


12-2004

Generating Linear Orders of Events for Geospatial Domains

Suzannah Hall

Follow this and additional works at: <http://digitalcommons.library.umaine.edu/etd>

 Part of the [Databases and Information Systems Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Hall, Suzannah, "Generating Linear Orders of Events for Geospatial Domains" (2004). *Electronic Theses and Dissertations*. 571.
<http://digitalcommons.library.umaine.edu/etd/571>

This Open-Access Thesis is brought to you for free and open access by DigitalCommons@UMaine. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DigitalCommons@UMaine.

**GENERATING LINEAR ORDERS OF EVENTS FOR
GEOSPATIAL DOMAINS**

By

Suzannah Hall

B.S. University of Maine, 2002

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

December, 2004

Advisory Committee:

Kathleen Hornsby, Assistant Research Professor, National Center for Geographic
Information and Analysis, Co-Advisor

Max J. Egenhofer, Professor of Spatial Information Science and Engineering,
Co-Advisor

M. Kate Beard-Tisdale, Professor of Spatial Information Science and Engineering

© 2004 Suzannah Hall

All Rights Reserved

GENERATING LINEAR ORDERS OF EVENTS FOR GEOSPATIAL DOMAINS

By Suzannah Hall

Thesis Co-Advisors: Dr. Kathleen Hornsby
Dr. Max J. Egenhofer

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Spatial Information Science and Engineering)
December, 2004

Events in the world do not occur in neat chronological order, but may take place, for example, *during* or *overlapping* each other, or as simultaneous events. Efficient summaries of real-world events are important in many disciplines and require events to be modeled in a linear fashion. This thesis focuses on modeling events as intervals and using relations between the events to derive linear orders from more complex partially-ordered sets. A method is developed for mapping Allen's thirteen temporal relations to only two relations, *before* and *equals*, which allow a linear ordering of all events present in the set. This mapping requires additional constraints to preserve semantics of the original relations as the orders are generated. Depending on the relations present, several linear orders may be possible, and methods are discussed of filtering the possible orders so as to present only the most plausible orders to a user. The result is a methodology that allows plausible linear orders to be automatically generated from partially-ordered sets of events.

DEDICATION

This thesis is dedicated to Laura Payeur

ACKNOWLEDGMENTS

First I would like to thank my advisors. I thank Kathleen Hornsby for her immense help, encouragement, and patience. I thank Kate Beard for her support all through my undergraduate and graduate career, and Max Egenhofer for his help and suggestions.

Financial support from the University of Maine Provost Fellowship, and the National Geospatial-Intelligence Agency under grant number NMA201-00-1-2009, is gratefully acknowledged.

I thank my parents, Stephen and Roxanna Hall, and my sister and brother, Kate and Adam, for everything they have ever done for me! There is no other way to thank them for all that they have contributed to the completion of this thesis. The support of my extended family is also very much appreciated.

Eeva, Connie, and Karen – I thank them for all the help they have given me over the years, in so many areas! Their support has been invaluable.

I thank my friends, especially Kristin, Melanie, Amber, Katie and Matt, Julie, Kathy, Sarah, Rose and Dan (and Daniel!). For the love they have shown me, their prayers, and the laughter we've shared – I owe them all so much.

And I thank Maggie Wilcox for being the first person to encourage me to pursue a graduate degree.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	xi
Chapters	
1 INTRODUCTION	1
1.1 Motivation and background for research	1
1.2 Research questions	4
1.3 Goal and hypothesis	5
1.4 Scope of thesis	6
1.5 Thesis structure	6
2 MODELING EVENTS IN INFORMATION SYSTEMS	8
2.1 Modeling events as intervals	11
2.2 Ordering event intervals	13
2.2.1 Linear and partial orders	13
2.2.2 Topological sorts	16
2.3 Summary	20
3 GENERATING LINEAR ORDERS OF EVENTS	21
3.1 Mapping event interval relations to <i>before</i>	21
3.2 Ordering events	27
3.2.1 Generating an initial topological sort	27
3.2.2 Performing a topological sort to generate all possible orders	32
3.3 Summary	34

4	REFINING ORDERS BASED ON RELATION SEMANTICS.....	36
4.1	Relation semantics	37
4.1.1	Semantics of <i>before</i> and <i>after</i>	38
4.1.2	Semantics of <i>during</i> and <i>contains</i>	38
4.1.3	Semantics of <i>meets</i> and <i>met_by</i>	40
4.1.4	Semantics of <i>overlaps</i> and <i>overlapped_by</i>	40
4.1.5	Semantics of <i>starts</i> and <i>started_by</i>	41
4.1.6	Semantics of <i>ends</i> and <i>ended_by</i>	41
4.1.7	Semantics of <i>equals</i>	42
4.2	Ordering events with constraints	44
4.3	Summary.....	49
5	USING EVENT LOCATION IN DETERMINING ORDERS OF EVENTS	51
5.1	Event location granularities	51
5.2	Determining spatial relevancy of events.....	56
5.2.1	Hierarchical clustering	56
5.2.2	Cases where event locations are not specified	59
5.3	Filtering orders based on event locations	60
5.3.1	Detecting a linear pattern in event location data.....	60
5.3.2	Evaluating orders for correspondence with a linear trend	62
5.3.3	Cases where event locations are not specified	65
5.4	Summary.....	65
6	EVALUATION OF RESULTING ORDERS.....	66
6.1	Comparing orders	67
6.2	First test scenario: World War I events	69
6.3	Second test scenario: Iron Curtain.....	73
6.4	Hypothesis evaluation.....	76
6.5	Summary.....	76

7	CONCLUSIONS.....	78
7.1	Summary.....	78
7.1.1	Constraints based on semantics of relations.....	80
7.1.2	Filtering orders based on spatial information.....	81
7.1.3	Evaluation	82
7.2	Conclusions.....	83
7.3	Future work.....	85
7.3.1	Use of explicit temporal information in the ordering process.....	85
7.3.2	Detection of other spatial patterns of event locations	85
7.3.3	Further evaluation of the <i>overlaps/overlapped_ by</i> relations.....	86
7.3.4	Automatic extraction of event-relation combinations.....	87
	BIBLIOGRAPHY	88
	BIOGRAPHY OF THE AUTHOR.....	93

LIST OF TABLES

Table 4.1	Mappings of all thirteen temporal interval relations to <i>l_before</i> , and constraints based on the semantics of each relation.	43
Table 6.1	Events in World War I test scenario.	69
Table 6.2	Events in Iron Curtain test scenario.	74
Table 7.1	Mapping rules to <i>l_before</i> and <i>l_equals</i> for each of the thirteen relations.	79
Table 7.2	Mapping rules and constraints for each event interval relation.	81

LIST OF FIGURES

Figure 1.1	Thirteen temporal interval relations (after Allen, 1983).	3
Figure 2.1	Timelines summarize the occurrence of events.....	14
Figure 2.2	Two partial orders (after Allen, 1991).....	15
Figure 2.3	A cyclic graph (a), and an acyclic graph (b).	16
Figure 3.1	Six cases where $A R B$ maps to $A l_before B$	23
Figure 3.2	Cases where $A R B$ maps to $B l_before A$	24
Figure 3.3	A equals B maps to A equals B	25
Figure 3.4	A matrix E generated from the relations present in the set of event-relation combinations from the example weather scenario, and their corresponding inverse relations.	29
Figure 3.5	The matrix E after removing LowPressureMoves and Rain	31
Figure 4.1	Given A meets B and A starts C , possible orders are $A B C$ and $A C B$	45
Figure 4.2	(a) Given A meets B , A and B may be separated in a linear order if another event occurs during event A . (b) Given A meets B , A and B may be separated in a linear order if B is during another event which does not contain event A	46
Figure 5.1	Hierarchy of event location granularities.	52
Figure 5.2	The $(n+1) \times (n+1)$ proximity matrix D , showing the Euclidean distance between each pair of event locations.....	57
Figure 5.3	Arrangement of point locations in (a) nonlinear and (b) linear formations.....	60
Figure 5.4	Perpendicular lines from event locations to a line of best fit, showing each event's assigned position along the line.....	63

Figure 5.5 Plot showing event locations and line of best fit.....64

Chapter 1

INTRODUCTION

1.1 Motivation and background for research

In every discipline, there is a need for access to information. The basis for access to information is often abstractions from more detailed sources. News analysts and journalists, for example, are interested in summaries of news articles (Allan et al. 2001; Mani and Maybury 2001), and epidemiologists are interested in being notified of new outbreaks of infectious diseases (Grishman et al. 2002). Happenings in the world are dynamic; for example, events occur one after another, during other events, or overlapping each other. Studies of the dynamic nature of real-world events in a computational setting have focused on modeling *events*, that is, changes to an entity over time. Representing real-world happenings as events facilitates the processing and conveying of information to a user (Zacks and Tversky 2001).

In the context of databases, events are often modeled as being instantaneous, such as an update to a bank account or the transmission of an electronic message (Hinze and Voisard 2002). Events modeled in this way are essentially changes of state, and have no duration. Alternatively, events are modeled as having duration but in order to simplify processing, each event is associated with a specific point in time, usually the point in time at which the event finishes (Motakis and Zaniolo 1995; Galton and

Augusto 2002). In cognitive psychology and linguistics, events are generally modeled as durative (Larson 1999; Pedersen and Wright 2002), and human perceptions of their durations are of interest to researchers in these fields.

Automatic extraction of events, and temporal information about them, from texts such as news and journal articles, is also a topic of interest to researchers. Events extracted from text are used to build summaries, or they may be added to a databases (Allan et al. 2001; Grishman et al. 2002; Pustejovsky et al. 2003).

Because all events have a temporal component, reasoning about events often involves arranging them in a sequence or *order* according to their temporal relations to each other (Alfonseca and Manandhar 2002). Though the relations between all events in a set may or may not be known (Pustejovsky et al. 2003), a user often requires a simple, linear order of events in which for any pair of events, A and B , either A is *before* B , B is *before* A , or both (in which case A and B are simultaneous) (Frank 1998). Allen (1983) developed a temporal logic involving an exhaustive set of thirteen possible relations between two temporal intervals (Figure 1.1). Events that are durative (i.e., take place over an interval of time), are referred to in this thesis as *event intervals*. Relations that hold between temporal intervals also hold between event intervals. In this thesis, scenarios of events are modeled using event intervals. The event intervals are related to each other by combinations of the thirteen interval relations. From these event intervals and relations between them, a method is presented that automatically generates linear orders. The semantics associated with the original scenario of events

are retained as much as possible with the resulting orders, generating a set of *plausible* linear orders.

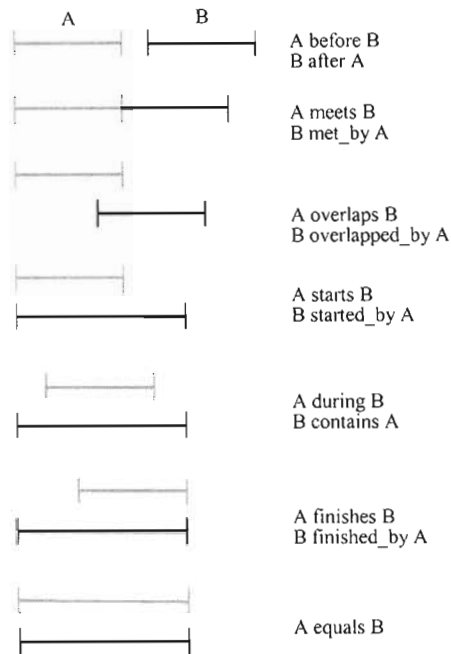


Figure 1.1 Thirteen temporal interval relations (after Allen, 1983).

A method of automatically generating orders of events is necessary. This approach will be of use to researchers in areas such as natural language processing, event extraction, databases, and question answering, as well as professionals in fields where informative summaries are needed and time is limited.

Given a description of a scenario of events, ordering the events begins with mapping the thirteen interval relations to either *before* or *equals*. This mapping process can also be referred to as *flattening*, as it essentially forces relations into one dimension. Once this flattening is performed, linear orders are generated. There may be many possible linear orders generated from a given set of events. It is important to ensure the

plausibility of the orders which are presented to a user. That is, all of the orders should be consistent with the knowledge present in the original set of events and relations. Preserving the semantics attached to the original thirteen event interval relations, as well as checking for temporal and spatial consistency, are some ways of increasing the overall plausibility of the orders generated.

1.2 Research questions

The automatic generation of linear orders of events from event descriptions, and the implications of the ordering procedure, involve a number of open research questions:

- What criteria should be used to determine which event or pair of events begins an order?
- How are subsequent event intervals placed in the order?
- Do the orders generated retain the semantics of the original event scenario?

When the relations in an event description are flattened to *before* and *equals*, the result is a simpler rendition of this event description. Orders will be generated using this simpler rendition. The results of the ordering process bring their own questions, for example:

- Are all orders equally plausible? Which orders should be presented to a user?
- Are there meaningful filters that could be applied, either to the event description or to the orders generated from it, that would render the results more meaningful to a user?

1.3 Goal and hypothesis

The goal of this thesis is to understand how partial orders of events can be used in generating plausible linear orders, for the purposes of simplification and summarization of a scenario of events. A mapping is developed based on reducing the thirteen event-interval relations to two relations, *before* and *equals*, and the result of the mapping is a revised (simpler) description of the scenario. This revised description serves as the foundation for a linear ordering of events.

Numerous linear orders often result from partially ordered sets of events. *Partial orders of events* refer to sets where it is not known how every event relates to every other event. While in some cases a complete set of possible orders may be necessary, there is also a possibility that a user may be presented with more orders than can be easily understood and, therefore, more orders are generated than are really useful. For this reason, methods for filtering the orders must also be developed. These methods of filtering include the use of constraints based on the semantics of the original event interval relations, and spatial filtering in cases where locations of the events are available.

The hypothesis of this thesis is:

Constraints that enforce the original semantics of the event interval relations are necessary in the generation of plausible linear orders.

1.4 Scope of thesis

This thesis describes an approach for mapping the thirteen event interval relations to *before* and *equals*, the result of which is a revised description of the scenario of events. A systematic method of generating linear orders from this revised description is presented. The semantics of the original thirteen relations are preserved in the ordering process by assigning to each relation a constraint (or set of constraints) that prevents orderings that would be possible given only the information in the revised set, but are not plausible based on the original description of the scenario. Methods of filtering multiple orders are presented, based on the temporal and geographic information available about the events.

The work in this thesis assumes a linear model of time. Cases of cyclic, branching, or other structures of time are not addressed. This thesis does not describe methods for extracting events from, for example, text. We assume an existing dataset, consisting of events and event interval relations, is available for the construction of orders.

1.5 Thesis structure

The remainder of this thesis is structured as follows:

Chapter 2 provides an overview of previous work that has been done with event modeling and event ordering. It also provides an overview of a topological sorting algorithm, and its relevance to this research. Chapter 3 introduces a methodology for generating linear orders from complex scenarios of events, using the topological sorting algorithm. This results in a set of all possible orders of the events in the

scenario. Chapter 4 discusses approaches to filtering these orders, using constraints developed based on the semantics of the thirteen temporal interval relations. For example, the semantics of the *meets* relation indicate that intermediate events should not come between two events in a linear order that are known to *meet* each other. In Chapter 5, two filtering methods based on event locations are presented. One method evaluates the spatial relevance of events, and eliminates events that are not spatially relevant to other events in the set. The second method evaluates the set of event locations for evidence of a linear trend, and, if such a trend is found, highlights orders that correspond to this geographic linearity. Chapter 6 describes a procedure of evaluating the validity of the final set of orders. A reference order, provided by an external source, is compared to the orders in the final set of generated orders. High levels of similarity between the reference order and the orders in the set would indicate that the orders in the final set are plausible, and the ordering and filtering methodologies are producing good results. Chapter 7 presents the major results and conclusions derived from this research, and outlines future work in this area.

