Drawing TimeML Relations with T-BOX

Marc Verhagen

Computer Science Department
Brandeis University
Waltham, USA

Abstract. T-BOX is a new way of visualizing the temporal relations in TimeML graphs. Currently, TimeML's temporal relations are usually presented as rows in a table or as directed labeled edges in a graph. I will argue that neither mode of representation scales up nicely when bigger documents are considered and that both make it harder than necessary to get a quick picture of what the temporal structure of a document is. T-BOX is an alternative way of visualizing TimeML graphs that uses left-to-right arrows, box-inclusions and stacking as three distinct ways to visualize precedence, inclusion and simultaneity.

1 Introduction

In the early days of TimeML, the TimeBank corpus was created as an illustration of the temporal annotation proposed by TimeML.[1] The first version of TimeBank was annotated almost exclusively with the Alembic Workbench [2]. Alembic is very useful for annotation of non-relational tags. But it does not deal neatly with highly relational information like the temporal links (TLINKs) of TimeML. In Alembic, TLINKs can be added as rows to a table where the columns denote the events and times that are linked and the relation type of the TLINK (before, after, includes etc). This works fine when an annotator sweeps through the text linearly and creates TLINKs between events and times that are close to each other in the text. It makes it impossible however to get a picture of what the temporal structure of a document is. In addition, annotation of TLINKs proved to be sensitive to certain errors like the direction of the relation.

In 2003, a new tool named Tango [6, 9] was developed in order to make TimeML annotation more intuitive. Tango is a graphical annotation tool that uses a graph to display the various links in a TimeML document. Annotation was expected to be more intuitive because with Tango it involves direct manipulation of a timeline. And indeed, adding a TLINK does not require elaborate manipulation of a table, but proceeds by drawing arrows between events and times that are displayed on a two-dimensional pane, as shown in figure 1. It turned out that Tango made annotation of TLINKs more reliable and that it

[1] See [5] for an overview of TimeML and [3] for a description of TimeBank. TimeML and TimeBank were created in the context of the ARDA workshops TERQAS and TANGO [4, 6].

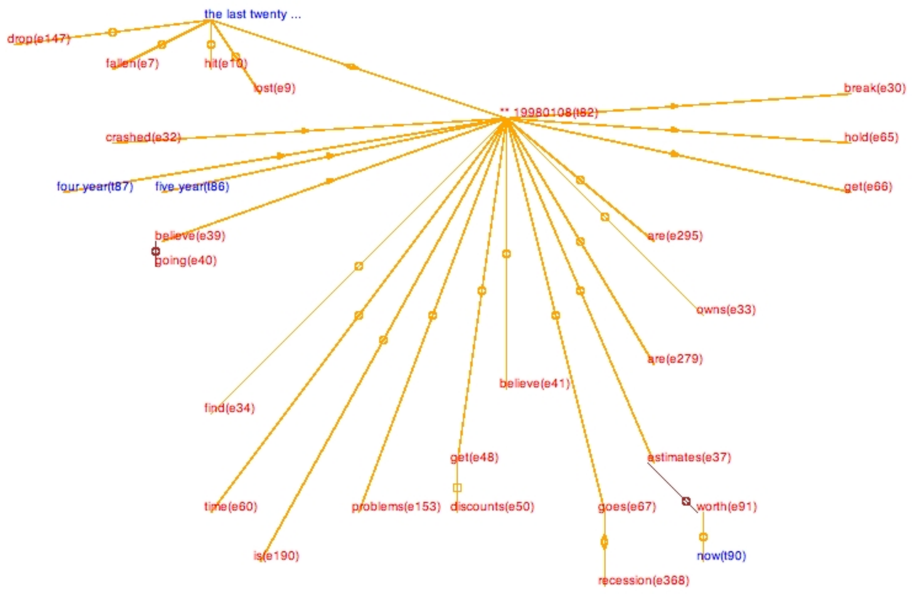


Fig. 1. A fragment from TimeBank, as displayed by Tango

invited the annotator to explore the temporal structure of a document more thoroughly. Annotation with Tango also appears to result in a TimeML graph where the events and times are more tightly connected. There are a couple of limitations though. The main problem is that it is still hard to quickly capture the temporal structure of the document. This is partly due to the fact that the labels of TLINKs are hard to read. But the problems remains even with clearer labels and better spacing. This is because there is no clear semantics associated to the relative positions of events. The annotator has complete freedom to place events and times where she likes them to be. Typically, some kind of left-to-right ordering is adopted but one can not rely on that. Another problem is that the Tango display is simply not that clear when a lot of links are involved. Larger documents can contain hundreds of links and graph clutter makes it hard to see the big picture.

