# Times Between The Lines

## Embedding a Temporal Closure Component in a Mixed-Initiative Temporal Annotation Framework

A Dissertation

Presented to
The Faculty of the Graduate School of Arts and Sciences
Brandeis University
Department of Computer Science
James Pustejovsky, Advisor

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

by
Marc Verhagen

November 2004

This dissertation, directed and approved by Marc Verhagen's Committee, has been accepted and approved by the Faculty of Brandeis University in partial fulfillment of the requirements for the degree of:

**DOCTOR OF PHILOSOPHY**

Adam Jaffe, Dean of Arts and Sciences

Disseration Committee:

Prof. James Pustejovsky, Chair, Dept. of Computer Science

Prof. Richard Alterman, Dept. of Computer Science

Prof. Ray Jackendoff, Dept. of Linguistics

Prof. Inderjeet Mani, Georgetown University

Dr. David Day, The MITRE Corporation

# Acknowledgements

There would be no times between no lines if it weren't for the TERQAS and TANGO projects. It was a pleasure to be involved in both of them. Thanks to all participants, that is, thanks to Luc Belanger, Jose Castaño, David Day, Lisa Ferro Robert Gaizauskas, Patrick Hanks, Linda van Guilder, Bob Ingria, Graham Katz, Bob Knippen, Marcia Lazo, Inderjeet Mani, James Pustejovsky, Dragomir Radev, Anna Rumshishky, Antonio Sanfilippo, Roser Saurí, Andrew See, Andrea Setzer and Beth Sundheim.

Thanks to Nicholas Asher, Jerry Hobbs and Frank Schilder for listening to me trying to reinvent the wheel and giving valuable feedback, and to Paul Buitelaar for his remarks on the closure chapter.

I have to quote Robert Gallager: "If something is worth doing then it is worth doing badly." I took the liberty to interpret his advice slightly differently so I was not forced to write something bad. I will not quote Ann Gallager, but I'm grateful she read most words of *Times Between The Lines* and alerted me to any weird usage of the English language. I was too stubborn to take all her hints so remaining unclarities are all mine.

Thanks to all my committee members for being a very pleasant and helpful audience when I proposed this dissertation. Inderjeet Mani provided great feedback on an early executive summary and helped with the evaluation chapter. Thanks to my advisor James Pustejovsky who gave me plenty of time and freedom and who helped greatly with the overall structure and presentation.

And finally, thanks to Ann and Nell, the significant women in my life. They kept each other company and me sane.

# Abstract

High-quality temporal annotation is not possible or realistic when we rely on human annotators alone. Temporal annotation is complex, and manual annotation of temporal links is slow, produces inconsistencies, and does not provide for a complete annotation. Unsupervised automatic annotation is not able to produce high-quality annotation either. Instead, we can combine the strengths of person and machine and let them cooperate in a mixed-initiative annotation effort. A mixed-initiative annotation framework combines high-precision preprocessing, manual annotation, temporal closure, and machine learning techniques.

This dissertation focuses on the temporal closure component and its interaction with the human annotator. The human adds temporal relations that need not be supported by textual clues; the temporal closure component adds implied temporal relations and helps the annotator add those temporal relations needed to create a complete annotation. This approach makes a complete annotation possible while making sure that the time complexity of the annotation task is linear to the size of the document.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

*The Times They Are A-Changing*

Bob Dylan, 1963

News articles typically present a story that develops over time. Events and times are introduced and the reader understands what the sequence of events is. Simple questions like "What happened after the kidnapping?" can only be answered if information about events and the temporal relations between events is available. A document needs to be annotated automatically or manually to provide this information.

The main theme of this dissertation is how a temporal closure component can be embedded in a temporal annotation environment. Temporal closure takes known temporal relations in a text and derives new implied relations from them, in effect making explicit what was implicit. I will claim that a temporal closure component helps to create an annotation that is complete and consistent. The four-step argument that lays out the motivation for this work proceeds as follows:

1. We need explicit temporal annotation for natural language processing applications like question answering and text summarization.

2. It is not yet possible to automatically create high-quality temporal annotation

3. So we have to rely on manual annotation. But manual temporal annotation is rather cumbersome and we cannot realistically expect human annotators to pick up all of the slack

4. The solution is to explore how human and computer can work together to produce high-quality temporal annotation. Temporal closure is an important aspect of the interaction.

The central idea is to let both human and computer do what they do best. The human can quickly see how events relate in time without there being any single clear and explicit textual marker; the computer can process large amounts of tedious data and skillfully perform simple reasoning tasks. In the rest of this introductory chapter, I will elaborate on the four points above.

## 1.1 Explicit Temporal Information

The notion of time is essential for talking about change. Take for example the question-answering task. We want to be able to answer questions that involve events occurring at certain times or events happening in a certain order. To do that we need to make explicit the temporal information in a text. Consider the following fragment from an Associated Press newswire.[1]

(1) Turkey (AP) _ Some 1,500 ethnic Albanians marched Sunday in downtown Istanbul, burning Serbian flags to protest the killings of ethnic Albanians by Serb police in southern Serb Kosovo province. The police barred the crowd from reaching the Yugoslavian consulate in downtown Istanbul, but allowed them to demonstrate on nearby streets.

This text is easy to understand and we all know what happened and when things happened. But what do we need to know exactly when we answer specific questions? Take the three questions below.

(2) What happened on Sunday?

(3) Were Serbian flags burned before the killings?

(4) Were ethnic Albanians killed during the demonstration?

[1]Taken from TimeBank document APW19980308.0201.

The first one is the simplest. The text contains a temporal expression *Sunday* and an event-denoting verb *marched* right next to it. It is not too much of a stretch to put these two together. Question (3) is more complicated. But if we have encoded that a pattern "doing X to protest Y" implies that X is during Y or after Y, then we can give an answer. Question (4) is even harder because it requires ordering of events that do not occur in near proximity to each other, and there are no obvious markers that give us the needed information.

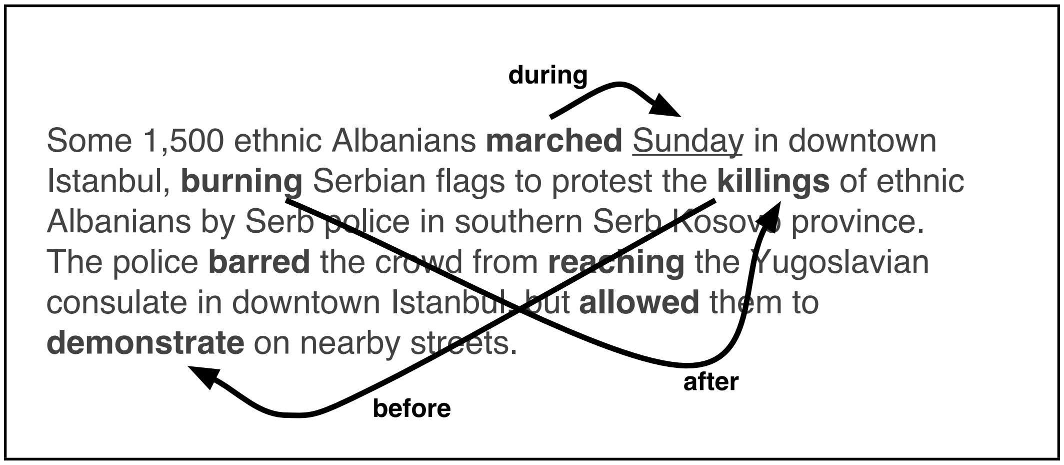

during
Some 1,500 ethnic Albanians **marched** Sunday in downtown Istanbul, **burning** Serbian flags to protest the **killings** of ethnic Albanians by Serb police in southern Serb Kosovo province. The police **barred** the crowd from **reaching** the Yugoslavian consulate in downtown Istanbul, but **allowed** them to **demonstrate** on nearby streets.
after
before

Figure 1.1: Enriching a text with temporal information

In any case, a document can be marked up to provide the information we need. We can add tags to the text that mark the events and time expressions as well as the temporal relations between them. The text in example (1) can be marked up with temporal information as in figure 1.1. The event *marched* is now explicitly anchored to the time expression *Sunday*. In addition, the events in the pairs *burning-killings* and *killings-demonstrate* are now ordered relative to each other. Once all events are anchored and ordered we can effectively create a timeline and graphically display the sequence of events in the document.

## 1.2 Automatic Tagging of Temporal Information

A document could be processed automatically to achieve the results in figure 1.1. The last couple of years have seen some significant research on event extraction, extraction of time expressions, and event anchoring and ordering. Some of this recent work is described below in a short overview.

(Aone and Ramos-Santacruz, 2000) describe REES (Relation and Event Extraction System), a large-scale, end-to-end relation and event extraction system. It extracts 100 different types of events and relations. Each event and relation is associated with a template. Lexical entries with subcategorization frames and generic syntactic patterns are used to populate slots in the template. The event templates can have a slot TIME, which provides for the opportunity to anchor the event if syntactic clues are available. The relations are not temporal relations, but rather pairs like Place-Country, Person-Age, Organization-Location, etc. One relation implicitly involves a temporal relation, namely Person-BirthDate.

Inderjeet Mani and George Wilson (Mani and Wilson, 2000) use hand-written and machine-learnt rules to resolve time expressions like *now*, *today* and *two weeks ago* by assigning them ISO values. For example, the expression *two weeks ago* should be mapped to the ISO time value 20:04:10:02 if the reference time is October 16th 2004. In addition, the authors describe some highly preliminary work on event anchoring and ordering, where tense-based heuristics are used to derive temporal ordering of events relative to speech time or reference time.

Elena Filatova and Eduard Hovy (Filatova and Hovy, 2001) describe a procedure for arranging the contents of news stories in a timeline by assigning timestamps to all events. Overt time markers are used when available, otherwise tense is used to map the event to an open-ended time interval.

Frank Schilder and Christopher Habel (Schilder and Habel, 2001) create a semantic tagging system for temporal expressions and methods to extract temporal information conveyed by those expressions. Their ultimate goal is to establish temporal relations between all events in an article, but for now they focus on anchoring temporal expressions in a timeline. They use finite state techniques to parse carriers of temporal information like prepositional phrases and adverbs. Event-denoting expressions are extracted using a lexicon of event-denoting verbs and nouns. Temporal information is then derived using unification over semantic attributes of events and time expressions.

(Pustejovsky et al., 2002) use finite state techniques to extract time expressions, events and temporal relations between events and time expressions. Temporal relations are extracted using a small set of syntactic heuristics.

Inderjeet Mani, Barry Schiffman and Jianping Zhang (Mani et al., 2003) employ a machine learning approach to temporal anchoring and partial ordering of events. Their algorithm creates anchorings of events to explicit or implicit time values (tvals). The tvals themselves are partially ordered and the anchorings of events to tvals can be used to create a partial ordering of events.

The research above is promising and some components, most notably the recognition of time expressions, are of a high quality. Nevertheless, it is too early to comfortably use state-of-the-art automatic generation of temporal relations because it does not yet exhibit high enough precision and recall[2]. (Mani and Wilson, 2000) report 59% precision for their highly experimental event anchoring procedure. (Filatova and Hovy, 2001) report 80% precision for theirs. (Schilder and Habel, 2001) claim 92% precision and 94% recall for simple time expressions and 87% precision and 91% recall for complex expressions (prepositional phrases); they do not give precision figures for the event anchoring stage. Finally, (Mani et al., 2003) report 84% precision for event anchoring, and 74% precision and 78% recall for event ordering (where the recall of temporal relation is relative to relations added by a human annotator, not the total number of relations possible).

Some temporal information is hard to extract automatically. We saw this earlier in example (1) and question (4) where the two events in question, *killed* and *demon-*

[2]Precision is defined as the percentage of correct answers. Recall is defined as percentage of correct answers relative to all possible correct answers.

*stration* were not obviously related. Sometimes there are overt markers like tense or temporal adverbs, but more often we have to rely on lexical information and knowledge of how things are in the world. We are perhaps quite far away from the time when machines can reliably extract all interesting temporal relations.

## 1.3 The Problem with Manual Annotation

Manual annotation needs to be a part of the total annotation effort, given the precision and recall figures for machine annotation. But manual annotation comes with its own set of practical challenges. The task is a complex one, characterized by high density, low markup speed, hard-to-avoid inconsistencies, and low inter-annotator agreement.

The high density is due to the fact that the set of possible temporal relations is essentially quadratic to the number of events and time expressions in a document. If a document has $N$ events and time expressions, then there are $N(N-1)/2$ possible temporal relations. A typical TimeBank document contains about 50 temporal objects, which implies 1225 possible temporal relations. Larger documents with about 150 time objects (events and time expressions) have over 10,000 relations. An annotation that contains all temporal relations is clearly impractical by human means alone.

Annotation of temporal relations requires more reflection than for example annotation of part-of-speech tags, and is therefore slower. Syntactic tags and many semantic tags, such as entity tags and event tags, can be added in a strictly linear fashion. Temporal relations are different because they require us to specify attributes of pairs of objects, and the objects involved may not be close to each other in the text. Annotating a mid-sized newspaper article can therefore take up to an hour.

Experience with consistency checking tools showed that it is hard to annotate a one-page document without introducing inconsistencies. An inconsistency can occur because the choice for a particular temporal relation often restricts subsequent choices. For example, if an annotator decides that X is before Y and Y is before Z, then the choice of temporal relations between X and Z is constrained. But even trained annotators are liable to introduce relations that clash with previous choices. This is sometimes the result of vague or ambiguous temporal relations between events, and sometimes the result of a fuzzy interpretation of a particular relation (for example, does X includes Y mean that X and Y may share a beginning point, or not?). But often plain fatigue is to blame.

A manually annotated document is sparse in temporal information given the high number of potential relations. More often than not, two annotators choose to add different temporal relations simply because the space they can pick from is so large, as depicted in figure 1.2.

It is unreasonable to expect an annotator to add much more than a hundred links in a two-page document. On average, annotators annotate about 1-5% of all possible relations. In only about 20% of the cases two annotators choose to add

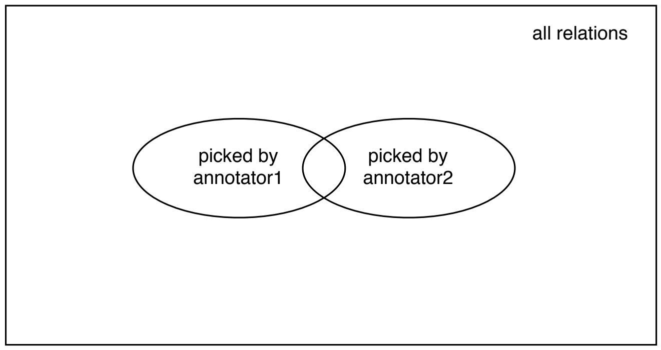


Figure 1.2: Two annotators marking up different temporal relations

temporal relations between the same two time objects.[3] This is problematic because low inter-annotator agreement scores are perceived by many to indicate an ill-defined annotation task (Hirschman et al., 1998; Setzer, 2001).

## 1.4 Mixed-Initiative Annotation

In the previous sections we saw that neither machine nor human can produce a high-quality annotation that is consistent and complete. Manual temporal annotation is expensive and time-consuming and clearly impractical if a complete annotation is needed. Fully automatic temporal annotation is not yet up to the task and exhibits precision and recall figures that are not high enough.

A central theme in this work is that machine and human need to focus on those

[3]This is probably a rather pessimistic figure since it is based on a small experiment with naive annotators. Trained annotators that have memorized a solid set of annotation guidelines may choose to add the same relations more often.

tasks they do best and interact to improve each other's decisions. This idea that has been considered before in work on mixed-initiative interaction in multi-agent systems.[4] In such systems, agents grab control of a task when they feel they are best suited to deal with it. In linguistic annotation, the Alembic Workbench (Day et al., 1997) was positioned as a mixed-initiative annotation framework. Mixed-initiative temporal annotation is a hybrid approach that goes some way towards meeting the practical challenges outlined above. It includes a range of modules:

- automatic pre-processing for those tasks that have high precision, most notably recognition of events and time expressions
- manual annotation
- a user-assisted temporal closure algorithm
- machine learning techniques

There are many open questions with regard to mixed-initiative temporal annotations. There are questions about what sort of machine processing to use and what the characteristics of the human-machine interaction are (when, how long and how often?). In addition, it is not entirely clear what the optimal division of labor between man and machine is.

This dissertation touches upon some of the issues enumerated above. It focuses on the closure component and its interaction with the annotator. Temporal closure

[4]See (Hearst et al., 1999) for an introduction to this field.

takes known temporal relations in a text (given by manual annotation or high-quality preprocessing) and derives new implied facts from them. It makes higher coverage possible without adding strain to the human annotator and makes it easier to create a consistent temporal annotation because it constrains choices and finds inconsistencies in a set of relations added before closure applied. In addition, temporal closure can be employed in a user-assisted mode where the user is asked to fill in temporal relations and the machine continues to add facts after each user-added relation. I will show that this approach makes it feasible to achieve a near-complete annotation because temporal closure will derive about 95% of the temporal relations. In other words, temporal annotation helps the annotator to find those 5% of temporal relation in the total relation space that make all other relations implicit.

What we in effect do is to apply easily codable domain knowledge within a tool in order to expand the annotation and promote its consistency; this may involve methods that ensure a fair trade-off between level of completeness and human annotator stress.

## 1.5 Context

Temporal annotation is set in the broader area of temporal interpretation of natural language. In this context, temporal annotation is an attempt to capture temporal information in real texts. As mentioned before, this is a hard task, not only because of the density and the complexity but also because of vagueness at several levels.

For example, when thinking about temporal annotation we could ask ourselves the following questions:

1. What are the primitives to work with (that is, what events should be selected from a text)?
2. How precise can and should the temporal relations between the primitives be?
3. What temporal relations out of all possible relations should be annotated?

Fortunately, time is a well-structured domain and a machine can exploit this structure and aid a human annotator when he or she creates temporal annotations. We will see that the structure of time can be exploited to structure the annotation task.

Much of the research in this dissertation is inspired by or based on work by Andrea Setzer (2001). She set up an environment for temporal annotation that includes a temporal closure component and a way for annotators to interact with the closure component. But *Times Between the Lines* differs in several major respects.

Setzer uses a rather coarse grained annotation language that only distinguishes between three temporal relation types: precedence, inclusion and simultaneity. These relation types are rather vague, especially the simultaneity relation, which is interpreted as “roughly at the same time”. As a result, Setzer’s inference rules are not precise and are in fact almost guaranteed to derive wrong inferences. This work, on the other hand, takes TimeML (Pustejovsky et al., 2003a) as the annotation language. TimeML defines its set of temporal relations more precisely and makes much finer distinctions between temporal relations possible, as a result, it allows a more precise inferencing system.

The small set of inference rules at the core of Setzer's algorithm is not sound and not complete: incorrect inferences can be generated and inconsistencies in the input are not necessarily detected. SputLink, the closure component described in here, uses an exhaustive set of composition rules and its algorithm is sound and complete, that is, it is guaranteed to find all inferences possible and all inconsistencies in its input.

Some possibly arbitrary choices are made in the course of the present work regarding the characteristics of the interaction between annotator and machine and the characteristics of the annotation that is the result of the interaction. These choices can be formulated as a list of desiderata:

1. reduce and simplify the task of the annotator as much as possible
2. ensure consistency of an annotation
3. allow an initial phase of annotation, that is, a phase where there is no interaction with a closure component and where the closure component does not generate restriction for the annotator
4. view annotation as a one-person affair.

All these requirements and choices guide the eventual setup of the closure component and its interaction with the user. For example, the wish to simplify the task of the annotator results in limitations on the interaction between annotator and closure component.[5] And to allow an initial annotation without closure means that inconsistencies can be generated at that some mechanism is required to spot and remove those inconsistencies (because that is required by the wish to ensure consistency).

A final note before I lay out the structure of the remainder of the dissertation. Many higher-level observations are made about the interaction between closure algorithm and annotator. But there is scant attention to the more physical aspects of this interaction. That is, there are few observations on visualization issues and other physical aspects of the human-computer interface. While worthwhile, these are beyond the scope of *Time Between the Lines*.

Here is how the rest of this dissertation is set up. Chapter 2 gives an overview of related work, most notably the research that *Times Between The Lines* takes as its point of departure. Section 2.1 gives an overview of some of the issues in temporal reasoning and conceptualizations of time. In section 2.2, I describe the TimeML annotation scheme, which is used as the representation language for temporal relations.

[5]To be more precise, it results in limitations on what temporal relations the annotator is asked to provide by the user-assisted closure component.

Section 2.3 acknowledges that annotation occurs in an environment and presents the tools used in this disseration. Chapters 3 and 4 are the meat of the dissertation. In chapter 3, I discuss in more detail some of the work that informed the present approach to temporal closure, and present the SputLink algorithm and its embedding in an annotation environment. Chapter 4 investigates the claims made in this dissertation and evaluates what best way to embed temporal closure into an annotation environment. Finally, chapter 5 ties it all up and gives some hints for future work.

# Chapter 2

# Times, Tags and Tools

*We must use time as a tool, not as a couch.*

John Fitzgerald Kennedy

This chapter gives an overview of some of the prior research that is relevant for the task at hand. The temporal closure module SputLink in chapter 3 is built upon well-established work on interval algebra and point algebra, section 2.1 places the current research in its theoretical context. Temporal closure as conceived here is embedded in an annotation environment and therefore the annotation language is the vehicle on which the temporal closure component operates. TimeML is the annotation laguage of choice and is described in section 2.2. Finally, section 2.3 presents the annotation tools that constitute the annotation environment.

## 2.1 Temporal Reasoning

Many approaches for reasoning over time have been proposed. The Minimal Tense Logic $K_t$ (Prior, 1957; Prior, 1967) uses temporal modal operators over propositions. The situation calculus (McCarthy and Hayes, 1969; Reiter, 2001) was perhaps the most widely adopted early attempt to model action and change in AI. Where $K_t$ uses a modal operator to provide for the temporal interpretation of an expression, the situation calculus introduces an additional argument. The situation calculus represents actions and their effects on the world and models the world as a set of situations which model the possible configurations of the world at a particular time. An event changes the configurations. For example, the states described in sentence (5) could be encoded as in (6).

(5) Sue gave an apple to Eve

(6) a. Have(s1,Sue,Apple)

b. Have(s2,Eve,Apple)

(McDermott, 1923) takes over the most general assumptions of the interval calculus but abandons the notion of events as fact changers. Instead, he defines an event as a set of intervals, intuitively those intervals in which the event happens once. The event calculus (Kowalski and Sergot, 1986; Kowalski, 1992) was introduced as a logic programming formalism for representing events and their effects. Events are treated as primitives that update the state of the world.

But the most direct mapping to our annotation task is provided by James Allen's Interval Algebra (Allen, 1983; Allen, 1984). Allen defines events in very much the same way as McDermott: as intervals during which a certain proposition holds. The intervals are considered the primitives, and temporal constraints are expressed as relations between intervals. Allen's work is described in more detail in chapter 3.