Chapter 2

MODELING EVENTS IN INFORMATION SYSTEMS

Modeling dynamic happenings or *events* has been an important area of study and the basis for new computing models in geographic information science (Frank 1998; Yuan 2001; Grenon and Smith 2004), databases (Gehani et al. 1992; Chakravarthy and Mishra 1994; Galton and Augusto 2002), artificial intelligence (Allan et al. 2001; Alfonseca and Manandhar 2002; Knight and Marcu 2002), cognitive psychology (Larson 1999; Zacks and Tversky 2001; Pedersen and Wright 2002) and linguistics (Grishman et al. 2002; Pustejovsky et al. 2003). In 2002, a week-long international workshop, ACTOR, was held in Holden, Maine on action-oriented approaches in geographic information science (<http://www.spatial.maine.edu/~actor2002/>). Participants at the conference discussed motivations for modeling the dynamic nature of events and processes that exist in the real world, as well as methods of building these models. A research agenda was established for using and developing these approaches in geographic information science.

Grenon and Smith (2004) propose an ontology that is a combination of reasoning based on space and time as separate entities (states of the world at a particular place and time) with reasoning based on objects as inherently spatio-temporal (changes and transformations in the world). This is because the state of a spatial entity at a particular moment in time, and the changes that a spatial entity may undergo over time, are both

important. Answering questions about one should not have to happen in a system designed for the other. In Grenon and Smith's ontology, entities are not said to exist within a specific geographic space, but in a specific *spacetime*. Thus, a geographic entity is part of both a specific place and a specific time. Changes to geographic entities are entities themselves, and are referred to as processes. Events are defined as boundaries of processes, or the boundaries of transitions within processes.

Grenon and Smith's separation between processes and events is similar to Allen and Ferguson's (1994) work on actions and events. In their temporal logic, an action is something a person or robot might do, such as walking or running a program, whereas an event refers to some occurrence or change of state, such as arriving at a destination or accomplishing a task. Events are often the result of some action.

In this thesis, the ideas of events and actions are treated as one, and an event is defined simply as a change to an entity over time. This can involve an action as defined by Allen and Ferguson, such as "the car crossed the bridge" or an occurrence that is the result of an action, such as "the car arrived on the other side of the bridge," which in the above models would be considered an event.

Database approaches to event modeling often use active databases in a wide variety of applications, such as network management, engineering design, and air traffic control (Chakravarthy and Mishra 1994). Active databases have been used in research on detecting common or important sequences of events and returning a set of actions to take with little or no immediate human input (Gehani et al. 1992; Chakravarthy and Mishra 1994). Database researchers have treated two types of events: primitive events

and composite events. Primitive events are those that are defined within the system, these are usually occurrences such as creation or deletion of an object in the database, or changes to a database object. Composite events are more complex, comprising several primitive events. A composite event occurs when the last of its component primitive events is complete (Gehani et al. 1992; Chakravarthy and Mishra 1994).

In this work, events are modeled as primary events, and do not formally encompass other events.

In the language processing field, there is significant interest in research on extracting events from text in order to gather important information from the text without requiring a whole article or series of articles to be read by one individual (Yang et al. 1999; Koen and Bender 2000; Alfonseca and Manandhar 2002; Grishman et al. 2002; Knight and Marcu 2002; Pustejovsky et al. 2003). Many early efforts in automatic text summarization involved extracting the most important sentences or clauses from a text (Knight and Marcu 2002). However, the results of this process have been shown to be less coherent than a person's summarization of the same article. Articles do not always present events in the order in which they happened. While extracted sentences may describe the events themselves effectively enough, they can leave out temporal clues that are important to a reader's comprehension of the article as a whole. Therefore, researchers have focused on using sentence compositions to create more effective summaries from texts (Alfonseca and Manandhar 2002).

One method of generating summaries from text, for example, news articles, is to search a document for sentence parts (e.g., nouns, verbs, etc.), key words dealing with

temporal relations between events (e.g., before, after, during, meanwhile), and key words that refer to the temporal relation between an event and the time that the document was written (e.g., yesterday, two weeks ago). Verbs and some nouns can then become events, and the temporal information is used to date the events (Pustejovsky et al. 2003). In *TimeML*, a specification language for event and temporal descriptions in text (Pustejovsky et al. 2003), temporal relations between objects are identified and mapped to one of the thirteen temporal relations identified by Allen. This has the potential to allow question-answering for such queries as those beginning with “Who/what is currently...” or “When did...” (Pustejovsky et al. 2003).

The remainder of this chapter discusses event modeling with respect to representing events as intervals rather than instantaneous entities, and provides definitions and implications of partial and linear orders of events. A description of topological sorts and their relevance to this research concludes the chapter.

2.1 Modeling events as intervals

When performing reasoning based on temporal knowledge, treating all events as instantaneous leads to some conclusions that are counterintuitive (Allen 1983). For example, the assumption that the transition between a door being open and the door being closed has no duration, leads us to conclude that there was either a time at which the door was neither open nor closed, or that there was a time at which it was both open and closed. One solution to this difficulty is to model states (e.g., door is open, door is closed) as intervals that are closed on one end and open on the other. This effectively

gives each state only one endpoint, however, which is unreasonably artificial (Allen, 1983). If events are modeled as intervals rather than instantaneous entities, it becomes possible to allow a moment for the door to close and this problem is avoided. An added benefit of representing events as intervals rather than points is that temporal relations between events can be better preserved (Fujimoto 1999). Relations such as *during* and *overlaps*, which are not meaningful when applied to instantaneous events, can be applied to intervals (and thus, events modeled as intervals). This, in turn, enriches the potential knowledge base about events and how they relate to each other.

In cognitive psychology and linguistics, events are generally modeled as durative (Larson 1999; Zacks and Tversky 2001; Pedersen and Wright 2002). The duration of a given event can be very relevant in such scenarios as a court case where a witness observed a suspect an estimated half hour after the crime was committed, and the suspect was captured 50 miles away exactly 40 minutes after the crime. Is the witness sure of the time? Is the witness sure that it was the suspect that she saw? Psychologists are interested in how a different description of an event may affect a person's perception and memory of the event, and also knowing how much exposure to an event is necessary to accurately remember events as they happened (Larson 1999; Pedersen and Wright 2002).

In some military simulation algorithms, modeling events as instantaneous results in a processor having to handle all events in a scenario in a very precise sequence. When events are modeled as intervals, however, some overlapping of the events and slightly looser requirements for when they must be processed are possible (Fujimoto 1999).

The uncertainty of when an event actually should take place is exploited to make processing more efficient.

In this thesis, events are modeled as having duration. Each event has a start point and an end point, and while the start point and end point may be nearly at the same time, the start point and end point of a single event are never exactly simultaneous. The start point of the event always precedes the end point of the event.

2.2 Ordering event intervals

There are different types of orders, based on the amount of information available about the relations between elements in the order. The first type of order we discuss is a linear order, where all elements in the order are arranged sequentially. A more complex type of order is a partial order, in which some relation information about events is unknown.

2.2.1 Linear and partial orders

A linear order refers to a sequence in which the relations between all elements in the order are known. For any events A , B in a linear order, it holds that either $A \prec B$, $B \prec A$, or $A = B$, read as A precedes B, B precedes A, or A is equal to B, respectively (Frank 1998). A linear order of event intervals (i.e., a set of events arranged sequentially, one after another) is a useful reasoning tool for many scenarios, such as Fujimoto's military simulation case (Fujimoto 1999). It is also a natural way of thinking of and explaining events (e.g., I ate lunch and then I went to the store) (Allen

1991). A linear order of events is an effective tool for summarizing information. As an example, consider a timeline in a history book as such a summary (Figure 2.1). The timeline allows a reader to gain a basic understanding of a historical sequence of events at a glance, without reading all the details of an entire paragraph or chapter.

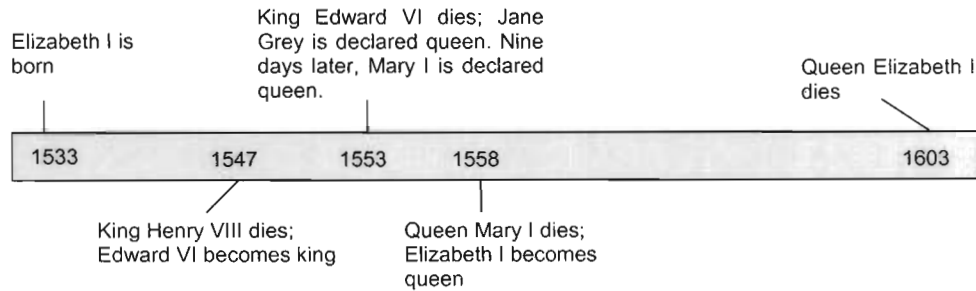


Figure 2.1 Timelines summarize the occurrence of events.

A partial order is defined as any set of events on which there is a binary relation between events that is reflexive, antisymmetric, and transitive (Kainz et al., 1993). In contrast to a linear order, a partial order is one in which some relations between events may be missing and not known (Lamport 1978; Allen 1991; Proveti 1996). This occurs, for example, if the same events are observed by different witnesses (Frank 1998). In this case, certain events are known to have happened, but their effects on, or relations to each other, are not necessarily known. Any orders generated from partial knowledge will essentially be orders that *could* happen (Lamport 1978).

Reasoning is possible based on orders that are generated from only partial information, but care must be taken to avoid unwittingly assigning relations among events that were not present in the original knowledge base, potentially resulting in orders that could not have happened (Allen 1991). For example, assume that event *e*l

took place between time $t1$ and $t3$, and event $e2$ took place between time $t6$ and $t7$. Given that events $e3$ and $e4$ both take place between $e1$ and $e2$, but without knowing their times, it would be possible to assume that both $e3$ and $e4$ took place between times $t4$ and $t5$. When diagramming this scenario (Figure 2.2), the sequence $e1, e3, e2$ is a partial ordering, as is the sequence $e1, e4, e2$. Separately, they are consistent with the knowledge given. However, when the two orders are compared it seems to show that $e3$ and $e4$ are simultaneous. While it is *possible* that $e3$ and $e4$ are simultaneous, it was not part of the original knowledge base and, therefore, should not be assumed. As more events are added and more uncertainties exist, additional situations like this will arise and care must be taken not to make unsound assumptions.

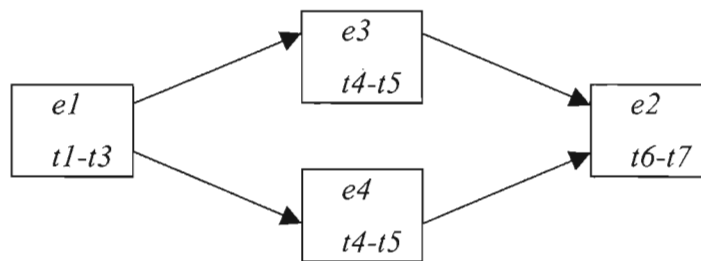


Figure 2.2 Two partial orders (after Allen, 1991).

Information about a scenario may in some cases be sufficient to allow the generation of a linear order, but often the available information is incomplete. It is important to be able to generate orders, even partial ones, which accurately represent the available information about a scenario. Partial orders are important to this research because they are a way of representing the kind of information that is usually available.

2.2.2 Topological sorts

A partial order can be visually represented in the form of a directed acyclic graph, or *DAG* (Allen 1983). A directed graph is one in which the links between the nodes (or vertices) of the graph have a direction. Thus any pair of nodes that are directly linked to each other may also be represented as an ordered pair, such as (a, b) if there is a link going from node a to node b in the graph. In this case, node a is called the origin of node b , and node b is called the destination of a . To say that a graph is acyclic means that it contains no cycles. This means that, beginning at a node a and following the links according to their direction, one must not be able to reconnect to node a . Examples of a cyclic and acyclic graph are shown in Figure 2.3. Notice that in Figure 2.3a there is a cycle, $a-d-c-b-a$, whereas in Figure 2.3b there are no cycles present.



Figure 2.3 (a) A cyclic graph, and (b) an acyclic graph.

When representing a scenario of events as a DAG, each event in the scenario maps to a node, and the links between the nodes are temporal relations. A DAG representing a scenario of events would in effect have thirteen types of links, one for each temporal relation.

A *topological sort* is a method of arranging all of a DAG's nodes in an order such that for every ordered pair present in the graph, the origin node comes before the

destination node (Lipschutz and Lipson 1997; Skiena 1998). Topological sorts are used in applications where efficient traversal of graphs is important, such as working with Bayesian networks (Charniak 1991), search algorithms (Knuth and Szwarcfiter 1974), or planning/decision-making algorithms (Allen 1991; Skiena 1998). For the acyclic graph (Figure 2.2b), one topological sort would be the order $a-d-b-c$. Since in this case d and b have the same origin and destination and no other links, the order $a-b-d-c$ is equally valid.

Algorithms for generating a single valid topological sort are well documented (Lipschutz and Lipson 1997; Skiena 1998). In one algorithm, all nodes with no origin are put at the beginning of a list, then those nodes and all links attached to them are deleted from the graph. Some of the remaining nodes in the graph will then have no origin, and these nodes are added to the list and are deleted from the graph along with all links attached to them. These steps are repeated until all nodes in the original graph are in the list. The list is a topological sort of the original graph (Lipschutz and Lipson 1997).

There is often more than one possible topological sort of any given DAG, and the problem of computing all topological sorts is complex (Knuth and Szwarcfiter 1974; Varol and Rotem 1981; Skiena 1998). As Skiena (1998) notes, the problem is actually NP-hard, meaning that it has no predictable, exact solution. One type of algorithm to generate all topological sorts of a DAG is similar to the algorithm for a single topological sort, but involves backtracking to nodes that in the single sort algorithm would remain deleted. This type of program to generate all topological sorts generally

involves recursion, which can be difficult for certain processing systems to manage (Knuth and Szwarcfiter 1974; Skiena 1998). Knuth developed an iterative solution to the problem in his structured program to generate all topological sorting arrangements (Knuth and Szwarcfiter 1974).

Another type of iterative algorithm was introduced in 1981 by Varol and Rotem. This algorithm takes as input one valid topological sort of all the objects in the DAG, and all the known information about their relations to each other in the form of ordered pairs. The objects in the input topological sort are systematically transposed such that all possible arrangements of the objects that do not violate the known relations are eventually generated (Varol and Rotem 1981). In this thesis, a topological sorting algorithm developed by Ruskey (1995), based on the Varol and Rotem algorithm, is used to generate all possible orders of a set of events, given the available relations between events in the form of ordered pairs. As an illustration, consider four objects, a , b , c , d where there is a single known relationship, (a, b) , meaning that in any valid sort, a must precede b . It is also known that a, b, c, d is a valid topological sort of the objects. The first step in generating all topological sorts is to transpose the last object in the order with all of its left-hand neighbors, until it has been in every valid position. This first cycle of transpositions results in four sorts:

- 1 a, b, c, d
- 2 a, b, d, c
- 3 a, d, b, c
- 4 d, a, b, c

The next step is to transpose the next two objects to the left in the first sort of the previous cycle (which in this case is the original order, numbered 1), and then repeat the process of transposing the last object into all valid positions. So, b and c in sort 1 are transposed, and the first sort of the second cycle becomes a, c, b, d . When d is transposed with all objects to its left, another four sorts result:

5 a, c, b, d

6 a, c, d, b

7 a, d, c, b

8 d, a, c, b

Again, the next two objects to the left in the first sort of the last cycle, in this case sort 5, are transposed, making the first sort of the third cycle c, a, b, d . The last object is transposed with its left-hand neighbors, resulting in four additional sorts:

9 c, a, b, d

10 c, a, d, b

11 c, d, a, b

12 d, c, a, b

Since no new arrangements would result from any additional transpositions of sort 5, the next step would be to return to sort 1 and transpose objects a and b . However, since it is known that a must precede b in all valid sorts, it is not necessary to proceed with this step. All valid topological sorts of objects a, b, c , and d have been generated.

The sorts generated by such an algorithm may also be referred to as partial orders. They represent arrangements of the objects which *could* happen, but where the relationships between some objects remain unknown.