2 Drawing TimeML Relations with T-BOX

The central idea of T-BOX is that relative placement of two events or times is completely determined by the temporal relations between them. Each event or time expression is placed in a box, also called a T-BOX.[2] A T-BOX has a default

[2] Much of this work was inspired by an email from Nick Chubrich, who proposed many ways to improve on the Tango display. One of his ideas was to introduce a

size, but can be stretched as needed. Events and times have the same ontological status. That is, both participate in TLINKS and both are placed in boxes in the TimeML graph. But times are distinguished from events by color-coding them. As mentioned before, T-BOX uses arrows, box inclusion and stacking instead of the labeled edges of Tango. Figure 2 shows the T-BOX representation of the TimeBank fragment that was displayed before in figure 1. Interpretating figure 2 is much easier than interpreting figure 1. And, as a result, T-BOX representations are much more likely to reveal something fishy in an annotation. Figure 2 for example, may suggest that the annotator over-used the simultaneous relation a bit.

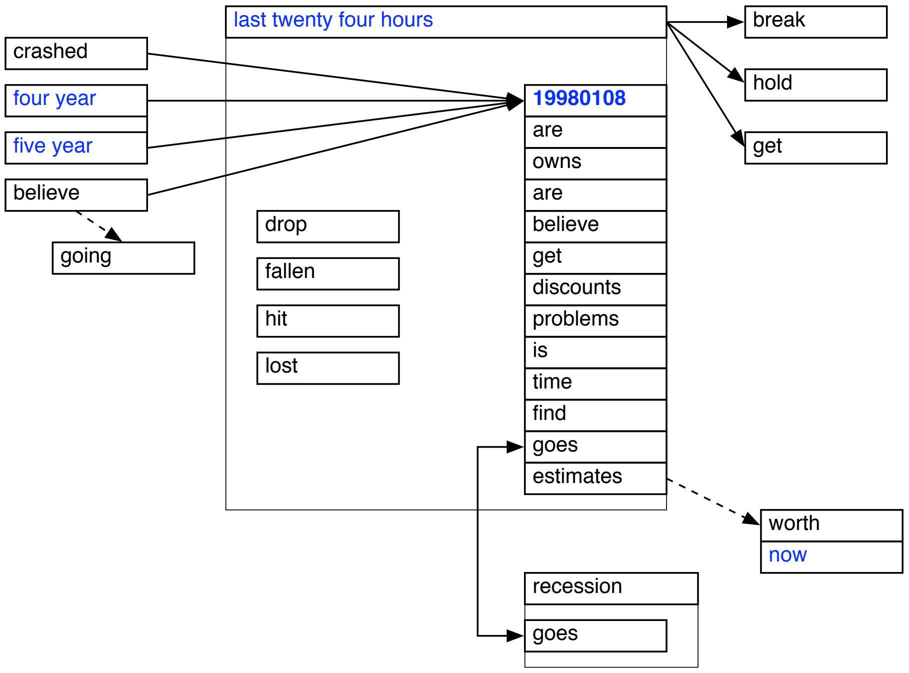


Fig. 2. Same TimeBank fragment, now in T-BOX style

There are four rules that determine placement of two events or times relative to each other:

1. If event X is *before* event Y, then X's box will always be displayed to the left of Y's box and there is a sequence of arrows that leads from one box to the

mechanism that allows annotators to select a whole group of events and use only one link to state that every event in this group stands in a particular temporal relation to another event or timex. The T-BOX derives in a crooked way from this.

other. X and Y are not necessarily displayed at the same vertical position. A variation of this theme is when X is immediately before Y (X ibefore Y). In that case, the arrow is replaced by a line ending in a solid dot. Note that due to the non-transitive nature of the ibefore relation there will never be a sequence of dotted arrows that is longer than one.