One of the decisions that must be made when designing a system for temporal reasoning is which temporal unit to take as primitive: the point or the interval. James Allen came down firmly on the interval side and argued that there is no need to use points since all events can be decomposed. This is true even for those events that appear to be a precise point in time, as in *finding a letter*. This event may include looking at a particular spot and then realizing that one is looking at the lost letter. If events that seem to be points can be decomposed, then it is perfectly alright to use very short intervals in the logic.

Choosing the interval as the primitive relieves us from the Divided-Instant Problem (van Benthem, 1983). Suppose there is a property, P, of the light being on, an interval, X, where P holds, and an interval, Y, where P does not hold. What happens at the point A where these two intervals meet? Or, to frame it differently, are the intervals open or closed? If X and Y are closed (that is, they include their boundaries) then property P is both true and false at point A, if intervals are closed then P is neither true nor false at point A. If the interval is the primitive then there is no need to resolve this logical problem.

The main problem with the interval algebra is that inconsistency tracking is not tractable. Fortunately, a subset of the interval algebra can be converted to a point algebra where inconsistency checking is tractable (Vilain et al., 1990; van Beek, 1992). The work described in this dissertation is at its core an application of the restricted interval algebra. A more detailed description is provided in sections 3.2 and 3.3.

## 2.2 Temporal Annotation

Temporal annotation is an essential part of many text-understanding efforts. Recent efforts within TIDES (Translingual Information Detection, Extraction, and Summarization), STAG (Sheffield Temporal Annotation Guidelines) and TimeML all aim to provide a markup language for temporal annotation. TIDES (Ferro et al., 2001) defined a set of guidelines for annotating time expressions with a canonicalized representation of the times they refer to. STAG (Setzer, 2001; Setzer and Gaizauskas, 2001) provides guidelines for annotating events and temporal information in newswire Texts. TimeML (Pustejovsky et al., 2002; Pustejovsky et al., 2003b) builds upon TIDES and STAG and provides an XML-compliant annotation scheme for times and events.

In this section I will introduce the TIDES and STAG annotation efforts and then lay out the TimeML annotation scheme, which is taken as the representation language for temporal information and the input on which temporal closure operates.

### 2.2.1 TIDES and STAG

The TIDES annotation standard (Ferro et al., 2001) for temporal expressions calls for a single XML tag <TIMEX2>. This tag is an extension to the <TIMEX> tag used in the Message Understanding Conferences (Grishman and Sundheim, 1996; MUC7, 1998): it replaces the TYPE versus DATE categorization attribute with a series of attributes to represent the actual time or date of the expression. The standard includes guidelines for annotating temporal expressions with a canonicalized representation of the times they refer to, using the ISO 8601 standard. Two examples of TIMEX2 tags are presented below.

(7)
```
<TIMEX2 VAL="1999-FA">Fall 1999</TIMEX2>

<TIMEX2 VAL="2004-04-23">today</TIMEX2>
```

Temporal expressions are viewed as stand-alone targets for annotation and extraction, and the <TIMEX2> tag is intended to support a variety of applications. (Mani and Wilson, 2000) created TEMPEX, an automatic time tagger that adds <TIMEX2> tags to documents.

Robert Gaizauskas and Andrea Setzer proposed STAG (Sheffield Temporal Annotation Guidelines) as a means to annotate events, time expressions and the relations between them (Setzer, 2001; Setzer and Gaizauskas, 2001). STAG classifies events in four groups: occurrences, perception events, reporting events and aspectuals. States are not annotated in STAG. In general, subcategorisation frames of events are ignored, except for reporting, perception and aspectual events, which usually have an other event as an attribute.

Events and time expressions are related with the `relatedToEvent`, `relatedToTime`, and `relType` tags. The values of the first two are references to other events or time expressions, the third contains a value for the relation type and is restricted to one of five values: `BEFORE`, `AFTER`, `INCLUDES`, `IS_INCLUDED` and `SIMULTANEOUS`. This last relation type is intended to be fuzzy and include all kinds of overlaps. An example of STAG annotation is in example (8).

(8) *The plane crashed on Wednesday*

```
The plane
<event eid=9 class=OCCURRENCE tense=PAST relatedToTime=5
relType=IS_INCLUDED>crashed</event>
<timex tid=5>Wednesday</timex>
```

In addition to this annotation scheme, (Setzer, 2001) proposes a three-step annotation process for temporal relations. The first phase consists of annotating relations that are made explicit by linguistic markers in the text. In the second phase relations are identified that are not explicitly marked. The last phase is comprised of a user-assisted closure loop in which a temporal closure component infers new relations and the user is asked to provide a new relation when the closure component cannot derive any more new relations. This is described in greater detail in section 3.4.2.

### 2.2.2 TimeML

TimeML is a general annotation scheme, a markup language capable of capturing all salient temporal information in a text. It was first developed during two ARDA workshops called TERQAS and TANGO (Pustejovsky et al., 2002; Pustejovsky et al., 2003b). With TimeML, the two workshops attempted to provide a metadata standard for markup of events, their temporal orderings, and how they are temporally related to each other. TimeML adopted the core of the Sheffield Temporal Annotation Guidelines and remained compliant to the TIDES time expression annotation.[1]

The notions of temporal object and temporal relation are central to TimeML. Temporal objects are marked up directly in the text, with XML-style tags surrounding the text sequences that denote a temporal object. Links are encoded by markup tags that consume no input.

#### Temporal Objects

There are two kinds of temporal objects: time expressions and events. These are marked up with the <TIMEX3> and <EVENT> tags. The <TIMEX3> tag is based on the <TIMEX2> tag and marks up temporal objects like *today*, *last Wednesday*, *about three years ago*, etc.

The <EVENT> tag is used to annotate those elements that mark the semantic

[1]The version of TimeML described here is 1.0. Some changes have been made since but none of them have a significant impact on the work described in this dissertation.

events in a text. Typically, events are verbs, although certain nominals, such as *crash* in *killed by the crash*, are also annotated as events. There are six event classes: occurrence, perception, reporting, aspectual, state, i_state, and i_action. Events are also annotated for tense and aspect. An example of TimeML event annotation is given in (9).

(9) *The agencies fear they will be unable to crack those codes to eavesdrop on spies and crooks.*

```
The agencies
<EVENT eid="e1" class="I_STATE">fear</EVENT>
they will be
<EVENT eid="e2" class="I_STATE">unable</EVENT>
to
<EVENT eid="e3" class="OCCURRENCE">crack</EVENT>
those codes to
<EVENT eid="e4" class="OCCURRENCE">eavesdrop</EVENT>
on spies and crooks.
```

TimeML distinguishes between event tokens and event instances, where the event instance is the actual realization of the event. This distinction is illustrated by examples like *John taught on Monday and Tuesday* where there are two teaching events. In order to annotate such cases, TimeML introduces a <MAKEINSTANCE> tag which is used alongside the <EVENT> tag to create the two instances of *taught*, as shown in example (10) below.

(10) *John taught on Monday and on Tuesday*

```
John
<EVENT eid=e1 class=OCCURRENCE>taught</EVENT>
on
<TIMEX3 tid=t1>Monday</TIMEX3>
and
```

```
<TIMEX3 tid=t2>Tuesday</TIMEX3>
<MAKEINSTANCE eiid=ei1 eventID=e1/>
<MAKEINSTANCE eiid=ei2 eventID=e1/>
```

Here, the event token with event ID `e1` is mapped to two event instances, with event instance IDs `ei1` and `ei2`. We will see in the next section that in TimeML events are temporally linked with other temporal objects through their instances. It should be noted that in the vast majority of cases there is only one instance for each event token.

**Temporal Links**

Temporal links encode temporal relations between events and time expressions. TimeML distinguishes three kinds of links: TLINKs encode temporal relations proper, ALINKs encode aspectual relations, and SLINKs encode modality, negation and factuality. These three kinds of links each have their own markup tag: <TLINK>, <ALINK>, and <SLINK>. Temporal relations of an event are not annotated on the event itself, but in a separate non-input-consuming tag that links events and time expressions to each other. The different kinds of links will be explained presently, but first consider the TLINKs that can be added to the annotation in (10), shown below in example (11).

(11)
```
<TLINK eventInstanceID=ei1 relatedToTime=t1 relType=is_included/>
<TLINK eventInstanceID=ei2 relatedToTime=t2 relType=is_included/>
```

The two instances of the *teaching* event are both linked to a time expression, one to *Monday* and one to *Tuesday.* For events, the link is always through the instance, even if there is only one.

The aspectual ALINKs and the subordinating SLINKs are almost always intra-sentential and often encode an argument relation between the two events that are linked. Some examples of ALINKs and SLINKs are given in (12) and (13).

(12) a. John *should* have *bought* wine

b. John *managed* to *leave* the party

c. John *said* he *bought* some wine

(13) a. John *started* to *read*

b. John *kept talking*

Modal SLINKs are introduced mostly by modal verbs, as in (12a). Verbs that entail the veracity of their arguments, like *managed* in (12b), give rise to factive SLINKs. Evidential SLINKs are often generated by reporting verbs, as in (12c). Aspectual links represent the relationship between an aspectual event and its event argument. They can indicate initiation, culmination, termination and continuation. Two examples of aspectual links are in (13).

The TLINKs are the temporal links proper. Unlike the other two types of links, they are not necessarily local and there need not be a syntactic relation between the two events or time expressions that are linked. They are also the links that provide the input to the closure engine[2] and will therefore be discussed in more detail. All TLINKs have a relation type attribute `relType`. TimeML distinguishes fourteen relation types, some of them inverses of each other.

simultaneous

Two events are judged simultaneous if they happen at the same time or if they are so close together that distinguishing their times makes no difference to the temporal interpretation of the text.

before,after

For temporal precedence of events and times.

[2]It is definitely the case that ALINKs and SLINKs carry temporal information and that this information could be exploited in a temporal closure component. We should separate temporal closure from the mapping of implicit temporal information in SLINKs and ALINKs to explicit temporal information. In other words: a separate module can create TLINKs from SLINKs and ALINKs and the closure component can take these new TLINKs as extra input.

ibefore,iafter

Like `before` and `after`, but indicates that one event is immediately before or after the other, as in *All passengers **died** when the plane **crashed** into the mountain.*

includes,is_includes

As with the relation between the temporal expression and the event in *John **arrived** in Boston on **Thursday**.*

holds,held_by

These are much like the `simultaneous` relation. The difference is that `holds` is a relation between an event and a time: an event holds during a particular time, as in *John was **CEO** for **two years***. The relation type `held_by` is almost never added by the annotators, it was added to TimeML mostly because the closure engine required that all relations could be reversed and it was convenient to have a name for the inverse.

begins,begun_by

As is the case between the event and the first temporal expression in *John was **in the gym** from **5pm** till 6pm.*

ends,ended_by

As is the case between the event and the second temporal expression in *John was **in the gym** from 5pm till **6pm**.*

identity

Event identity is also annotated as a TLINK even though it is not a temporal relation proper.

unknown

This relation was added for the sake of user-prompting in the user-assisted phase (see section 3.6.1). It is often not possible to specify a temporal relation between two random events in a text. If a user is forced to provide a temporal relation then he or she should be able to state that the relation is not known. This way, we can make a distinction between relations that have not yet been considered by the annotator and relations that were considered but have no value.

TimeML has no `overlap` relation, this is motivated by the observation that this relation does not naturally occur in real texts. The relations above are intended to be mutually exclusive, but the guidelines do acknowledge that especially the `simultaneous` relation can be a bit fuzzy. This point will be taken up again in chapter 4.

### TimeBank

The Terqas and Tango workshops also produced TimeBank, a TimeML-annotated corpus of about 200 articles from the *New York Times*, *Wall Street journal*, *Associated Press* and others. TimeBank was conceived as a gold standard annotated in strict compliance with the TimeML annotation guidelines. It contains about 8000 events, 1400 time expressions, 300 ALINKs, 2600 SLINKs, and 6000 TLINKs. See (Day et al., 2003) for more details. TimeBank was used in this dissertation as a test bed for the temporal closure engine.

## 2.3 Annotation Tools

There is a dearth of annotation tools nicely attuned to the task of marking up temporal relations. Some of the tools that can be used for temporal annotation are listed in this section. Special attention goes towards those tools that were used for experiments in the present work.

The Alembic WorkBench (Day et al., 1997) is a widely used mixed-initiative annotation tool. Text-extents can be marked up with user-defined tags, and a relation table lets annotators define relations between text extents. This table feature was originally introduced to accomodate coreference chains but could easily be adapted to deal with temporal links.

Annotate (Plaehn and Brants, 2000) is another mixed-initiative annotation tool. It is a syntactic annotation tool that interacts with a tagger and a parser. It contains trees with labeled terminal nodes for POS tags and morphology, labeled edges for grammatical functions, and labeled non-terminal nodes for phrase categories. It also alllows secondary links that can be drawn between arbitrary nodes for anaphor resolution. This last feature can be exploited to add temporal links.

(Setzer, 2001) created an annotation tool for STAG. It lets users add attributes to events that indicate what other events they are related to. In addition, it has a simple user-prompting interface in which the tool asks the user to provide the temporal relation between two unrelated events.

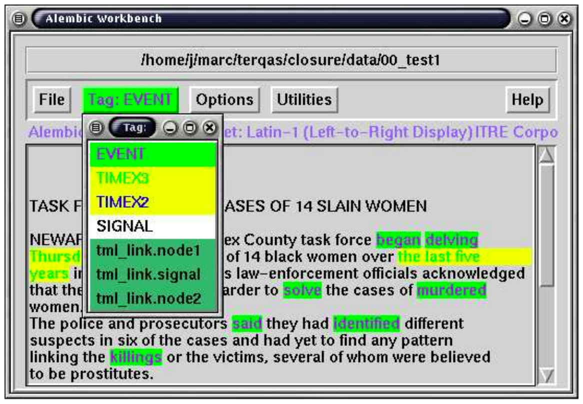
None of these tools is a perfect match for the link annotation task. They do not make adding large amounts of links easy and they do not provide a graphical interface that displays annotation results in an intuitive fashion.[3] Recently, two tools were created to make the annotation task less tedious and more intuitive: the Event Diagram and Tango. The Event Diagram is an addition to Alembic that was developed during the TERQAS workshop (Pustejovsky et al., 2002). It featured a semi-graphic display of events, and a user-prompting component. Tango (Pustejovsky et al., 2003b) is a graphic tool that provides functionality close to timeline-editing.

The following subsections contain descriptions of the tools used in this dissertation. Section 2.3.1 lays out Alembic and its extension the Event Diagram; section 2.3.2 presents Tango.

[3]Video editing tools like Ontolog (Heggland, 2002) do have a nice graphical display of a timeline but lack functionality that affords annotation of relations between events at the semantic level. A similar point can be made for ANVIL (Vintar and Kipp, 2001; Kipp, 2001), a multi-modal dialog annotation tool.

### 2.3.1 Alembic and the Event Diagram

Alembic is a general-purpose annotation tool that has been successfully used for many linear annotation tasks. Alembic makes it easy to add tags like <EVENT> and <TIMEX3> to text spans. As mentioned above, Alembic uses a table metaphor to annotate relations between tags. Screen shots of Alembic's text and relation table windows are provided in figures 2.1 and 2.2.

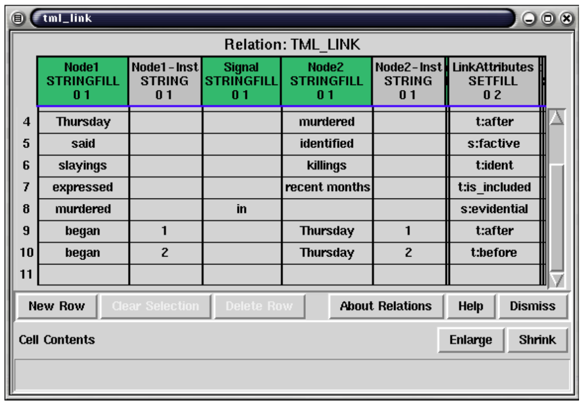


TASK F ASES OF 14 SLAIN WOMEN

NEWAF ex County task force began delving
Thursd of 14 black women over the last five
years ir s law-enforcement officials acknowledged
that the arder to solve the cases of murdered
women.
The police and prosecutors said they had identified different
suspects in six of the cases and had yet to find any pattern
linking the killings or the victims, several of whom were believed
to be prostitutes.

Figure 2.1: Alembic's Text Window

The problem is that the relation table is essentially a linear interface, but the task of annotating temporal links is by no means linear. In practice, the table display proved unintuitive and it was hard to track what relations had already been added. It was nearly impossible to manage the high density of temporal annotation. Adding <TLINK> tags in Alembic is a rather clumsy affair requiring a long click sequence. Nevertheless, experienced annotators generally developed a relatively fast way of an-



Relation: TML_LINK

| | Node1 STRINGFILL 0 1 | Node1 - Inst STRING 0 1 | Signal STRINGFILL 0 1 | Node2 STRINGFILL 0 1 | Node2 - Inst STRING 0 1 | LinkAttributes SETFILL 0 2 |
|---|---|---|---|---|---|---|
| 4 | Thursday | | | murdered | | t:after |
| 5 | said | | | identified | | s:factive |
| 6 | slayings | | | killings | | t:ident |
| 7 | expressed | | | recent months | | t:is_included |
| 8 | murdered | | in | | | s:evidential |
| 9 | began | 1 | | Thursday | 1 | t:after |
| 10 | began | 2 | | Thursday | 2 | t:before |
| 11 | | | | | | |

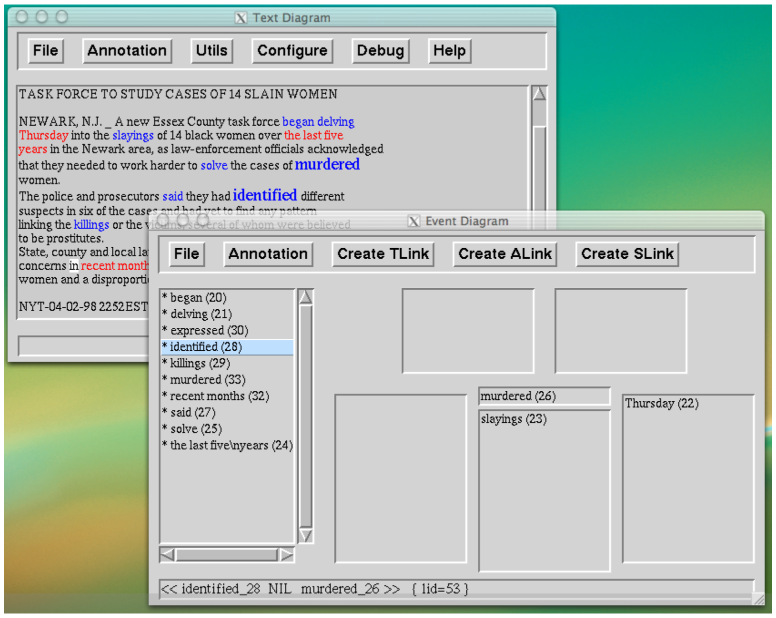
Cell Contents

Figure 2.2: Alembic's Relation Table Window

notating temporal relations and Alembic was successfully used in the creation of much of TimeBank.

The Event Diagram was developed for the Terqas project. It is a semi-graphic link annotation tool that shows how a temporal object relates temporally to all other objects. The display was designed to resemble a timeline as closely as possible, for example, all the events that occurred before the event in focus would be displayed in a widget to the left. The tool also afforded a less linear way of adding links: any event in the text could be put in the focus position easily and be linked to other events. The Event Diagram is tightly integrated with a closure module and implemented the user-prompting and text-segmented closure approach introduced in section 3.6.2. A screen shot of the Event Diagram is in figure 2.3.

Annotators reported that the Event Diagram allowed faster link creation than Alembic and that it encouraged a more semantic exploration of a text. A disadvantage



TASK FORCE TO STUDY CASES OF 14 SLAIN WOMEN

NEWARK, N.J. _ A new Essex County task force began delving
Thursday into the slayings of 14 black women over the last five
years in the Newark area, as law-enforcement officials acknowledged
that they needed to work harder to solve the cases of murdered
women.
The police and prosecutors said they had identified different
suspects in six of the cases and had yet to find any pattern
linking the killings or the victims, several of whom were believed
to be prostitutes.
State, county and local la
concerns in recent month
women and a disproporti

NYT-04-02-98 2252EST

* began (20)
* delving (21)
* expressed (30)
* identified (28)
* killings (29)
* murdered (33)
* recent months (32)
* said (27)
* solve (25)
* the last five\nyears (24)

murdered (26)
slayings (23)
Thursday (22)

<< identified_28 NIL murdered_26 >> { lid=53 }

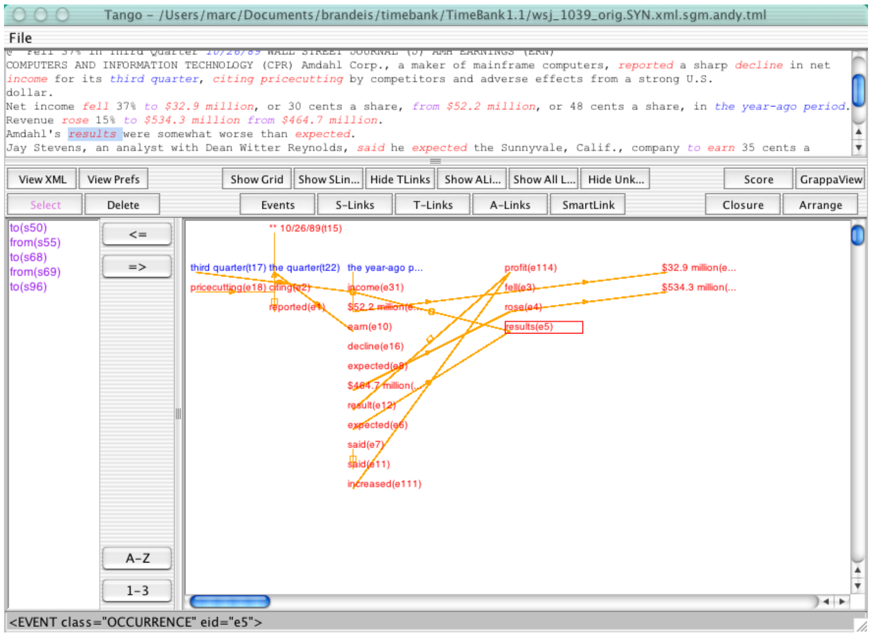
Figure 2.3: The Event Diagram

was that it was hard to get used to the incomplete spatial metaphor.

### 2.3.2 Tango

The high density of temporal annotation almost requires some form of graphical annotation. Tango provided an editable fully graphic display of events and temporal relations on a timeline (see figure 2.4). Tango also incorporated a closure component as well as most functionality in the Event Diagram. A bit surprisingly, Tango did not make the annotation process faster then it was with Alembic, but it did reduce the number of mistakes made by the annotators.