2.3 Summary

In this thesis, events are modeled as a change to an entity over time. Events are always assumed to be durative, that is, the start point and end point of an event may not be simultaneous. Scenarios of events can be presented in the form of linear *orders*. A *linear order* of events is possible in cases where all relations between events in the scenario are known. When some information about relations between events in the set is missing, *partial orders* are used to describe the scenario. A topological sorting algorithm is used to derive all possible linear orders of events, given a set of event pairs with relations between them, and one topological sort of the events in the scenario.

The next chapter discusses the operations necessary for putting a set of events related by a possible thirteen different relations into the form of the ordered pairs described above. The results of running the Varol-Rotem algorithm on a set of events is also discussed.

Chapter 3

GENERATING LINEAR ORDERS OF EVENTS

Chapter 2 reviewed the Varol-Rotem (1981) algorithm for generating all topological sorts given a set of ordered pairs. In this chapter, an ordering algorithm generates all possible orders of events, given a set of events related by temporal interval relations. As the ordering algorithm is based on the Varol-Rotem method, the input is similar and includes a set of ordered pairs. To allow a set of events related by temporal relations to take the form of a set of ordered pairs, this chapter also introduces a method for mapping each of the thirteen temporal interval relations to *before*.

3.1 Mapping event interval relations to *before*

When summarizing a series of events, it is common to express events that happened over a certain period of time as having happened one after the other. For example, a morning's activities might be summarized as, "I ate breakfast, read the paper, went shopping, and got a phone call from my friend," when in fact the person began reading the paper while eating, but finished after breakfast, and the phone call was received while the person was shopping.

A scenario involving temporal relations between such events may be expressed as a series of statements referred to in this thesis as *event-relation combinations*. An event-relation combination is of the form $A R B$, where A and B are two event intervals, and

R is one of the thirteen temporal interval relations (Section 1.1). “FinishedBreakfast *before* ReadNewspaper” and “PhoneCall *during* Shopping” are examples of event-relation combinations.

When all known relations between events in a scenario are expressed as event-relation combinations, the result is a set of event-relation combinations that completely describe the scenario. In this thesis, we assume that relevant event-relation combinations for a given scenario are known and, therefore, do not treat their derivation.

In a linear order of events, for any two events A and B , either A is *before* B or B is *before* A . For the relation *before*, the start point and end point of one event precede both the start point and end point of the second event. In a set of event-relation combinations describing a scenario in detail, however, events may be *during* or *overlapping* each other, or one event may *start* or *end* another. These relations are not strictly linear, because both the start point and end point of one event do not precede both the start and end point of the other. To place all events in a set of event-relation combinations into a linear order, we map each of the thirteen relations to *before*. Such a mapping transforms the thirteen interval relations from one domain into the summary domain containing only three relations, *before*, *after*, and *equals*. To distinguish the relations from the two domains, those in the summary domain (i.e., the linear orders) are called l_before , l_after , and l_equals . This mapping is done on the basis of start points, except where the start points of two events are simultaneous, which occurs in the relation *starts*. In this case, the mapping is based on the end points of the events.

When both the start points and end points of two events are simultaneous, the events remain equal in a linear ordering.

Given an event-relation combination involving two event intervals, A and B, if the relation between A and B is such that the start point of A is before or equal to the start point of B, and the end point of A is before the end point of B, then the relation maps to $A \textit{ l_before } B$. This holds for *A before B*, *A meets B*, *A overlaps B*, and *A starts B*.

A relation also maps to $A \textit{ l_before } B$ if the start point of A is before the start point of B, and the end point of B is *before* or *equal* to the end point of A. This applies in the cases of *A contains B* and *A ended_by B*. Figure 3.1 illustrates the six interval relations that map to $A \textit{ l_before } B$.

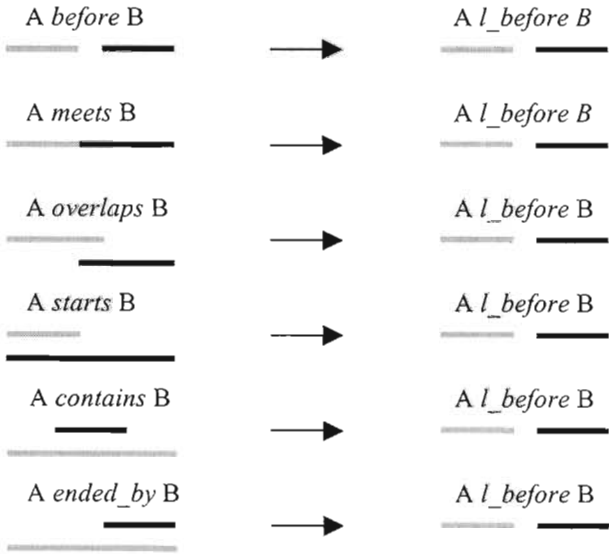


Figure 3.1 Six cases where $A R B$ maps to $A \textit{ l_before } B$.

For the case where the start point of A is after the start point of B, and the end point of A is before or equal to the end point of B, event B will map to *l_before* A. This holds for *A after B*, *A met_by B*, *A overlapped_by B*, *A ends B*, and *A during B*.

A relation also maps to *B l_before A* if the start point of B is before or equal to the start point of A, and the end point of B is before the end point of A, as in the relation *A started_by B*. The six cases that map to *B l_before A* are shown in Figure 3.2.

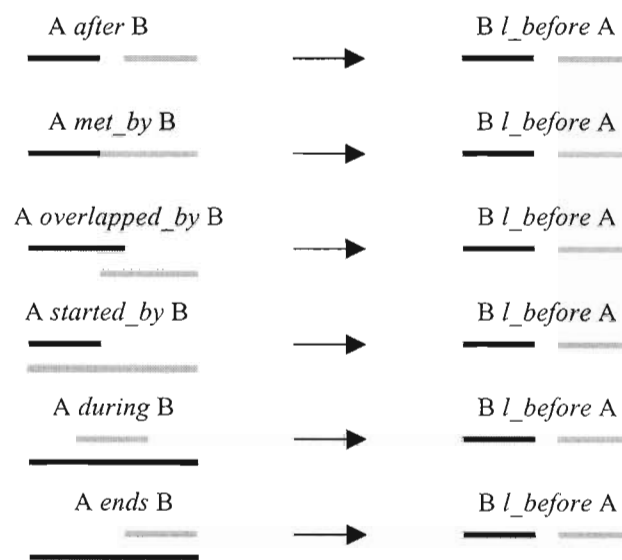


Figure 3.2 Cases where $A R B$ maps to $B l_before A$.

In the case of *A equals B*, both the start points and the end points of A and B are the same. For this case A and B will remain equal in a linear ordering, and the relation maps to *A l_equals B*, as shown in Figure 3.3.



Figure 3.3 *A equals B* maps to *A l_equals B*.

Thus, an order of events captures simultaneous occurrences of events as well as events that come one before another in a linear fashion.

When these mapping rules are applied to a set of event-relation combinations, the result is a revised set of event-relation combinations that now involve only the relations *l_before* or *l_equals*. As an example, consider a set of event-relation combinations involving weather events:

- LowPressureMoves *equals* Rain
- LowPressureMoves *contains* ColdFrontMoves
- ColdFrontMoves *starts* ScatteredShowers
- WestWinds *ends* LowPressureMoves
- NewLow *after* WestWinds
- NewLow *starts* Showers

In this set of event-relation combinations, there are n event intervals, and m relations in the set, where $m \leq 13$ is the number of possible event interval relations. In this example, the six event-relation combinations describe relations between $n=7$ different events and $m=5$.

Each event-relation combination in the set is mapped onto a relation involving only *l_before* or *l_equals*. The result is a revised description of the scenario, in which n remains the same, and m is reduced to either 1 or 2, depending on whether both

l_before and *l_equals* are involved. In the weather example, this mapping produces the following revised set of event-relation combinations where $n=7$ and $m=2$:

LowPressureMoves *l_equals* Rain
LowPressureMoves *l_before* ColdFrontMoves
ColdFrontMoves *l_before* ScatteredShowers
LowPressureMoves *l_before* WestWinds
WestWinds *l_before* NewLow
NewLow *l_before* Showers

This set of event-relation combinations is a linear description of the weather events, in contrast to the original set of event-relation combinations in which non-linear relations such as *contains*, *starts*, and *ends* were present. In the case where a set of event-relation combinations involves only *l_before* and *l_equal* relations, no mapping step is necessary.

Given a set of event-relation combinations involving only *l_before* and *l_equals* relations, it is possible to use the set as input for a sorting algorithm such as the one described in section 2.2, in which the input is a set of ordered pairs. An ordering algorithm based on the topological sorting concept is introduced in the following section. The algorithm generates all possible orders of the events from a set of event-relation combinations containing only the relations *before* and *equals*.

3.2 Ordering Events

Given a set of event-relation combinations involving only *before* and *equals*, an order is developed in which the n events are arranged sequentially according to their relations to each other. This is accomplished using the Varol-Rotem algorithm (Section 2.2.2) for generating all topological sorts. The *before* and *equals* relations between events are used as input for the algorithm, and all orders generated are consistent with these relations. For example, in the weather scenario described above, the relationship “LowPressureMoves *before* ColdFrontMoves,” as well as the other five relationships in the set of event-relation combinations, will be maintained for every one of the orders generated. That is, the algorithm will not generate any orders in which LowPressureMoves is not *before* ColdFrontMoves, or in which the relation in any other event-relation combination is not maintained.

In addition to the *before* and *equals* relations between events, input for the sorting algorithm includes one initial topological sort of the n events in a set of event-relation combinations. The method for generating this valid topological sort is described in the following section.

3.2.1 Generating an initial topological sort

Algorithms for generating a topological sort of a directed, acyclic graph accomplish their task by searching for nodes of the graph that have no incoming vertices. By creating a matrix containing the relations present in the set of event-relation

combinations, a similar method may be employed for generating an initial topological sort of a scenario of events.

A square matrix E is created by combining the n events from the set of event-relation combinations pairwise, such that each of the n events in the set of event-relation combinations is at the head of one row and one column. Thus the matrix has rows i and columns j , where $i=1...n+1$ and $j=1...n+1$. Cells in E are denoted as e_{ij} . The contents of the matrix are the m relations present in the set of event-relation combinations, as well as their converses. Given a set of event-relation combinations including *A before B*, for example, the relation *before* would be placed in the cell where the row for event A intersects the column for event B, and the relation *after* would be placed where the row for B intersects the column for A. Given *A starts C*, *starts* would be placed at the intersection of the row for A and the column for C, and *started_by* would be placed at the intersection of the row for C and the column for A. By including the converse relations in the matrix, a single row can be searched to find all available information about a given event. Cells that represent a combination for which no relation is known are populated with the ‘~’ symbol. The matrix formed based on the event-relation combinations in the weather scenario is shown in Figure 3.4.

To generate a topological sort of the events in the set of event-relation combinations, the matrix E generated from the set is searched for any rows in which there are no relations R that, given $A R B$, map to B *before* A . That is, E is searched for rows that do not contain any *after*, *met_by*, *overlapped_by*, *started_by*, *during*, or *ends*

relations. Any relations present in such a row will map to A before B , and the event at the head of the row, when the mapping is performed, is only related to other events by *before*. For all rows found meeting this condition, the event at the head of the row is added to a list S , which becomes the initial topological sort.

$E =$

	LowPressureMoves	Rain	ColdFrontMoves	ScatteredShowers	WestWinds	NewLow	Showers
LowPressureMoves	~	<i>equals</i>	<i>contains</i>	~	<i>ended_by</i>	~	~
Rain	<i>equals</i>	~	~	~	~	~	~
ColdFrontMoves	<i>during</i>	~	~	<i>starts</i>	~	~	~
ScatteredShowers	~	~	<i>started_by</i>	~	~	~	~
WestWinds	<i>ends</i>	~	~	~	~	<i>before</i>	~
NewLow	~	~	~	~	<i>after</i>	~	<i>starts</i>
Showers	~	~	~	~	~	<i>started_by</i>	~

Figure 3.4 A matrix E generated from the relations present in the set of event-relation combinations from the example weather scenario, and their corresponding converse relations. The symbol \sim denotes the universal relation.

The initial topological sort is used as input for the algorithm that generates all possible orders of the events in a set of event-relation combinations. The rows and columns for the events added to S are deleted from the matrix. This process is repeated until E is empty, and all events in the set of event-relation combinations are included in the list S . The result is a single valid topological sort containing all the events from the set of event-relation combinations. Pseudocode for this process is below:

```

While  $E$  is not empty
{
  For all rows  $r=1..maxRows(E)$ 
  {
    Does row  $r$  contain any after, met_by, overlapped_by,
    started_by, during, or ends relations?
    if no
    {
      does  $r$  contain any equals relations?
      if no, add event at head of  $r$  to  $S$ 
      if yes
      {
        does the row(s) for the event(s) to which it
        is equal contain any after, met_by,
        overlapped_by, started_by, during, or ends
        relations?
        {
          if no, add event at head of  $r$  to  $S$ 
        }
      }
    }
  }
  Delete from  $E$  rows and columns for events that have been added
  to  $S$ 
}

```

The first pass through the matrix E for the example scenario returns two rows that contain no *after*, *met_by*, *overlapped_by*, *started_by*, *during* or *ends* relations. At the head of these rows are the events LowPressureMoves and Rain. Both rows contain an *equals* relation, but since they are *equal* to each other and neither row contains any *after* relations, they are both added to S . Thus after the first loop of searching the matrix for rows without *after* relations, $S=(LowPressureMoves, Rain)$. The two events, LowPressureMoves and Rain, are removed from the matrix (Figure 3.5). When events that are *equal* to each other are added to S , they are enclosed in parentheses.

$$E =$$

	ColdFrontMoves	ScatteredShowers	WestWinds	NewLow	Showers
ColdFrontMoves	~	<i>l_before</i>	~	~	~
ScatteredShowers	<i>l_after</i>	~	~	~	~
WestWinds	~	~	~	<i>l_before</i>	~
NewLow	~	~	<i>l_after</i>	~	<i>l_before</i>
Showers	~	~	~	<i>l_after</i>	~

Figure 3.5 The matrix E after removing LowPressureMoves and Rain

The search is repeated on the smaller, revised matrix, and returns the rows headed by ColdFrontMoves and WestWinds. These events are added to S , and their rows and columns are deleted. Now, $S=(LowPressureMoves, Rain), ColdFrontMoves, WestWinds$.

After another search through the matrix, S is updated to $S=(LowPressureMoves, Rain), ColdFrontMoves, WestWinds, ScatteredShowers, NewLow$. At this point, Showers is the only event left in the matrix, and there are no *after*, *met_by*, *overlapped_by*, *started_by*, *during*, or *ends* relations present in its row. Adding this last event to S , the matrix is emptied and the topological sort is complete. In this case, $S=(LowPressureMoves, Rain), ColdFrontMoves, WestWinds, ScatteredShowers, NewLow, Showers$.

The initial topological sort S is used as input for the ordering algorithm presented in the next section, which generates *all* possible orders of the events.

3.2.2 Performing a topological sort to generate all possible orders

Using the initial valid topological sort, as well as the set of event-relation combinations containing only the *l_before* and *l_equals* relations between events in the scenario, the ordering algorithm systematically transposes events in the initial topological sort until all orders that conform to the known relations have been generated (Section 2.2.2). The result is a set O , containing all possible orders of the n events in the set of event-relation combinations.

A scenario containing n events may generate at most $(n-1)!$ possible orders. For any set of n items, there are $n!$ permutations of the items. However, in this work there are relations between the n events (items) which put some constraints on the number of orders (permutations) that are possible. In any consistent set of event-relation combinations, there will be at least one event with no *after*, *met_by*, *overlapped_by*, *started_by*, *during*, or *ends* relations in its row in the matrix E . This event, then, is first in the order. If the remaining $(n-1)$ events have minimal constraints and are only related, for example, to the first event, there will be $(n-1)!$ possible orders.