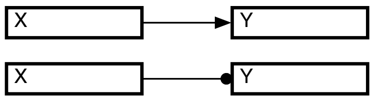
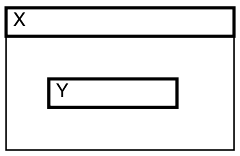
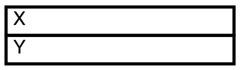


Fig. 3. Rule 1: before

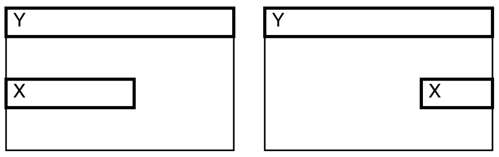
2. If X *includes* Y then the T-BOX of X is extended with a box that has thinner lines. The included event Y is placed inside this box. If needed, the including event X can be stretched so that it has space for Y. Y does not touch any side of the box.



3. If X and Y are *simultaneous* then their boxes will be stacked directly on top of each other or there is a series of boxes between X and Y that are stacked similarly. If X and Y both include events, then these would be placed in a shared extension underneath Y. Simultaneity and identity are displayed differently. If two events are identical, then they will be placed together in the same box.



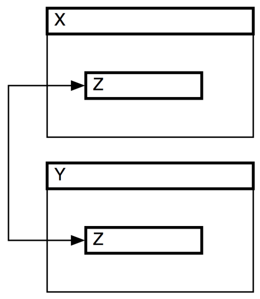
4. If X *begins* Y then X is placed inside Y's extended box and X will hug the left side of Y's box. The case is similar if X ends Y.



These four rules cover all TimeML relations.[3] If no rule governs placement of two events X and Y, then none of the configurations above will occur. X could be above, below, to the right or to the left of Y, but X cannot be inside the extension of Y, nor can there be a sequence of arrows between the two, nor can X and Y be stacked in any way. It cannot be stressed enough that vertical and horizontal placement by themselves don't mean a thing. They only mean something in connection with arrows, box inclusion or stacking. This incidentally means that the timeline metaphor is abandoned. A timeline strongly suggest that vertical placement under a time or date actually means something. For T-BOX, it doesn't.

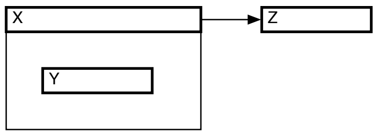
2.1 Massaging the Rules

The four rules above do need some additions. A special case occurs when one event is included in two unrelated events. That is, [X includes Z] and [Y includes Z] and there is no clear TimeML relation between X and Y. The T-BOX way to represent this is to print Z twice and convey with an arrow that the two Z's are the same thing.



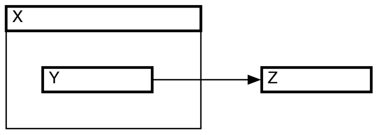
X and Y do not have to line up horizontally, but the two Z's have to line up (and therefore X and Y will at least have some overlap). The internal structure of X will only be displayed on the Z that is embedded in X. There are related special cases for begin and end relations, as well as for certain mixes of includes, begins and ends.

The simple formulation of rule 1 (placement of two events where one is before the other) fails to correctly account for the interplay of inclusion and precedence relations. Take the case where [X includes Y] and [X before Z].



[3] Note however that the rules assume an interval interpretation of TimeML events, similar to the one proposed by James Allen in [1].

We would like to display this case as above, but technically another arrow is needed from Y to Z because the display has not made explicit that Y is before Z because there is no sequence of arrows between the two. So the rule should state that X before Y can also be expressed by X being included in a box that has a chain of arrows to Y. The reversed case is not problematic. If [X includes Y] and [Y before Z] then there should be no arrow from X to Z and the display should not strongly imply that X is before Z (it does strongly imply that X is at least not after Z).