New TLINKs can be created by drawing lines between events and time expressions. Any event or time expression can be selected to be the focus element, and dropping



```
@ Fell 37% in Third Quarter 10/26/89 WALL STREET JOURNAL (J) AMH EARNINGS (ERN)
COMPUTERS AND INFORMATION TECHNOLOGY (CPR) Amdahl Corp., a maker of mainframe computers, reported a sharp decline in net
income for its third quarter, citing pricecutting by competitors and adverse effects from a strong U.S.
dollar.
Net income fell 37% to $32.9 million, or 30 cents a share, from $52.2 million, or 48 cents a share, in the year-ago period.
Revenue rose 15% to $534.3 million from $464.7 million.
Amdahl's results were somewhat worse than expected.
Jay Stevens, an analyst with Dean Witter Reynolds, said he expected the Sunnyvale, Calif., company to earn 35 cents a
```

```
<EVENT class="OCCURRENCE" eid="e5">
```

Figure 2.4: Annotating with Tango

one or more objects to the left or right of the focus element automatically creates `before` or `after` links. Tango also provides an auto-arrange feature that lines up events with the time expressions they are anchored to. Tango also includes some code that automatically creates TLINKs between certain time expressions, mainly those with specific ISO values. At the time of this writing, user-prompting had not yet been incorporated.

# Chapter 3

# Temporal Closure

*We need closure on this.*

John Clippinger, entrepreneur

In chapter 1 it was suggested that we need to better manage the complexity of temporal annotation and that a mixed-initiative annotation environment including a closure component can increase annotation density, ensure consistency and boost inter-annotator agreement. This chapter introduces SputLink, a temporal closure module that is embedded in the TANGO annotation tool. SputLink ensures consistency and yields an annotation that is more precise and has higher coverage.

In section 3.1, I will first expand a bit on the holy grails of completeness and consistency as related to an annotation. Then I will describe approaches from temporal reasoning and temporal annotation on which this work is based. James Allen's

Interval Algebra will be discussed in section 3.2. Ways to constrain the interval algebra are presented in 3.3. Section 3.4 describes two annotation environments that use a version of temporal closure. Finally, SputLink is presented in sections 3.5 and 3.6, where the latter section focuses on how SputLink is embedded in an annotation environment.

## 3.1 Completeness and Consistency

The temporal annotation of a document should be as informative as possible. Ideally, we want reliable temporal relations between all events and time expressions. That is, we want each and every bit of information to be represented in the annotation and furthermore we want it represented correctly.

An annotation is a tuple $<N, R, L>$ where $N$ is a set of nodes consisting of the union of all events and time expressions, $R$ is the set of relation types from TimeML (`before`, `includes`, `simultaneous`, `unknown`, etc.), and $L$ is a set of TLINKs where each link is of the form $\langle x\ r\ y \rangle$ where $x, y \in N$ and $r \in R$. A small example of an annotation is listed in (14).

(14) $N = \{\ e_1, e_2, e_3\ \}$

$L = \{\ \langle e_1 \text{ before } e_2 \rangle\ ,\ \langle e_2 \text{ simultaneous } e_3 \rangle\ \}$

Completeness over a document means that each temporal object (event or time expression) in the document is temporally related to all other time objects. More formally:

(15) An annotation $<N, R, L>$ is complete iff for every pair $\langle x\ y \rangle \in N^2$ with $x \neq y$, there is a link $\langle x\ r\ y \rangle$ with r ∈ R.

Note that it is often not possible for the annotator to establish the temporal relation between two events. This does not necessarily prevent an annotation from being complete because *unkown* is in the set of relation types.

An annotation that is consistent does not contain conflicting relations. Imagine that an annotator added the following three links:

(16) ⟨x before y⟩

⟨y before z⟩

⟨x after z⟩

Clearly the first two relations clash with the third relation. It would be a boon for an annotation environment to have a module capable of catching such inconsistencies. A closure component is such a module. In the example in (16), the closure algorithm will generate the link ⟨x before z⟩ which is directly at odds with ⟨x after z⟩.

It should be noted that consistency is not the same as correctness. Take for example the following sentence and its set of links:

(17) She walked$^{e1}$ into the room, switched$^{e2}$ on the light and read$^{e3}$ a book.

$L = \{ \langle \text{e1 after e2} \rangle , \langle \text{e2 after e3} \rangle \}$

This annotation is consistent but incorrect. We can infer ⟨e1 after e3⟩ from the two links in (17), but this link does not conflict with any other link in $L$. But it is clear that the two links in (17) do not reflect the temporal ordering expressed in the sentences.

A few remarks on the terminology are in order before continuing. The terms *(temporal) relation* and *(temporal) link* will be used somewhat interchangeably, but *link* is used mostly in the context of the annotation task, whereas *relation* is more general. A *relation type* is either one of the relation type values from TimeML's TLINK or the set of relations types that features in a relation. This set may also be called the *temporal constraint* on the relation. The definition at the beginning of this section did not allow any kind of disjunction in the relation type, but soon we will see that interval algebra and point algebra allow disjunctions and that the closure component makes use of them.

## 3.2 Interval Algebra

Allen's interval calculus (Allen, 1983; Allen, 1984) has been very influential in the field of temporal reasoning. The starting point is the acknowledgment that there are thirteen basic temporal relationships between two intervals, as depicted in figure 3.1, which shows seven relations and the inverses of six of them.

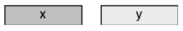
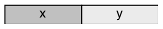
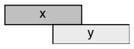
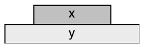
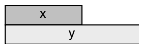
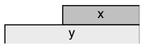
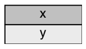
| Relation | Symbol | Inverse | Example |
|---|---|---|---|
| X before Y | $<$ | $>$ |  |
| X meets Y | m | mi |  |
| X overlaps Y | o | oi |  |
| X during Y | d | di |  |
| X starts Y | s | si |  |
| X finishes Y | f | fi |  |
| X equal Y | $=$ | $=$ |  |

Figure 3.1: The Thirteen Basic Relations

The temporal relations between intervals can be maintained in a graph where the nodes are the intervals and the arcs are labeled by arbitrary disjunctions over the thirteen basic relations. Allen assumes that the network always maintains complete information about how its intervals could be related. When a new temporal relation between two intervals is added, all consequences are generated by computing the transitive closure of the temporal relations. Each new fact adds a constraint about how its two intervals could be related, which may in turn introduce new constraints between other intervals through the transitivity rules governing the temporal relations.

A fragment of Allen's 13×13 transitivity table that models the transitive behavior of all relation pairs is given in table 3.1. The composition operator $\odot$ is used as another

| $\odot$ | $<$ | $>$ | d | di |
|---|---|---|---|---|
| $<$ | $<$ | all | $<$ o m d s | $<$ |
| $>$ | all | $>$ | $>$ oi mi d f | $>$ |
| d | $<$ | $>$ | d | all |
| di | $<$ o m di fi | $>$ oi di mi si | o oi d s f di si fi = | di |

Table 3.1: The transitive behavior of basic relations

way to denote lookup in the transitivity table: $r_1 \odot r_2$ is the cell $(r_1, r_2)$.

If a new fact ⟨i during j⟩ is added, and j is before k, then it is inferred from the table that i must be before k. The new constraint can be a disjunction, for instance, if the arc $\langle i, j \rangle$ is labeled $\{<\}$ and the arc $\langle j, k \rangle$ is labeled {d}, then the arc $\langle i, k \rangle$ can be constrained to the set {< o m d s}. In any case, the new fact is added to the network, possibly introducing further constraints on the relationships between other intervals. A slightly adapted version of Allen's constraint propagation algorithm is given in figure 3.2.

In this algorithm, `R(i,j)` is the new basic relation or set of basic relations just added between `i` and `j`, and `N(i,j)` is the existing set of basic relations between `i` and `j`. The propagation algorithm creates new arcs (or further constrains existing arcs) that bypass the two intervals on both sides of the newly added relation `R(i,j)`.

It is easy to see that the time complexity of Allen's algorithm is $O(N^3)$ where $N$ is the number of intervals. Adding one arc to the network is linear and the number of modifications that can be made is 13 times the number of binary relations between all nodes, which is $O(N^2)$.

```
Add R(i,j):
 add R(i,j) to ToDo
 while notEmpty ToDo do
     get next R(i,j) from ToDo
     N(i,j) := R(i,j)
     foreach node k do
         R(k,j) := N(k,j) ∩ Constraints(N(k,i),R(i,j))
         if R(k,j) ⊂ N(k,j) then
            add R(k,j) to ToDo
         R(i,k) := N(i,k) ∩ Constraints(R(i,j),N(j,k))
         if R(i,k) ⊂ N(i,k) then
            add R(i,k) to ToDo

Constraints(R1,R2):
 Result := ∅
 foreach r1 in R1 do
     foreach r2 in R2 do
         Result := Result ∪ r1 ⊙ r2
 return Result
```

Figure 3.2: Allen's Constraint Propagation Algorithm

A rather nasty problem with the procedure is that while it does not generate inconsistencies, neither does it detect all inconsistencies in its input. That is, it is sound but not complete. The constraint propagation never compares more than three arcs at a time, and there are temporal networks where each subgraph of three arcs is consistent but where there is no consistent labeling for the whole graph (see the example given in (Allen, 1983)). The algorithm becomes exponential when complete consistency checks are incorporated.

Also note that Allen's procedure was not written with temporal annotation in mind. The procedure assumes that constraints are continuously added in a monotonic fashion. But in a real-life annotation environment annotators can change their minds. Some kind of backtracking mechanism would need to be added to allow for this. A final observation here is that there is an obvious mapping from TimeML temporal relations to the basic relations used in Allen's interval algebra. This mapping will be exploited later.

## 3.3 Constraining Interval Algebra

The main problem with Allen's interval algebra is that it in no way restricts the labels on the arcs; any disjunction of basic relations is allowed. And it is not obvious how to introduce principled restrictions because the intervals and basic relations are treated as primitives.

Each interval can be represented as a pair of points where one precedes the other. For example, the interval X could be rewritten as $x_1$ - $x_2$, where $x_1$ is the begin point, $x_2$ is the end point and $x_1 < x_2$. All basic relations can also be rewritten using precedence and equality relations on begin and end points, as shown for a few of the basic relations in table 3.2.

| | |
|---|---|
| X before Y | $x_2 < y_1$ |
| X starts Y | $x_1 = y_1 \wedge x_2 < y_2$ |
| X during Y | $x_1 > y_1 \wedge x_2 < y_2$ |
| X overlap Y | $x_1<y_1 \wedge x_2>y_1 \wedge x_2<y_2$ |

Table 3.2: Mapping interval relations to point relations

James Allen acknowledged that intervals can be represented as pairs of points but claimed that certain logical inferences were easier to make in an approach that uses intervals as the primitives. In Allen's words (Allen, 1983):

> "A point-based representation is too uniform and does not facilitate structuring the knowledge in a way which is convenient for typical temporal reasoning tasks."

For example, we can build a hierarchy of `during` relations in which propositions can be inherited. Such a hierarchy also allows reasoning processes to be easily localized: if one is concerned with what is true today, then only those intervals that are during today need to be considered.

However, an undeniable advantage of the point-based approach is that principled restrictions on the set of possible arc labels can be made using a point algebra. In addition, using a restricted algebra makes inconsistency detection tractable. The next few sections describe the point algebra of Vilain, Kautz and van Beek and the idea of conceptual neighborhood as presented by Freksa.

### 3.3.1 Point Algebra

A tractable restricted subset of the interval algebra was proposed by Marc Vilain, Henry Kautz and Peter van Beek (Vilain et al., 1990). They used relations on points to restrict the $2^{13} = 8192$ different labels that interval algebra allows. The point algebra is defined by the four point relations between the beginning and end of two intervals. Any basic relation between intervals can be represented by defining the four relations R1 through R4 as shown in figure 3.3.

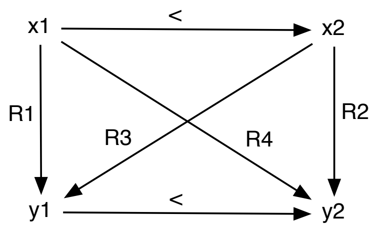


Figure 3.3: Decomposing an interval relation

The labels R1 through R4 on the point relations are taken from the set $\{< = >\}$, so instead of thirteen basic relations there are now only three. An interesting table emerges when all thirteen basic relations from the interval algebra are ordered according to the point relations assigned to the four relations above (see figure 3.4).

A convex relation is a relation between the four points where the following labels are allowed: {<}, {=}, {>}, {<=}, {>=}, and {<=>}. Convex relations map to disjunctions of interval relations, but not all disjunctions of interval relations can be expressed by a convex relation. For example, $x_2 <= y_1$ maps to the disjunction $\{< m\}$, but there is no convex relation that covers the disjunction $\{< si >\}$, as can easily be verified by inspecting figure 3.4. Ordering and restricting the unlimited disjunctions of Allen this way gives us a set of 82 convex relations. Schilder (1997)

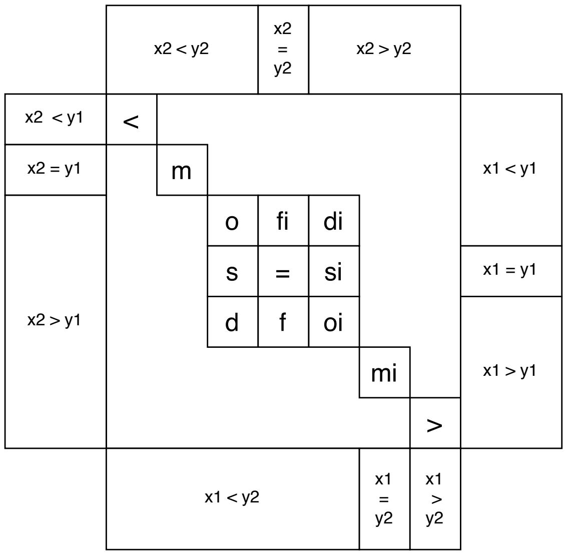


Figure 3.4: Interval relations and point relations

ordered these in a hierarchy based on the subset relation.

This point algebra of convex relations can be mapped to a subset of Allen's interval algebra by simply translating the point relation assignments to disjunctions of basic relations between intervals, using table 3.4. This interval algebra, with 82 rather than $2^{13}$ possible labels, has the property that detecting inconsistencies is now tractable. Indeed, (Vilain et al., 1990) proved that Allen's constraint propagation algorithm is sound and complete if the reduced set of labels is adopted.

The proof provides a model for the point algebra where the real numbers model time points and the relations $<$, $>$ and $=$ model the relations between time points. The constraint propagation algorithm[1] partitions the set of points in one or more

[1](Vilain et al., 1990) also proved that the constraint propagation algorithm and inconsistency detection are equivalent in the sense that there are polynomial mappings between them.

partial order graphs. The graphs contain clusters of points that are related by the `equal` relation. One characteristic of these graphs is that there are no cycles of precedence relations. As a result, each graph has one or more bottom nodes or clusters of nodes, that is, nodes that are preceded by no other nodes. The mapping to the model involves picking a bottom cluster and assigning a real number to it. The cluster is then removed from the graph and the process proceeds with a new real number greater than the first and a new bottom cluster. Eventually, all points are mapped to real numbers and the relations between points can be interpreted as precedence and equality relations between real numbers.

By the way, the intractability of inconsistency detection in the full interval algebra was proved NP-hard because there is a polynomial mapping of the 3SAT problem to inconsistency checking.

### 3.3.2 Conceptual Neighborhood

Christian Freksa (Freksa, 1992) proposes another subset of the interval algebra. He argues that Allen does not introduce a good mechanism for coarse temporal information because his disjunctions of basic relations are not at all restricted. In addition, Allen's representation and algorithm become more complex when less information is available on the arcs. Coarse temporal information is needed to properly describe indefinite temporal information in discourse, as exemplified in example (18) below.[2]

[2]This example was taken from (Schilder, 1997).

(18) "Mary stared$^{e1}$ at Peter. He gave$^{e2}$ her pizza back."

Event $e1$ can occur before $e2$, it can meet $e2$ or it can overlap with $e2$. Allen's scheme needs the disjunction $\{<\ \mathtt{m}\ \mathtt{o}\}$ to capture this information and requires a loop over the transitivity table to compute how constraints propagate through the network. To more concisely capture this kind of coarse temporal knowledge, Freksa introduced the notion of conceptual neighborhood. Two relations between intervals are conceptual neighbors if they can be directly transformed into one another by deforming the intervals (that is, shortening or lengthening), as in figure 3.5.

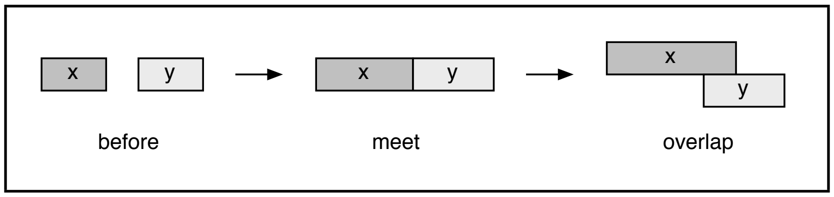


Figure 3.5: Deforming intervals

The `before` and `meet` relations are conceptual neighbors, but `before` and `overlap` are not because the transformation is indirect via the relation `meet`. The thirteen basic temporal relations can be ordered in a network according to their conceptual neighborhood. The resulting graph (figure 3.6) looks very similar to the table with point relation assignments in the previous section (figure 3.4).

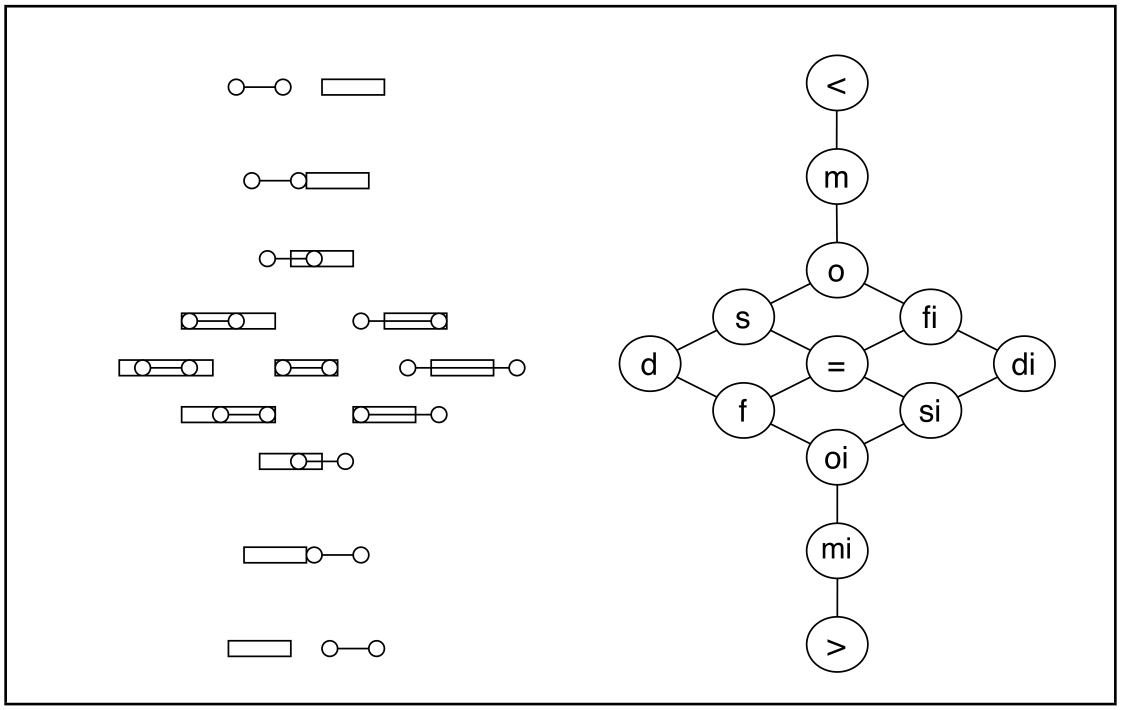


Figure 3.6: The Basic Relations in their Neighborhood

The lines between the relations represent the direct one-step transformations of the intervals. A conceptual neighborhood is defined as a set of relations that are path-connected through conceptual neighbor relations. For example, the set {< `m o fi =`} is a conceptual neighborhood but {< `o`} is not. Note that all convex relations are conceptual neighborhoods but that the reverse is not true. The figure above presents another way of defining convex relations. A convex relation *Rel* has a top element $r_1$ and a bottom element $r_2$ such that $Rel = \{r | r_1 \subseteq r \subseteq r_2\}$. So {`oi =`} is not a convex relation because, according to the definition above, `f` and `si` should also be included.

Freksa continues by identifying ten conceptual neighborhoods that are the basis for coarse temporal reasoning. He selected the neighborhoods in such a way that finer relations (the Allen relations) can be expressed as conjunctions of the coarse relations. Two examples of these neighborhoods are shown in table 3.3.

| label | mnemonic | Allen | point relations |
|---|---|---|---|
| tt | tail to tail with | fi = f | $x_2 = y_2$ |
| oc | older contemporary of | o fi di | $x_1 < y_1 \wedge x_2 > y_1$ |

Table 3.3: Two Neighborhoods

He then creates the transitivity table for these ten relations and shows that using this table generates the same inferences as Allen's transitivity table. The only difference is that Allen's algorithm creates disjunctions when reasoning over coarse information whereas Freksa's uses conjunctions when reasoning over fine information. Finally, Freksa creates a 29 $\times$ 29 table that is closed under neighborhood-based reasonings, that is, composition of any two of the 29 neighborhoods results in one of the 29 neighborhoods. These 29 relations are a subset of the 82 convex relations defined by (Vilain et al., 1990) and therefore the algebra inherits the tractability of the point algebra with convex relations.

## 3.4 Closure in an Annotation Environment

The quality of an annotated document and the quality of an annotation specification and its guidelines are often measured by comparing the annotations of two or three annotators. The complication is that, unlike for example with part-of-speech annotation, temporal annotations need to be compared at the semantic level and not the syntactic level. The two pairs of temporal networks in figure 3.7 should be considered pairwise identical because they convey the same meaning, even though they do not contain the same temporal facts.

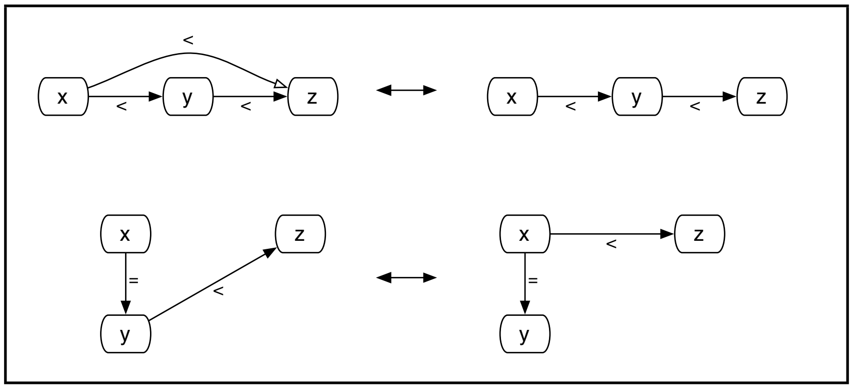


Figure 3.7: Two pairs of identical annotations

Some way is needed to compare temporal networks in a meaningful manner. In the next sections I describe one annotation effort that uses a model-theoretic approach and one that includes an explicit temporal closure component for the purpose of annotation comparison.