The set of event-relation combinations for the weather example contained $n=7$ events, therefore the maximum number of possible orders would be $(7-1)! = 720$ orders. However, the events in the set of event-relation combinations in this case were related to each other such that the result of running the ordering algorithm on this example is a set O of only ten possible orders of the seven weather events.

$O =$

- 1 (LowPressureMoves, Rain), ColdFrontMoves, WestWinds, ScatteredShowers, NewLow, Showers
- 2 (LowPressureMoves, Rain) ColdFrontMoves, WestWinds, NewLow, ScatteredShowers, Showers
- 3 (LowPressureMoves, Rain) ColdFrontMoves, WestWinds, NewLow, Showers, ScatteredShowers
- 4 (LowPressureMoves, Rain) ColdFrontMoves, ScatteredShowers, WestWinds, NewLow, Showers
- 5 (LowPressureMoves, Rain) WestWinds, ColdFrontMoves, ScatteredShowers, NewLow, Showers
- 6 (LowPressureMoves, Rain) WestWinds, ColdFrontMoves, NewLow, ScatteredShowers, Showers
- 7 (LowPressureMoves, Rain) WestWinds, ColdFrontMoves, NewLow, Showers, ScatteredShowers
- 8 (LowPressureMoves, Rain) WestWinds, NewLow, ColdFrontMoves, ScatteredShowers, Showers
- 9 (LowPressureMoves, Rain) WestWinds, NewLow, ColdFrontMoves, Showers, ScatteredShowers
- 10 (LowPressureMoves, Rain) WestWinds, NewLow, Showers, ColdFrontMoves, ScatteredShowers }

The two events LowPressureMoves and Rain are enclosed in parentheses to show that they are *l_equal* to each other. Note that in all of the orders, LowPressureMoves and Rain are at the beginning of the order, followed by either ColdFrontMoves or WestWinds. The last event in every order is either Showers or ScatteredShowers. These kinds of consistencies are the result of relations between events that effectively tie an event to a range of positions within an order.

Differences between orders are the result of a comparative lack of information regarding the relations between a given event and other events. For example, NewLow is found in various positions in the orders, as is *ColdFrontMoves*. All orders in set O , however, conform to the known *before* relations present in the revised set of event-relation combinations.

For a given set O of possible orders generated from a set of event-relation combinations, though all known *before* relations between events are adhered to, it may

be that not all of the generated orders are equally plausible. For example, in the set of event-relation combinations for the detailed weather example (prior to mapping the relations to *l_before*) it was known that *ColdFrontMoves starts ScatteredShowers*. One would expect an event that *starts* another to immediately precede it in a linear order, with no intermediate events between the two. However, in six of the ten orders generated for the weather example (orders 1, 2, 3, 6, 7, and 9), there are one or more intermediate events between *ColdFrontMoves* and *ScatteredShowers*. The orders in which this is not the case are more plausible than the orders in which there are intermediate events, because they preserve the semantics of the relations between events. Using the semantics of the original relations can give insight on which of the orders are better and should be presented to a user.

3.3 Summary

This chapter discusses terminology and procedures necessary for collapsing the thirteen event interval relations to *before* and *equals* based on start and end points. It also introduces the steps involved in producing one initial topological sort based on the information given about a scenario. This initial topological sort is used as input for the algorithm that generates all possible orders of the events in the scenario. An example scenario is introduced as an illustration for the procedures discussed throughout the chapter. The end result, a set of several linear orders, is analyzed briefly and the necessity of further filtering is established.

Chapter 4 presents constraints based on the semantics of each of the thirteen relations. These constraints, when applied to the full set of orders generated by the ordering algorithm, will improve the quality of a final set of orders to be presented to a user. An algorithm is also introduced, which tests the orders in a set O for compliance with the constraints.

Chapter 4

REFINING ORDERS

BASED ON RELATION SEMANTICS

In the previous chapter, a methodology was developed for mapping all the relations in a set of event-relation combinations to *l_before* and *l_equals*. The resulting set of event-relation combinations was used as input for an ordering algorithm, which generated the set of all possible orders of the events. Though all of these orders are *possible* given the known information about relationships between events, not all of the orders are *plausible*. For example, while it was known from the set of event-relation combinations that ColdFrontMoves *starts* ScatteredShowers, and one would expect ScatteredShowers to immediately follow ColdFrontMoves in a linear order, there were several orders in the set O in which there were one or more intermediate events between ColdFrontMoves and ScatteredShowers. This chapter introduces a set of constraints that use the semantics of the known relations between events to eliminate less meaningful orders from the set O , increasing the overall plausibility of the resulting set of orders.

4.1 Relation semantics

Each temporal interval relation has semantics associated with it. For example, given a set of event-relation combinations including the relation *A meets B*, the start point of B exactly coincides with the end point of A (Allen 1984). When A and B are placed in a linear order, therefore, A should immediately precede B. An order in which there is an intermediate event between A and B would not be plausible. An intermediate event between A and B may result, however, when both *A meets B* and *A before C* are present in a set of event-relation combinations. In this case, two orders are possible: A B C and A C B. In the second order, event C falls between A and B. Similarly, given *A starts B*, it is intuitive that in a linear order A should immediately precede B and there should be no intermediate events between them. However, if the set of event-relation combinations includes *A starts B* and *C before B*, one of the two orders that results, A C B, has an intermediate event between A and B.

The remainder of this section discusses these types of semantics that are associated with each of the thirteen temporal interval relations. Semantics for each pair of converse relations are treated separately, beginning with *before* and *after*. We show how the semantics of the relations can be used to eliminate some of the more unlikely orders, and thus increase the plausibility of the final set of orders that will be presented to a user.

4.1.1 Semantics of *before* and *after*

The relation *before* is defined such that given A *before* B , both the start point and end point of A precede the start point and end point of B . This is consistent with the relationship between events in a linear order (Section 3.1). Given the relation A *before* B , in an order containing both A and B the start point and end point of A will precede the start point and end point of B , and the placement of other events within the order will not interfere with this relationship. That is, the *before* relation between A and B will hold in every order generated, regardless of whether there are intermediate events between A and B . Thus A *before* $B \rightarrow A \prec B$, read A *before* B maps to A *l_before* B . The same reasoning holds for the relation A *after* B , except that in this case the start point and end point of B precede both the start point and end point of A , and A *after* $B \rightarrow B \prec A$.

4.1.2 Semantics of *during* and *contains*

The relations *during* and *contains* capture cases where one event both starts and ends within the time that another event is occurring. That is, given A *during* B , the start point of B precedes the start point of A , and the end point of A precedes the end point of B . According to the mapping rules defined in Chapter 3, A *during* B maps to B *l_before* A in a linear order. Because event B takes place during the time that event A is happening, B should immediately follow A in a linear order. In order to preserve this semantics, and prevent intermediate events from coming between B and A , a constraint is applied stating that given A *during* B , no event may come between B and A in a

linear order. Formally, $A \text{ during } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$, read $A \text{ during } B$ maps to $B \text{ l_before } A$ and there does not exist a C such that $B \text{ l_before } C \text{ l_before } A$.

For cases where more than one event occurs *during* another event, for example, if $A \text{ during } B$ and $C \text{ during } B$, then *both* A and C cannot immediately follow B in a linear order. If the relationship between A and C is known, then A and C will immediately follow B in the order in which they occur. That is, if it is known that A is *l_before* C , then the order is $B A C$. If the relationship between A and C is unknown, however, then two orders are possible, $B A C$ and $B C A$.

In the case of nested *during* relations, such as $A \text{ during } B$ and $C \text{ during } A$, the events may be put in order and the semantic preserved for all events involved, as $B A C$.

The constraint for the *contains* relation is $A \text{ contains } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$.

In this thesis, the semantic of the *during/contains* relation is considered stronger than the semantics of other relations. Therefore, the constraint for the *during* relation takes precedence over constraints for other relations, that is, if the constraint for the *during* relation is in conflict with the constraint for another relation (Sections 4.1.3-4.1.7) the constraint for *during* will be upheld. The constraints for all relations except *during* and *contains* are considered to be of equal weight.

4.1.3 Semantics of *meets* and *met_by*

The relation *meets* is defined such that, given A *meets* B , the end point of A and the start point of B coincide. It is implausible that a third event would occur between A and B . Thus, a constraint is applied such that A *meets* $B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$.

An exception to this constraint occurs when a third event occurs *during* A . The event-relation combination A *meets* B maps to A *l_before* B according to the mapping rules described in Chapter 3, and D *during* A maps to A *l_before* D . Since the constraints for *during* take precedence over those for *meets*, in this case we say that if there exists an event or set of events D that occurs *during* event A , then D comes between A and B in a linear order. Formally, if $\exists D \mid D$ *during* A , then $A \prec D \prec B$.

If A *meets* B and B occurs *during* an event E , and it is not known that event A is also *during* E , E is allowed to come between A and B in a linear order. Thus, if $\exists E \mid B$ *during* E , and not A *during* E , then $A \prec E \prec B$.

In the case of A *met_by* B , the constraints are as follows: A *met_by* $B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If A *met_by* B and $\exists D \mid D$ *during* B , then $B \prec D \prec A$, and if $\exists E \mid A$ *during* E , and not B *during* E , then $B \prec E \prec A$.

4.1.4 Semantics of *overlaps* and *overlapped_by*

When one event *overlaps* another, the duration of the overlap is often unknown. It is possible that the overlap is very slight, and the two events almost *meet*. It is also possible that the overlap is large and one event almost *starts* or *ends* the other. Or the two events may nearly coincide, making it similar to an *equals* relation. Because of the

uncertainty of the nature of an *overlaps* relation, and because the semantics for each type of overlap are different, we say that for an *overlaps* relation, no additional constraints are applied. It is possible for intermediate events to come between two *overlapping* events in a plausible linear order. Thus $A \text{ overlaps } B \rightarrow A \prec B$, and in the case of *overlapped_by*, $A \text{ overlapped_by } B \rightarrow B \prec A$.

4.1.5 Semantics of *starts* and *started_by*

When one event *starts* another, the start points of the two events are simultaneous and the end point of one is before the other. The relation $A \text{ starts } B$ maps to $A \text{ l_before } B$ according to the mapping rules defined in Chapter 3. It is implausible, however, that in a linear order there should be a third event between A and B. Thus a constraint is applied to prevent this, and $A \text{ starts } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$.

An exception to this constraint occurs when, given $A \text{ starts } B$, there is an event *during* event A. Because the constraint for *during* takes precedence over that of *starts*, if there exists an event or set of events D such that $D \text{ during } A$, D comes between A and B in a linear order. That is, if $A \text{ starts } B$ and $\exists D \mid D \text{ during } A$, then $A \prec D \prec B$.

In the case of *started_by*, $A \text{ started_by } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ started_by } B$ and $\exists D \mid D \text{ during } B$, then $B \prec D \prec A$.

4.1.6 Semantics of *ends* and *ended_by*

In an *ends* relationship, two events have simultaneous end points, and the start point of one event precedes the start point of the other event. According to the mapping rules

defined in Chapter 3, the event-relation combination $A \text{ ends } B$ maps to $B \text{ l_before } A$. Another event cannot occur between B and A in a linear order, and a constraint is applied to keep this from happening. $A \text{ ends } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$.

However, given $A \text{ ends } B$, in the case where a third event or a set of events occurs *during* event B , the *during* constraint takes precedence and the *ends* constraint is revised to allow the third event or set of events to come between B and A in a linear order. If $A \text{ ends } B$ and $\exists D \mid D \text{ during } A$, then $B \prec D \prec A$.

For the case of *ended_by*, the constraint is $A \text{ ended_by } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $A \text{ ended_by } B$ and $\exists D \mid D \text{ during } B$, then $A \prec D \prec B$.

4.1.7 Semantics of *equals*

The *equals* relation, in which the start points and end points of two events are simultaneous, is preserved in linear orderings. Since the two events are temporally identical, relations that apply to one also apply to the other. The two events are listed together in a linear order, and placed according to the known relations of both events. This applies only to events that are listed as *equal* in the set of event-relation combinations; events that are not listed as *equal* in the set of event-relation combinations will not be *equal* in the set of linear orders. That is, $A \text{ equals } B \rightarrow A=B$, and if not $A \text{ equals } C$, then $\nexists C \mid A=C$.

The constraints for each of the thirteen temporal interval relations are summarized in Table 4.1.

Relation	Maps to Relation	Constraints
<i>A before B</i>	<i>A l_before B</i>	$A \text{ before } B \rightarrow A \prec B$
<i>A after B</i>	<i>B l_before A</i>	$A \text{ after } B \rightarrow B \prec A$
<i>A during B</i>	<i>B l_before A</i>	$A \text{ during } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$
<i>A contains B</i>	<i>A l_before B</i>	$A \text{ contains } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$
<i>A meets B</i>	<i>A l_before B</i>	$A \text{ meets } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $\exists D \mid D \text{ during } A$, then $A \prec D \prec B$. If $\exists E \mid B \text{ during } E$, and not $A \text{ during } E$, then $A \prec E \prec B$.
<i>A met_by B</i>	<i>B l_before A</i>	$A \text{ met_by } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ met_by } B$ and $\exists D \mid D \text{ during } B$, then $B \prec D \prec A$. If $\exists E \mid A \text{ during } E$, and not $B \text{ during } E$, then $B \prec E \prec A$.
<i>A overlaps B</i>	<i>A l_before B</i>	$A \text{ overlaps } B \rightarrow A \prec B$
<i>A overlapped_by B</i>	<i>B l_before A</i>	$A \text{ overlapped_by } B \rightarrow B \prec A$
<i>A starts B</i>	<i>A l_before B</i>	$A \text{ starts } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $A \text{ starts } B$ and $\exists D \mid D \text{ during } A$, then $A \prec D \prec B$.
<i>A started_by B</i>	<i>B l_before A</i>	$A \text{ started_by } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ started_by } B$ and $\exists D \mid D \text{ during } B$, then $B \prec D \prec A$.
<i>A ends B</i>	<i>B l_before A</i>	$A \text{ ends } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ ends } B$ and $\exists D \mid D \text{ during } A$, then $B \prec D \prec A$.
<i>A ended_by B</i>	<i>A l_before B</i>	$A \text{ ended_by } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $A \text{ ended_by } B$ and $\exists D \mid D \text{ during } B$, then $A \prec D \prec B$
<i>A equals B</i>	<i>A l_equals B</i>	$A \text{ equals } B \rightarrow A=B$, and if not $A \text{ equals } C$, then $\nexists C \mid A=C$.

Table 4.1 Mappings of all thirteen temporal interval relations to *l_before*, and constraints based on the semantics of each relation.

4.2 Ordering events with constraints

The constraints described above outline characteristics that a plausible order should follow, based on the semantics of the relations. For example, if a set of event-relation combinations includes *A meets B*, then there should be no events between A and B in an order.

Orders that comply with these constraints are consistent with the inherent semantics of the temporal interval relations. As such, orders that comply with the constraints are more plausible than those generated based only on the *l_before* and *l_equals* relations derived from the set of event-relation combinations.

Ideally, orders should comply with the constraints for every relation. However, situations may arise when it is impossible for a set of possible orders to comply with all the constraints. This type of situation occurs when the constraint for one relation makes it impossible for another relation's constraint to apply. For example, given *A meets B* and *A starts C*, the constraint for *meets* requires that event B immediately follow A, and the constraint for *starts* requires that event C immediately follow A. A linear order cannot comply with both of these constraints. We say that no constraint takes precedence over another except for *during*, and as a result, *multiple orders* are possible given that one of the constraints remains unviolated and the others are complied with as closely as possible. In the example of *A meets B* and *A starts C*, two possible orders result, *A B C* and *A C B* (Figure 4.1). In *A B C*, the constraint for *A meets B* is met, and event C follows immediately after B. In the order *A C B*, the constraint for *A starts C* is met, and B follows immediately after C.