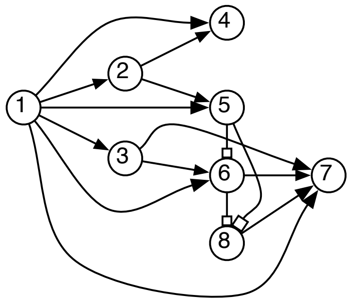
Note that the display rules do not by themselves force a minimal and clear representation of a TimeML graph. What they do is to provide the basic building blocks for an intuitive visualization. The next section presents a procedure that creates a minimal T-BOX drawing from a maximal TimeML annotation.

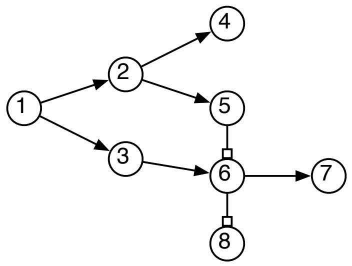
3 A Procedure to Display TimeML Relations

The input to the procedure is a TimeML annotation that is complete, that is, any temporal relation that can be inferred from other relations is expressed by a TLINK. This assumes a constraint propagation algorithm as described in [1, 10] and applied as a temporal closure component for TimeML annotation in [7, 8]. The main ingredients in the T-BOX procedure are graph reduction and a bottom-up process to replace parts of an AVM with T-BOX representations.

3.1 Reducing the Graph

First we map a complete TimeML graph to a unique minimal representation. One characteristic of this minimal graph is that the complete graph can be created simply by running a temporal closure algorithm. It takes three steps to map the graph on the left to the graph on the right. In these graphs, arrows indicate before links and lines that end in open squares indicate `includes` links.





Create equivalence classes The relations `identity`, `simultaneous`, and `during` are all equivalence relations.[4] We can group events and times in equivalence classes and select one event or time to be the class representative. All TLINKs from elements in the equivalence class to elements outside it are deleted except for relations from the class representative. This representative is placed at the top of the box.

Normalize non-equivalence relations It is very likely that a complete TimeML graph contains cycles, yet the display procedure requires an acyclic graph. Cycles can be removed by selecting a set of normalized relations and mapping the inverse relations to elements of the selected set. For example, [X after Y] can be mapped to [Y before X].

Remove derivable relations The closure algorithm in [7, 8] uses a complete set of compiled out composition rules. These rules can be used to delete relations that can be derived. For example, the before arrow between X and Z can be removed given the following composition rule and graph fragment:

[X before Y] $\odot$ [Y before Z] = [X before Z]

$$\begin{tikzcd}
\boxed{X} \arrow[r] \arrow[rr, bend left] & \boxed{Y} \arrow[r] & \boxed{Z}
\end{tikzcd}$$

This reversed closure operation results in a unique minimal graph because all nodes that stand in equivalence relations to each other have been conflated into single nodes.

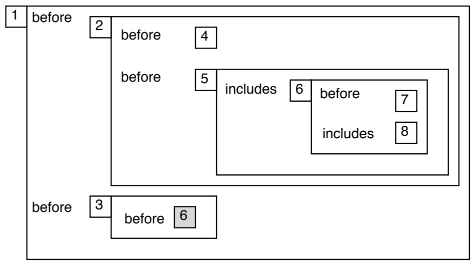
3.2 Creating a Pseudo AVM

The minimal directed acyclic graph from the previous section can be trivially mapped to an AVM with re-entrancies and list values:

$$\begin{bmatrix} \text{node} & 1 \\ \text{before} & \left\langle \begin{bmatrix} \text{node} & 2 \\ \text{before} & \left\langle \begin{bmatrix}\text{node} & 4\end{bmatrix}, \begin{bmatrix} \text{node} & 5 \\ \text{includes} & \boxed{6} \begin{bmatrix} \text{node} & 6 \\ \text{before} & \begin{bmatrix}\text{node} & 7\end{bmatrix} \\ \text{includes} & \begin{bmatrix}\text{node} & 8\end{bmatrix}\end{bmatrix}\end{bmatrix}\right\rangle\end{bmatrix}, \begin{bmatrix}\text{node} & 3 \\ \text{before} & \boxed{6}\end{bmatrix}\right\rangle \end{bmatrix}$$

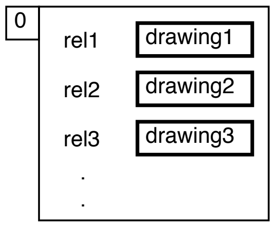
[4] That is, they are all reflexive, symmetric and transitive. This is not quite true though. TimeML's `during` is definitely not symmetric since an event is during a time and not the other way around. Also, TimeML does not stipulate anything about reflexivity. Using equivalence as a notion is valid however because we choose to interpret all TimeML relations as basic Allen relations between intervals. TimeML's `simultaneous`, `identity` and `during` are all mapped to `equals`, which is an equivalence relation.