### 3.4.1 Model Theoretic Interpretation: Katz & Arosio

Graham Katz and Fabrizio Arosio (Katz and Arosio, 2001) propose a simple temporal annotation language for intra-sentential precedence and inclusion relations between verbs. The language has labels $<$ and $>$ for precedence relations and $\subseteq$ and $\supseteq$ for inclusion relations. Each sentence also includes an indexical reference $\circ$ to the speech time which can be temporally related to the verbs. Figure 3.8 is an example where the speech time and two verbs are temporally linked.

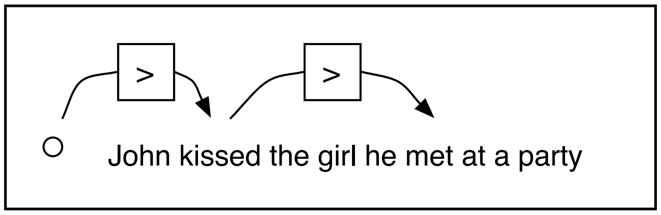


Figure 3.8: An intra-sentential annotation

Annotations are provided with a model theoretic interpretation: they are interpreted relative to a structure $D, <, \subseteq$ where $D$ is the set of verbs in the corpus and $<$ and $\subseteq$ are binary relations on $D$. Models for this structure are assignments of pairs from $D$ to $<$ and $\subseteq$, satisfying the following axioms:

(19) a. $\forall_{x,y,z}$: $x < y \wedge y < z \Longrightarrow x < z$

b. $\forall_{x,y,z}$: $x \subseteq y \wedge y \subseteq z \Longrightarrow x \subseteq z$

c. $\forall_{w,x,y,z}$: $x < y \wedge z \subseteq x \wedge w \subseteq y \Longrightarrow z < w$

d. $\forall_{w,x,y,z}$: $x < y \wedge y < z \wedge x \subseteq w \wedge z \subseteq w \Longrightarrow y \subseteq z$

Consistency and inconsistency of annotations are now defined in terms of satisfiability in models. In addition, Katz and Arosio use the notions of minimal model and distance measure. They report results from an experiment where two annotators annotated 50 complex sentences. The annotations were syntactically identical in only 70% of the cases, but they were semantically consistent in 85% of the cases.

### 3.4.2 Temporal Closure in STAG

Andrea Setzer and Robert Gaizauskas aimed to capture temporal information in newswire texts. To that end, they defined an annotation language and a set of annotation guidelines called STAG: Sheffield Temporal Annotation Guidelines (Setzer, 2001; Setzer and Gaizauskas, 2001). Events and times are connected using five temporal relations: `before`, `after`, `includes`, `included` and `simultaneous`. The first four have the expected interpretations, although it should be noted that `before` is similar to Allen's label {< `m`} and `after` to {> `mi`}. The fifth relation, `simultaneous`, is the fuzziest and means something like "roughly at the same time", which can be considered similar to Allen's label {`o s fi d = di f si oi`} or the point relation assignment { $x_1 < y_2$ , $x_2 > y_1$ }.

As hinted at before, Setzer and Gaizauskas used a deductive closure component to get more reliable inter-annotator agreement figures. They use a domain $E \cup T$ (events and time expressions) which has three binary relations defined on it. B, I and

S are the sets of all pairs in the domain that are assigned the `before`, `includes` and `simultaneous` relations respectively. A normalization step was included where `after` and `included` are mapped onto `before` and `includes` by taking the inverse of the relation. As a result, the set of inference rules was smaller and the process clearer. Ten inference rules were derived from the formal properties of the STAG relations: `simultaneous` is an equivalence relation while `before`, `includes` and their inverses are transitive, asymmetric and irreflexive. The inference rules are:

(20)
1. $\forall x, y, z \in (E \cup T) : (x, y) \in S \Rightarrow (y, x) \in S$
2. $\forall x, y, z \in (E \cup T) : (x, y) \in B \wedge (y, z) \in B \Rightarrow (x, z) \in B$
3. $\forall x, y, z \in (E \cup T) : (x, y) \in I \wedge (y, z) \in I \Rightarrow (x, z) \in I$
4. $\forall x, y, z \in (E \cup T) : (x, y) \in B \wedge (y, z) \in I \Rightarrow (x, z) \in B$
5. $\forall x, y, z \in (E \cup T) : (x, y) \in I \wedge (x, z) \in B \Rightarrow (y, z) \in B$
6. $\forall x, y, z \in (E \cup T) : (x, y) \in S \wedge (y, z) \in S \Rightarrow (x, z) \in S$
7. $\forall x, y, z \in (E \cup T) : (x, y) \in B \wedge (y, z) \in S \Rightarrow (x, z) \in B$
8. $\forall x, y, z \in (E \cup T) : (x, y) \in I \wedge (y, z) \in S \Rightarrow (x, z) \in I$
9. $\forall x, y, z \in (E \cup T) : (x, y) \in S \wedge (y, z) \in I \Rightarrow (x, z) \in I$
10. $\forall x, y, z \in (E \cup T) : (x, y) \in B \wedge (x, z) \in S \Rightarrow (z, y) \in B$

Now standard precision and recall measures can be applied to the domain after computing the deductive closure of B, I and S.

Measuring the inter-annotator agreement was Setzer and Gaizauskas' main application for the deductive closure component, but they proceeded to use it to increase the number of temporal facts in a text. They introduced two stages of annotation. In the first stage, the annotator manually marks up explicit and implicit temporal relations in the text. In the second stage, relations are normalized and all inferences that can be drawn are added to the set of relations. Then the system enters a loop where the user is prompted to specify the relation between two events and time expressions that have not yet been related. Each time the user adds a fact, the closure component tries to add new inferences. This loop continues untill all relations are specified.

### Discussion

The first thing to notice is that the inferencing approach is not sound due to the fuzzy nature of the `simultaneous` relation, rules 6 through 10 can actually derive incorrect facts. Setzer acknowledges this problem with her inferencing approach in her thesis and gives an example that illustrates it, here presented in figure 3.9.

Event x is before event y and event y is simultaneous with event z ("roughly at the same time"), yet event x is not before event z, thereby violating rule 7. These incorrect inferences are not necessarily a problem when the closure component is

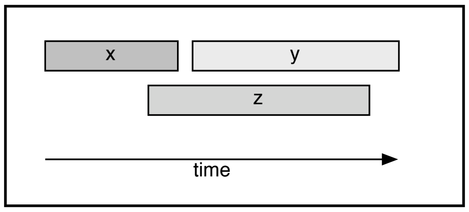


Figure 3.9: An incorrect inference in Setzer's closure algorithm

used to more correctly measure inter-annotator agreement. But if temporal closure is used to achieve a complete annotation, then it should not be at the price of lower precision.[3]

Another feature of this closure component is that the set of inference rules is not complete. To see this, let's put the ten inference rules in a transitivity table similar to the ones used by Allen and Freksa. First we need to map Setzer's relations to disjunctions of basic relations. This mapping is shown in table 3.4.[4]

We can now create the 5 $\times$ 5 transitivity table in 3.5 and plot the 10 inference rules.

Some care needs to be taken since relations were normalized in order to reduce the set of rules. In the table above, the cells contain an inference rule number and

[3]This begs the empirical question of how often false inferences are drawn. A small number of these false hits and slightly lower precision may be acceptable as long as there would be a significant increase in recall.

[4]This mapping is open to debate. One could argue that the `includes` relation should not be narrowed down to the inverse of Allen's `during` relation, but that it should include non-proper inclusion relations like `starts`. Changing the interpretation this way does not significantly change the point I'm making.

| relation | shorthand | disjunction |
|---|---|---|
| before | B | $<$ m |
| after | $B_i$ | $>$ mi |
| includes | I | di |
| included | $I_i$ | d |
| simultaneous | S | oi fi di si = s d f o |

Table 3.4: Mapping STAG relations to basic relations

| $\odot$ | B | I | S | $I_i$ | $B_i$ |
|---|---|---|---|---|---|
| B | 2: B | 4: B | 7: B | | |
| I | | 3: I | 8: I | | |
| S | $10_r$: B | 9: I | 6: S | $8_{lr}$: $I_i$ | $7_{lr}$: $B_i$ |
| $I_i$ | $5_l$: B | | $9_{lr}$: $I_i$ | $3_{lr}$: $I_i$ | $4_{lr}$: $B_i$ |
| $B_i$ | | $5_r$: $B_i$ | $10_l$: $B_i$ | | $2_{lr}$: $B_i$ |

Table 3.5: Transitivity table for STAG relations

the value of the composed relation. For example, in cell (B,B) we find rule 2, which created a new `before` relation. Some rules have subscripts indicating that the left (l) and/or right (r) needed to be inversed so the rule fits into the appropriate slot in the table. For example, rule 8 fit in the (I,S) box with no problems, but for it to fit in (S,$I_i$) relations had to be inversed, as shown in (21) below.

(21) Rule 8: $(x, y) \in I \wedge (y, z) \in S \Rightarrow (x, z) \in I$

Rule $8_{lr}$: $(z, y) \in S \wedge (y, x) \in I_i \Rightarrow (z, x) \in I_i$

The transitivity table is not completely filled in and not all possible inferences are being made. Some compositions will not derive any information. For example, all basic relations are possible when B and $B_i$ are composed. And some compositions result in the kind of coarse information that Setzer and Gaizauskas are not dealing with. For example, when I an B are composed, the result is the disjunction {`fi di o m <`} (which Freksa labeled `ol`, with mnemonic `older`). Using the available labels from STAG would create an incorrect inference, which should be avoided for the non-fuzzy relations. There seems to be only one genuine omission from the list of inference rules: composing I and I$_i$ generates S, which ought to be added as an extra rule:

(22) $\forall x, y, z \in (E \cup T) : (x, y) \in I \wedge (z, y) \in I \Rightarrow (x, z) \in S$

Nevertheless, the approach of Setzer and Gaizauskas inspired the work developed in this dissertation. The remainder of this chapter describes how to use temporal closure to achieve a consistent, high-quality, and high-coverage annotation.

## 3.5 Launching SputLink

The temporal closure module SputLink is an implementation of Allen's interval algebra but it restricts the set of possible labels using insights from point algebra. It is embedded in TANGO (Pustejovsky et al., 2003b), a graphical annotation tool for temporal annotation using TimeML (Pustejovsky et al., 2003a).

It is assumed that all TimeML relations can be mapped onto Allen relations and relations between points. A translation of all TimeML relations is given in table 3.6. Note that there is no need for a mapping to the basic relations `o` and `oi` since TimeML has no overlap relation. In addition, TimeML relations are not intended to be as

| TimeML relation | Allen relation | relations between points |
|---|---|---|
| A before B | $<$ | $a_2 < b_1$ |
| A after B | $>$ | $a_1 < b_2$ |
| A ibefore B | m | $a_2 = b_1$ |
| A iafter B | mi | $b_2 = a_1$ |
| A includes B | di | $a_1 < b_1 \wedge a_2 > b_2$ |
| A is_included B | d | $a_1 > b_1 \wedge a_2 < b_2$ |
| A identity B<br>A simultaneous B<br>A holds B<br>A held_by B | $=$ | $a_1 = b_1 \wedge a_2 = b_2$ |
| A begins B | s | $a_1 = b_1 \wedge a_2 < b_2$ |
| A begun_by B | si | $a_1 = b_1 \wedge a_2 > b_2$ |
| A ends B | f | $a_1 > b_1 \wedge a_2 = b_2$ |
| A ended_by B | fi | $b_1 > a_1 \wedge a_2 = b_2$ |

Table 3.6: Mapping TimeML relations to basic relations

precise as Allen relations. There is a certain amount of fuzziness built into some of the relations, although this fuzziness is not even close to the fuzziness of STAG's `simultaneous` relation. As a result, a TimeML closure engine that uses the precise relations behind the scenes may introduce incorrect links in a similar way to some of the inference rules of Gaizauskas and Setzer. How often this happens is an empirical question. By the way, it is not trivial to translate back from Allen relations to TimeML relations since there are four relations that are mapped onto the = relation: `identity`, `simultaneous`, `holds`, and its inverse `held_by`. This issue will be taken up in section 3.5.3.

Allen's interval algebra is restricted in much the same way as portrayed in section 3.3. A restricted set of labels is adopted and a transitivity table constructed that compiles out all possible compositions of the allowed relations.

To run the algorithm we need to map an annotation graph with TimeML objects onto a graph with intervals and relations between intervals. This amounts to reducing events and times to intervals. In other words, we abstract away from all properties of events and times and view them as time intervals only. The algorithm treats events and times the same. No ontological difference is made between anchoring an event to a time on the one hand and ordering an event with respect to another event on the other. Both anchoring and ordering are simply seen as a temporal relation between time objects and the algorithm takes these temporal relations as its input. It is trivial to re-introduce the distinction after the algorithm runs. For example, the TANGO tool displays times separately from events in its display and intuitively the temporal relations between events and times can be interpreted as anchors, rather than orderings.

The core SputLink constraint propagation algorithm is presented in figure 3.10. It is very similar to Allen's algorithm in figure 3.2. The main difference is that there is no `Constraints` procedure that loops over a $13 \times 13$ composition table of basic relations but a single lookup $\odot$ in a $29 \times 29$ composition table of convex relations. The creation of this table will be discussed presently.

Consistency checking proceeds as follows. Each time an additional constraint is proposed between $N_i$ and $N_j$, it is checked whether the intersection of the new

```
Add R(i,j):
 add R(i,j) to ToDo
 while notEmpty ToDo do
     get next R(i,j) from ToDo
     N(i,j) := R(i,j)
     foreach node k do
         R(k,j) := N(k,j) $\cap$ $\odot$(N(k,i),R(i,j))
         if R(k,j) $\subset$ N(k,j) then
            add R(k,j) to ToDo
         R(i,k) := N(i,k) $\cap$ $\odot$(R(i,j),N(j,k))
         if R(i,k) $\subset$ N(i,k) then
            add R(i,k) to ToDo
```

Figure 3.10: SputLink's Constraint Propagation Algorithm

constraint with the old constraints generates the empty set. If so, there is no possible relation type for $\langle N_i \; N_j \rangle$ and an inconsistency error is raised. The complexity of the algorithm is identical to the complexity of Allen's algorithm. It runs in polynomial time with a time complexity of $O(n^3)$ and a space complexity of $O(n^2)$.

### 3.5.1 Generating the Transitivity Table

As shown above, each TimeML relation can be translated into a set of precedence and equality statements between points-in-time. We can actually set up closure by simply reasoning over these points and the $<$, $>$ and $=$ relations between them. The algorithm in figure 3.10 can be applied using the simple transitivity table in 3.7.

In SputLink all reasoning occurs on the interval level. However, point-based reasoning is used to create the transitivity table. There are two steps in the procedure for generating the table: (i) identify all possible relations, and (ii) identify the com-

| $\odot$ | $<$ | $=$ | $>$ | ? |
|---|---|---|---|---|
| $<$ | $<$ | $<$ | ? | ? |
| $=$ | $<$ | $=$ | $>$ | ? |
| $>$ | ? | $>$ | $>$ | ? |
| ? | ? | ? | ? | ? |

Table 3.7: Transitivity table for point relations

position for all relation pairs.

**Identify all possible relations**

Table 3.8 shows how all configurations of point relations for two intervals $x_1$ - $x_2$ and $y_1$ - $y_2$ can be enumerated using $<$, $=$, $>$ and ? as allowed labels on point relations.

| | $x_1$ | $x_2$ | $y_1$ | $y_2$ |
|---|---|---|---|---|
| $x_1$ | $=$ | $<$ | {<,>,=,?} | {<,>,=,?} |
| $x_2$ | $>$ | $=$ | {<,>,=,?} | {<,>,=,?} |
| $y_1$ | {<,>,=,?} | {<,>,=,?} | $=$ | $<$ |
| $y_2$ | {<,>,=,?} | {<,>,=,?} | $>$ | $=$ |

Table 3.8: Configurations of point relations

Two points-in-time can be equal, one point can occur before the other, or the relation between the two points can be unknown. The top-left and bottom-right quadrants are filled in by virtue of the properties of an interval, the top-right and bottom-left quadrants are mirror images of each other. So there are $|\{<, >, =, ?\}|^4 = 256$ possible configurations of relations between begin and endpoints of two intervals.

Every one of these assignments can be seen as a graph with 4 nodes and 6 edges. The assignment is well-formed if running the closure algorithm using the $4 \times 4$ transitivity table does not derive an inconsistency error. There are 96 well-formed configurations of point relations between $x_1$ - $x_2$ and $y_1$ - $y_2$. Two assignments are equivalent if they are identical under closure. There are 29 equivalence classes, each corresponding to a disjunction of basic interval relations as well as to an assignment of point relations. Allen's thirteen basic relation types are amongst these 29; the other sixteen are shown in figure 3.11.[5]

These 29 well-formed minimal configurations are a subset of the 82 convex relations defined by (Vilain et al., 1990). The difference is that Vilain et al allowed {<=} and {>=} as labels on the point relations, thereby allowing many more assignments than in the above approach. Interestingly, the 29 relations between intervals are the same that Freksa (1992) identified.

[5]The names and icons of these coarse relations are adopted from (Freksa, 1992). The icons are small versions of the graph in figure 3.6. For example, the relation{< m o} is depicted with three solid dots sitting on top of a thin circle, which itself sits on a thin vertical bar.

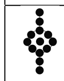
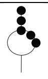
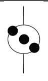
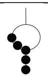
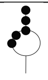
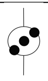
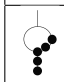
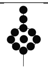
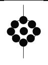
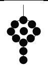
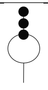
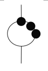
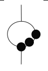
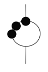
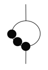
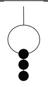
| icon | name | mnemonic | Allen relations | point constraints |
|---|---|---|---|---|
|  | ? | | all | |
|  | ol | older | < m o fi di | $x_1 < y_1$ |
|  | hh | head to head with | si = s | $x_1 = y_1$ |
|  | yo | younger | d f oi mi > | $x_1 > y_1$ |
|  | sb | survived by | < m o s d | $x_2 < y_2$ |
|  | tt | tail to tail with | fi = f | $x_2 = y_2$ |
|  | sv | survives | di si oi mi > | $x_2 > y_2$ |
|  | bd | born before death of | < m o fi di si = s d f oi | $x_1 < y_2$ |
|  | ct | contemporary of | o fi di si = s d f oi | $x_1 < y_2 \wedge x_2 > y_1$ |
|  | db | died after birth of | o fi di si = s d f oi mi > | $x_2 > y_1$ |
|  | ob | older and survived by | < m o | $x_1 < y_1 \wedge x_2 < y_2$ |
|  | oc | older contemporary of | o fi di | $x_1 < y_1 \wedge x_2 > y_1$ |
|  | sc | surviving contemporary of | di si oi | $x_1 < y_2 \wedge x_2 > y_2$ |
|  | bc | survived by contemporary of | o s d | $x_2 > y_1 \wedge x_2 < y_2$ |
|  | yc | younger contemporary of | d f oi | $x_1 > y_1 \wedge x_1 < y_2$ |
|  | ys | younger and survives | oi mi > | $x_1 > y_1 \wedge x_2 > y_2$ |

Figure 3.11: Sixteen Coarse Relations

Figure 3.12 shows how the relations between intervals can be plotted in a hierarchy by using the subset relation.

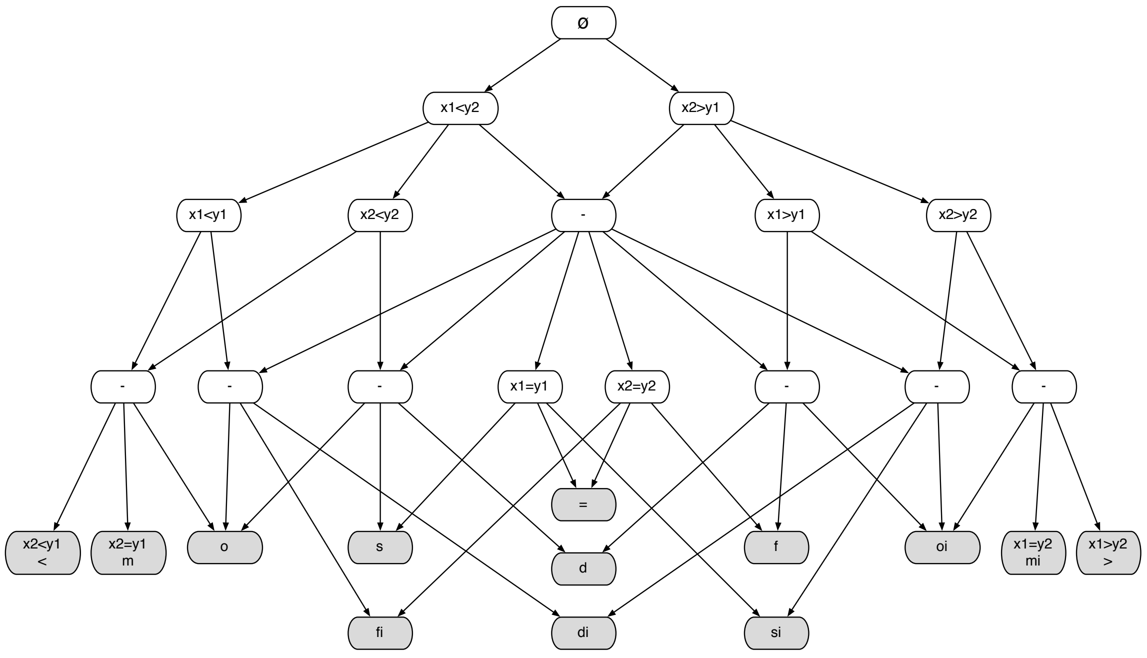


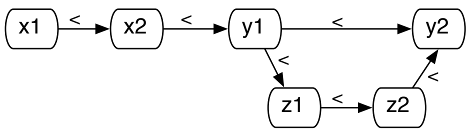
Figure 3.12: The hierarchy of relations

For display purposes, not all point relations for a node $N$ are printed in the picture above, only the ones that are not included in one of the nodes that subsume $N$ are presented. For example, the node $\{x2 > y2\}$ can also be written as $\{x2 > y2\ ,\ x2 > y1\}$. There are five layers in the lattice, corresponding to relations between intervals where 0, 1, 2, 3 or 4 relations between points are known. The bottom layer presents those relations where all relations between begin and endpoints are specified. Under-specification increases when going up the lattice. The nodes at the bottom layer map to Allen's basic relations, whereas nodes on higher layers map to a disjunction of relations. Note how the bottom layer resembles the conceptual neighborhood graph in section 3.3.2. The hierarchy is similar to, yet smaller than the hierarchy presented in (Schilder, 1997).