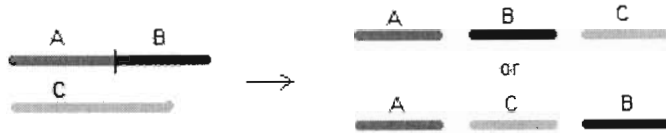


Figure 4.1 Given *A meets B* and *A starts C*, possible orders are *A B C* and *A C B*.

When constraints dictate that two events *A* and *B* should immediately follow one another in a linear order, another constraint may allow an intermediate event or set of events *C* to come between them. However, another event or set of events may not come between *A* and *B*, even if there is already an intermediate event between them. For example, given *A meets B* and *B during C*, *C* is allowed to come between *A* and *B*, but other events are not. Event *B* should follow as closely after event *A* as possible.

Due to the fact that the constraints cannot be tested one at a time, but depend on the other relations present in the set of event-relation combinations, orders are tested for compliance with the constraints once the ordering algorithm has generated the set *O* of all possible orders. The matrix *E* generated from the set of event-relation combinations is used to check the orders in *O* against the relation constraints. Beginning at the top left-hand corner of the matrix *E*, in cell $e_{1,1}$, the lower diagonal half of the matrix is parsed through to detect relations present in the set of event-relation combinations that have constraints that must apply in the set of plausible orders. Only the lower diagonal half of *E* needs to be searched, because the matrix is diagonally symmetrical. For every relation encountered, each order in the set *O* is checked to see that constraints for the relation are not violated. The entire matrix *E* is used in this checking process.

There are two cases in which an order may contain a violation of a constraint and still be maintained in the set of plausible orders. The first case is when an exception to a constraint applies, for example, given the event-relation combination *A meets B*, A and B should not be separated from each other in a linear order unless there is an event or set of events *during* A (Figure 4.2a), or if B is *during* another event which does not *contain* A (Figure 4.2b).

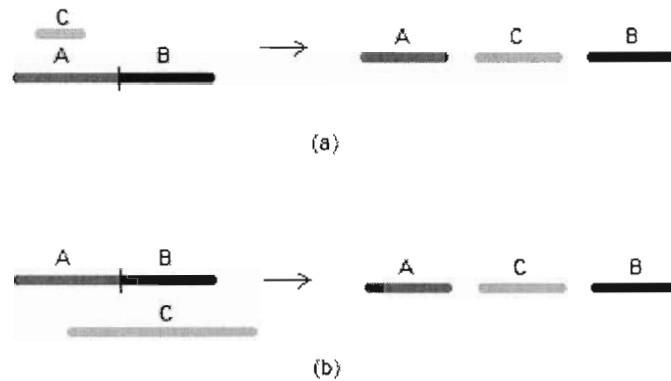


Figure 4.2 (a) Given *A meets B*, A and B may be separated in a linear order if another event occurs *during* event A, (b) given *A meets B*, A and B may be separated in a linear order if B is *during* another event which does not contain event A.

The second case is where the combination of two relations with constraints does not allow both constraints to apply, such as the case of *A meets B* and *A meets C*. In this case two orders are possible, A B C and A C B, both of which violate one constraint for the relation *meets*, but both of which are plausible.

The algorithm for checking the orders in O against the relation constraints has two parts, reflecting the two types of constraint violations that are permitted. When an order is found in which a constraint is violated, the first part of the algorithm checks the

matrix for exceptions to the constraint. For example, in the weather example introduced in Chapter 3, order 2 in the set O is $(LowPressureMoves, Rain) ColdFrontMoves, WestWinds, NewLow, ScatteredShowers, Showers$. One event-relation combination in the original set of event-relation combinations, however, was $NewLow$ *starts* $Showers$. Therefore, according to the constraints for the *starts* relation, no intermediate events should come between $NewLow$ and $Showers$ unless the intermediate event is related to either $NewLow$ or $Showers$ by a relation stronger than that of the constraint for the *starts* relation. However, in order 2 shown above, the event $ScatteredShowers$ is between $NewLow$ and $Showers$. In this situation, checks are performed to determine whether $ScatteredShowers$ was *during* $NewLow$, or whether $ScatteredShowers$ *contains* $NewLow$ and not $Showers$. Since the converse relations are present in the matrix, only one row or column needs to be searched for each of these checks. For example, in this scenario, the row for the event $ScatteredShowers$ would be searched for a *during* relation in its intersection with $NewLow$'s column, and for a *contains* relation in the intersection with $Showers$ ' column. If one of these exceptions applies, then the order is retained and the next order is checked for compliance with the constraint. If none of the constraint exceptions applies, the second part of the algorithm, a function called *EqualWeight*, is called.

The function *EqualWeight* checks other relations involving $ScatteredShowers$. First, a check is performed to determine whether event $ScatteredShowers$ relates to either $NewLow$ or $Showers$ by a relation with a constraint that is equal to or takes precedence over the relation between $NewLow$ and $Showers$, which would allow it to fall between

NewLow and Showers in an order. If this is the case, then the order is retained in the set of plausible orders. Where this is not the case, however, the order is not immediately discarded. The row in the matrix E headed by event ScatteredShowers is searched for any relations that have constraints of equal or greater weight than the relation between events NewLow and Showers. If none are found, then the order is discarded. Otherwise, the events to which ScatteredShowers is related are checked for relations to NewLow and Showers. This process is recursive, and is repeated until a dead end or a connection to NewLow or Showers is found. If a connection is found, the order is retained in the set of plausible orders. The order is discarded if a dead end is reached.

Pseudocode for both parts of the algorithm is shown below.

```

For each relation  $r$  in left diagonal half of matrix, where  $e1 \ r \ e2$ :
{
  If  $r = (meets, met\_by, during, contains, starts, started\_by,$ 
     $ends, ended\_by)$ 
    For each order in  $O$ 
      {
        Is there an intermediate event between  $e1$  and  $e2$ ?
        If yes, for each violating intermediate event
          {
            Does  $e$  qualify for an exception to the constraint?
            {
              If no, run EqualWeight( $e, e1, e2, r$ )
              {
                If Exception=False, discard order
              }
            }
          }
        }
      }
}

```

```
EqualWeight(events e, e1, e2, relation r)
```

Is there a relation r_1 in the row for e with a constraint equal to or stronger than that of r , where $e \ r_1 \ e_3$, and e_3 has not been checked?

```
{
  If no, Exception=False, end
  If yes, for each  $r_1$  in the row for  $e$ 
  {
    Is the relation to either  $e_1$  or  $e_2$ ?
    {
      If yes, Exception=True, end
      If no, run EqualWeight( $e_3, e_1, e_2, r$ )
    }
  }
}
```

The result of running this algorithm is a set O_I of all orders that preserve the semantics of each of the temporal interval relations present in the set of event-relation combinations.

When all the constraints are applied to the weather example introduced in Chapter 3, the resulting set O_I contains two orders. These were orders numbered 1 and 4 in the set O of all possible orders generated by the ordering algorithm:

$O_I =$

1 (LowPressureMoves, Rain), ColdFrontMoves, WestWinds, ScatteredShowers, NewLow, Showers

4 (LowPressureMoves, Rain), ColdFrontMoves, ScatteredShowers, WestWinds, NewLow, Showers }

4.3 Summary

In this chapter, implausible orders are eliminated from the set O based on semantics of the relations present in the set of event-relation combinations. Semantics of each of the

thirteen temporal interval relations were examined, and constraints based on the semantics developed. For example, the relation *A during B* maps to *B before A* and there does not exist a *C* such that *B before C before A*. Orders inconsistent with the constraints are considered implausible. These orders are removed from the set of orders presented to a user.

The method for applying the constraints to a set O of orders was presented, and pseudocode given and discussed. The constraints were applied to the set O of possible orders from the weather example introduced in Chapter 3, and to the burglary example. In both cases the resulting sets O_I were found to contain fewer orders than the sets O , and consisted only of orders that were plausible based on the established semantics of the temporal relations between events.

In Chapter 5, additional methods of filtering orders based on event locations will be discussed.

Chapter 5

USING EVENT LOCATION

IN DETERMINING ORDERS OF EVENTS

In previous chapters, we have discussed ordering scenarios of events in terms of the temporal relations between events. In Chapter 4, the semantics of temporal relations between events were used to refine the orders in the set O , increasing the plausibility of the orders derived. However, it is also possible that in addition to the semantics associated with temporal relations, there is spatial information associated with events. In cases where spatial information about events is available, event locations may be used to further refine the orders in O_I . This chapter explores two approaches for using event locations in order refinement. A strategy for dealing with mixed event location granularities is introduced, and the first filtering approach prunes out events such that only events within a certain area of spatial relevance are included in the orders. The second filtering approach tests for spatial patterns in the occurrence of event locations, and filters out less plausible orders from O_I based on these patterns.

5.1 Event location granularities

We extend the set of event-relation combinations to include the spatial location of events. For example, an event-relation combination describing the relationship between

a car theft and an armed robbery that occurred in San Antonio, Texas would appear as “CarTheft [San Antonio, TX] *after* ArmedRobbery [San Antonio, TX].”

An event’s location may be captured at different levels of detail: for example, a country, a state, a city, or a street address. In this work, we assume that any event location belongs to one of five categories: street address, landmark, city, state, or country. A landmark is defined in this work as a location that is coarser than a street address, but is not a city or town. For example, a park, a city’s downtown area, and a geographic feature such as a mountain would be considered landmarks. In location-based reasoning, locations at the street address, landmark, or city level are treated as point data, and locations at the state or country level are treated as area data. Event locations may also be unknown or missing, a topic which is discussed in further detail in Sections 5.2.2 and 5.3.3. Granularities of the five categories of known event locations vary from coarse to fine (Figure 5.1).

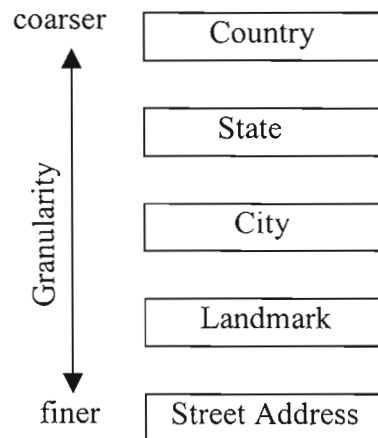


Figure 5.1 Hierarchy of event location granularities.

In order to perform computations based on event locations, all locations in a set of event-relation combinations must be translated to the same granularity. All event locations can simply be translated to the coarsest level represented in the set, but this may result in significant information loss. For example, there may be a case in which there are events whose locations are known at the city granularity, and other events that are described only at the granularity of country. Translating all the city locations to country takes away considerable detail. In this work, all event location granularities are coarsened only to the *most common location granularity* in the set of event-relation combinations, that is, the *modal* granularity.

The mode of a set of data is defined as the most frequently occurring value within the set. Every event location in a set falls into one of the five potential values of granularity level: *street address*, *landmark*, *city*, *state*, and *country*. Each of these is assigned a variable: *street_address*, *landmark*, *city*, *state*, and *country*, respectively. The value of each of these variables is the number of occurrences of that granularity level within the set. To determine the values of *street_address*, *landmark*, *city*, *state*, and *country*, for each of the n events in the set of event-relation combinations, the event location granularity is evaluated and the number of the appropriate granularity level variable is incremented.

```

For each event
  If granularity = street address
    Then street_address++;
  If granularity = landmark
    Then landmark++;
  If granularity = city
    Then city++;
  If granularity = state
    Then state++;
  If granularity = country
    Then country++;

```

At the end of the process, the granularity level whose variable has the highest value is the modal granularity.

As an example, consider the following set of event-relation combinations, describing an afternoon's activities:

```

PicnicLunch (Seal Harbor Village) meets TakePhoto (10 Lower Dunbar Road, Seal Harbor, ME)
PicnicLunch (Seal Harbor Village) before HorseRental (Wildwood Stables)
TakePhoto (10 Lower Dunbar Road, Seal Harbor, ME) before Homework (Fogler Library)
HorseRental (Wildwood Stables) contains Rest (Jordan Pond)
HorseRental (Wildwood Stables) contains MeetBikers (Eagle Lake)

```

The orders generated from this set of event-relation combinations are:

```

 $O_i = \{$  PicnicLunch TakePhoto HorseRental MeetBikers Rest Homework,
PicnicLunch TakePhoto HorseRental Rest MeetBikers Homework,
PicnicLunch TakePhoto Homework HorseRental MeetBikers Rest,
PicnicLunch TakePhoto Homework HorseRental Rest MeetBikers  $\}$ 

```

In this example, there are $n=6$ events and, therefore, there are also six event locations in the set. Five of these locations – Seal Harbor Village, Wildwood Stables, Fogler Library, Jordan Pond, and Eagle Lake – are at the **landmark** granularity level. One event location, 10 Lower Dunbar Road, Seal Harbor, ME, is at the **street address** level. Thus, the final variable values are *landmark=5* and *street_address=1*. Variables *city*, *state*, and *country* are equal to zero since no locations in this set of event-relation combinations were at those granularity levels. The modal granularity in this set is **landmark**, since *landmark=5 > street_address=1*. The event location at the **street address** granularity, 10 Lower Dunbar Road, Seal Harbor, ME, is coarsened to the **landmark** granularity, and becomes Seal Harbor Village. The set of event-relation combinations is revised to:

PicnicLunch (Seal Harbor Village) *meets* TakePhoto (Seal Harbor Village)
PicnicLunch (Seal Harbor Village) *before* HorseRental (Wildwood Stables)
TakePhoto (Seal Harbor Village) *before* Homework (Fogler Library)
HorseRental (Wildwood Stables) *contains* Rest (Jordan Pond)
HorseRental (Wildwood Stables) *contains* MeetBikers (Eagle Lake)

In this example, there were no event locations whose granularity is coarser than the modal granularity. In cases where there are locations at a coarser granularity than the modal granularity, however, these events are processed as though no location information is available, and are not included in location-based operations. In the case where there is more than one modal granularity (i.e., two levels of granularity are equally represented), event locations are translated to the coarsest granularity represented in the scenario.

5.2 Determining spatial relevancy of events

Given a set of event-relation combinations extended to include locations, we first determine whether all events are located such that they are spatially relevant to each other, or whether there are some events that are spatial outliers. In this work, a hierarchical clustering algorithm is applied to identify outlying event locations. This section describes the hierarchical clustering process as applied to event locations, and discusses the effect of its results on the resulting set of linear orders of events.

5.2.1 Hierarchical clustering

Hierarchical clustering is an agglomerative clustering technique. That is, each element is initially assigned its own cluster and sets of the two most similar clusters are merged until the appropriate number of clusters is reached. In the form of hierarchical clustering used in this work, *single linkage* clustering, the similarity between elements (event locations), is based on Euclidean distance between event locations (Johnson 1967; Bailey and Gatrell 1995). For each pair of event locations $el1$ and $el2$, this Euclidean distance is referred to as $d[el1,el2]$. Two clusters a and b are considered most similar if the distance between them is the minimum distance between any two clusters, that is, $\forall el1,el2 \mid el1 \neq el2: d[(a),(b)] = \min d[el1,el2]$. Distances between clusters are stored in an $n \times n$ proximity matrix D , which shows the Euclidean distance between each pair of event locations. Since the matrix D is symmetrical, only the top diagonal half of the matrix is populated. For the example introduced in the previous section, the matrix D is shown in Figure 5.2, with distances between locations in miles.

	Seal Harbor Village	Seal Harbor Village	Wildwood Stables	Jordan Pond	Eagle Lake	Fogler Library
D= Seal Harbor Village		0	1.5	2.4	4.2	53.9
Seal Harbor Village			1.5	2.4	4.2	53.9
Wildwood Stables				1	2.6	53
Jordan Pond					1.9	52.8
Eagle Lake						52.4
Fogler Library						

Figure 5.2 The $(n+1) \times (n+1)$ proximity matrix D , showing the Euclidean distance between each pair of event locations.

