checking must be done. If the new value is `Unknown`, no action is taken, since this simply means that the original information about the relation between the entities was potentially more informative than the new information. If the new value is not `Unknown`, however, but a real value, and it is not the same as the original value, this indicates a conflict, and a message is output. This design for the system means that consistency checking is not a separate stage in the temporal closure process, parallel to the three closure routines. Instead, consistency checking takes place each time the graph is updated.

If the user has requested normalization only (labeled in the program as "symmetric closure" for legacy reasons), the reflexive closure function is called once, followed by symmetric closure. If the user has also requested transitive closure, then reflexive closure is performed, followed by symmetric closure, followed by transitive closure, and then by symmetric closure again. This ensures that the reflexive and symmetric closure feed the transitive closure, but that there is also symmetric closure performed on any new TLINKs added by the transitive closure routine.

4 Results

We evaluated our program against the document stock containing 186 news wire articles from the TimeBank document stock.

4.1 Density results

After the document has been parsed, the TLink extractor created a table relating pairs of events with a TLink connecting them. There are three situations that we distinguished for our measurements. First, the TLinks recorded might be one of the five expressive TLinks. Second, the TLink might be `UNKNOWN` happening when one of those TimeML TLinks were found in the document that could not be associated to any of the basic TLinks. And finally, there might be no TLink at all in the document meaning the pair is marked with `EMPTY`.

It is fairly obvious that there can maximally be as many TLinks as the squared number of entities in the document. Starting from this fact, we set up two density ratios. First, the total ratio, is the number of non-`EMPTY` TLinks (the five basic and `UNKNOWN`) divided by the maximal possible number of TLinks in the document. The second is the expressives density ratio being

the ratio between the number of expressive TLinks in the document and the maximally possible number of TLinks.

We have already mentioned that the closure algorithm increases the density. We investigated this effect with the help of the previously defined ratios. Thereby we distinguished between the change of density within the normalization phase, and the total closure. We measured the change of the density as the ratio between the density after the normalization/closure and the density before.

After the normalization the total density ratio increased by 277.83 % in average, while the expressives density ratio increased by 307.98 % in average. The difference between the density ratio for the five TLinks and the density ratio for all basic TLinks clearly shows that UNKNOWN relations have been replaced by expressive relations. After the transitive closure the total density ratio increased by 305.00 %, while the expressives density ratio increased by 307.99 %. A similar argument for the differences between those two density ratios holds for the transitive closure as well. Furthermore, both density ratios increased from the normalization to the transitive closure after normalization as expected. However, the step from the pre-normalization to the normalization is significantly greater than the step from normalization to transitive closure. The reason may be that chained relations like e_1 BEFORE e_2 BEFORE e_3 are rare in the documents. This may be due to the structure of news articles and due to a sparse annotation of the documents.

4.2 Inconsistencies

The transitive closure seems to be far more vital for consistency checks. We have already showed in Section 4.2 how transitively closing TLinks may expose flaws in the annotation. Our program runs consistency checks during the closure procedure, and outputs a message like below if it found an inconsistency.

```
Inconsistent graph! (ei286,ei286) was already Includes;  
trying to set it to Simultaneous.
```

In the TimeBank document stock we found the four flawed documents ABC19980304.1830.1636.tml.xml, wsj_0325_orig.tml.xml, wsj_0918_orig.tml.xml, and wsj_1011_orig.tml.xml.

Because the affected pair of events is displayed, it is an easy matter to actually trace down the flaw. Taking for example ABC19980304.1830.1636.tml.xml which the following message has been displayed for :

```
Inconsistent graph! (ei286,ei286) was already Includes;  
trying to set it to Simultaneous.  
Inconsistent graph! (ei286,ei286) was already Includes;  
trying to set it to IsIncluded.
```

Looking up the event pair (ei286,ei286) in the document tells that the annotator considered ei286 to include itself, which clearly contradicts the semantics of INCLUDES.

```
<TLINK relatedToEventInstance="ei286" eventInstanceID="ei286"  
relType="INCLUDES" />
```

If the TLink relation is closed reflexively for ei286 then ei286 SIMULTANEOUS ei286 is inferred, disagreeing with INCLUDES. Furthermore, if the relation is closed symmetrically then for the same pair IS-INCLUDED is inferred, incompatible to the existing annotation.

Similarly, does wsj_0325_orig.tml.xml contain the evidential inconsistent line

```
<TLink eventInstanceID="ei199" relType="BEFORE"
relatedToEventInstance="ei199" />
```

and wsj_0918_orig.tml.xml contains

```
<TLink eventInstanceID="ei2049" relType="BEFORE"
relatedToEventInstance="ei2049" />
```

and wsj_1011_orig.tml.xml contains the two contradictory lines

```
<TLink relType="BEFORE" timeID="t57" relatedToEventInstance="ei2048" />
<TLink eventInstanceID="ei2048" relType="BEFORE" relatedToTime="t57" />
```

which have obviously been caught by symmetric closure.

In our opinion, logging the closure procedure which was being undertaken at the time the inconsistency was detected could support an annotator in debugging larger documents. The implementation of this feature, however, lies beyond the scope of this assignment.

5 Conclusions

We have written a program that reads in a TimeML document, extracts TLink information in form of a TLink relation table, simplifies the relations, closes the table reflexively, symmetrically, and transitively. Among statistical values the program outputs the closed TLink relation in form of TimeML compatible TLink tags.

We have seen that even normalizing, i.e. reflexively and symmetrically closing, TLink relation dramatically increases the density of the table, especially of expressive TLinks. Furthermore, we have demonstrated how to find inconsistent annotations in TimeML documents and how to trace them down, supporting the argument for the use of the closure operations.

References

- [1] James Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*. 1983. <http://doi.acm.org/10.1145/182.358434>
- [2] Laura Rimell, Christian Kissig. 2005. Temporal Linking of Events
- [3] Andrea Setzer. Temporal Information in Newswire Articles. An Annotation Scheme and Corpus Study. PhD thesis, University of Sheffield, 2001. <http://www.andrea-setzer.org.uk/PAPERS/thesis.pdf>
- [4] Andrea Setzer and Robert Gaizauskas. A pilot study on annotating temporal relations in text. In *ACL Workshop on Temporal and Spatial Information Processing*, 2001. <http://www.andrea-setzer.org.uk/PAPERS/acl2001.pdf>.
- [5] TimeML specification language. <http://www.timeml.org/timemldocs.html>