**Identify the composition of all relation pairs**

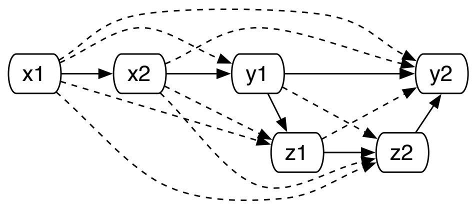
Having established that SputLink's transitivity table contains 29 rows and columns, it is time to fill in the $29^2 = 841$ cells. This can again be done by simply using the algorithm in figure 3.10 with the $4 \times 4$ transitivity table for $<$, $>$, $=$ and ?. Each relation pair creates its own little graph, as depicted below for the relations [ $x_2 < y_1$ ] and [ $y_1 < z_1$, $z_2 < y_2$ ].

(23) Graph for $\langle$X $<$ Y$\rangle$ and $\langle$Y di Z$\rangle$



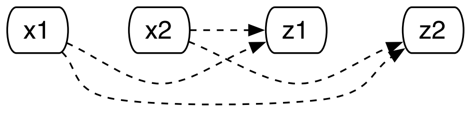
Now we simply run transitive closure on all the relation pairs. The result is in (24), where links derived by closure are dotted and all edge labels have been omitted (edges are implicitly labeled $<$).

(24) Graph after closure



Finally, we need to figure out the new constraints between $x_1$ - $x_2$ and $z_1$ - $z_2$. This amounts to a simple mapping from the point relations between $x_i$ and $z_j$ to one of the 29 relations. To continue the running example, below is a graph with just $x_1$ - $x_2$ and $z_1$ - $z_2$ and all the relations between the begin and end points.

(25) Graph after $y_1$ - $y_2$ and associated links have been removed



This configuration can be mapped onto one of the 29 equivalence classes and by extension onto a TimeML relation or a disjunction of TimeML relations. In the example we started with the relations [ $x_2 < y_1$ ] and [ $y_1 < z_1$ , $z_2 < y_2$ ], which can be mapped onto the TimeML relations `before` and `includes` respectively. After closure we created a new relation [ $x_1 < z_1 \wedge x_1 < z_2 \wedge x_2 < z_1 \wedge x_2 < z_2$ ], which is identical to $x_2 < z_1$ and can be mapped onto the TimeML relation `before`. In TimeML terms, we have now generated the inference rule

$$\texttt{X before Y} \wedge \texttt{Y includes Z} \Longrightarrow \texttt{X before Z}$$

All cells of the 29 × 29 transitivity table can be filled in a similar way. Of the 841 compositions, 638 do actually constrain a relation beyond assigning the disjunction of all basic relations. Only about 20% of these are expressed as a single basic relation.

### 3.5.2 Adding Points

Taking an interval-based approach assumes that intervals are the primitives for the purpose of temporal closure over the annotation. Allen originally claimed that even events or time expressions that look like points-in-time can in fact be treated as very short intervals and that interval-based reasoning was more efficient than point-based reasoning. Antony Galton (Galton, 1990) argued that the neglect of time instants results in a formalism that is too crude for representing facts about continuous change. To amend that, points and intervals need to be treated on equal footing.

What matters for the present purpose is which inferences can be drawn between nodes, not the ontological status of those nodes. And it should be noted that some inferences may be missed if no distinction between points and intervals can be made. For example, let's look at the annotation in figure 3.13 and the inferences it should allow. Here, L is the set of links added by the annotator. L, represented graphically

They had a massive **lunch**$^{e1}$ from 13.00$^{t1}$ till 16.00$^{t2}$, as always they started with **deviled eggs**$^{e2}$.

L = { ⟨t1 begins e1⟩ , ⟨t2 ends e1⟩ , ⟨e2 begins e1⟩ }

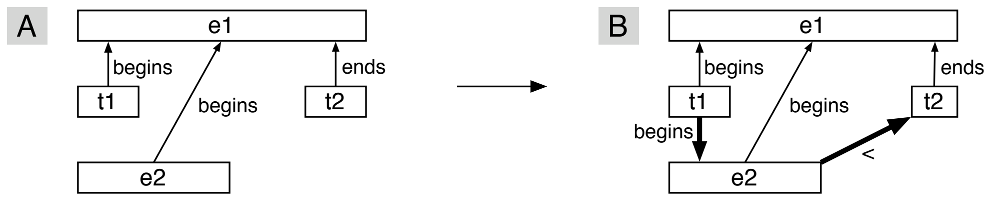


Figure 3.13: Potentially missed inferences

in A, should ideally allow the inferences shown in B. But if we cannot assume that t1 and t2 are points, then the links ⟨t1 begins e2⟩ and ⟨e2 before t2⟩ (shown in B with thicker arrows) cannot be derived, although in both cases a coarser label can be derived. It is not clear how many inferences are actually being missed because of this. But it is clear that using points and intervals or introducing some degree of quantitative reasoning would help in finding these inferences.

SputLink as described above has no concept of points but the 29 × 29 composition table can easily be expanded to allow for temporal relations between points and intervals and relations between points. For example, to take care of point-interval relations we can take the square in figure 3.3 with four relations between points and reduce it to the triangle in figure 3.14 with two point relations.

We can then create eight convex relations between point and interval: five basic ones (`before`, `starts`, `in`, `ends`, and `after`), and three disjunctions. These disjunc-

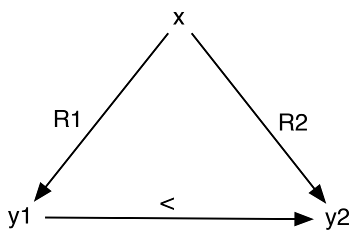


Figure 3.14: Decomposing a point-interval relation

tions are: {`before,starts,in`}, for when the point comes before the end point of the interval, {`in,ends,after`}, for when the point comes after the begin point of the interval, and {`before,starts,in,ends,after`}, the totally underspecified relation. In addition, there are eight relations between intervals and points (the inverses of the eight relations above) and four relations between points ($<$, $>$, `=`, and `?`).

The composition table as described in section 3.5.1 composes a new relation $\langle$X r Z$\rangle$ from two existing relations $\langle$X $r_1$ Y$\rangle$ and $\langle$Y $r_2$ Z$\rangle$. But it assumes that X, Y and Z are all intervals and therefore we can simply write $\odot(r_1, r_2) = r$ and ignore the types of the temporal objects that are related. The rows and columns of the extended composition table are not simply labeled by relations anymore but by tuples $\langle$x r y$\rangle$, where x is the type (point or interval) of the first argument of the relation, y is the type of the second argument of the relation, and r is the relation type between x and y. The extended composition table has 49 rows and columns rather than 29 since in addition to the 29 interval-interval relations, it also has to consider 8 interval-point relations, 8 point-interval relations and 4 point-point relations.

The cells of this extended $49 \times 49$ table can be filled in the same way as the $29 \times 29$ composition table for interval-interval relations. Note however that half the cells are not filled in due to type mismatches. For example, a temporal relation between a point and an interval can never be composed with a temporal relation between two points.

The composition operator $\odot$ in the SputLink algorithm of figure 3.10 can now be made sensitive to the types of the temporal objects involved and missed inferences as in figure 3.13 can be dealt with.

### 3.5.3 From Intervals to Events and Timexes

Recall that there is no one-to-one mapping from TimeML relation types to Allen's basic relations. The basic relation `=` can be mapped onto `identity`, `simultaneous`, `holds` or `held_by`, written here as $=^i, =^s, =^h$ and $=^{hb}$ respectively. Now which TimeML relation should be assigned to a relation type if the closure component generates a constraint on a temporal link that includes Allen's `equal` relation?

The easiest solution would be to simply leave the TimeML link underspecified and use a disjunction of the four relation types above. But this is not acceptable because TimeML relation types like `identity`, `simultaneous` and `holds` are not intended to be ambiguous, and if a more specific inference can be made then it should be made. This is for example possible if we have $\langle \mathrm{x} =^i \mathrm{y} \rangle$ and $\langle \mathrm{y} =^s \mathrm{z} \rangle$. We know that this derives $\langle \mathrm{x} =^s \mathrm{z} \rangle$, and a disjunction of TimeML relation types would not capture this.

Another solution is to expand the transitivity table to one that is TimeML friendly. Each cell $(r_1, r_2)$ where $\odot(r_1, r_2)$ includes $=$ has to be replaced by a 4 by 4 table compiling out all pairwise compositions of elements from $\{ =^i , =^s , =^h , =^{hb} \}$.[6] This is possible but rather clunky and not very intuitive. This can be exemplified by building the $4 \times 4$ table in 3.9 that would replace the contents of the cell (=,=).

| $\odot$ | $=^i$ | $=^s$ | $=^h$ | $=^{hb}$ |
|---|---|---|---|---|
| $=^i$ | $=^i$ | $=^s$ | $=^h$ | $=^{hb}$ |
| $=^s$ | $=^s$ | $=^s$ | $=^h$ | (a) |
| $=^h$ | $=^h$ | (b) | $\emptyset$ | $=^s$ |
| $=^{hb}$ | $=^{hb}$ | $=^{hb}$ | $=^s$ | $\emptyset$ |

Table 3.9: Composing the equality relations

What is disturbing is that some cells cannot have a value and that the values of other cells depend on the type of intervals that are linked. The $\emptyset$ in two of the cells reflects the fact that it is impossible to have these configurations of temporal relations. For example ⟨x holds y⟩ implies that y is a timex, but ⟨y holds z⟩ implies that y is an event. Some cells have a value that is determined by the type of the intervals linked by the relation. Let $x^e$ mean that the interval $x$ is an event and $x^t$ that $x$ is a timex. Now cell (a) can be filled in as in (26) and (b) as in (27).

[6]This assumes that we do not need to replace $=$ with all non-empty subsets of $\{ =^i , =^s , =^h , =^{hb} \}$.

(26) $=^{hb}$ if $\langle x^t =^s y^t \rangle$ and $\langle y^t =^{hb} z^e \rangle$

$=^s$ if $\langle x^e =^s y^t \rangle$ and $\langle y^t =^{hb} z^t \rangle$

(27) $=^h$ if $\langle x^e =^h y^t \rangle$ and $\langle y^t =^s z^t \rangle$

$=^s$ if $\langle x^t =^h y^t \rangle$ and $\langle y^t =^s z^e \rangle$

The best solution is to separate the closure component as much as possible from the component that translates back to TimeML relation types. The thing to realize is that the closure component reduces events and timexes to intervals whose only characteristics are their begin and end points, and that the choice of TimeML relation type depends on factors beyond the position of the events or timexes in the time partial order. Which TimeML relation to take depends on (i) determination of whether the input relations were proper TimeML relation, and (ii) the types of intervals that are linked.

The `identity` relation from TimeML is not really a temporal relation, rather it marks that two objects in the text refer to the same event or time. It was included in TimeML for pragmatic reasons. For the temporal structure of a document, it is important to know that two text extents refer to the same event or time, especially if, in the long run, we're going to tie in the argument structures of events. If `identity` is not a real temporal relation then perhaps it should not be treated as such. It is important to make a distinction between text extents and temporal objects. In the text below, the event-denoting noun *agreement* occurs twice, referring to the exact same event.

(28) Poughkeepsie said it is continuing to try to sell itself, under a June **agreement** with a dissident-shareholder group. The bank also said its effort would continue past the Nov. 1 deadline set in that **agreement** and that the litigation between the two sides might resume as a result.
*Text taken from TimeBank, article wsj_0736.*

An annotator can identify this identity with a TLINK, but should then be able to view the two linked events as one event, and only add links from other events to one of them. The temporal closure module should also keep the two levels apart and not try to deal with compositions that involve an identity link, but instead collapse two nodes that are linked by TimeML's `identity` relation. An additional benefit here is that temporal closure will speedup somewhat because the number of nodes gets smaller.

The situation with `simultaneous` and `holds` is an altogether different animal. The choice between these two relation types is determined by the events and timexes that are linked by the temporal relation. `Simultaneous` is perceived as a relation between events and `holds` and `held_by` are relations between an event and a timex: an event `holds` during a timex (where the timex is an interval and not a date), and a timex is `held_by` and event. Each time a constraint is created that includes an =, it is simply a matter of checking what types of intervals are linked and choosing the appropriate type.

| | event | timex |
|---|---|---|
| event | simultaneous | holds |
| timex | held_by | simultaneous |

Note that the annotator needs to be constrained in the same way. For example, he or she should not be allowed to draw a `holds` relation type between two events.

## 3.6 Embedding SputLink

SputLink's closure algorithm does not run in a vacuum, but rather is embedded in a mixed-initiative temporal annotation environment. This section explores some of the issues about how the algorithm interacts with other components and how the algorithm can be employed to achieve consistency and (near) completeness.

There are basically two ways for the algorithm to be embedded: (i) as a separate stage in the annotation and (ii) as a process that constantly runs in the background.

In the first case, manual or automatic annotation occurs before any activity from the temporal closure component, which runs in a separate stage afterwards. This is the approach taken by (Setzer, 2001). She also introduced a user-prompting stage where the user is asked to fill in a relation type for a link that has none, this is followed by another application of the closure component. The cycle continues till all event/timex pairs are visited. Note that the first time that closure runs, it may discover inconsistencies.

In the second case, temporal closure runs each time a temporal relation gets added or is further constrained. This has the advantage that the annotation is guaranteed to be consistent at any time, but it may not always be possible to use this mode (for example, when some temporal links are generated automatically in a pre-processing stage).

### 3.6.1 User-Prompting

Closure by itself does not guarantee anything near completeness. Consider the strategy where a manual annotation stage is followed by application of the closure module. This strategy was used in the creation of the TimeBank corpus. TimeBank annotators typically mark up only 1% of all possible temporal links, and closure ramps this up to just over 5%.[7] User-prompting as used by (Setzer, 2001) does guarantee completeness. But unfortunately user-prompting is potentially quadratic to the number of events and timexes in the document and therefore becomes unbearable for all but the shortest documents and most unflappable annotators.

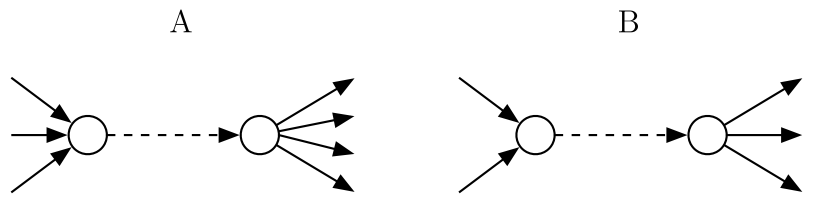
SputLink includes a routine that identifies which temporal facts are not yet known. The annotator is prompted with one of these unspecified links and asked to fill in

[7]See chapter 4 for more complete statistics.

the relation type, that is, the annotator creates a link. This new link is added to the queue and transitive closure starts again. Only when all links are specified (possibly with a relation type `unknown`) will the annotator be left in peace.

Prompting does not need to be performed randomly. The prompting order could reflect textual occurrence or derivational potential. The first simply means that prompting tries to follow the annotator's strategy. Annotators tend to move through a text sequentially, adding events, timexes and temporal links sentence by sentence. However, it is pretty much impossible for a prompting module to stick to this strategy since at least some unspecified links that are prompted for need to be massively global. That is, they link events from remote corners of the document.

The second approach is to let prompting reflect derivational potential. The user-prompting module would maintain a set of all time object pairs that do not have a temporal link defined between them or a link with an underspecified relation type. And for each time object the module would maintain how many links it participates in. When the user has to be prompted, it will be with a pair of time objects where the individual time objects participate in the most links. Let's look at an example. The picture below shows two fragments of an annotation graph. The dotted arrows represent links with an underspecified relation type or no relation type at all.



In case A, letting the user fill in the prompted link would allow a total of 3 + 4 = 7 compositions (that is, look ups in the transitivity table); in case B, it would allow 2 + 3 = 5 compositions. So the derivational potential is higher for case A and hence the prompting mechanism should present this one first.

Weights can be assigned to each composition using the transitivity table for the 29 relation types. The idea is to count the number of information carrying relations (that is, relations that are not completely underspecified) in each row and divide them by 29. Take for example the situation where we have a link ⟨x < y⟩ and a proposed link for prompting ⟨y z⟩. We now check all the values in the transitivity table next to < and find 23 cells where the composition yields a non-? relation type, so we arrive at a weight of 23/29 = 0.79. This means that in 79% of the cases, a new constraint is generated if we compose ⟨x < y⟩ and ⟨y r z⟩.[8]

Some compositions generate more specific constraints than others, and this can be reflected in the weights by adding up the cardinalities of the weights in a row. Let $r_1$ through $r_{29}$ be the 29 relation types. Now the weight for a relation r can be

[8]This assumes a random distribution of relation pairs, which is most certainly not the case. The weights can be further adjusted to account for this.

computed as follows:

$$r^w = \frac{n - \sum_{i=1}^{29} |r \odot r_i|}{n} \tag{29}$$

Here, $n$ is the maximum cardinality in a row, which is equal to 29 $\times$ the cardinality of the totally underspecified relation `?`, which is 13. The lower the cardinality, the more likely it is that the composition of r with another relation type will generate a constraint. The table 3.10 has weights for some of the 29 rows.

| | derivations | weight | cardinality | inverse | weight |
|---|---|---|---|---|---|
| $<$ | 23 | 0.79 | 141 | 236 | 0.63 |
| m | 26 | 0.90 | 123 | 254 | 0.67 |
| tt | 26 | 0.90 | 171 | 206 | 0.55 |
| db | 13 | 0.45 | 284 | 93 | 0.25 |
| ? | 0 | 0.00 | 377 | 0 | 0.00 |

Table 3.10: Weights for composing $<$, `m`, `tt`, `db` and `?` with another relation type

A similar table compiles out the weights for relation types that appear to the right of the composition operator. Now instead of simply counting the links, we add up the weights.

### 3.6.2 Text-Segmented Closure

None of the user-prompting schemes in the previous section will reduce the potential worst-case time complexity of the task from quadratic to linear. The solution is to constrain user-prompting using text-segmented closure. The basic idea is that we relax the requirement (or strong wish) for completeness and settle for local completeness, which is defined informally as follows:

(30) A *locally complete temporal annotation* of a document is an annotation where each event is linked to all events and time expressions within its local context and where each time expression is linked to all events within its local context.

This relaxed completeness does not require the annotator to fill in all the relations that the closure algorithm cannot derive axiomatically. Instead, the only relations that the annotator is prompted for are relations between events and timexes that are adjacent in the text. A segment is defined as a sequence of N sentences or time objects, where sentence boundaries are defined by punctuation markers. For example, a segment could consist of three sentences. Segments overlap, that is, with three-sentence segments the first segment of a document contains the first three sentences, the second segment contains sentences two through four, and so forth.

The annotator in the prompting phase is faced with a sliding window that moves down the text. The window starts out covering just the first segment and the user is prompted for new relation types inside this window. Each time the annotator adds a relation, temporal closure will compute the consequences (including the non-local ones). The cycle continues until all event and timex pairs in the first segment are specified. Then the window slips down one sentence. A benefit of this is that the annotator always has easily available all she needs to determine the relation type, no scrolling is required.

Text-segmented closure is not globally complete, only locally complete. But as we will see later in chapter 4, text-segmented closure with user-prompting can derive more than 90% of all possible links, yet display linear annotation time because user-prompting is now linear to the number of events instead of quadratic. With local prompting instead of global prompting, the worst case time complexity of the task drops from $O(N^2)$ to $O(n^2 s)$, where $N$ is the number of time objects in a document, $n$ the number of time objects in a segment and $s$ the number of segments. This means that if the segment size is fixed across documents, temporal annotation is reduced to a linear task. Table 3.11 has a few example figures that illustrate the difference between the number of global links and the number of local links in a document (the first column is the number of time objects in the document, the second column the number of segments in the document).

| time objects | segments | global links | local links |
|---:|---:|---:|---:|
| 50 | 5 | 2500 | 500 |
| 100 | 10 | 10,000 | 1000 |
| 200 | 20 | 40,000 | 2000 |

Table 3.11: Number of global and local links

Note the distinction between the complexity of the annotation task and the complexity of the closure algorithm. The first task is linear, whereas the second is cubed to the number of time objects. This seems like a fair division of labour.

### 3.6.3 Dealing with Inconsistencies

Obviously, the best way to deal will inconsistencies is to make sure that they never come into existence. Recall that one way to embed a closure component is to have it run in the background, constantly computing all consequences each time a new TLINK is added. This is not always realistic though. For example, an unattended pre-processing routine cannot easily be integrated with a closure component. It is easier to let the pre-processing module run in batch mode and let an annotator use the closure mechanism afterward and check for inconsistencies. Also, sometimes we may want to check the consistency of a TimeML annotation created with an annotation tool that did have a closure component or that included a closure component that did not help to resolve the inconsistencies. For these cases we would do well to adopt a three-phase annotation approach:

1. automatic pre-processing or legacy manual annotation
2. running closure and resolving inconsistencies
3. further manual annotation with closure running in background

The first phase may generate inconsistencies, which are then resolved in the second phase. How do we resolve them? A rather unreliable approach is to let the closure component make an educated (or, more likely, an uneducated) guess as to which temporal link is better. This only works if TLINKs come with reliable confidence scores or if there is some global preference system.

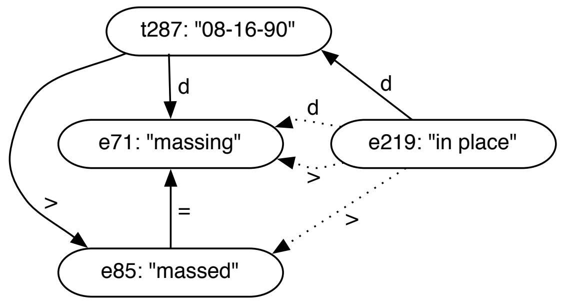
A more reliable approach is to let the annotator decide which link is the right choice. It is fairly trivial to print out the history of how an inconsistency came about. This is essentially a stack trace that leads the annotator back to all the links that were not added by the closure engine and that somehow featured in creating the inconsistency. An example of an inconsistency in an annotated file from the 1990 Associated Press is shown below:

```
{d} <"in place" e219 - "massing" e71>
    {d} <"in place" e219 - "08-16-90" t287>
    {d} <"08-16-90" t287 - "massing" e71>

{>} <"in place" e219 - "massing" e71>
    {>} <"in place" e219 - "massed" e85>
        {d} <"in place" e219 - "08-16-90" t287>
        {>} <"08-16-90" t287 - "massed" e85>
    {=} <"massed" e85 - "massing" e71>
```

But this information should be presented to the user in a more accessible way. Ideally, the annotation tool should present the subgraph involved in the inconsistency as well as the text positions of all involved events and timexes. An example of how this could be done is in figure 3.15.