To begin the clustering algorithm, each of the n event locations is initially assigned its own cluster. Thus, every row and column in D represents a cluster. A search is conducted in the matrix D for the two clusters with the minimum distance between them, and these two closest clusters are merged. D is then updated to show this new cluster configuration, by merging the rows and columns of the two clusters that have been merged. The distance $d[old, new]$ between an old cluster old and a new cluster new , where new comprises old clusters a and b , is the smaller of the distances $d[(a)(old)]$ and $d[(b)(old)]$.

A *stopping rule* is used to determine when the clustering process should be terminated. In this work, we use the stopping rule defined by Calinski and Harabasz (Milligan and Cooper 1985), referred to as the *CH index*. The CH index is the ratio of the distance between clusters and the distance between events within clusters, accounting for the total number of events and the number of events within each cluster. The stopping rule indicates that a clustering process should be terminated when the CH index is at its maximum value. The equation for determining this ratio is based on B ,

the average distance between clusters; W , the average distance between events within clusters; n , the number of events; and k , the number of clusters (Equation 5.1).

$$\text{CH index} = \frac{B(n-k)}{W(k-1)} \quad (5.1)$$

At the end of the clustering process, the distribution of clusters is analyzed. If there is one single cluster, then all events are considered spatially relevant. If there is more than one cluster, however, the events in the largest cluster are considered spatially relevant, but events in other clusters are considered outlying events. Spatially relevant events are retained in the set O_I of linear orders, but outlying events are eliminated from the orders in O_I . The outlying events are eliminated from the orders and not from the set of event-relation combinations because, though spatial information may make an event less relevant to the orders, temporal information about the event is no less valid. For this reason, the event is still important to the ordering process.

When hierarchical clustering is performed on the current example, the result is that events taking place at Seal Harbor Village, Wildwood Stables, Jordan Pond, and Eagle Lake are all part of one cluster. Remaining in its own cluster is the event taking place at Fogler Library, due to the large distance between this event and the other five events. Thus, the event that takes place at the Fogler Library, *Homework*, is eliminated from the orders. The removal of *Homework* results in two sets of orders becoming identical. The identical orders are collapsed, and the set O_I is reduced to only two orders:

$O_I = \{ \text{PicnicLunch TakePhoto HorseRental MeetBikers Rest,} \\ \text{PicnicLunch TakePhoto HorseRental Rest MeetBikers} \}$

In the case where event locations are modeled as area data rather than point data, distances are measured between the boundaries of the event locations. Each event location is initially in its own cluster, and clusters are merged until the CH index reaches its maximum. Events contained in the largest cluster are retained in the orders, while others are eliminated.

There may be cases where, instead of a single largest cluster, there are two or more clusters containing a similar number of event locations. In this case, unless additional information is available, these clusters are assumed to be of equal importance, and all events contained in these clusters are included in the orders.

5.2.2 Cases where event locations are not specified

It is possible that in a set of event-relation combinations, some locations of events are unknown. In the case where no locations are known for any of the events in a set of event-relation combinations, no filtering based on locations is performed. In other cases, there may be location information for some events, but not all. In these cases, events without locations are automatically included in the linear orders in the set O_I . The events with locations are used in filtering events based on location, as described above.

5.3 Filtering orders based on event locations

Once outlying events have been eliminated from the orders in O_I , the orders themselves may be further refined. Given a set of event-relation combinations in which the locations of events are known, two cases are possible. The events may occur in locations such that no spatial order is apparent (Figure 5.3a), or the locations may be such that they exhibit a pattern showing a spatial order of locations (Figure 5.3b). A spatial order of locations may be linear, as in Figure 5.3b, or occur in some other pattern. Other patterns include, for example, events occurring along a nonlinear roadway, such as a loop road, or events occurring in a circular pattern. In this work, we examine the case where event locations show a linear distribution, and analysis of other possible patterns is considered as a topic for future work (Section 7.3.2).

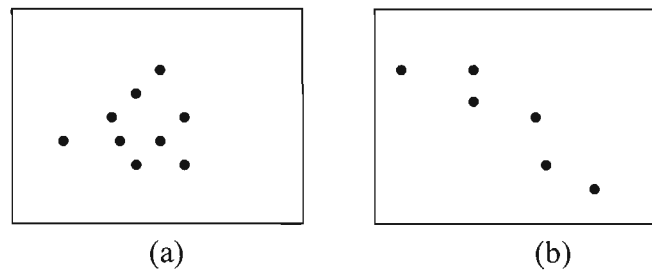


Figure 5.3 Arrangement of point locations in (a) nonlinear and (b) linear formations.

5.3.1 Detecting a linear pattern in event location data

The first step in filtering orders of events based on the spatial arrangement of event locations is to determine the degree to which the event locations exhibit a linear trend. Linear regression is used to evaluate the degree of linearity shown by the event locations. Linear regression is a statistical method that is applied to determine the

equation of a *line of best fit*, based on the x and y values of a given set of points. A correlation coefficient r is determined, which indicates the quality of the linear relationship between the x and y values. We apply least-squares regression, which minimizes the sum of the squares of the distance from each data point to the line in order to determine the equation of the line of best fit. The equation, in the form $y=mx+b$, gives the slope of the line m , and the y -intercept b , where x and y are in this case, the latitude and longitude of the locations. The value for m is determined using Equation 5.2, where n is the number of data points in the set, and \bar{x} is the average of all values of x present in the set. Equation 5.3 shows the formula for determining the value of the y -intercept, b , and Equation 5.4 the formula for the correlation coefficient r , in which \bar{y} is the average of all y values present in the set.

$$m = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - n\bar{x}^2} \quad (5.2)$$

$$b = \frac{\sum y - m \sum x}{n} \quad (5.3)$$

$$r = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sqrt{(\sum x^2 - n\bar{x}^2)(\sum y^2 - n\bar{y}^2)}} \quad (5.4)$$

The absolute value of r is always between 0 and 1, with values close to 1 indicating a very high linearity, and values close to 0 indicating that the linear relationship between the x and y values is very weak. In this work, linear regression is applied to each set of event locations, and if the absolute value of the correlation coefficient is

greater than 0.7, then the linear trend exhibited by the locations is considered significant (Kiemele, Schmidt et al. 1997).

As discussed in the previous section, locations associated with events will not always be in point form, as in the case of locations at the region or country granularity levels. When location information is at the region or country level, and thus in the form of area data as opposed to point data, the centroid of each location is computed, and linear regression is applied using these values (Bailey and Gatrell 1995).

5.3.2 Evaluating orders for correspondence with a linear trend

If, once linear regression has been applied on a set of points, the value of r is greater than 0.7, then we say that the set of event locations corresponds to a linear trend. The next step is to evaluate whether the linear trend apparent in the event locations corresponds to any of the temporal orders in the set O_I , because correspondence between the linear trend and orders in O_I would be evidence of a spatio-temporal order of events. Two lists, l_p and l_n , are formed by listing the events according to their location in positive and negative directions along the line of best fit. These lists show the two possibilities of spatial ordering exhibited by the event locations. It seldom happens, however, that all events fall exactly along a line of best fit (that is, that $r=1$). In cases where $r \neq 1$, a method is needed to assign each location a position on the line before l_p and l_n can be generated. A perpendicular line is drawn between each event location and the line of best fit. The place at which a location's perpendicular line meets the line of best fit is that location's assigned position on the line (Figure 5.4).

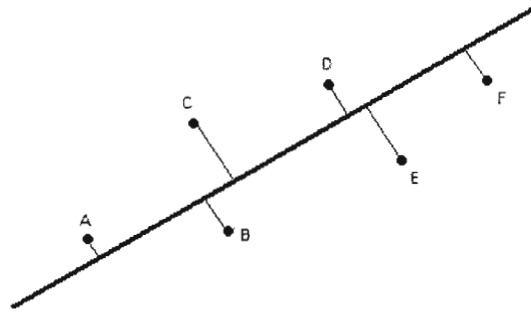


Figure 5.4 Perpendicular lines from event locations to a line of best fit, showing each event's assigned position along the line. The lists formed from these locations are $l_p = A, B, C, D, E, F$ and $l_n = F, E, D, C, B, A$.

The lists l_p and l_n are then compared with orders in the set O_I . If an order matches either l_p or l_n , then a spatial and temporal order coincide, and this is evidence of a *spatio-temporal sequence of events*. If there are orders in O_I that correspond with the spatial order of events, these orders are retained in O_I , but orders that do not correspond with the spatial order of events are eliminated from O_I . If none of the orders in O_I correspond with the spatial order, then the spatial linearity of the events is assumed to be unrelated to the events' temporal relations to each other. Events may occur in a linear arrangement due to other causes. For example, a series of robberies may occur in a line because several convenience stores are along a single road, but the stores may not have been robbed in order. In these cases, all orders are retained in O_I .

In the example of the afternoon's activities, when linear regression is run on the locations of the n events in the set of event-location combinations, $r=0.7557$. This is above the 0.7 significance threshold (Figure 5.5). The lists l_p and l_n , then, are:

$l_p = \text{MeetBikers, Rest, HorseRental, (PicnicLunch, TakePhoto)}$

$l_n = (\text{PicnicLunch, TakePhoto}), \text{HorseRental, Rest, MeetBikers}$.

Events *PicnicLunch* and *TakePhoto* occur at the same location.

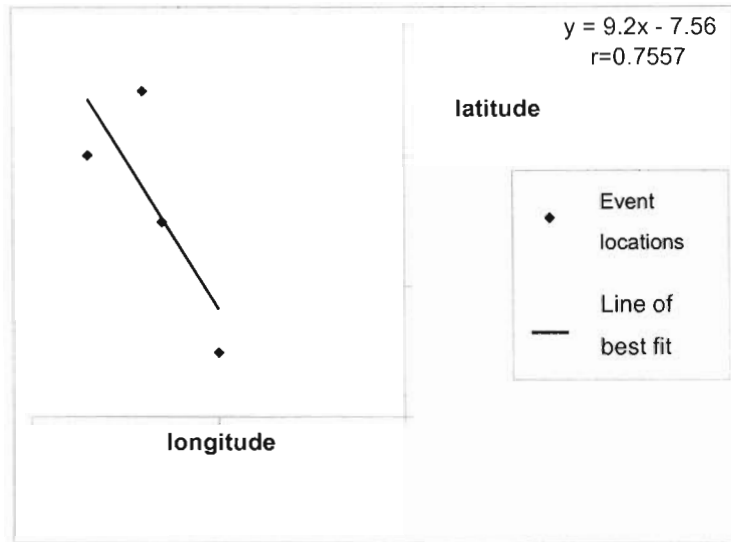


Figure 5.5 Plot showing event locations and line of best fit.

When the orders in O_I are compared with l_p or l_n , one order – *PicnicLunch TakePhoto HorseRental Rest MeetBikers* – is found to correspond with l_n . This order is retained in the set O_I , and the other order is eliminated. The set O_I now contains only one order:

$O_I = \{\text{PicnicLunch TakePhoto HorseRental Rest MeetBikers}\}$

5.3.3 Cases where event locations are not specified

In the case where no events have locations, all orders are retained in O_I . In cases where location information exists for some events, these event locations are used in testing for linearity. Events without locations are automatically included in the linear orders in the set O_I . This means that in the case of a significant linear trend in event locations, the lists l_p or l_n will not contain all the events present in the orders within O_I . Instead of checking for orders that match l_p or l_n , then, we check for orders containing l_p or l_n as a subset.

5.4 Summary

In this chapter, the methodology for determining linear orders of events was expanded to include event locations for refining orders. Hierarchical clustering was used to identify events that were not spatially relevant to other events, and these outlying events were eliminated from the orders in the set O_I . Linear regression was then used to identify linear trends in the distribution of event locations, and in cases where one or more orders in O_I corresponded with the spatial order of events, a spatio-temporal ordering of events was assumed. Orders in O_I that did not correspond with the spatial order were eliminated. The orders in the resulting set O_I contain only events that are spatially relevant to each other, and the possibility of a spatio-temporal sequence of events is accounted for.

The following chapter describes a method of evaluating the plausibility of orders in the set O_I , and presents conclusions based on the results of this evaluation.

Chapter 6

EVALUATION OF RESULTING ORDERS

In Chapter 3, a method was introduced for generating all possible orders of a set of events, given a set of event-relation combinations. Chapter 4 presented a series of constraints based on semantics of the temporal relations of events, which were used for refining the resulting set of orders. In Chapter 5, two methods are described for using event locations in further refinement of the orders. This chapter presents an evaluation of the results of these processes. Two test scenarios are used as the basis for evaluation, based on timelines found in *Timelines on File: The 20th Century* (Diagram 2000). A timeline is a linear order of events in which explicit temporal information is available for every event. The first test scenario includes sixteen events taking place at the end of World War I. Its set of event-relation combinations includes several *meets* and *during/contains* relations in addition to *before/after* relations. The second test scenario includes six events taking place before and during the Cold War. Its set of event-relation combinations includes *during/contains*, *starts*, and *ends* relations as well as *before/after* relations. Sets of event-relation combinations from each scenario are used to generate linear orders of events, and the constraints based on semantics of temporal intervals are applied. Each set of resulting orders is compared to the source's timeline, and an average percent similarity between orders in the set and the timeline is calculated. This measure describes the level to which orders in the set O_i agree with the

timeline. High percent similarity indicates that an order is very similar to the timeline. A low percent similarity indicates that an order is significantly different from the timeline, and suggests a lower level of plausibility.

6.1 Comparing orders

Each order in the set O_l is compared to the reference order (RO) based on the *longest common subsequence (LCS)* of events within the two orders. That is, a search is performed for the most events that occur in the same sequence within both orders. A set of events q is said to be a subsequence of an order r if the events in q occur sequentially in the order r . There may, however, be gaps between the events as they occur in r . For example, the set of events $q=A, B, D, F$ is a subsequence of the order $r=A, B, C, D, E, F$ because all the events in q occur sequentially within the order r . Formally, given $q=q_1, q_2, \dots, q_m$ and $r=r_1, r_2, \dots, r_n$, we say that q is a subsequence of r if there exists a set of events $1 < i_1 < i_2 < \dots < i_m < n$ such that for $1 < j < m$, $q_j = r_{i_j}$ (Eppstein 1996). This method of comparison is used in several disciplines, including molecular biology, where it is commonly used for comparison of genetic sequences (Needleman and Wunsch 1970; Smith and Waterman 1981; Booth et al. 2004).

The length of the longest common subsequence of two orders can be determined using an iterative algorithm (Needleman and Wunsch 1970). The algorithm creates a two-dimensional array, LCS, which has events from one order at the head of each column, and the events from the other order at the head of each row. Each cell in the array is populated, beginning at the bottom right-hand corner of the array. For a cell

$[i][j]$, the event at the head of column i and row j are compared, and if they are the same, then the value of $LCS[i][j]$ is $1+LCS[i+1][j+1]$. Otherwise $LCS[i][j] = LCS[i+1][j+1]$. When the array is full, the value in cell $LCS[0][0]$ is equal to the length of the longest common subsequence,

```

Length_LCS(Order r, Order s)
{
  For (i=m; i>=0; i--)
    For (j=n; j>=0; j--)
      {
        if (r[i] == s[j]), LCS[i][j] = 1 + LCS[i+1][j+1];
        Else L[i][j] = max(LCS[i+1][j] , LCS[i][j+1])
      }
  Return LCS[0][0];
}

```

Once the length of the longest common subsequence has been determined, a *percent similarity* between two orders r and s can be computed. This percent similarity is defined as the ratio of length of the longest common subsequence to the number of events in the orders (Equation 6.1).

$$PercentSimilarity(r,s) = \frac{Length[LCM(r,s)]}{Length(r)} \quad (6.1)$$

The percent similarity between two orders ranges from 0 to 1, with values close to zero indicating that the orders are nearly distinct, and values close to 1 indicating that the orders are very similar. In an evaluation of the closeness of the linear orders generated to the reference order, we would expect a similarity value close to 1.

6.2 First test scenario: World War I events

The first test scenario is a set of event-relation combinations based on a timeline of events taking place during the second half of World War I. There are fifteen events in this scenario, presented in Table 6.1 (Diagram Group 2000).