Let's print this AVM slightly differently. The lists are flattened out by repeating the attribute name, node names are printed as an index, and boxes are drawn for clarity. These are simple mechanical changes but they make the following steps more transparent. The AVM above now looks as follows (and is strictly no AVM any more):



3.3 From AVM to T-BOX

There is a bottom-up step-by-step process for replacing parts of a TimeML AVM with their corresponding T-BOX representations. The mapping from AVM to drawing is governed by a couple of rules.

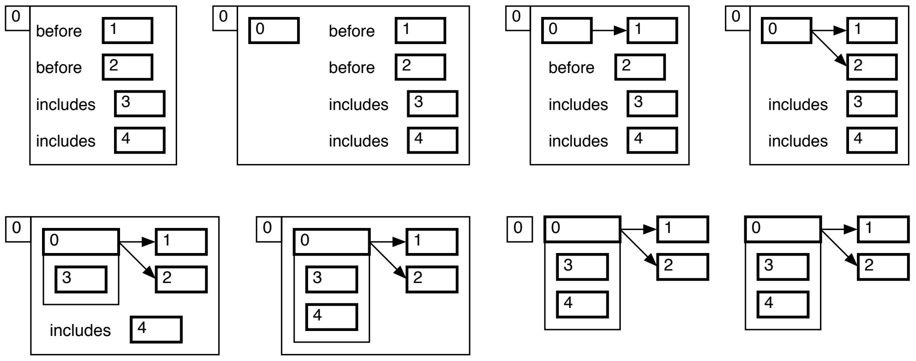


The basic AVM-to-drawing mapping rule for an AVM labeled 0 is as follows:

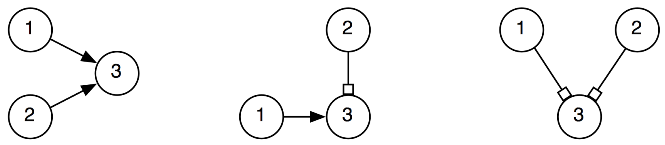
1. Draw a bar for 0 in the top left corner of the AVM
2. For every attribute equal to `before`, remove the attribute and draw an arrow from bar 0 to the drawing that is the value of `before`.
3. For every attribute equal to `includes`, remove the attribute and move the value of the attribute to the box underneath 0, draw the box if there isn't one yet. If the drawing
4. If there are no more attributes in the box, remove the border of the box.
5. Remove the label of the box if its ID is the same as the head of the drawing.[5]

[5] The head of the drawing is the bar that is at the top left, the one that dominates all others. The head of the drawing is typically the one that is referred to by the attribute to its left in the AVM. The bar that is referred to by the attribute is called the local target. Usually, the head and local target are the same but there are cases with re-entrancies where they aren't, these will be discussed later.

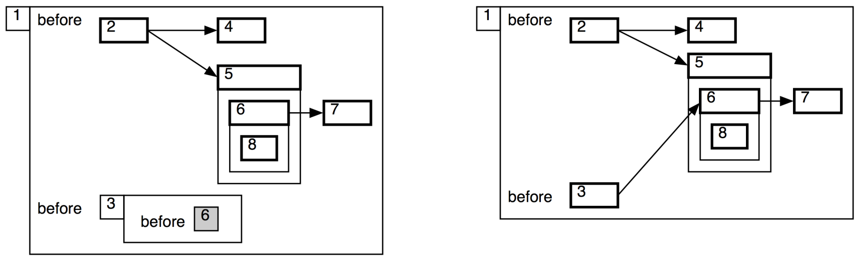
The workings of the basic rule are illustrated in the next eight AVMs/drawings:



Merging Branches Note that the basic rule only governs creation of drawings for those AVMs that do not include re-entrancies, they work for trees, not DAGs. There are three special cases to deal with nodes in the TimeML graph whose in-degree is higher than 1:

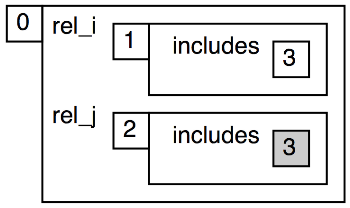


In most cases it is possible to simply draw an arrow if one merges in a re-entrancy inside a `before` relation:[6]

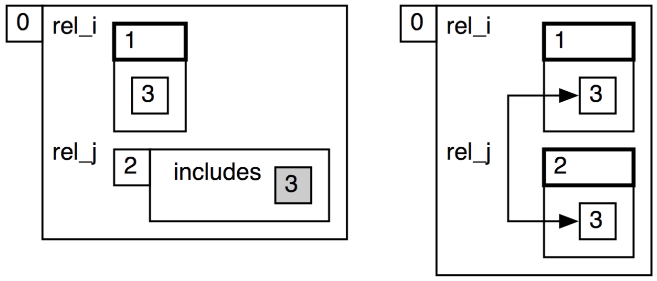


[6] Sometimes, this arrows will actually not point to the right. In that case, the drawing that is the target of the arrow needs to be moved to the right.

The most complicated case corresponds to the special rule in section 2.1: two events that are not necessarily related yet include the same event:

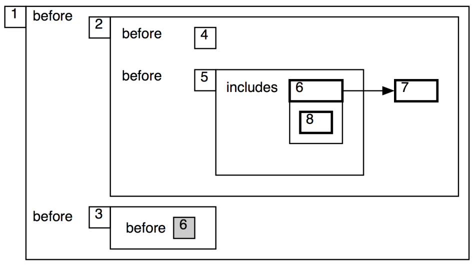


The solution is to line up horizontally the boxes of the included event, and add a connecting two-way arrow:

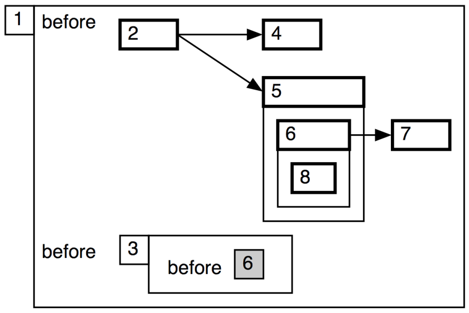


A Larger Example Here is how the pseudo AVM of the previous section fares under the drawing rules:

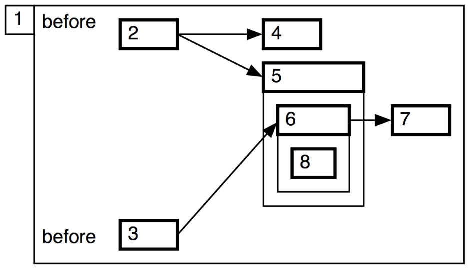
1. Replace the sub-AVM labeled 6 with a box labeled 6, which encloses a box labeled 8 and connects with an arrow to a box labeled 7:



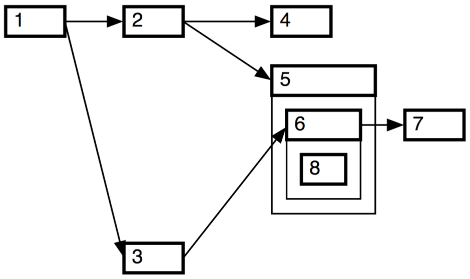
2. The next steps introduce another including box and before arrows from 2 to 4 and 5:



3. Insert the graph fragment for the re-entrancy replaces the index with an arrow to its original. Note that this requires that the requires that the target of the re-entrancy has already been drawn.