In this figure, the document creation time is printed in a box separate from the body of the article. Only the sentences that contain relevant events and timexes are shown. It may be necessary to have some mechanism to zoom out to other parts of the document. A quick inspection of figure 3.15 reveals that the problem was that the document creation time had two links with different relation types to two events that were considered equal, but in retrospect, it may be better to see the two



| AP-NR-08-16-90$^{t287}$ | |
|---|---|
| 6 | Pentagon sources in Washington meanwhile said the Bush administration plans to deploy 45,000 Marines to the region to back up the thousands of Army, Navy and Air Force troops already **in place**$^{e219}$ in the gulf and the Saudi desert. |
| 21 | He called U.S. soldiers **massing**$^{e71}$ in Saudi Arabia the real occupiers in the Persian Gulf. |
| 25 | In the largest U.S. military operation since Vietnam, an estimated 20,000 American GIs have already **massed**$^{e85}$ to defend Saudi Arabia. |

Figure 3.15: Presenting an Inconsistency

*massing* events as separate events, one occurring before the other.[9] The annotator can now change the relation type on ⟨e85 e71⟩, after which the closure component has to retract all consequences of the previous relation type and recompute closure.

[9]Note that the closure component detected this inconsistency when it created a clashing link from *in place* to *massing*, while links from *in place* themselves are not responsible for the inconsistency. This is due to indeterminism in the order in which closure derives new facts.

# Chapter 4

# Evaluation

*Evaluate what you want — because what gets measured, gets produced.*

James Belasco, Business Leadership Strategist

In this chapter, I examine more closely the claims made about temporal closure, showing that temporal closure detects inconsistencies and that, when coupled with user-prompting, it makes a near-complete annotation feasible. More specifically, I will present data on (i) the number of links added, (ii) the increase in average link span, (iii) the number of inconsistencies detected in TimeBank, and (iv) the impact of closure on inter-annotator agreement. In addition, I will investigate how the user-prompting in text-segmented closure helps the annotation task. Throughout the chapter, some preliminary comparisons with Setzer's closure component are offered. Few observations will be offered on the influence of visualization modes on temporal annotation. While interesting, those are beyond the scope of this dissertation.

Three generations of SputLink have been implemented:

1. An early version embedded in the Alembic Workbench and the Event Diagram, implemented in Tcl/Tk (Pustejovsky et al., 2002). This version was in essence an improvement of Setzer's closure implementation, using a larger (but still incomplete) set of inference rules that fit in with TimeML. It did include text-segmented closure functionality.
2. A version embedded in Tango, implemented in Java by Andrew See (Pustejovsky et al., 2003b). The algorithm in this version is very close to the one described in chapter 3, but it lacks an implementation of text-segmented closure and the prompting strategies of sections 3.6.2 and 3.6.1.
3. A standalone version, implemented as a set of Perl classes. This version is the 'purest' since it adheres most closely to the algorithm and features from chapter 3. Unlike the first two versions though, it lacks a graphical user interface. It should also be noted that this version does not yet incorporate the additions of section 3.5.2. That is, all events and times are considered intervals and there are no points.

Unless stated otherwise, the standalone version of SputLink is the one used for all evaluations in this chapter.

Recall that SputLink is embedded in an annotation environment and that three stages of annotation can be distinguished: (i) a phase of TLINK-annotation by the annotator, (ii) a phase of initial temporal closure, and (iii) a phase of interactive closure centered around a user-prompting and closure loop. This three-phase approach is presented graphically in figure 4.1.

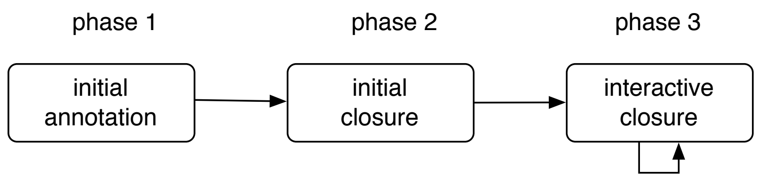


Figure 4.1: The three phases of annotation

The data on number of links added and average link span in section 4.1 and the inter-annotator agreement figures in section 4.2 are all relative to phases one and two of the annotation. The user-prompting evaluation results in section 4.3, on the other hand, also include phase three.

## 4.1 Closing TimeBank

A wealth of TimeML data is available in the TimeBank corpus (Day et al., 2003). TimeBank consists of 182 documents with 7962 events, 1422 timexes and 5681 TLINKs. Applying closure to TimeBank delivers solid statistics on numbers of links generated (section 4.1.1) and the non-local nature of TLINKs after closure (section 4.1.2). It also provides examples of how temporal closure identifies inconsistencies (section 4.1.3). In all sections except the section on inconsistencies only a subset of the 182 TimeBank documents was used: the 32 documents that contained inconsistencies were excluded from the sample.

### 4.1.1 Links Added

A first obvious characteristic of a corpus after initial closure is that its increased number of TLINKs. But always at the back of our mind is the loftier goal of a complete or near-complete annotation. This section explores how much closer initial closure gets us to that goal.

Running initial temporal closure over TimeBank more than quadruples the number of TLINKs, as shown in table 4.1. The density column deserves some explanation. It is convenient to have a measure that reflects completeness of an annotation. The density of an annotation is the percentage of TLINKs relative to all possible TLINKs in the corpus. For example, suppose we have an annotation graph with 10 events and 9 temporal links. The total number of possible links between the 10 events is 45.

|  | links | links/doc | share | density |
|---|---|---|---|---|
| added by initial annotation | 4243 | 28.3 | 24.2% | 1.28% |
| added by initial closure | 13306 | 88.7 | 75.8% | 4.02% |
| total | 17549 | 117.0 | 100.0% | 5.30% |

Table 4.1: Links added during the first two phases of TimeBank annotation

So the density is $9/45 \times 100\% = 20\%$. An annotation is complete if its density is 100%. The density is actually nothing more or less than the recall measure[1] where the key is a hypothetical annotation where all links are added and where we are not interested in the relation type of the links. However, using recall for the purpose of measuring completeness has proven to be a tad confusing, so here I'll use the term density. After closure, the density of TimeBank goes up from 1.28% to 5.30% with closure being responsible for almost 76% of all links.

It seemed like a good idea to check whether some characteristics of TimeBank documents had any impact on the number of TLINKs added by closure. The two characteristics that I investigated are (i) the size of the annotated document and (ii) the number and size of subgraphs in the annotation. The number of links before and after closure for seven classes of document size are printed in table 4.2. The document size is measured in the number of events and timexes in a document.

The percentage of derived TLINKs does not change much depending on the size of the document, except that smaller documents seem to provide less fodder for the closure component. Initial closure derived about two-thirds of the TLINKs for

[1]See (33) in section 4.2 for a definition.

| time objects | docs | links per document | | | density |
|---|---|---|---|---|---|
| | | annotated | derived | % derived | |
| <25 | 57 | 11.2 | 20.3 | 64.4% | 18.8% |
| 25–50 | 48 | 21.8 | 47.8 | 68.7% | 11.5% |
| 50–75 | 21 | 39.4 | 161.5 | 80.4% | 10.5% |
| 75–100 | 8 | 45.8 | 130.8 | 74.1% | 4.5% |
| 100–150 | 11 | 72.0 | 340.6 | 82.5% | 5.4% |
| 150–200 | 3 | 102.7 | 263.7 | 72.0% | 2.9% |
| >200 | 2 | 134.0 | 439.5 | 76.6% | 1.2% |
| all | 150 | 28.3 | 88.7 | 75.8% | 5.3% |

Table 4.2: Links before and after closure relative to document size

documents with less than 50 time objects and three-quarters of the TLINKs for larger documents. The density is much higher for small documents. This is consistent with the facts that the number of temporal links is quadratic to the document size and that the number of links in an initial annotation tends to be linear. It should be mentioned though that the figures hide large variations amongst individual files. For the class of documents with less than 25 time objects, the percentage derived by closure ranges from 0% to 86.9%. Similarly, the observed density ranges from 1.96% to 56.9%. These variations become gradually smaller with increased document size, and are virtually non-existant in documents with more than 150 time objects.

The number of subgraphs is one way to measure the connectedness of a graph. An annotation graph with the set of nodes {e1 e2 e3} and one link from e1 to e3 has two subgraphs: {e1 e3} and {e2}. If this same graph also had a link from e2 to e3, then it would be more connected and have only one subgraph {e1 e2 e3}. Table 4.3 shows

| subgraphs | docs | nodes/doc | % derived | density |
|---|---:|---:|---:|---:|
| <10 | 55 | 14.9 | 70.3 | 21.1% |
| 10–25 | 62 | 23.1 | 75.4 | 11.7% |
| 25–50 | 18 | 45.8 | 75.0 | 5.1% |
| 50–75 | 13 | 69.4 | 80.2 | 4.4% |
| >75 | 2 | 134.0 | 76.6 | 1.2% |

Table 4.3: Density after initial closure relative to number of subgraphs

how the number of subgraphs in an annotation impacts the density after closure. It is obvious that the smaller the number of subgraphs, the higher the density. But, not surprisingly, those documents with a higher number of subgraphs tend to be larger. In fact, the number of subgraphs and the document size in time objects have an almost identical impact on the density and can therefore be considered two faces of the same coin. Figure 4.2 shows the linear relation between documents size and number of subgraphs.

Intuitively, the measure that really matters is the size of the largest subgraph. Suppose we have a graph with eight events and two ways of carving it up into two subgraphs: [ {e1 e2 e3 e4} , {e5 e6 e7 e8} ] and [ {e1 e2 e3 e4 e5 e6 e7} , {e8} ]. Closure can never derive a link that connects two subgraphs because there already needs to be a path connecting the two events or timexes, so the number of links derived by closure is bound by the maximum number of links for the individual subgraphs. In the first case, the maximum number of links is $6 + 6 = 12$, in the second case it is $21 + 0 = 21$. In general, annotations with the largest subgraphs are favored to derive more links by closure because the number of links is quadratic to the number of time

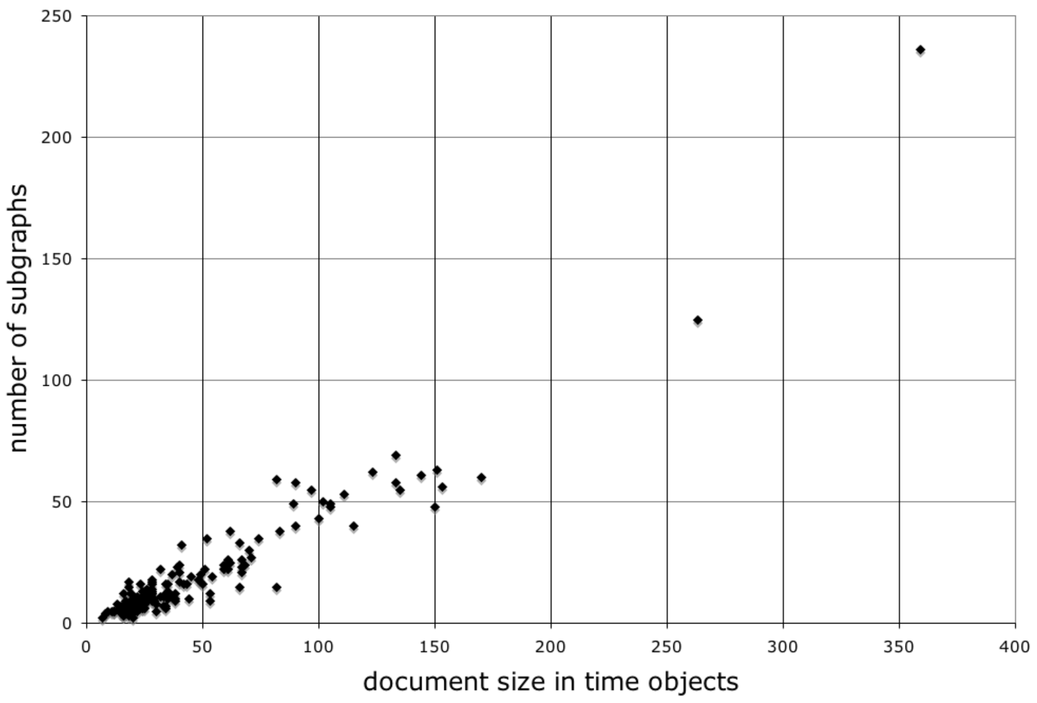


Figure 4.2: Correlation of document size and number of segments

objects.

The size of the largest subgraph is measured as the ratio of the size of the subgraph and the total number of time objects. For example, if an annotation graph has 28 events and timexes and the largest subgraph contains 12 elements, then the ratio is $12/28 = 0.43$. Table 4.4 shows that indeed the size of the largest subgraph has a major impact on the density after initial closure. And this impact cannot be explained by adjusting for document size. This means that an annotation strategy that maximizes the size of the largest subgraph is more likely to achieve higher density with fewer

| ratio | docs | nodes/doc | % derived | density |
|---|---|---|---|---|
| 0.00–0.25 | 53 | 35.9 | 67.1 | 2.5% |
| 0.25–0.50 | 60 | 24.6 | 75.4 | 9.3% |
| 0.50–0.75 | 34 | 24.1 | 85.1 | 19.2% |
| 0.75–1.00 | 3 | 14.0 | 83.5 | 53.8% |

Table 4.4: Density after initial closure relative to the largest subgraph

user-added links. This issue will be taken up again in section 4.3.

So the relative size of the largest subgraph before closure determines the density after closure. From this it follows that smaller documents are more likely to have a higher density because it is far more likely for a small document to have a large subgraph than it is for a large document.

Andrea Setzer (2001) also provides some data on percentage of links derived by closure. A comparison of Setzer's data with SputLink is given in table 4.5. For this table a subset of TimeBank was used; only the 45 documents with sizes similar to Setzer's document (that is, between 15 and 25 time objects) were used.

| | docs | links per document | | | density |
|---|---|---|---|---|---|
| | | annotated | derived | % derived | |
| Setzer | 1 | 14.0 | 25.0 | 64.1% | 20.5% |
| SputLink 45 | 25 | 12.6 | 24.6 | 66.2% | 18.8% |

Table 4.5: Comparing Setzer's closure module to SputLink

It looks like there is no big difference in how many links are derived by initial closure. However, a comparison is pretty much meaningless given the very small size of Setzer's sample and the observation made previously that there is a large variation hidden in the averages.

### 4.1.2 Average Link Span

In this section I present statistics showing that temporal closure adds non-local TLINKs to the annotation and that these links were mostly absent from the annotation before closure.

The annotators who marked-up TimeBank seemed to converge on similar annotation strategies, linking events to other events and timexes that were close in textual proximity. Take for example the TimeBank fragment in (31) and the TLINKs added by the annotator in (32).[2]

(31) DCT: $\underline{\text{02-27-98 0802EST}}^{t25}$

Both U.S. and British officials **filed**$^{e12}$ objections to the court's jurisdiction in $\underline{1995}^{t23}$, **claiming**$^{e13}$ Security Council **resolutions**$^{e20}$ **imposed**$^{e14}$ on Libya to **force**$^{e15}$ the suspects' extradition **overruled**$^{e16}$ a $\underline{1971}^{t24}$ Convention which gives Libya the right to try the men.

[2]The fragment in (31) was taken from TimeBank article APW19980227-0476. The annotation in (32) is simplified in the sense that the MAKEINSTANCE tag has been omitted.

(32) <TLINK eventID=e12 relatedToTime=t23 reltype=IS_INCLUDED/>

<TLINK eventID=e12 relatedToEvent=e13 reltype=IS_INCLUDED/>

<TLINK eventID=e20 relatedToEvent=e12 reltype=BEFORE/>

<TLINK eventID=e14 relatedToEvent=e20 reltype=SIMULTANEOUS/>

The four TLINKs shown are all the TLINKs that the annotator added for events and timexes in the sentence above. There were no TLINKs from events in this sentence to events elsewhere in the document. The fragment exhibits two kinds of TLINKs: a local anchoring of the *filed* event to the time expression *1995* and three TLINKs that establish local orderings of events. What is interesting are the TLINKs that are not there. There are no global anchorings from events to time expressions in other sentences and there is no ordering of events with events outside the sentence. The four events and the time expression in {e12 e13 e14 e20 t23} form a subgraph in the annotation graph of the whole article.

The average link span is the textual span between the two events or timexes that are linked; it is measured by the number of sentence boundaries that are crossed. A sentence is delimited by punctuation and may include a main clause and an embedded clause. If all TLINKs are intra-sentential, as in the example above, then the average link span is 0; if all TLINKs cross one sentence boundary, then the average link span is 1. Table 4.6 has the link spans for TimeBank and two small corpora annotated with Tango. The baseline is the average link span if all links were annotated, that is, a complete TimeBank annotation would have an average link span of just over 13.

|  | TimeBank 1.1 |  | Tango Sample |  | Video Corpus |  |
|---|---|---|---|---|---|---|
| before closure | 2.42 | (0.88) | 3.59 | (1.86) | 0.59 | (0.59) |
| after closure | 6.89 |  | 5.96 |  | 6.93 |  |
| baseline | 13.35 |  | 7.74 |  | 9.57 |  |

Table 4.6: Link spans for TimeBank and two small corpora

For TimeBank, the average link span after initial annotation is 2.42. The numbers between brackets reflect the average link span when all links to the document creation time (DCT) are taken out. For TimeBank, taking out the DCT makes a big difference: 2.42 versus 0.88. This means that most cross-sentence links involve global anchoring to the DCT and that there is no significant global anchoring to other time expressions and no significant global ordering of events.

After initial closure, the link span rises from 2.42 to 6.89.[3] These numbers reveal that initial temporal closure adds a whole group of non-local links that are systematically missed by the annotators.

Recall that TimeBank was annotated with Alembic, a tool that uses a table metaphor to deal with non-local markup tags like <TLINK>. The experience is that such a tool leads annotators to an annotation strategy that involves predominantly local TLINK annotation, linking events to events that are nearby in the text. A more

[3]I have no data for average link span without DCT after closure. It is meaningless to take the closed TimeBank and compute the average link span by ignoring all derived links. This is because we would still count links that were derived using other links that contain the DCT. It would be interesting to see what happens when all links with the DCT are taken out before closure. In all likelihood, average link span and density after closure will be lower.

graphically oriented tool like Tango could be expected to facilitate higher average link spans. To calculate the average link spans for a graphical tool, I took two smallish corpora that were annotated with Tango: the Tango sample and the video corpus. The Tango sample was created for evaluation purposes at the end of the Tango project; the video corpus was created at Georgetown University[4] and was used to analyze how native and non-native speakers transcribe a video clip.[5] The data in table 4.6 show that there is no conclusive evidence that Tango does indeed better facilitate the creation of links that span many sentences. The Tango sample has a higher link span than TimeBank, but this sample contains only two documents and one of them had a lower link span than TimeBank. The video corpus has a significantly lower link span, but this corpus has no DCT (which would drive up the link span). In addition, the video corpus was much closer to the narrative convention for which a purely local TLINK annotation is perhaps the most appropriate. In any case: it is clear that, no matter what the tool, the link span increases after closure.

The average link span is obviously very sensitive to document length. To see whether the averages in table 4.6 hide anything interesting I calculated the link span for seven document sizes; this is shown in table 4.7, where the document size is given in number of events and timexes. The average link span increases after closure for each document size. More interestingly, the link span is pretty similar throughout

[4]With thanks to Inderjeet Mani for providing the corpus.

[5]The clip was taken from a Charlie Chaplin movie.

| time objects | docs | before closure | | after closure | baseline |
|---|---|---|---|---|---|
| <25 | 57 | 1.05 | (0.72) | 1.77 | 2.08 |
| 25-50 | 48 | 1.52 | (0.95) | 2.69 | 4.07 |
| 50-75 | 21 | 2.25 | (0.95) | 4.83 | 5.59 |
| 75-100 | 8 | 2.38 | (0.91) | 5.61 | 8.81 |
| 100-150 | 11 | 3.22 | (0.52) | 11.07 | 11.08 |
| 150-200 | 3 | 2.60 | (0.87) | 8.25 | 16.39 |
| >200 | 2 | 6.51 | (4.59) | 16.64 | 22.72 |
| all | 150 | 2.38 | (1.08) | 6.77 | 13.39 |

Table 4.7: Link spans for different document sizes in TimeBank

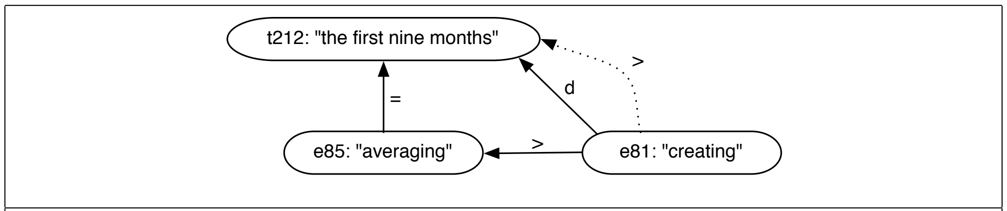
most document sizes when TLINKs that involve the DCT are left out. There is some variation, but it cannot be mapped straightforwardly to document size. It is not clear whether the outlier for the largest size is caused by the document size or by the fact that the two documents were the work of one annotator (who only annotated three documents overall). I strongly suspect that the latter is the case.

### 4.1.3 Inconsistencies

An inconsistency occurs when the relation type $\mathrm{r}_1$ of a TLINK $\langle \mathrm{x}\ \mathrm{r}_1\ \mathrm{y}\rangle$ clashes with the relation type $\mathrm{r}_2$ of a TLINK $\langle \mathrm{x}\ \mathrm{r}_2\ \mathrm{y}\rangle$, where $\mathrm{r}_1 \neq \mathrm{r}_2$ and $\langle \mathrm{x}\ \mathrm{r}_2\ \mathrm{y}\rangle$ is derived by closure from $\langle \mathrm{x}\ \mathrm{r}_3\ \mathrm{q}\rangle$ and $\langle \mathrm{q}\ \mathrm{r}_4\ \mathrm{y}\rangle$. An example from TimeBank[6] is displayed in figure 4.3.

One of the motivations for temporal closure is that it can catch inconsistencies like the one in 4.3 where two TLINKs, one added by initial annotation and one added by initial closure, are incompatible. The dotted line represents a TLINK $\langle \mathrm{e}81 < \mathrm{t}212\rangle$, which

[6]Source: TimeBank document NYT19980206.0460.



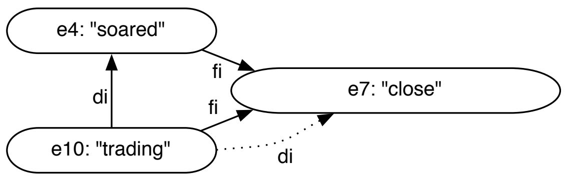
After accounting for a small downward revision Friday to December's figures, the economy has been **creating**$^{e81}$ jobs at a rate of 358,000 a month for the last four months and over 381,000 over the last three months after **averaging**$^{e85}$ 242,000 for <u>the first nine months</u>$^{t212}$ of 1997.