Feb-Dec 1916: Battle of Verdun – <i>Verdun</i>
May-Jun 1916: Battle of Jutland – <i>Jutland</i>
Jul-Nov 1916: Battle of the Somme – <i>Somme</i>
Feb 1917: Germany resumes unrestricted submarine warfare – <i>SubWarfare</i>
Apr 6, 1917: US declares war on Germany – <i>USDecWar</i>
Jun 1917: US troops begin landing in France – <i>USTroopsLand</i>
Jul-Nov 1917: Battle of Passchendaele – <i>Passchendaele</i>
Nov 1917: Russian Revolution – <i>Revolution</i>
Jan 8, 1918: Woodrow Wilson announces his Fourteen Points – <i>14Points</i>
Mar 1918: Germany launches first of its final three offensives – <i>Offensive</i>
May 1918: Allies stop Germans at Château-Thierry – <i>ChâteauThierry</i>
Jul-Aug 1918: 2 nd Battle of the Marne – <i>2ndMarne</i>
Sep 1918: Bulgaria surrenders to the Allies - <i>BulgariaSurrender</i>
Oct 1918: Allies defeat Austria-Hungary at Vittorio Veneto – <i>VittorioVeneto</i>
Nov 11, 1918: End of World War I – <i>EndWar</i>

Table 6.1 Events in World War I test scenario. Text in italics refers to short forms for the events.

Based on this timeline, the order in which events are presented in the source is:

{ Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points Offensive ChâteauThierry 2ndMarne BulgariaSurrender VittorioVeneto EndWar }

This order of events is the *reference order* (RO). A set of event-relation combinations are extracted from the temporal relations that are explicitly stated to hold

between these events, based on the temporal information explicit in the source. For example, the event *Verdun* occurred from February to December of 1916, and the event *Jutland* during May and June of 1916, thus *Verdun contains Jutland*. The event-relation combinations used to describe this scenario are:

<i>Verdun contains Jutland</i>	<i>Passchendaele before 14Points</i>
<i>Jutland meets Somme</i>	<i>Revolution ends Passchendaele</i>
<i>Somme before SubWarfare</i>	<i>Passchendaele before ChâteauThierry</i>
<i>Somme during Verdun</i>	<i>Passchendaele before Offensive</i>
<i>Verdun before SubWarfare</i>	<i>ChâteauThierry before 2ndMarne</i>
<i>SubWarfare before USDecWar</i>	<i>2ndMarne meets BulgariaSurrender</i>
<i>USDecWar before USTroopsLand</i>	<i>BulgariaSurrender meets VittorioVeneto</i>
<i>USTroopsLand meets Passchendaele</i>	<i>VittorioVeneto before EndWar</i>

The set of event-relation combinations that result from mapping all relations to *l_before* is:

<i>Verdun l_before Jutland</i>	<i>Passchendaele l_before 14Points</i>
<i>Jutland l_before Somme</i>	<i>Passchendaele l_before Revolution</i>
<i>Somme l_before SubWarfare</i>	<i>Passchendaele l_before ChâteauThierry</i>
<i>Verdun l_before Somme</i>	<i>Passchendaele l_before Offensive</i>
<i>Verdun l_before SubWarfare</i>	<i>ChâteauThierry l_before 2ndMarne</i>
<i>SubWarfare l_before USDecWar</i>	<i>2ndMarne l_before BulgariaSurrender</i>
<i>USDecWar l_before USTroopsLand</i>	<i>BulgariaSurrender l_before VittorioVeneto</i>
<i>USTroopsLand l_before Passchendaele</i>	<i>VittorioVeneto l_before EndWar</i>

An initial topological sort of the events is determined from this revised set of event-relation combinations based on the algorithm described in Section 3.2.1. The initial topological sort based on these event-relation combinations is:

```
{ Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele ChâteauThierry
14Points Revolution Offensive 2ndMarne BulgariaSurrender VittorioVeneto EndWar }
```

The topological sort algorithm is applied on this initial sort and the mapped event-relation combinations. The result is a set O of 336 possible orders. Applying the constraints based on the semantics of the temporal relations between events reduces the set of 336 possible orders to ten:

```
 $O_1 = \{1$  Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points
ChâteauThierry Offensive 2ndMarne BulgariaSurrender VittorioVeneto EndWar
```

```
2 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points
ChâteauThierry 2ndMarne BulgariaSurrender VittorioVeneto Offensive EndWar
```

```
3 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points
ChâteauThierry 2ndMarne BulgariaSurrender VittorioVeneto EndWar Offensive
```

```
4 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points
Offensive ChâteauThierry 2ndMarne BulgariaSurrender VittorioVeneto EndWar
```

```
5 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
14Points Offensive 2ndMarne BulgariaSurrender VittorioVeneto EndWar
```

```
6 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
14Points 2ndMarne BulgariaSurrender VittorioVeneto Offensive EndWar
```

```
7 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
14Points 2ndMarne BulgariaSurrender VittorioVeneto EndWar Offensive
```

```
8 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
2ndMarne BulgariaSurrender VittorioVeneto 14Points Offensive EndWar
```

```
9 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
2ndMarne BulgariaSurrender VittorioVeneto 14Points EndWar Offensive
```

```
10 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution ChâteauThierry
2ndMarne BulgariaSurrender VittorioVeneto EndWar 14Points Offensive }
```

When the orders in this final set O_I are compared with the RO (Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Revolution 14Points Offensive ChâteauThierry 2ndMarne BulgariaSurrender VittorioVeneto EndWar), the results are:

- Percent Similarity between the RO and order 4 is 100%.
- Percent Similarity between the RO and orders 1, 2, 3, and 5 is 93.3%.
- Percent Similarity between the RO and orders 6, 7, 8, 9 and 10 is 86.7%.

The average percent similarity between the orders in O_I and the RO is 90.7%. The average percent similarity between the RO and the initial 336 in the set O of all possible orders is 83.34%. The average percent similarity between the RO and the 326 orders removed as a result of the application of the semantic constraints is 83.1%. The average percent similarity is higher for the orders in O_I than for the orders in O . This indicates that the orders least similar to the RO were removed with the application of the semantic constraints.

Since locations of events in this scenario are known, location-based reasoning can be applied to this set of events. Based on the hierarchical clustering algorithm discussed in Section 5.2.1, two events are found to be less spatially relevant than the others. These events are Revolution and 14Points. When these two events are removed from the orders, three sets of orders become identical and are collapsed. These sets are orders 1 and 5, orders 2, 6, and 8, and orders 3, 7, 9 and 10. Thus, O_I becomes:

$O_I = \{1$ Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele ChâteauThierry Offensive
2ndMarne BulgariaSurrender VittorioVeneto EndWar

2 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele ChâteauThierry 2ndMarne
BulgariaSurrender VittorioVeneto Offensive EndWar

3 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele ChâteauThierry 2ndMame
BulgariaSurrender VittorioVeneto EndWar Offensive

4 Verdun Jutland Somme SubWarfare USDecWar USTroopsLand Passchendaele Offensive ChâteauThierry
2ndMame BulgariaSurrender VittorioVeneto EndWar }

The average percent similarity between orders in the revised O_I and the reference order is 81.7%. Events are not removed from the reference order when location-based reasoning is applied, and this means that the percent similarity between the reference order and the orders in O_I will decrease. This is because the orders in O_I now contain only thirteen events, while the reference order contains fifteen. The best possible similarity between the reference order and any order in O_I after location-based reasoning has been applied is 86.7%. A significant linear trend in the remaining event locations, based on linear regression as described in Section 5.3.1, is not found.

6.3 Second test scenario: Iron Curtain

The second test scenario involves six events, shown in Table 6.2 (Diagram Group 2000).

May 1945: End of World War II – <i>EndWWII</i>
1946-1949: Creation of Soviet satellite states in Eastern Europe – <i>SovietStates</i>
March 1946: Churchill gives a speech warning that Europe is being divided by an “Iron Curtain” - <i>IronCurtain</i>
1946-1991: Cold War – <i>ColdWar</i>
1989: Widespread antigovernment demonstrations in Eastern Europe – <i>AntiGov</i>
1990-1991: Iron Curtain descends – <i>EndIronCurtain</i>

Table 6.2 Events in Iron Curtain test scenario.

The reference order in this scenario is:

{EndWWII SovietStates IronCurtain ColdWar AntiGov EndIronCurtain}

Based on the dates given in the source, a set of event-relation combinations that describes this scenario is:

EndWWII *before* ColdWar
IronCurtain *starts* ColdWar
SovietStates *starts* ColdWar
AntiGov *during* ColdWar
EndIronCurtain *ends* ColdWar

When all relations are mapped to *before*, the revised set of event-relation combinations becomes:

EndWWII *l_before* ColdWar
IronCurtain *l_before* ColdWar
SovietStates *l_before* ColdWar
ColdWar *l_before* AntiGov
ColdWar *l_before* EndIronCurtain

The initial topological sort based on this set of event-relation combinations is {EndWWII IronCurtain SovietStates ColdWar AntiGov EndIronCurtain}.

When the revised set of event-relation combinations and the initial topological sort are input in the topological sorting algorithm, the result is a set of twelve possible orders:

$O = \{$ EndWWII SovietStates IronCurtain ColdWar AntiGov EndIronCurtain
 EndWWII SovietStates IronCurtain ColdWar EndIronCurtain AntiGov
 EndWWII IronCurtain SovietStates ColdWar AntiGov EndIronCurtain
 EndWWII IronCurtain SovietStates ColdWar EndIronCurtain AntiGov
 IronCurtain EndWWII SovietStates ColdWar AntiGov EndIronCurtain
 IronCurtain EndWWII SovietStates ColdWar EndIronCurtain AntiGov
 SovietStates EndWWII IronCurtain ColdWar AntiGov EndIronCurtain
 SovietStates EndWWII IronCurtain ColdWar EndIronCurtain AntiGov
 SovietStates IronCurtain EndWWII ColdWar AntiGov EndIronCurtain
 SovietStates IronCurtain EndWWII ColdWar EndIronCurtain AntiGov
 IronCurtain SovietStates EndWWII ColdWar AntiGov EndIronCurtain
 IronCurtain SovietStates EndWWII ColdWar EndIronCurtain AntiGov $\}$

The application of the constraints based on semantics of temporal interval relations reduces the set to only two:

$O_1 = \{$ 1 EndWWII SovietStates IronCurtain ColdWar AntiGov EndIronCurtain
 2 EndWWII IronCurtain SovietStates ColdWar AntiGov EndIronCurtain $\}$

For the Iron Curtain test scenario, similarity results are:

- Percent Similarity between the reference order and order 1 is 83.3%.
- Percent Similarity between the reference order and order 2 is 100%.

The average similarity in this scenario between the reference order and orders in O_1 is 91.7%. The average similarity between the reference order and the twelve possible orders in O is 75%. The average similarity of the ten orders in O that were not included in O_1 is 71.7%. Again, the percent similarity to the reference order was higher on average for orders in O_1 than for orders in O , showing that the least plausible orders were removed from O with the application of the semantic constraints.

6.4 Hypothesis evaluation

The hypothesis of this thesis is that *constraints that enforce the original semantics of the event interval relations are necessary in the generation of plausible linear orders.*

The results of this evaluation support the hypothesis; in both test scenarios, the average percent similarities between the reference orders and the orders in O_I were higher than the average percent similarity between the reference order and the orders in O before constraints were applied. The average percent similarity between the orders removed because of semantic constraints and the reference orders was in both cases lower than the average similarity between the reference order and the orders in set O , as well as the orders in the set O_I . The plausibility of the orders in the final set O_I was on average greater than the plausibility of orders prior to the constraints being applied.

6.5 Summary

This chapter introduced two example scenarios of events that had been put in order by an external source, and showed the results of generating possible linear orders from the sets of event-relation combinations describing these scenarios. Orders in the resulting set O were then compared to the reference order based on their longest common subsequence. A measure of similarity between orders and the reference order, percent similarity, was introduced and used to evaluate the average similarity of events in O with the reference order. The results of comparing the orders in the set O with the reference order showed that the average percent similarity was improved with the

application of the constraints based on the semantics of the temporal relations between events.

The following chapter presents a summary of the thesis and describes the methodology for generating linear orders from a set of event-relation combinations, as well as filtering that can be applied to the orders generated. These filters are based on the semantics of the temporal relations between events, and location information about events. Directions for future work, including the use of explicit temporal information in ordering, and the automatic extraction of event-relation combinations from text, are discussed.

Chapter 7

CONCLUSIONS

This thesis presents the steps necessary for generating linear orders of events, given a set of event-relation combinations that are partially ordered. A method of refining orders based on semantics of the temporal relations is also established, in order to increase the plausibility of the set of orders returned to a user.

This chapter summarizes the work described in this thesis, and concludes with a discussion of directions for possible future work.

7.1 Summary

Events in the world are not linear. For example, any two events may occur such that one is *during* the other, or the two events may *meet* or *overlap*. There are thirteen possible relations that may hold between two event intervals (Section 1.1). However, a linear order is a useful way of summarizing and describing scenarios of real-world events. This thesis describes the methods for automatically generating linear orders based on descriptions of geographic events.

Events in this thesis are assumed to be presented in the form of *event-relation combinations*, in the form $e1 R e2$, where $e1$ and $e2$ are events, and R is one of Allen's thirteen temporal interval relations. The event-relation combinations occur, most likely, as a partial order, i.e., not all information about the relations between all events in the

set is available. In order to generate a linear order, the relations between events must be mapped from one of the possible thirteen interval relations to l_before or l_equals . In this way, the set of event-relation combinations can be translated to a set of event-relation combinations in which the relations between events are all linear.

Event-relation combinations $e1 R e2$ in which the start point of $e1$ precedes the start point of $e2$ ($R = before, overlaps, contains, meets, \text{ or } ended_by$) map to $e1 l_before e2$. For event-relation combinations $e1 R e2$ in which the start point of $e2$ precedes the start point of $e1$ ($R = after, overlapped_by, during, met_by, \text{ or } ends$) the mapping is $e2 l_before e1$. In the case of event-relation combinations involving $starts$ and $started_by$, the start points of the events are equal. In this case, the end point of the events is used as the basis for the mapping. The event-relation combination $e1 starts e2$ maps to $e1 l_before e2$, and $e1 started_by e2$ maps to $e2 l_before e1$. In the case where two events are equal, as in $e1 equals e2$, the two events remain equal in our linear orderings. Therefore $e1 equals e2$ maps to $e1 l_equals e2$. The mapping of each event interval relation to l_before is shown in Table 7.1.

Relation	Maps to Relation	Relation	Maps to Relation
$A \text{ before } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ after } B$	$\rightarrow B \text{ } l_before \text{ } A$
$A \text{ contains } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ during } B$	$\rightarrow B \text{ } l_before \text{ } A$
$A \text{ meets } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ met_by } B$	$\rightarrow B \text{ } l_before \text{ } A$
$A \text{ overlaps } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ overlapped_by } B$	$\rightarrow B \text{ } l_before \text{ } A$
$A \text{ starts } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ started_by } B$	$\rightarrow B \text{ } l_before \text{ } A$
$A \text{ ended_by } B$	$\rightarrow A \text{ } l_before \text{ } B$	$A \text{ ends } B$	$\rightarrow B \text{ } l_before \text{ } A$
		$A \text{ equals } B$	$\rightarrow A \text{ } l_equals \text{ } B$

Table 7.1 Mapping rules to l_before and l_equals for each of the thirteen relations.

7.1.1 Constraints based on semantics of relations

When the set of event-relation combinations has been revised in this way, the set may be used, along with one topological sort of the events in the set, as input for a topological sorting algorithm. A topological sort is a linear order of events in a set such that, for any event-relation combination a *l_before* b , event a precedes event b in the order. The topological sorting algorithm generates a set of all possible orders of the events in the set of event-relation combinations, given the temporal relations between the events. There may be multiple possible orders; the highest possible number of orders generated by the topological sorting algorithm, given n events in the set of event-relation combinations, is $(n-1)!$.