4. Finally, take care of the last two remaining before links.

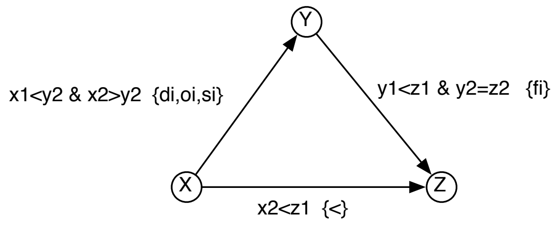


4 Consistency and Drawability

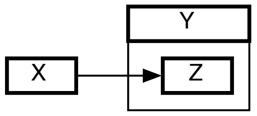
Temporal closure catches inconsistencies in an annotation. So does impossibility to arrange a graph with the rules above. Take the graph below, where each arrow indicates a before link.

$$\begin{tikzcd}
X \arrow[rr] & & Y \arrow[dl] \\
& Z \arrow[ul] &
\end{tikzcd}$$

Rule 1 in section 2 dictates that Z should be drawn both to the left of X and to the right of Y, which only Escher could have pulled off. Similarly, temporal closure will derive that given [X before Y] and [Y before Z], we should have [X before Z], but we have [Z before X]. Any consistent graph can be drawn using the procedure above. But drawability does not imply consistency when all we use is the display rules in 2. Some inconsistent graph can be drawn. As an example, take the graph below.



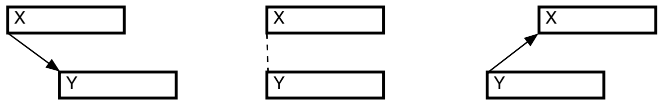
This graph can be drawn as follows.



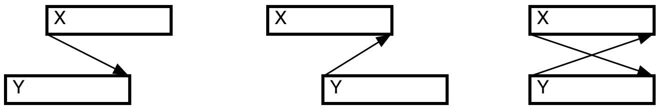
But with closure we can compose {di,oi,si} with {fi} and derive {di,oi,si}. And the intersection of {di,oi,si} with {<} is $\emptyset$, which indicates an inconsistency. I'll return to this example later.

5 Disjunctions

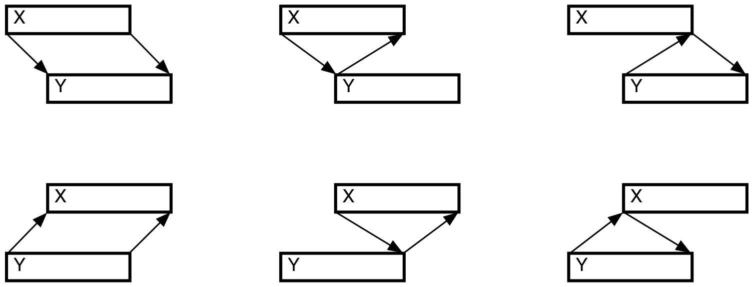
Certain disjunctions of TimeML relations can also be displayed easily. Temporal closure works with a subset of all possible disjunctions over the 13 basic relations. A total of 29 convex relations is defined by restrictions imposed by a point algebra. Of the 29 relations, 13 are Allen's basic relations and 16 are disjunctions of Allen relations. Three disjunctions describe how the begin points relate: [$x_1 < y_1$], [$x_1 = y_1$], and [$x_1 > y_1$]. The first of these corresponds to the disjunction of TimeML relations [`before` $\vee$ `ibefore` $\vee$ `ended_by` $\vee$ `includes`]. The graphs are depicted below.



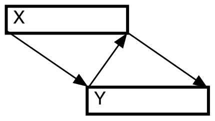
The main thing to note is that arrows are drawn from the corners of the boxes. Placement rules for arrows are unchanged, that is, the source of the arrow is placed to the left of the target of the arrow. A straight vertical dotted line connects two begin points that are equal. There are no restrictions on how the right corners of the boxes relate spatially, this is governed by the size of the boxes, which in turn is governed by the contents of the boxes. Three similar relations and graphics can be defined and drawn for relations between end points. Other disjunctions occur when the beginning of one event precedes the end of the other and vice versa ([$x_1 < y_2$], and [$y_1 < x_2$]), and when the beginnings of both events both precede the end of the other event ([$x_1 < y_2 \wedge y_1 < x_2$]).



Six other disjunctions that are defined by two point relations are printed below without comment.