Figure 4.3: Example inconsistency from initial TimeBank annotation

was derived by closure from ⟨e81 < e85⟩ and ⟨e85 = t212⟩. Clearly, ⟨e81 < t212⟩ clashes with ⟨e81 d t212⟩ because an event cannot be both during and before a certain time period. In this particular case, the annotator decided that "averaging" and "the first nine months" are simultaneous, but then continued stating that "creating" is both after one and during the other.

All TimeBank articles were checked for inconsistencies using SputLink. There were 32 documents with an inconsistency. Manual inspection showed that all inconsistencies could be reduced to three annotator-added TLINKs between three time objects. In three cases, the inconsistencies were first detected while generating an inconsistent TLINK to a fourth time object; one of these three was given as an example in figure 3.15 in the previous chapter. About half the inconsistencies were intra-sentential, the others crossed 1 to 20 sentence boundaries, with the vast majority only crossing one or two sentences. Those spanning more than 2 sentences almost always included the document creation time.

One interesting inconsistency example is presented in figure 4.4. It illustrates a tension between the annotator's intuition of some TimeML relations and the strict interpretation of the closure engine. In the text in 4.4, the annotator probably wanted to convey that the trading included the soaring and that both were ended by the close event, which seems very reasonable. But note that this requires a slightly liberal interpretation of inclusion, namely as a temporal relation that does not require proper inclusion (that is, the soaring could end at the same time as the trading). The closure engine is more strict though and strikes back with an inconsistency, it would find ⟨e10 fi e4⟩ acceptable, but if it is confronted with ⟨e10 di e4⟩, then it will require ⟨e10 di e7⟩ rather then ⟨e10 fi e7⟩.



Weisfield's shares **soared**$^{e4}$ on the announcement yesterday, closing up \$11 to **close**$^{e7}$ at \$50 in national over-the-counter **trading**$^{e10}$.

Figure 4.4: An inconsistency illustrating an interpretation mismatch

There are two solutions here. Annotators could be trained to put in the more specific relation `ended_by` instead of `includes`, but this might be forcing annotators to be more specific than they want to be and may slow down annotation even further. A better solution is to make SputLink a bit smarter. It could recognize that `includes` and `ended_by` are conceptual neighbors and replace offending relations of this kind with the relation that subsumes both of them in the relation hierarchy in figure 4.6, in this case the relation {di fi}. Once this is done, the inconsistency evaporates. The trick of course is to limit this functionality. This would require a careful study of annotators' intentions and their interpretations of TimeML relations.

SputLink can obviously help resolve the inconsistencies and thereby increase the number of correct TLINKs by 32 and marginally improve TLINK-precision[7], which, given the 5681 TLINKs in TimeBank, would increase by 0.56%.

The number of 32 inconsistencies is small considering the size of TimeBank. But small-scale experiments with the Alembic version of SputLink[8] have shown that more inconsistencies can pop up in the user-prompting phase, when the annotators are asked to reflect on temporal relations that are much less clear than those that they volunteer in an initial round of markup. This particular experiment showed that about 4-5 inconsistencies are generated during the user-prompting stage of a document with about 40 time objects. This has not been quantified thoroughly though.

The Tango project (Pustejovsky et al., 2003b) performed a mini-evaluation in

[7]Precision is defined in (33) in the next section.

[8]This version, due to the absence of a complete inference rule set, allowed inconsistencies to be generated during the user-prompting phase.

order to measure how Tango compared to Alembic. It turned out that the TLINK precision figures for Tango were higher than those for Alembic: precision with Tango was 76% and precision with Alembic was 64%.[9] Whether these results stand up under more scrutiny (that is, larger experiments) is yet to be determined, but the question remains whether a graphical tool like Tango reduces the number of inconsistencies introduced.

| corpus | tool | docs | TLINKs | inconsistencies | precision improvement |
|---|---|---|---|---|---|
| TimeBank 1.1 | Alembic | 182 | 5681 | 32 | 0.56% |
| Tango sample | Tango | 2 | 140 | 0 | 0.00% |
| Video corpus | Tango | 15 | 782 | 1 | 0.12% |

Table 4.8: Inconsistencies in Alembic and Tango

The number of inconsistencies for the two Tango-annotated samples are shown in table 4.8. There seems to be a significant reduction in number of inconsistencies for Tango. But recall that comparisons are hazardous due to the small size of the Tango sample and the nature of the text in the video corpus.

[9]This is not to say that overall TLINK-precision of TimeBank is 64%. The mini-evaluation did not include any inspection of initial annotations. For TimeBank, on the other hand, each annotated file was inspected and discussed by other annotators and several revisions were made between TimeBank versions.

## 4.2 Inter-Annotator Agreement

Inter-annotator agreement (IAA) gives a hint as to how well-defined an annotation task is: low IAA indicates an ill-defined task. However, as noted in section 3.4, a comparison of two TimeML annotations needs to take into account that two annotations can be syntactically different yet semantically the same. Temporal closure maps identical semantic annotations onto identical syntactic annotations and therefore has the potential to increase IAA scores, as was claimed previously by (Katz and Arosio, 2001) and (Setzer, 2001).

To calculate IAA, I adopt (Setzer, 2001), who, following (Hirschman et al., 1998), used pairwise comparisons of precision and recall figures.[10] For each text, one annotator is taken as the key and standard precision and recall, as defined in (33), are calculated with the other annotator as the response. Then annotators swap their key and response status and P&R are calculated again. Finally, we average over the two sets of data.

(33)
$$\text{Precision} = \frac{\text{matches in response}}{\text{entities in response}} \times 100$$

$$\text{Recall} = \frac{\text{matches in response}}{\text{entities in key}} \times 100$$

[10]An older standard measure to measure inter-rater agreement is the Kappa coefficient, which adjusts for the number of agreements that would have occurred by chance. This coefficient however, is not well suited for annotation tasks that cannot be construed as pure classification tasks.

The IAA data in this section are obtained from an experiment at Brandeis University. Eighteen documents[11] were each annotated by two people using the Alembic Workbench. The annotators had no prior exposure to Alembic and had no background in linguistics. They all received about two hours of training. Of the eighteen documents, ten were taken out of the sample because closure generated inconsistencies or because one of the two annotators did not add any TLINKs at all. The precision and recall measures before and after closure are in table 4.9. Precision and recall numbers are identical when the above calculation method is applied to pairs of annotators, the table therefore collapses recall and precision into one number. The time objects column contains the P&R for the presence of an event or timex at some text extent. This is rather liberal, because it is considered a match when the same extent is annotated as an event by one annotator and as a timex by the other. The links column is similar to the time objects column and reflects whether two text extents were connected by a TLINK by the two annotators. The relations column is sensitive to the relation type of the two TLINKs: a match requires the two relation types to be the same. The first percentage inside the links and relations columns reflects the P&R after the initial annotation phase, the second percentage the P&R after the initial closure.

As expected, inter-annotator agreement before closure was low, varying from 0% to 36%, with the average hovering around 20%. There was also considerable dis-

[11]These documents were not from TimeBank, but were taken from the same domain.

| document | precision & recall | | | | |
|---|---|---|---|---|---|
| | time objects | links | | relations | |
| AP900822-0016 | 87.1% | 25.8% | 27.1% | 17.8% | 20.3% |
| APW19980428.0729 | 83.1% | 35.9% | 42.9% | 14.4% | 8.6% |
| APW19980510.0720 | 78.9% | 24.1% | 16.7% | 16.1% | 10.9% |
| CNN19980104.1600.0033 | 68.6% | 8.7% | 14.7% | 0.0% | 2.9% |
| NYT19980212.0025 | 66.7% | 16.1% | 10.9% | 6.4% | 4.4% |
| PRI19980218.2000.0431 | 80.9% | 0.0% | 0.0% | 0.0% | 0.0% |
| SJMN91-06338157 | 79.5% | 31.6% | 36.3% | 6.3% | 16.1% |
| WSJ910627-0102 | 75.5% | 20.6% | 20.4% | 8.6% | 13.3% |

Table 4.9: IAA scores for 8 documents

agreement amongst annotators on the relation type. There were 104 instances where two annotators created a TLINK connecting the same text objects, but in only 50 of those the annotators added the same relation type attribute. IAA for relation type on average is about 9%, and about 48% if links that do not occur in both annotations are ignored.

What was not expected was that initial temporal closure has no obvious effect on inter-annotator agreement: it goes up for some documents but down for others. It may be unintuitive that temporal closure can actually lower inter-annotator agreement, especially given the assertion that temporal closure maps semantically identical but syntactically different annotations to syntactically identical annotations. Consider figure 4.5 for an example of how closure can reduce IAA. The solid arrows are TLINKs added in the initial annotation phase. The two annotations have one TLINK in common and differ on the other: IAA is 50%. Now enter the dotted line which represents a third TLINK added after initial closure. IAA is now down to 33%. In general, closure can generate both TLINKs that raise IAA and TLINKs that push down IAA and there are many annotation configurations where the latter is more prevalent than the former.

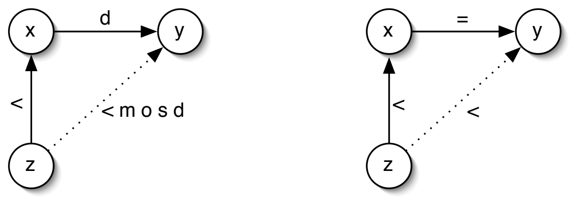


Figure 4.5: Example of how closure can reduce IAA

So there is no evidence that initial temporal closure has a positive (or negative) influence on IAA. This is contrary to results from (Katz and Arosio, 2001), who reported that annotations were syntactically identical in 70% of the case and semantically identical in 85% of the cases. It is not clear however whether they measured the same thing as I'm measuring here, partly because (Katz and Arosio, 2001) only looked at annotation within a sentence and used a much smaller set of relation types. A comparison with (Setzer, 2001) is not possible because she does not provide IAA figures after temporal closure. It should also be noted that the Brandeis experiment was performed by a very diverse group of naive, unpaid and possibly unmotivated annotators. A larger-scale experiment with less naive annotators is sorely needed.

But even given these tentative results, we can still speculate with good cause that IAA scores will go up when two annotations both have a sufficiently high density. In that case the P&R figures in the links column have no choice but going up, with the numbers in the relations column probably following in their wake.

One problem with the numbers in table 4.9 is that all relation types that are annotated differently are treated the same. For example, if the two annotators chose `before` and `ibefore` for a particular TLINK, then this disagreement will be punished as hard as if they had chosen `before` and `ends`, two relations that are much more dissimilar than `before` and `ibefore`. It is interesting to measure how reducing specificity on relation types increases P&R. The hierarchy from section 3.5.1, repeated with minor changes as figure 4.6, comes in handy here.[12]

IAA requirements can be relaxed by not using equality of relation types at the bottom level, but set membership at a higher level in the hierarchy. For example, say we take the second level from the bottom as our measuring stick. Now if the key contains the TimeML relation type `before`, then the response may contain `before` or `ibefore`, which are the TimeML relations that can be obtained from the set $\{<\ \mathrm{m}\}$. Note that if the key contains the relation type `begins`, then the response could be

[12]The changes made are for presentation purposes. The `o` and `oi` nodes, which are not relevant to TimeML, were removed. Also, the labels on the higher levels have been replaced with sets of basic relations. Note that these still have to be translated into TimeML relations.

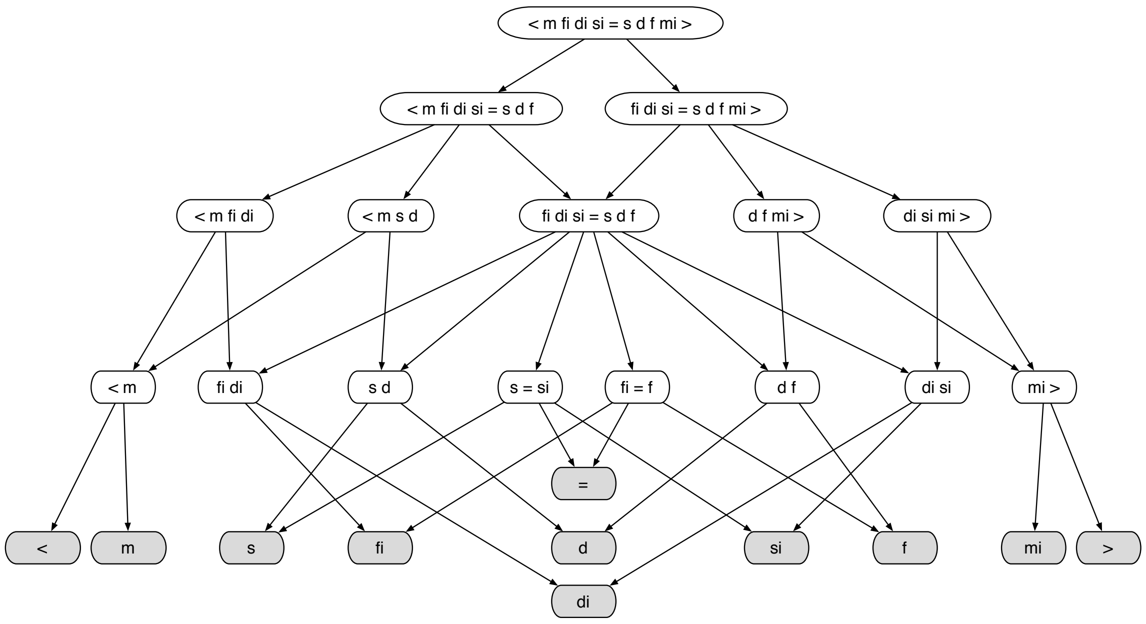


Figure 4.6: Hierarchy of relations

taken from {s d} or {s = si} depending on which subsuming node is taken, a choice that depends on whether we allow variation of the begin points or the end points of the intervals.

More interestingly, we can find those TimeML relation types that are most responsible for depressing the IAA score by simply tallying the mismatches. These relation types can be replaced in key and response with a disjunction from 4.6 by taking the lowest node that subsumes both the relation type from the key and the relation type from the response. For example, `before` and `ibefore` (< and `m`) would both be replaced by {< m}, `before` and `ends` (< and `f`) by {< m fi di si = f d s}, and `begins` and `included_by` (`s` and `d`) by {s d}. Precision and recall can now be measured again, using the sets rather than the single TimeML relations.

Table 4.10 has all mismatches that occur more than once.[13] The third column has the most specific type that is more general than both of the relations in the mismatch. Obviously all mismatches should be analyzed in full. Some are probably just due to inexperience with the Alembic tool or relative unfamiliarity with TimeML. For example, the confusion of AFTER and BEFORE is likely the result of an annotator accidentally switching the events that are related.

| n | relation type pair | | more general type |
|---|---|---|---|
| 8 | INCLUDES–SIMULTANEOUS | (di , =) | fi di si = s d f |
| 8 | IS_INCLUDED–SIMULTANEOUS | (d , =) | fi di si = s d f |
| 8 | INCLUDES–IS_INCLUDED | (di , d) | fi di si = s d f |
| 4 | IDENTITY–SIMULTANEOUS | (= , =) | = |
| 2 | BEGINS–BEGUN_BY | (s , si) | fi di si = s d f |
| 2 | AFTER–SIMULTANEOUS | (> , =) | fi di si = s d f mi > |
| 2 | BEFORE–IBEFORE | (< , m) | < m |
| 2 | DURING–SIMULTANEOUS | (= , =) | = |
| 2 | BEFORE–SIMULTANEOUS | (< , =) | < m fi di si = s d f |
| 2 | AFTER–BEFORE | (> , <) | all |
| 2 | AFTER–IAFTER | (> , mi) | mi > |

Table 4.10: Most frequent relation type mismatches across annotators

The data in table 4.10 make it clear that getting rid of mismatches involves com-

[13]It is no accident that mismatches seem to occur in pairs. For normalization purposes, each time a TLINK is added to the annotation graph, the reverse TLINK is also added. As a consequence, if there is a mismatch BEFORE–SIMULTANEOUS, then there is also a mismatch AFTER–SIMULTANEOUS.

paring relations at a higher level and that for some mismatches we have to go higher up the hierarchy than for others. For example, eliminating the AFTER–IAFTER mismatch requires going up one level whereas the BEFORE–SIMULTANEOUS mismatch requires us to go up three levels. The three most frequent mismatches all involve annotator disagreement on event containment: is event A simultaneous to event B, is A contained in B or is B contained in A. The most precise relation from the hierarchy in 4.6 that abstracts away over these choices is {fi di si = s d f}, which Freksa called `contemporary of`. Obviously, when comparisons for IAA are made on that high a level, which effectively only distinguishes between overlap and precedence, then the IAA will rise significantly, going up from 48% to 78%.[14] Note however that this approach massages away disagreement on any pair of relations from {fi di si = s d f}, including for example the pair ⟨fi =⟩ (the TimeML relations `ended_by` and `simultaneous`) and we surely don't want to go as far as saying that `ended_by` and `simultaneous` cannot be distinguished from each other.

The reason that I dwelled on this issue so long is that there are actually some interesting implications for temporal closure here. If annotators routinely confuse some relations than closure will generate questionable inferences. But if closure uses less specific relations as its input then these unwanted inferences will not be made.

[14]It is interesting that the number of 78% is very close to the number reported by (Katz and Arosio, 2001). This suggests that granularity of the annotation language is a major factor in determining the inter-annotator agreement.

Suppose that for a TimeML annotation the relations `simultaneous`, `includes`, and `is_included` do not enter the closure engine as the basic relations `=`, `di`, and `d`, but all as {fi di si = s d f}. In that case, the potential incorrectness will not propagate through the graph while the information that the annotator added a specific TimeML relation will remain available. The disadvantage is that less inferences will be made and completeness will be harder to achieve.

## 4.3 Interactive Closure

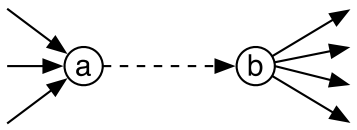
Recall that in chapter 3 initial closure was not expected to provide a complete annotation. And indeed the results in section 4.1.1 show that only one in twenty links are made available after initial closure. The data in this section concentrate on phase three of the annotation: the interactive closure with user-prompting (see figure 4.1 earlier in this chapter). I will show that a near-complete annotation can be obtained in linear time without having to ask the annotator to supply non-local temporal relations.

Some of the data in this section are derived using a human annotator actually answering the prompting. But the vast majority of data was collected using a simulation. In this simulation, an added component to SputLink stands in for the user and provides the relation type. This relation type is generated randomly, but relative weights were used to model the observation that some relations are more frequent than others. Relation type distribution data from TimeBank provided the relative frequencies of TimeML relations. The only exception was the `unknown` relation, which was added to TimeML predominantly to allow for underspecification in the prompting phase. Distribution data for this relation type were gleaned from the manual prompting experiments, which indicated that about 24% of user prompts result in the addition of an `unknown` relation type.

The simulation was set up because there were not nearly enough data to make any significant comparative statements. Properly evaluating the claims about TSC, the optimal segment size and the optimal prompting strategy would require a large-scale annotation fest with a medium-sized group of annotators (about 5-10) annotating a range of articles using an array of different user-prompting setups. There were simply not enough resources to do this and the next best option was to set up a simulation, with some human assisted closure as a sanity check. Using a simulation is acceptable in the present case because, unlike with for example inter-annotator agreement or precision, the data that I'm trying to generate here are quantitative, that is, I'm interested in how many TLINKs are derived rather than which TLINKs.

### 4.3.1 Prompting Strategies

The user-prompting strategies outlined in chapter 3 have been evaluated using a simulation. The four user-prompting strategies from section 3.6.1 are repeated here in table 4.11. The results of the simulation are displayed in table 4.12. It shows the den-



text position
: choose a and b on their text position

link counts
: choose a and b to maximize the number of incoming and outgoing links

derivation weights
: choose a and b to maximize the weights on the incoming and outgoing links, weights reflect the number of derivations possible

cardinality weights
: choose a and b to maximize the weights on the incoming and outgoing links, weights reflect the number of derivations possible and the cardinality of the derivation

Table 4.11: The four user-prompting strategies

sity achieved for each prompting strategy relative to the number of prompts allowed. The first column contains the baseline: the density before any user-prompting.[15] There are no big differences in density when there are five or less prompts, but with 10 or 20 prompts the simple text position based prompting strategy outperforms the more complicated strategies.

This may seem counter-intuitive so let's look at some more numbers for the 20-prompt column. The interesting figure is the number of subgraphs in the annotation after 20 prompts (recall that the number of subgraphs is a good indicator

[15]This simulation was ran on a subset of TimeBank, excluding some large documents. Therefore, the density figures are higher than they would be if all documents were used in the simulation.

|  | number of prompts |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | 0 | 1 | 2 | 5 | 10 | 20 |
| text position | 13.2 | 16.4 | 20.0 | 31.0 | 51.3 | 78.4 |
| link counts | 13.2 | 16.6 | 20.7 | 29.0 | 43.7 | 69.9 |
| derivation weights | 13.2 | 16.7 | 21.0 | 30.9 | 46.9 | 70.8 |
| cardinality weights | 13.2 | 17.5 | 21.4 | 31.1 | 46.0 | 72.1 |

Table 4.12: Densities achieved with four global prompting strategies

for the density achieved[16]). For text position prompting the number of subgraphs is 2.3, for link counts 4.2, for derivation weights 4.0 and for cardinality weights it is 3.8. The text position prompting strategy apparently is more successful in connecting subgraphs and therefore achieves higher densities. It turns out that prompting schemes that count incoming and outgoing links or the weights on those links tend to favor adding a link to a subgraph, as depicted in figure 4.7.

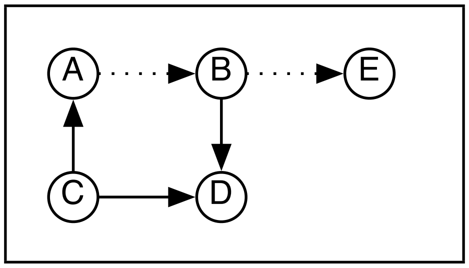

$$\begin{tikzcd}
A \arrow[r, dotted] & B \arrow[r, dotted] \arrow[d] & E \\
C \arrow[u] \arrow[r] & D &
\end{tikzcd}$$

Figure 4.7: Two choices for user-prompting

In this figure, existing TLINKs are drawn with a solid arrow and two choices for

[16]To be more precise, it is the ratio of the largest subgraph to the number of time objects that is the most significant. But when comparing identical document sets we can use the number of subgraphs as well.

user prompting are drawn with dotted arrows. A prompting strategy that stresses the degree of nodes will favor ⟨A B⟩ and never pick ⟨B E⟩, but a simple strategy that focuses on text position may still pick ⟨B E⟩. As a result, the simple strategy is more likely to connect subgraphs and will therefore have a better chance to achieve a higher density.

It would be interesting to experiment with the four prompting schemes above a bit further. We could for example only allow user prompts that connect subgraphs and see how the prompting strategies perform in this mode. This was left for future experimentation.