Not all orders generated by the topological sorting algorithm, however, are equally plausible. In some orders, for example, there may be intermediate events between two events that are related to each other by *meets*. Similarly, two events related by a *during* relation may be separated by one or more intermediate events in an order. This is contrary to the semantics of the relations *meets* and *during*, as one would not expect intermediate events to occur between events related by either *meets* or *during*. To prevent this kind of implausibility, a set of constraints is developed, based on the semantics of the thirteen temporal relations. These constraints are presented in Table 7.2. When these constraints have been applied to the set of all possible orders of events, the result is a set of orders, O , in which the semantics of the original relations between events have been preserved.

Relation	Maps to Relation	Constraints
<i>A before B</i>	<i>A l_before B</i>	$A \text{ before } B \rightarrow A \prec B$
<i>A after B</i>	<i>B l_before A</i>	$A \text{ after } B \rightarrow B \prec A$
<i>A during B</i>	<i>B l_before A</i>	$A \text{ during } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$
<i>A contains B</i>	<i>A l_before B</i>	$A \text{ contains } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$
<i>A meets B</i>	<i>A l_before B</i>	$A \text{ meets } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $\exists D \mid D \text{ during } A$, then $A \prec D \prec B$. If $\exists E \mid B \text{ during } E$, and not $A \text{ during } E$, then $A \prec E \prec B$.
<i>A met_by B</i>	<i>B l_before A</i>	$A \text{ met_by } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ met_by } B$ and $\exists D \mid D \text{ during } B$, then $B \prec D \prec A$. If $\exists E \mid A \text{ during } E$, and not $B \text{ during } E$, then $B \prec E \prec A$.
<i>A overlaps B</i>	<i>A l_before B</i>	$A \text{ overlaps } B \rightarrow A \prec B$
<i>A overlapped_by B</i>	<i>B l_before A</i>	$A \text{ overlapped_by } B \rightarrow B \prec A$
<i>A starts B</i>	<i>A l_before B</i>	$A \text{ starts } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $A \text{ starts } B$ and $\exists D \mid D \text{ during } A$, then $A \prec D \prec B$.
<i>A started_by B</i>	<i>B l_before A</i>	$A \text{ started_by } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ started_by } B$ and $\exists D \mid D \text{ during } B$, then $B \prec D \prec A$.
<i>A ends B</i>	<i>B l_before A</i>	$A \text{ ends } B \rightarrow B \prec A$ and $\nexists C \mid B \prec C \prec A$. If $A \text{ ends } B$ and $\exists D \mid D \text{ during } A$, then $B \prec D \prec A$.
<i>A ended_by B</i>	<i>A l_before B</i>	$A \text{ ended_by } B \rightarrow A \prec B$ and $\nexists C \mid A \prec C \prec B$. If $A \text{ ended_by } B$ and $\exists D \mid D \text{ during } B$, then $A \prec D \prec B$
<i>A equals B</i>	<i>A l_equals B</i>	$A \text{ equals } B \rightarrow A=B$, and if not $A \text{ equals } C$, then $\nexists C \mid A=C$.

Table 7.2 Mapping rules and constraints for each event interval relation.

7.1.2 Filtering orders based on spatial information

There may be cases in which there is spatial as well as temporal information about some or all events in a set of event-relation combinations. In these cases it is possible to further refine the orders in the set O . A hierarchical clustering technique is used to prune events that are not spatially relevant to the scenario from the generated orders.

In some cases, events may occur in a spatial order as well as a temporal order. An example is the case where events occur along a linear trajectory. This kind of spatial order of events can be used to further filter orders from the set O . Linear regression is used to determine whether there is a significant linear trend in the locations of events within a set, and to calculate the equation of a line of best fit among the events. If a significant linear trend is present, the spatial orders of events (one in each direction along the line of best fit) are compared to the orders in O . If any orders in O match one of the spatial orders of events, they show evidence of a spatio-temporal sequence of events, and are considered to be plausible. These orders are retained in the final set of orders presented to a user.

7.1.3 Evaluation

A method of comparing the results of the ordering methodology (the orders in the set O) with a reference order was developed to evaluate the veracity of the orders in the final set O . The ratio of the length of the longest subsequence of events between the two orders and the length of the reference order was used as a measure of similarity between two orders. Ratios close to one indicate a high degree of similarity, while ratios close to zero indicate a low level of similarity. Orders that are very similar to the reference order are considered more plausible than orders with a low degree of similarity.

7.2 Conclusions

The major results of this thesis are:

- The set of constraints based on the semantics of temporal intervals, which assist in filtering orders from the set of all possible linear orders of events.
- The set of mapping rules, which translate any of the thirteen temporal interval relations to *l_before* or *l_equals*.
- In certain cases, such as sequences of events that occur in a linear trajectory, it is necessary to account for the spatial locations of events *in addition to* their temporal relations in order to prune irrelevant orders from the set of results.

These results can be used to answer the research questions posed in Chapter 1. The beginning questions were: *What criteria should be used to determine which event or pair of events begins an order? How are subsequent event intervals placed in the order?* and *Do the orders generated retain the semantics of the original event scenario?* The topological sorting algorithm determines which event in a set of event-relation combinations should begin an order, and how subsequent events should be placed in the order. The semantics of the original event scenario, however, are not always preserved in the resulting set of all possible linear orders of events, and so for these cases additional filtering is necessary.

This brings us to the next two research questions posed in Chapter 1, *Are all orders equally plausible?* and *Which orders should be presented to a user?* Some, but not all,

orders in the set of all possible linear orders retain the semantics of the original scenario of events. Thus, all orders generated by the topological sorting algorithm are not equally plausible. Tests are applied to be sure that only the orders which do retain the semantics of the original scenario of events are presented to a user.

Another research question, *Are there meaningful filters that could be applied, either to the event description or to the orders generated from it, that would render the results more meaningful to a user?*, requires a solution to the problem of unequal plausibility within the set of all possible linear orders. In this case, constraints based on semantics of the thirteen temporal interval relations were used to filter out less plausible orders, resulting in a set O_I containing only the orders that retain the semantics of the original scenario of events. Spatial information about events, when available, can also be used for filtering events and orders from the set O_I , such that the resulting set of orders contains only spatially relevant events. If the event locations exhibit a significant linear trend, the set of orders can be pruned to include only orders which show evidence of a spatio-temporal sequence.

The veracity of the results of these procedures was evaluated by running the topological sorting algorithm on test scenarios where an order was provided by an external source, and applying the filtering methods outlined in this thesis on the resulting set of possible linear orders. Comparison of the orders in the resulting set O_I with the pre-existing order of events showed that the orders in the set O_I after all filtering methods have been applied show a consistently higher average percent similarity with the reference order than the orders in the set O . For this reason, we

conclude that constraints based on semantics of event interval relations do increase the plausibility of orders presented to the user. This supports the hypothesis of this thesis: *Constraints that enforce the original semantics of the event interval relations are necessary in the generation of plausible linear orders.*

7.3 Future work

This thesis considered methods of generating plausible linear orders based on a set of event-relation combinations that are partially ordered. Methods of generating all possible orders were established, and filtering procedures were developed to increase the plausibility of orders presented to a user. This section discusses directions for future work that builds on the work presented in this thesis.

7.3.1 Use of explicit temporal information in the ordering process

In this thesis, it was assumed that no explicit temporal information about events, such as dates or times, were available. However, there may be cases where some temporal information is available, such as the case where certain events are time-stamped. In these cases, temporal information should be used to further increase the plausibility of orders returned to a user.

7.3.2 Detection of other spatial patterns of event locations

In this work, linear regression was used to determine whether event locations in a set of event-relation combinations exhibited a significant linear trend. The orders in the set O

were then tested for evidence of a spatio-temporal sequence. However, spatial patterns of event locations are not always in the form of a straight line. For example, events may occur along a network such as street or electrical network, in a curved or looped pattern, or back and forth between two locations. Methods of detecting these kinds of spatial patterns would provide additional opportunities for filtering orders from the set O .

7.3.3 Further evaluation of the *overlaps/overlapped_by* relations

The *overlaps* and *overlapped_by* relations were treated in this thesis in the same manner as the *before* and *after* relations, i.e., no additional constraints are applied. The justification for this is that when two events are related by *overlaps* or *overlapped_by*, there are many uncertainties as to the semantics of the relation between the two events. In the case where the overlap between the two events is very small, the *overlaps* relation may closely resemble a *meets* or *before* relation. In cases where the overlap is large, however, the relation between the two events may closely resemble a *starts*, *ends*, or *equals* relation. Since each of these possibilities comes with its own semantics, no constraints based on the semantics of *overlaps* or *overlapped_by* were applied in this work. However, further research into the patterns of semantics of the *overlaps/overlapped_by* relation would be beneficial, leading to additional constraints that may be applied to the orders generated by the topological sorting algorithm, and further increasing the plausibility of the orders presented to a user.

7.3.4 Automatic extraction of event-relation combinations

The work in this thesis assumed the existence of a set of event-relation combinations describing a scenario of geographic events that are partially ordered. Methods to derive the set of event-relation combinations from a narrative, for example, were not discussed. While there is a growing literature on extracting this type of information from text (Pustejovsky et al. 2003), automatic event extraction remains an open and challenging research area, and one that will be essential in developing on the ideas presented in this thesis.

BIBLIOGRAPHY

- Alfonseca, E. and S. Manandhar (2002). *A framework for constructing temporal models from texts*. FLAIRS Conference, Pensacola Beach, FL, AAAI Press: 456-460.
- Allan, J., R. Gupta and V. Khandelwal (2001). *Temporal Summaries of News Topics*. SIGIR'01, New Orleans, Louisiana, ACM: 10-18.
- Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11): 832-843.
- Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence* 23: 123-154.
- Allen, J. (1991). *Planning as temporal reasoning*. Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA: 3-14.
- Allen, J. (1991). Time and time again: the many ways to represent time. *International Journal of Intelligent Systems* 6(4): 341-355.
- Allen, J. and G. Ferguson (1994). Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation* 4(5): 531-579.
- Bailey, T. and A. Gatrell (1995). *Interactive Spatial Data Analysis*. Essex, Longman Group Limited.
- Booth, H., S. MacNamara, O. Nielsen and S. Wilson (2004). An Iterative Approach to the Longest Common Subsequence. *Bioinformatics* (submitted).
<http://wwwmaths.anu.edu.au/~booth/mcap_paper.pdf>

- Chakravarthy, S. and D. Mishra (1994). Snoop: An Expressive Event Specification Language for Active Databases. *Knowledge & Data Engineering Journal* 14(1): 1-26.
- Charniak, E. (1991). Bayesian networks without tears. *AI Magazine* 12(4): 50-63.
- Diagram Group (2000). *Timelines on File*. New York, Facts on File.
- Eppstein, D. (1996). Longest Common Subsequences.
<<http://www.ics.uci.edu/~eppstein/161/960229.html>>.
- Frank, A. (1998). Different types of "times" in GIS. *Spatial and Temporal Reasoning in Geographic Information Systems*. M. J. Egenhofer and R. G. Golledge, eds. Oxford, Oxford University Press: 40-62.
- Fujimoto, R. (1999). *Exploiting temporal uncertainty in parallel and distributed simulations*. Workshop on Parallel and Distributed Simulation, Atlanta, GA: 46-53.
- Galton, A. and J. Augusto (2002). *Two approaches to event definition*. 13th International Conference on Database and Expert Systems Applications (DEXA'02), Aix en Provence, France: 547-556.
- Gehani, N., H. Jagadish and O. Shmueli (1992). *Event specification in an active object-oriented database*. 1992 ACM SIGMOD International Conference on Management of Data, San Diego, CA: 81-90.
- Grenon, P. and B. Smith (2004). SNAP and SPAN: Towards dynamic spatial ontology. *Spatial Cognition and Computation* 4(1): 69-103.

- Grishman, R., S. Huttunen and R. Yangarber (2002). *Real-Time Event Extraction for Infectious Disease Outbreaks*. Proceedings of Human Language Technology Conference, San Diego, CA.
- <<http://nlp.cs.nyu.edu/publication/papers/grishman-hlt02.pdf>>
- Hinze, A. and A. Voisard (2002). *A flexible parameter-dependent algebra for event notification services*. Technical Report Number tr-b-02-10, Freie Universität Berlin.
- Johnson, S. (1967). Hierarchical Clustering Schemes. *Psychometrika* 32(3): 241-254.
- Kainz, W., M. Egenhofer and I. Greasley (1993). *Modeling spatial relations and operations with partially ordered sets*. International Journal of Geographical Information Systems 7(3): 215-229.
- Kiemele, M., S. Schmidt and R. Berdine (1997). *Basic Statistics*. Colorado Springs, Air Academy Press.
- Knight, K. and D. Marcu (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence* 139: 91-107.
- Knuth, D. and J. Szwarcfiter (1974). A structured program to generate all topological sorting arrangements. *Information Processing Letters* 2(6): 153-157.
- Koen, D. and W. Bender (2000). Time Frames: Temporal augmentation of the news. *IBM Systems Journal* 39(3&4): 597-616.
- Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 21(7): 558-565.

- Larson, R. (2003). Time and event measure. *Philosophical Perspectives 17, Language and Philosophical Linguistics*. J. Hawthorne and D. Zimmerman, eds. Oxford, Oxford University Press: 247-258.
- Lipschutz, S. and M. Lipson (1997). *Schaum's outline of theory and problems of discrete mathematics*. New York, McGraw-Hill.
- Mani, I. and M. Maybury, Eds. (2001). *Advances in Automatic Text Summarization*. Cambridge, Massachusetts, MIT Press.
- Milligan, G. and M. Cooper (1985). An examination of procedures for determining the number of clusters in a dataset. *Psychometrika* 50(2): 159-179.
- Motakis, I. and C. Zaniolo (1995). A formal semantics for composite temporal events in active database rules. *Journal of Systems Integration* 7(3-4): 291-325.
- Needleman, S. and C. Wunsch (1970). General method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48: 443-453.
- Pedersen, A. and D. Wright (2002). Do differences in event descriptions cause differences in duration estimates? *Applied Cognitive Psychology* 16: 769-783.
- Proveti, A. (1996). Ordering events: Intervals are sufficient, more general sets are usually not necessary. *Reliable Computing* 2(3): 321-327.
- Pustejovsky, J., J. Castano, R. Ingria, R. Sauri, R. Gaizauskas, A. Setzer and G. Katz (2003). *TimeML: Robust Specification of Event and Temporal Expressions in Text*. AAAI Spring Symposium, Palo Alto, CA, AAAI Press: 28-34.
- Ruskey, F. (1995). Combinatorial Object Server: Linear Extensions. <<http://www.theory.csc.uvic.ca/~cos/inf/pose/LinearExt.html>>.

- Skiena, S. (1998). *The Algorithm Design Manual*. Santa Clara, CA, TELOS - the Electronic Library of Science.
- Smith, T. and M. Waterman (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147: 195-197.
- Varol, Y. and D. Rotem (1981). An algorithm to generate all topological sorting arrangements. *The Computer Journal* 24(1): 83-84.
- Yang, Y., Carbonell, R. Brown, T. Pierce, B. Archibald and X. Liu (1999). Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems* 14(4): 32-43.
- Yuan, M. (2001). Representing complex geographic phenomena in GIS. *Cartography and Geographic Information Science* 28(2): 83-96.
- Zacks, J. and B. Tversky (2001). Event Structure in Perception and Conception. *Psychological Bulletin* 127(1): 3-21.

BIOGRAPHY OF THE AUTHOR

Suzannah Hall was born on May 28, 1980 in Bangor, Maine. She graduated from Mt View High School in Thorndike, ME in June 1998. She obtained her Bachelor of Science in Spatial Information Engineering at the University of Maine in May 2002, and began work on her Master's degree in Spatial Information Science and Engineering in the summer of 2002. There she worked as a graduate research assistant with the Department of Spatial Information Science and Engineering. Suzannah is a candidate for the Masters of Science degree in Spatial Information Science and Engineering from the University of Maine in December of 2004.