Those who were counting may have noticed that so far only 26 different disjunctions have been accounted for. The ones that are missing are (i) the completely underspecified relation, which imposes no graphical ordering constraint, and (ii) the overlaps relation and its inverse, which does not exist in TimeML but which could be drawn as follows:

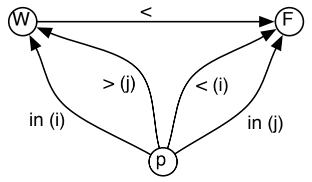


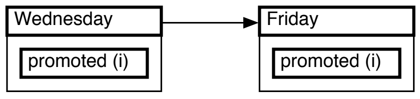
So every convex relation between events can be drawn. The question is whether the rules in section 2 should be expanded. This is an empirical question depending on (i) simplicity of design, (ii) potential for increased clutter for each display relation, and (iii) added convenience and clarity of the display. For example, adding overlap to the display is unlikely to scale up gracefully when three or more events stand in overlap relations. On the other hand, adding lines between begin points may be a viable option.

Finally, there are disjunctions that cannot be expressed with the 29 convex relations but that can occur in language. For example, what should we do with the following sentence:

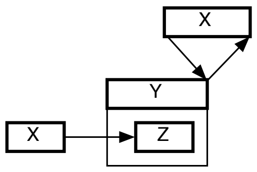
He got promoted on Wednesday or Friday

There is no neat way to localize or encapsulate disjunctions in T-BOX representations. The best we can do is to use multiple positions:





Disjunctions and Consistency Recall the example in section 4. It showed that the display rules in section 2 allow you to draw inconsistent annotations. But the example given cannot be drawn if we expand our rule base to include all 29 convex relations. The relation between X and Y now has a visual display, using arrows from the corners and an overlap of the horizontal extent:



It is clear that X now has to be at two places at the same time and that the rules that govern placement are incompatible. Consistency and drawabilty are the same thing for the 29 convex relations.

6 Conclusion

I have presented T-BOX as a viable and attractive alternative to the table-based and graph-based display modes of Alembic and Tango. It is not my intention that T-BOX represenations replace tables and graphs. Rather, I would like to see an annotation environment where the annotator can switch freely between the modes, using the display mode that seems most comfortable at a given time. T-BOX will be implemented as an addition to Tango in the near future and till then there will be no empirical data on temporal annotation with T-BOX displays.

References

1. James Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

2. David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-Initiative Development of Language Processing Systems. In *Fifth Conference on Applied Natural Language Processing Systems*, pages 88–95, Washington D.C., U.S.A., 1997.
3. David Day, Lisa Ferro, Robert Gaizauskas, Patrick Hanks, Marcia Lazo, James Pustejovsky, Roser Saurí, Andrew See, Andrea Setzer, and Beth Sundheim. The TimeBank Corpus. *Corpus Linguistics*, March 2003.
4. James Pustejovsky, Luc Belanger, Jose Castaño, Robert Gaizauskas, Patrick Hanks, Bob Ingria, Graham Katz, Dragomir Radev, Anna Rumshishky, Antonio Sanfilippo, Roser Saurí, Andrea Setzer, Beth Sundheim, and Marc Verhagen. TERQAS Final Report. Technical report, The MITRE Corporation, Bedford, Massachusetts, 2002.
5. James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5 Fifth International Workshop on Computational Semantics*, 2003.
6. James Pustejovsky, Inderjeet Mani, Luc Belanger, Linda van Guilder, Robert Knippen, Andrew See, Jon Schwarz, and Marc Verhagen. TANGO Final Report. Technical report, The MITRE Corporation, Bedford, Massachusetts, 2003.
7. Marc Verhagen. *Times Between The Lines.* PhD thesis, Brandeis University, Waltham, Massachusetts, USA, 2004.
8. Marc Verhagen. Temporal Closure in an Annotation Environment. In James Pustejovsky and Robert Gaizauskas, editors, *Language Resources and Evaluation.* Springer, Forthcoming.
9. Marc Verhagen and Robert Knippen. TANGO: A Graphical Annotation Environment for Ordering Relations. In James Pustejovsky and Robert Gaizauskas, editors, *Time and Event Recognition in Natural Language.* John Benjamin Publications, Forthcoming.
10. Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms: A revised report. In D. S. Weld and J. de Kleer, editors, *Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufman, San Mateo, California, 1990.