### 4.3.2 Text-Segmented Closure

In section 3.6.2 of the previous chapter, I proposed text-segmented closure (TSC) as a way to reduce the complexity of the user-prompting phase of the annotation task and to create an annotation that was locally complete. In this section I will examine that claim. I will also look at the level of global density given to us by local completeness and the optimal segment size for TSC.

Consider the data in table 4.13, which was filled using a simulation of user-prompting with a segment size of 3 sentences where the segments always included the document creation time.[17]

[17]The figures for initial annotation are slightly lower here than in table 4.1. This is because for the simulation experiments the five largest documents were taken out of TimeBank.

| | | links/doc | share | density |
|---|---|---|---|---|
| phase 1 | annotated links | 22.1 | 2.8% | 2.6% |
| phase 2 | initial closure | 60.7 | 7.6% | 7.1% |
| phase 3 | prompted links | 23.8 | 3.0% | 2.8% |
| | interactive closure | 695.0 | 86.7% | 81.7% |
| total | | 801.6 | 100.1% | 94.2% |

Table 4.13: Number of TLINKs derived with simulated user-prompting

As you can see, initial and interactive closure together derive a little over 94% of all TLINKs, which is significantly higher than the 76% share of derived TLINKs after phase 2 (cf. table 4.1). This particular local prompting setup delivers a global density of just over 94%. Note that all that's needed to achieve a density of 94% is an average of about 22 + 24 = 46 user-specified TLINKs per document. The massive variation in density we saw before user-prompting is not observed here. About 95% of all documents have densities over 80%, 44% have densities over 98% and 20% have a density of 100%.

Let's compare the figures in table 4.13 to the prompting statistics reported by (Setzer, 2001). She gives closure figures for six newswire articles from the New York Times from 1996. The comparison is in table 4.14. The difference in link-generating capacity of the two closure engines (82% versus 94%) may tentatively be explained as follows:

1. SputLink has a bigger rule base. Setzer uses an incomplete set of 10 inference rules (cf. section 3.4.2), SputLink employs a complete relation composition table with 638 entries.

|  | Setzer |  | SputLink |  |
|---|---|---|---|---|
|  | links/doc | share | links/doc | share |
| annotated | 18.5 | 4.0% | 22.1 | 2.8% |
| prompted | 63.5 | 13.8% | 23.8 | 3.0% |
| derived | 387.3 | 82.2% | 755.7 | 94.2% |
| total | 469.3 | 100.0% | 801.6 | 100.0% |

Table 4.14: Comparison of user-prompting stages

2. The smaller sample size of Setzer's corpus results in a higher variation of closure percentage, potentially skewing the results. For example, one of Setzer's articles had only 67.4% of links derived by closure. Individual data for the other files were not available.
3. Density is 100% for Setzer and 94.2% for SputLink. The last 6% may have taken more prompting-cycles to complete.
4. The simulation skews the results.

### 4.3.3 Optimal Segment Size

Two of the parameters that need to be set for text-segmented closure are the segment size unit and the number of units in a segment. There are two obvious choices for the unit: the sentence and the event or timex. The sentence is here again understood as a text extent between two punctuation markers. An additional question is whether the document creation time (which is a bit like a global time expression) should be included in the segment or not. There is always going to be a trade-off between effort (number of prompts) and result (density), but the ultimate goal is to get a high enough density with a number of user prompts that is feasible for human annotation, that is, it should definitely not be quadratic to the size of the document.

| | Number of sentences in segment | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| with DCT | | | | | | | | | |
| density in % | 77.5 | 90.4 | 94.3 | 95.3 | 97.5 | 98.2 | 98.4 | 98.4 | 99.2 |
| prompts/doc | 17.5 | 21.9 | 23.2 | 24.8 | 25.9 | 28.8 | 26.2 | 26.2 | 26.4 |
| without DCT | | | | | | | | | |
| density in % | 28.9 | 75.7 | 90.2 | 95.4 | 97.4 | 97.5 | 98.4 | 98.4 | 98.7 |
| prompts/doc | 13.7 | 21.4 | 24.0 | 24.4 | 25.5 | 26.2 | 26.7 | 26.3 | 26.3 |

Table 4.15: Prompting simulation results for sentence segments

Table 4.15 shows how link density and number of prompts are related when the sentence is the unit of measurement. The top half of the table presents the figures when the DCT is included in each segment; for the bottom half the DCT is left out of all segments. The data in this table support two significant observations:

1. All that's needed to achieve near-completeness of TimeBank is about 25 to 30 user prompts per document. Not shown in the table are the numbers for individual documents. Not surprisingly, the amount of prompts is higher for larger documents. But the number of prompts was lower than the number of time objects for all documents which confirms that user prompting in text-segmented closure is linear relative to the document size.

2. Local prompting within a window of three to four sentences supports a density in the mid nineties. This is a nice result because it means that text-segmented closure does not need to degrade to large segments if high density is required. So prompting can remain essentially local. The table also shows that allowing restricted global prompting by including the DCT gives much better results for the smaller segments. This effect evaporates when the segments are larger than three sentences.

The picture does not change when we take segment sizes to be determined by number of nodes (events and timexes) rather than sentences, as displayed in table 4.16. But using the node as the unit is in some sense more pure because it will not allow extremely long or short sentences to skew the results. That this indeed happens with sentence-sized segments becomes clear when we look at the range of measurements for individual files. For example, the average density when the segment size is three sentences (with the DCT included) is 94.3%, but this hides the fact that there are considerable variations. Sixteen of the documents had a density below 90%, six below 80%, and one had a density of 17%. If the segment size is set to ten nodes then the spread is much smaller: seven documents with density below 90%, two below 80%, and none below 65%. To frame this in more standard statistical terms we can use the standard deviation $\sigma$ which indicates how tightly measurements are clustered around the mean and which is defined as $\sigma = \sqrt{\frac{\sum(x-\mu)^2}{n}}$ where $\mu$ is the mean and $n$ is the number of measurements. The standard deviation for the three-sentence segments is

12.2, for the ten-node segments it is 5.1.

| | Number of nodes in segment | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 7 | 10 | 15 | 20 |
| with DCT | | | | | | | | | |
| density in % | 62.3 | 77.4 | 83.5 | 86.2 | 89.3 | 92.5 | 94.6 | 96.7 | 97.8 |
| prompts/doc | 11.8 | 18.3 | 19.6 | 21.1 | 22.0 | 23.2 | 24.8 | 28.8 | 26.3 |
| without DCT | | | | | | | | | |
| density in % | 9.7 | 35.8 | 57.9 | 74.3 | 79.3 | 90.5 | 95.1 | 97.2 | 97.8 |
| prompts/doc | 0.0 | 16.9 | 19.6 | 20.9 | 22.2 | 23.4 | 24.5 | 25.5 | 26.1 |

Table 4.16: Prompting simulation results for node segments

As previously discussed, table 4.15 shows that user-prompting is linear to the document size in time objects. And recall that in section 3.6.2 of chapter 3 I claimed that global user prompting is potentially quadratic to the size of the document. If this were true then the number of prompts per document would go up much faster then actually happens in tables 4.15 and 4.16. What we see instead is that density and number of prompts go up pretty much evenly and that the relation is quite linear. And indeed, setting the segment size to infinity results in 100% density with only 27.1 prompts per document. The documents used for this simulation did not show any outliers, in other words, the worst-case scenario of quadratic user-prompting did not occur in any of the documents. Figure 4.8 illustrates the linearity of the relation between prompts and density.

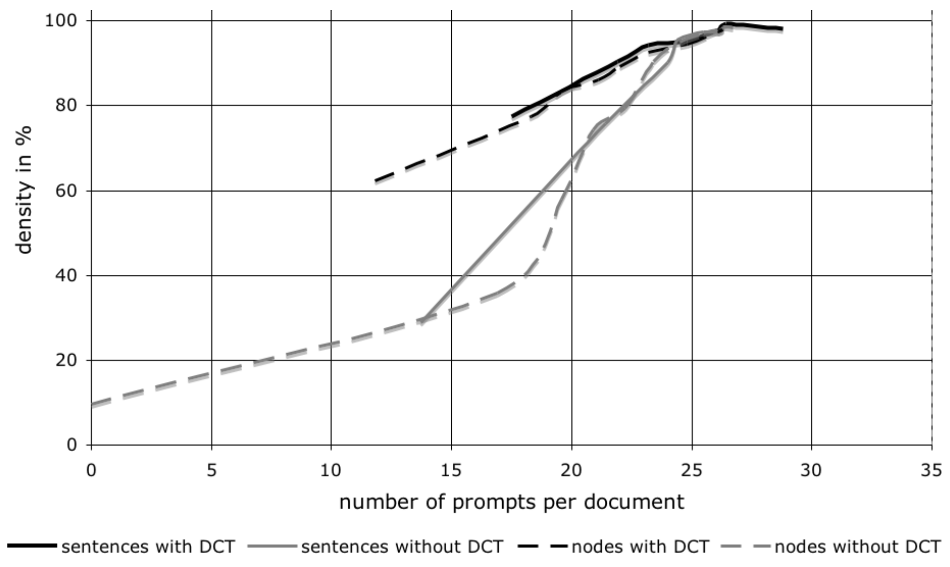


Figure 4.8: The relation between number of prompts and density

So the simulation results do not display the worst-case scenario of quadratic user-prompting. This was corroborated by a couple of manual experiments where the segment size was set to infinity and where the annotator was not subjected to quadratic user-prompting at all. This result can be explained by the fact that when the density of an annotation increases then the chance that there is a path between any two nodes X and Y also increases. And if there is a path between X and Y, then closure is often able to draw a TLINK directly from X to Y.

All in all, text-segmented closure has proven to be a viable approach to the interactive closure phase of the annotation effort. It provides for near complete annotations with a linear annotation effort while only prompting the user for local temporal relations, which simplifies the annotation task.

# Chapter 5

# Conclusion and Further Research

*Now what?*

The Puffer Fish in 'Finding Nemo'

This dissertation will probably not fundamentally change anyone's view of the world. But the previous 121 pages should have made it clear that a temporal closure component can greatly enhance temporal annotation. I have introduced SputLink, a temporal closure component based on Jalmes Allen's interval algebra and embedded in an annotation environment. I have shown that SputLink increases the density of an annotation and helps reduce the locality of manual annotation: that is, closure generates temporal relations between events that are not close to each other in the document.

With closure, it is possible to ensure consistency and much easier to achieve near completeness. Densities of over 90% are possible with interactive closure and user-prompting. Experiments show that temporal annotation is a task that is linear to the size of the document. Text-segmented closure simplifies the annotation process in the sense that the annotator will never be required to specify temporal relations between events that are not close in textual proximity. Yet text-segmented closure does not have a large negative impact on the annotation density.

The main goal in any annotation strategy should be to create a fully connected annotation graph. Running closure over a fully connected graph generates more new temporal relations than running closure over a fragmented graph. Text-segmented closure is one way to achieve that connectedness.

So what are the implications of all these results? I've mentioned before in the introduction and in chapter 3 that time is a well-structured domain and that an exhaustive inference system can be built that uses the compositionality of temporal relations. It is now clear that using the structure of time allows us to structure the annotation task. Indeed, the density results of chapter 4 show that the annotation task reduces to creating a connected subgraph of temporal objects and that text-segmented closure is an annotator-friendly way of achieving that.

We should also realize that these results are not just relevant for manual annotation. In fact, the closure component does not care whether it is a human or a program that provides the initial annotation or answers the prompts. What is important is that a large enough number of temporal relations is made available. Automatic generation of TLINKs tends to find local orderings and local anchorings as well as anchorings to a document-level time expression like the document creation time. And the results from text-segmented closure show that there is no need for long-distance temporal relations if we aim for a density of about 90-95%. That is, there is no need for an automatic link generator to reliably generate those links that are hardest to find.

Obviously there are many ways to create a connected graph. One could imagine a component that selects from the available temporal relations those that are needed to create a connected graph. Now suppose that the temporal relations are generated automatically and that each generated relation is associated with a confidence score. The component above could then favor those relations that have a high confidence level. This amounts to selecting a minimal set of temporal relations that maximizes the overall reliability of those relations. And this minimal set can be used as input to the closure component and give rise to a closed annotation graph that does not compromise high density.

(Setzer, 2001) also discussed this *minimal set problem*, which she defined as the problem of finding the minimal set of temporal relations from which we can infer all temporal information in a text. For her, finding the minimal set was of little practical value since a computationally feasible solution was not available. The problem here is slightly different though because rather than finding a minimal set from all temporal information, the task is to find a minimal set within an already constrained set of temporal relations (that is, those relations that are generated automatically).

There are some unavoidable loose ends and directions for future research. More data and research are needed on the relation between inter-annotator agreement and temporal closure. Section 4.2 showed that the effect of closure on IAA is a bit murky and that temporal closure has no clear positive effect on IAA. This is contrary to earlier findings by (Katz and Arosio, 2001) and this discrepancy deserves further study.

Related to this IAA issue is the question of how fine-grained an annotation language should be. Recall that the inter-annotator scores for TimeML are lower than the scores for the simpler language that Katz and Arosio use, but that the scores are similar when two TimeML annotations are compared at a higher level in the hierarchy of 29 convex relations. This may suggest that using the coarser relations simplifies the annotation task. But note that not enough data were presented to back this up firmly. Also, IAA scores for TimeML only went up significantly when some very coarse relations were used. And these relations lump together fine TimeML relations like `ended_by` and `simultaneous` and we may not want to lump these together.

(Setzer, 2001) argued that an annotation language should not be too precise and that vague relations like her `simultaneous` relation are necessary. TimeML, on the other hand, gave annotators the means to capture all possible temporal relations that are actually expressed in text. (Freksa, 1992) and (Schilder, 1997) end up somewhere in the middle, allowing both fine and coarse relations. Both viewpoints (using fine grained relations versus using coarse relations) can be defended with examples from real texts. More data on IAA and inconsistencies may shed light on what coarse relations actually help the consistency of an annotation.

Another thing to study is whether temporal closure allows the annotator to focus on the precise and easy to define temporal relations. It may be the case that many vague temporal relations can be inferred by the closure component.

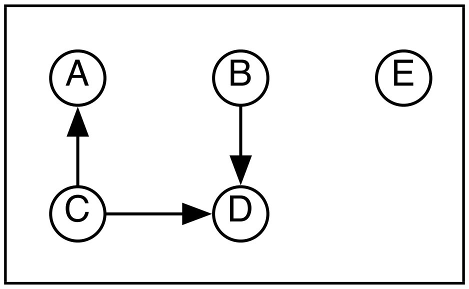


Figure 5.1: Connecting subgraphs and prompting strategies

The global prompting strategies introduced in section 3.6.1 did not appear to be useful since they were not successful in connecting subgraphs. But those same strategies could be considered as secondary strategies where they determine how subgraphs are connected. So with the annotation graph in figure 5.1 we always first look to connect subgraphs. This will constrain our choices for user-prompting to ⟨A E⟩, ⟨B E⟩, ⟨C E⟩, and ⟨D E⟩. Then the strategies can be employed to choose between these four. I expect that density figures would go up a bit.

SputLink was deliberately kept as simple as possible. It ignores ALINKs and SLINKs but both these link types can carry temporal information and in some cases this could be exploited by automatically adding TLINKs between events that are linked by an SLINK or ALINK. For example, the TLINK *after* is often implied in a SLINK with relation type evidential and a subordinating event in past tense, as in example (34) below.

(34) The janitor *said* that he *saw* the burglar around midnight.

We could add a component to SputLink that knows what to do with these subordinating links, but it is probably cleaner to relegate that component to a pre-processing stage.

Another simplification is that SputLink engages in qualitative reasoning only. But TimeML does include some quantitative information, most notably in the durations. Adding that information to SputLink would facilitate more inferences.

The effects of genre on the interaction between closure and annotation needs to be studied as well. Section 4.1.2 presented some data on the video corpus, a genre that is much closer to the narrative convention than the newswire articles in TimeBank. It turned out that the average link span after initial annotation is shorter for the video corpus, but that average link span after closure is similar (in fact, for the video corpus it was much closer to the baseline of a complete annotation than it was for the TimeBank articles). It seems that some genres make it easier to create a connected annotation graph.

This dissertation did not investigate the many issues pertaining to the interface between computer and human. One could expect that the role of the closure component in an annotation environment is different if the user interface is different. For example, I expect that a fully graphical annotation tool like Tango would make it easier to create a connected graph than a table-based annotation tool like Alembic. No evidence was presented to support this. Also, there is no indication that the average link span differs depending on the tool used. That is, a graphical tool does not necessarily help the annotator create non-local temporal links. More experimental data are needed.

SputLink operates at a purely extensional level and TLINKs are interpreted purely extensionally, that is, TLINKs can be created (by annotator or SputLink) between any two events, no matter whether those events actually occurred or not. So no difference is made between reasoning in extensional contexts and reasoning in intensional contexts.

TimeML provides intensional information in its SLINKs and these SLINKs can be used to mark those events that occur in an intensional context. It may be worthwhile to look at ways to isolate these intensional contexts from the global application of relation composition, that is, no TLINKs will be drawn from an extensional event to an event in an intensional context. My take on this is that it is okay to not restrict TLINK creation as long as intensional events are marked as such. That way, a reasoning component can use information about how an intensional event relates temporally to other events, yet still have access to the fact that an event is intensional.

Finally, I have not discussed at all the storage requirements for complete annotations. Clearly, the space complexity is quadratic to the number of events if all temporal information is made explicit. This is tedious for larger documents and, more importantly, not feasible if the temporal information from many thousands or millions of documents are merged.

So some mechanism to condense the results is needed as well as a mechanism to access implicit information in the condensed representation. One way to do this is to create a minimal annotation, that is an annotation that has all the facts that SputLink needs to expand the annotation back to its complete or near complete state.[1] Queries can now not be matched directly to a temporal fact in the annotation, but a single-pair shortest path algorithm can be used to connect two events temporally.

[1]A real simple way to condense an annotation is to delete all TLINKs that were generated by closure and keep only those TLINKs that were added by the annotator during initial annotation or during the user-prompting phase. This is not necessarily a minimal annotation though.

Another way to condense the annotation is to use the notion of a chronicle (also called trajectory or biography). A chronicle is defined as a subgraph of an annotation where all events contain a certain entity, for example a person or a state. Chronicles can be used to answer questions like the following.

(35) Did Albert Einstein ever meet Marilyn Monroe?

The Einstein and Monroe chronicles can be merged by merging the times that the events in the two chronicles anchor to and by merging identical events in the two chronicles. Temporal closure can be applied to the merged graph and all temporal information available is made explicit.

# References

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

James Allen. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23(2):123–155, July.

Chinatsu Aone and Mila Ramos-Santacruz. 2000. REES: A Large-Scale Relation and Event Extraction System. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP-2000).*

David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. 1997. 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.

David Day, Lisa Ferro, Robert Gaizauskas, Patrick Hanks, Marcia Lazo, James Pustejovsky, Roser Saurí, Andrew See, Andrea Setzer, and Beth Sundheim. 2003. The TimeBank Corpus. *Corpus Linguistics*, March.

Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. 2001. TIDES Temporal Annotation Guidelines, version 1.0.2. Technical report, The MITRE Corporation, McLean, Virginia. Report MTR 01W0000041.

Elena Filatova and Eduard Hovy. 2001. Assigning Timestamps to Event-Clauses. In *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*, pages 88–95, Toulouse, France.

Christian Freksa. 1992. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54(1):199–227.

Antony Galton. 1990. A Critical Examination of Allen's Theory of Action and Time. *Artificial Intelligence*, 42:159–188.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference 6: A Brief History. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING-96).*

Marti A. Hearst, James F. Allen, Eric Horvitz, and Curry I. Guinn. 1999. Mixed-Initiative Interaction. *IEEE Intelligent Systems*, 14:14–23.

Jon Heggland. 2002. OntoLog: Temporal Annotation Using Ad Hoc Ontologies and Application Profiles. URL: citeseer.nj.nec.com/542250.html.

Lynette Hirschman, Patricia Robinson, John Burger, and Marc Vilain. 1998. Automatic Coreference: The Role of Annotated Training Data. In *AAAI 1998 Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 1419–1422, Stanford, USA.

Graham Katz and Fabrizio Arosio. 2001. The Annotation of Temporal Information in Natural Language Sentences. In *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*, pages 104–111, Toulouse, France. Association for Computational Linguistics.

Michael Kipp. 2001. ANVIL: A Generic Annotation Tool for Multimodal Dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, Aalborg, September.

Robert Kowalski and Marek Sergot. 1986. A Logic-Based Calculus of Events. *New Generation Computing*, 4:67–95.

Robert Kowalski. 1992. Database Updates in the Event Calculus. *Journal of Logic Programming*, 12:121–146.

Inderjeet Mani and George Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, pages 69–76, New Brunswick, New Jersey.

Inderjeet Mani, Barry Schiffman, and Jianping Zhang. 2003. Inferring Temporal Ordering of Events in News. In *Proceedings of the Human Language Technology Conference (HLT-NAACL'03)*.

John McCarthy and Patrick J. Hayes. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press. reprinted in McC90.

Drew McDermott. 1923. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, 6:101–155.

MUC7. 1998. Proceedings of the Seventh Message Understanding Conference (MUC-7). Available at
http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html.

Oliver Plaehn and Thorsten Brants. 2000. Annotate – An Efficient Interactive Annotation Tool. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Arthur Prior. 1957. *Time and Modality.* Clarendon Press, Oxford.

Arthur Prior. 1967. *Past, Present and Future.* Clarendon Press, Oxford.

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. 2002. TERQAS Final Report. Technical report, The MITRE Corporation, Bedford, Massachusetts.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5 Fifth International Workshop on Computational Semantics.*

James Pustejovsky, Inderjeet Mani, Luc Belanger, Linda van Guilder, Robert Knippen, Andrew See, Jon Schwarz, and Marc Verhagen. 2003b. TANGO Final Report. Technical report, The MITRE Corporation, Bedford, Massachusetts.

Raymond Reiter. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, Cambridge, Massachusetts.

Frank Schilder and Christopher Habel. 2001. From Temporal Expressions To Temporal Information: Semantic Tagging Of News Messages. In *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*, pages 65–72, Toulose, France, July.

Frank Schilder. 1997. *Temporal Relations in English and German Narrative Discourse.* Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Andrea Setzer and Robert Gaizauskas. 2001. A Pilot Study on Annotating Temporal Relations in Text. In *ACL 2001, Workshop on Temporal and Spatial Information Processing.*

Andrea Setzer. 2001. *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study.* Ph.D. thesis, University of Sheffield, Sheffield, UK.

Peter van Beek. 1992. Reasoning about Qualitative Temporal Information. *Artificial Intelligence*, 58:279–326.

Johan van Benthem. 1983. *The Logic of Time.* Kluwer Academic, Dordrecht.

Marc Vilain, Henry Kautz, and Peter van Beek. 1990. 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.

Špela Vintar and Michael Kipp. 2001. Multi-Track Annotation of Terminology Using Anvil. In *Proceedings of the Workshop on Multi-layer Corpus-based Analysis*, Eurolan.